CS 4476/5413 Lecture Notes

TOPICS IN

APPLIED COMBINATORICS

Ruizhong Wei

Department of Computer Science
Lakehead University

Winter, 2013

Contents

Chapter 1

Orthogonal Arrays

1.1 Latin squares

Definition 1.1.1 *A* latin square of side n *(or order n) is an $n \times n$ array in which each cell contains a single element from an n-set S, such that each element occurs once in each row and exactly once in each column.*

A *transversal* in a latin square of side n is a set of n cells, one from each row and column containing each of the n symbols exactly once.

An example of latin square of side 8:

```
01234567
10345672
23506741
45617023
56720314
67452130
72163405
```

Theorem 1.1.2 *[3] Let $L(n)$ denote the number of distinct latin squares of side n. Then $L(n) \sim (e^{-2}n)^{n^2}$ for $n \to \infty$.*

A latin square is equivalent to an algebraic object called quasigroup, as define below.

Definition 1.1.3 *Let X be a finite set of cardinality n, and let $\circ$ be a binary operation defined on X. We say that the pair $(X, \circ)$ is a quasigroup of order n provided that the following properties are satisfied:*

1. *For every $x, y \in X$, the equation $x \circ z = y$ has a unique solution for $z \in X$.*

2. *For every $x, y \in X$, the equation $z \circ x = y$ has a unique solution for $z \in X$.*

The operation table of a binary operation $\circ$ defined on X is the $|X| \times |X|$ array $A = (a_{x,y})$, where $a_{x,y} = x \circ y$. It is not difficult to see that a latin square is equivalent to a operation table of a quasigroup.

A group is a quasigroup (but the inverse is not necessary true). Therefore we can use a group to get a latin square. For example, the group $(\mathbb{Z}_n, +)$ defines a latin square of order n. The following latin square is from $(\mathbb{Z}_6, +)$.

```
012345
123450
234501
345012
450123
501234
```

Definition 1.1.4 *Two latin squares of side n, $L = (a_{i,j})$ on symbol set S and $L' = (b_{i,j})$ on symbol set S' are* orthogonal *if every element in $S \times S'$ occurs exactly once among the n^2 pairs $(a_{i,j}, b_{i,j}), 1 \leq i, j, \leq n$.*

As an example, the following is orthogonal latin squares of order 3.

```
123     123
231     312
312     231
```

The superposition of the above two latin squares is:

```
(1,1) (2,2) (3,3)
(2,3) (3,1) (1,2)
(3,2) (1,3) (2,1)
```

Theorem 1.1.5 *If $n > 1$ is odd, then there exist orthogonal latin squares of order n.*

Proof. We define two latin squares L_1 and L_2 of order n with entries from $\mathbb{Z}_n$:

$$\begin{aligned} L_1(i,j) &\equiv (i+j) \bmod n \\ L_2(i,j) &\equiv (i-j) \bmod n. \end{aligned}$$

It is easy to check that L_1 and L_2 are latin squares. To prove they are orthogonal, we just need to show that for any $(x,y) \in \mathbb{Z}_n \times \mathbb{Z}_n$, there is a solution of the following system:

$$\begin{aligned} i+j &\equiv x \bmod n \\ i-j &\equiv y \bmod n \end{aligned}$$

Since n is odd, there is 2^{-1} in $\mathbb{Z}_n$ (i.e., $\frac{n+1}{2}$). So

$$\begin{aligned} i &\equiv (x+y)2^{-1} \bmod n \\ j &\equiv (x-y)2^{-1} \bmod n. \end{aligned}$$

Hence L_1 and L_2 are orthogonal. ∎

A pair of orthogonal latin square of order 5 using the above proof is displayed below.

```
01234     04321
12340     10432
23401     21043
34012     32104
40123     43210
```

Now we give a recursive construction of latin squares which is called *direct product*. Suppose that L and M are latin squares of order n and m (respectively) defined on symbol sets X and Y (respectively). Then the direct product of L and M, denoted $L \times M$, is an $mn \times mn$ array defined as follows:

$$(L \times M)((i_1,i_2),(j_1,j_2)) = (L(i_1,j_1), M(i_2,j_2)).$$

Let L and M be latin squares of order 3 and 2 as follows:

```
3 1 2       1 2
2 3 1       2 1
1 2 3
```

Then $L \times M$ is as follows:

```
(3,1)(1,1)(2,1)(3,2)(1,2)(2,2)
(2,1)(3,1)(1,1)(2,2)(3,2)(1,2)
(1,1)(2,1)(3,1)(1,2)(2,2)(3,2)
(3,2)(1,2)(2,2)(3,1)(1,1)(2,1)
(2,2)(3,2)(1,2)(2,1)(3,1)(1,1)
(1,2)(2,2)(3,2)(1,1)(2,1)(3,1)
```

We can rewrite (change the notation of symbols) this latin square as

```
1 2 3 4 5 6
3 1 2 6 4 5
2 3 1 5 6 4
4 5 6 1 2 3
6 4 5 3 1 2
5 6 4 2 3 1
```

Theorem 1.1.6 *If there exist orthogonal latin squares of orders n_1 and n_2, then there exist orthogonal latin squares of order $n_1 n_2$.*

Proof. Suppose that L_1 and L_2 are orthogonal latin squares of order n_1 on set X, and M_1 and M_2 are orthogonal latin squares of order n_2 on set Y. Then we can prove that $L_1 \times M_1$ and $L_2 \times M_2$ are orthogonal latin squares of order $n_1 n_2$.

Consider an ordered pair of symbols, $((x_1, y_1), (x_2, y_2))$. We want to find a unique cell $(i_1, i_2), (j_1, j_2))$ such that

$$\begin{aligned} (L_1 \times M_1)((i_1, i_2), (j_1, j_2)) &= (x_1, y_1), \text{ and} \\ (L_2 \times M_2)((i_1, i_2), (j_1, j_2)) &= (x_2, y_2). \end{aligned}$$

This is equivalent to

$$\begin{aligned} L_1(i_1, j_1) &= x_1 \\ M_1(i_2, j_2) &= y_1 \\ L_2(i_1, j_1) &= x_2 \\ M_2(i_2, j_2) &= y_2 \end{aligned}$$

Since L_1 and L_2 are orthogonal, and M_1 and M_2 are orthogonal, the above solution for i and j is unique. □

A set of latin squares $L_1, L_2, \ldots, L_m$ is *mutually orthogonal*, or a set of *MOLS*, if for every $1 \leq i < j \leq m$, L_i and L_j are orthogonal. We will denote it as MOLS(n) if the order of the latin squares is n.

An example of MOLS(4) is as follows.

Example 1.1.7 *3 MOLSs of side 4 (or order 4).*

```
1234   1234   1234
4321   3412   2143
2143   4321   3412
3412   2143   4321
```

Using the previous examples and theorems, we can prove the following.

Theorem 1.1.8 *There exist orthogonal latin squares of order n, if $n \not\equiv 2 \bmod 4$*

The proof of this theorem is left as an exercise.

Theorem 1.1.9 *There do not exist n MOLSs of order n if $n > 1$.*

Proof. Suppose we have s MOLSs of order n $L_1, L_2, \ldots, L_s$. We may assume, w.l.o.g., that the first row of each square is: $1\ 2\ 3 \cdots n$. Next we consider the s values $L_1(2,1), L_2(2,1) \ldots, L_s(2,1)$. They must be different because for any pair of L_i and L_j, all the pairs $(x,x), x \in \{1, 2, \cdots, n\}$ appear in the superposition of the first rows. Also $L_i(2,1) \neq 1$. Therefore $s \leq n-1$. □

Theorem 1.1.10 *If there exist s MOLS(n_i), $1 \leq i \leq l$, then there exists s MOLS(n), where $n = n_1 n_2 \cdots n_l$.*

Proof. The result follows from Theorem 1.1.6. □

Theorem 1.1.11 *If $q = p^e$ is a prime power, then there are $q-1$ MOLS(q).*

Proof. Let $\mathbb{F}_q$ be the finite field of order q. For each $\alpha \in \mathbb{F}_q \backslash \{0\}$, define the latin square $L_\alpha(i,j) = i + \alpha j$, where $i, j \in \mathbb{F}_q$ and the algebra is performed in $\mathbb{F}_q$. It is easy to check that each pair of the latin squares are orthogonal. □

Theorem 1.1.12 (MacNeish's Theorem) *Suppose that n has prime power factorization $n = p_1^{e_1} \cdots p_l^{e_l}$, where the p_is are distinct primes and $e_i \geq 1$ for $1 \leq i \leq l$. Let*

$$s = \min\{p_i^{e_i} - 1 : 1 \leq i \leq l\}.$$

Then there exists s MOLS(n).

When n is not a prime power, most cases of maximum number of MOLS(n) are not determined. We only know that there is no MOLS(6). The smallest unknown case is $n = 10$. We have 2 MOLS(10), but don't know if 3 MOLS(10) exists or not.

1.2 Orthogonal arrays

Definition 1.2.1 *Let $k \geq 2$ and $n \geq 1$ be integers. An orthogonal array OA(k, n) is an $n^2 \times k$ array, A, with entries from a set X of cardinality n such that, within any two columns of A, every ordered pair of symbols from X occurs in exactly one row of A.*

An OA(k, n) is equivalent to $k - 2$ MOLS(n). We have the following theorem.

Theorem 1.2.2 *Suppose that $n \geq 1$ and $k \geq 3$ are integers. Then $k - 2$ MOLS(n) exists if and only if an OA(k, n) exists.*

Proof. Suppose that $k - 2$ latin squares are $L_1, L_2 \ldots, L_{k-2}$ which are all defined on symbol set $\{1, 2, \ldots, n\}$. For every $i, j \in \{1, 2, \ldots, n\}$, construct an k-tuple:

$$(i, j, L_(i, j), L_2(i, j), \ldots, L_{k-2}(i, j).$$

Then form an array A whose rows consist of these n^2 k-tuples. We can show that A is an OA(k, n). To do that, we need to show that every ordered pair of symbols occurs in any two columns a and b, where $1 \leq a < b \leq s + 2$. We consider the following cases:

1. If $a = 1$ and $b = 2$, then clearly it is true.

2. If $a = 1$ and $b \geq 3$, then we get every ordered pair because every row of L_b is a permutation of $\{1, 2, \ldots, n\}$.

3. If $a = 2$ and $b \geq 3$, then we get every ordered pair because every column of L_b is a permutation of $\{1, 2, \ldots, n\}$.

4. If $a \geq 3$, then we get every ordered pair because L_a and L_b are orthogonal.

The above construction can easily be reversed. Suppose we have an OA(k, n) A, $k \geq 3$, defined on set $\{1, 2, \ldots, n\}$. For $1 \leq h \leq k-2$ and $1 \leq r \leq n^2$, define

$$L_h(A(r, 1), A(r, 2)) = A(r, h + 2).$$

We will prove that these are MOLS(n). First we can see that each L_h is a latin square. The entries in row i of L_h are the set:

$$\{A(r, h + 2) : A(r, 1) = i\}.$$

These symbols are all distinct because every ordered pair occurs exactly in columns 1 and $h + 2$ of A. A similar argument proves that each column of L_h is a permutation of $\{1, 2, \ldots, n\}$. To see two latin squares L_h and L_g are orthogonal, we notice that each ordered pair appears once in a row of the columns $h + 2$ and $g + 2$. □

From Example 1.1.7 we have the following OA$(5, 4)$ (it is written as transposed of OA to save some space).

```
1111222233334444
1234123412341234
1234432121433412
1234341243212143
1234214334124321
```

Form Theorem 1.1.11 we have the following result.

Corollary 1.2.3 *For any prime power $q = p^n$, there is an $OA(q + 1, q)$.*

We can generalize the definition of orthogonal arrays as follows.

Definition 1.2.4 *Let t, v, k and λ be positive integers such that $k \geq t \geq 2$. A t-(v, k, λ) orthogonal array (denoted t-(v, k, λ)-OA) is a pair (X, D) such that the following properties are satisfies.*

1. *X is a set of v elements called points.*
2. *D is a $\lambda v^t \times k$ array whose entries are chosen from the set X.*
3. *Within any t columns of D, every t-tuple of points is contained in exactly λ rows.*

Our previous definition of OA is just the case $t = 2$ and $\lambda = 1$. An orthogonal array (X, D) is a linear orthogonal array if $X = \mathbb{F}_q$ for some prime power q and the rows of D form a subspace (of the vector space $(\mathbb{F}_q)^k$) having dimension $\log_q |D|$.

Theorem 1.2.5 *Let l and n be positive integers, and let q be a prime power. Let M be an $l \times n$ matrix of element from the finite field $\mathbb{F}_q$ such that every set of t columns of M is linearly independent. Define D to be the $q^l \times n$ matrix whose rows consist of all the linear combinations of the rows of M. Then $(\mathbb{F}_q, D)$ is a t-(q, n, λ)-OA, where $\lambda = q^{l-t}$.*

Proof. Choose t columns of D, say the ones labeled $c_1, c_2 \ldots, c_t$. Let $(y_1, y_2, \ldots, y_t)$ be an arbitrary t-tuple of elements of $\mathbb{F}_q$. We want to determine the rows i of D such that $D(i, c_j) = y_j$ for $1 \leq j \leq t$.

A row of D is constructed as $\mathbf{r}M$, where $\mathbf{r} = (r_1, \ldots r_l) \in (\mathbb{F}_q)^l$. Let $\mathbf{c}_j$ denote the jth column of M for $1 \leq j \leq n$. We want to determine all vectors $\mathbf{r}$ such that

$$\mathbf{r}\mathbf{c}_{i_j} = y_{i_j}, 1 \leq j \leq t.$$

The column vectors $\mathbf{c}_{i_1}, \ldots, \mathbf{c}_{i_t}$ are linearly independent by assumption. Therefore, the above is a system of t independent linear equations in l unknowns, and it has a solution space of dimension $l - t$. The number of solutions $\mathbf{r}$ is q^{l-t}. ∎

Using above theorem, we can get several results OAs.

Corollary 1.2.6 *Let $l \geq 2$ be a positive integers and let q be a prime power. Then there exists a 2-$(q, (q^l - 1)/(q - 1), q^{l-2})$-OA.*

Proof. Excluding the zero vector, there are $q^l - 1$ distinct l-tuples of elements of $\mathbb{F}_q$. Each l-tuple has $q-1$ nonzero scalar multiples, so that the $q^l - 1$ nonzero vectors are partitioned into $(q^l - 1)/(q - 1)$ subspaces each of dimension equal to one. Arbitrarily pick one vector from each subspace, and let these vectors be the columns of M. Then apply Theorem 1.2.5. ∎

Corollary 1.2.7 *Let $t \geq 2$ be an integer, and let q be a prime power. Then there exists a t-$(q, q, 1)$-OA.*

Proof. For every $x \in \mathbb{F}_q$, construct the vector $\mathbf{x} = (1, x, x^2, x^3, \ldots, x^{t-1}) \in (\mathbb{F}_q)^t$. Transpose these q vectors to form the columns of M. We need to prove that any t of the vectors are linearly independent. Suppose that this is not the case. Then there exists a $t \times t$ sub-matrix of of M, say M_0, whose columns are linearly dependent. Denote

$$M_0 = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ y_1 & y_2 & y_3 & \cdots & y_t \\ y_1^2 & y_2^2 & y_3^2 & \cdots & y_t^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_1^{t-1} & y_2^{t-1} & y_3^{t-1} & \cdots & y_t^{t-1} \end{pmatrix}$$

where $y_1, y_2, \ldots, y_t$ are distinct elements of $\mathbb{F}_q$.

If the columns of M_0 are linearly dependent, then the rows of M_0 are also linearly dependent. So there exist $a_1, a_2 \ldots, a_t \in \mathbb{F}_q$, not all equal to 0, such that $(a_1, a_2, \ldots, a_t)M_0 = (0, 0, \ldots, 0)$. Define the polynomial

$$a(x) = a_1 + a_2 x + \cdots + a_t x^{t-1};$$

then $a(y_j) = 0$ for $1 \leq j \leq t$. However, a polynomial of degree $t - 1$ cannot have t roots. ∎

An example of 3-$(5, 5, 1)$-OA from the above construction: The matrix M is as follows:

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 1 & 4 & 4 & 1 \end{pmatrix}.$$

The 125 rows that are the linear combinations (over $\mathbb{Z}_5$) of the three rows of M comprise the desired orthogonal array.

An orthogonal array (X, D) is a *linear orthogonal array* if $X = \mathbb{F}_q$ for some prime power q and the rows of D form a subspace (of the vector space $\mathbb{F}_d)^k$) having dimension $\log_q |D|$.

The above construction results linear orthogonal arrays.

Theorem 1.2.8 *Let q be an odd prime power. For $a, b \in \mathbb{F}_q$, define $f_{a,b} : \mathbb{F}_q \to \mathbb{F}_q$ by the rule*

$$f_{a,b}(x) = (x + a)^2 + b.$$

Then the $q^2 \times q$ array $D = (d_{i,j})$, where $d_{i,j} = f_{a,b}(j)$ $(i = (a,b) \in (\mathbb{F}_q)^2, j \in \mathbb{F}_q)$, is a 2-$(q,q,1)$-OA.

Proof. Let $x_1, x_2 \in \mathbb{F}_q$ (where $x_1 \neq x_2$) and let $y_1, y_2 \in \mathbb{F}_q$. We need to show that there is exactly one ordered pair $(a,b) \in (\mathbb{F}_q)^2$ such that

$$(x_1 + a)_b^2 = y_1$$

and

$$(x_2 + a)_b^2 = y_2.$$

Subtracting the two equations, we have

$$a = \frac{y_1 - y_2}{2(x_1 - x_2)} - \frac{x_1 + x_2}{2}.$$

Then for the given a, we can get a unique solution for b. ∎

An example of 2-(3,3,1)-OA constructed using the above theorem:

$$\begin{array}{cc} & 012 \\ \hline f_{0,0} & 011 \\ f_{0,1} & 122 \\ f_{0,2} & 200 \\ f_{1,0} & 110 \\ f_{1,1} & 221 \\ f_{1,2} & 002 \\ f_{2,0} & 101 \\ f_{2,1} & 212 \\ f_{2,2} & 020 \end{array} \rightarrow \begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 2 & 1 \\ 0 & 0 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \\ 0 & 2 & 0 \end{pmatrix}$$

This is not a linear orthogonal array.

Theorem 1.2.9 (Gilbert-Varshamov Bound) *Let l, t and n be positive integers such that $2 \leq t \leq l$, and let q be a prime power. Suppose that*

$$\sum_{i=0}^{t-1} \binom{n-1}{i} (q-1)^i < q^l. \tag{1.1}$$

Then there exists a linear t-(q,n,λ)-OA, where $\lambda = q^{l-t}$.

Proof. We will use Theorem 1.2.5 to prove this theorem by indicating the existence of an $l \times n$ matrix M satisfying the hypotheses of Theorem 1.2.5.

First let M_l be the $l \times l$ identity matrix so that any t columns of M_l are linearly independent. Next we can obtain matrix M_j which satisfying the hypotheses of Theorem 1.2.5, where $l+1 \leq j \leq n$, as follows. Since the number of linear combinations of at most $t-1$ columns of M_j is

$$\sum_{i=0}^{t-1} \binom{j}{i} (q-1)^i.$$

By the condition 1.1,

$$\sum_{i=0}^{t-1} \binom{j}{i} (q-1)^i < q^l.$$

So there is a column vector c which is not one of these linear combinations. Adjoining this column to obtain the matrix M_{j+1}. The matrix M_n is what we want. □

The above theorem is not a constructive result, but gives us a lower bound of the existence of orthogonal arrays.

1.3 Authentication codes

One important topic of network security is authentication. When Alice sends a message to Bob (encrypted or not), how can Bob be sure that it was Alice who sent the message, and how does he know that the message was not altered by someone else during its transmission? One solution is using authentication code.

Before we give a formal definition of authentication code, we make some settings first. There are three participants in this situation: Alice, Bob and Oscar. Alice and Bob want to communicate over an insecure channel. Oscar has the ability to introduce his own messages into the channel and/or modify existing messages. We consider two types of attacks by Oscar. When Oscar places a message m' into the channel, it is called *impersonation*. When Oscar sees a message m and changes it to s different message $m' \neq m$, it is called *substitution*.

The goal of an authentication code is to allow Bob to detect with high probability when such an attack has taken place.

Definition 1.3.1 *An authentication code is a four-tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{K}, \mathcal{E})$*, where the following conditions are satisfied.*

1. $\mathcal{S}$ *is a finite set of source states.*
2. $\mathcal{A}$ *is a finite set of authenticators.*
3. $\mathcal{K}$ *is a finite set of keys.*
4. *For each* $K \in \mathcal{K}$*, there is an authentication rule* $e_K : \mathcal{S} \to \mathcal{A}$*.*

We let the authentication code work as follows. Alice and Bob jointly choose a secret key $K \in \mathcal{K}$ at random. They do this ahead of time, either when they are together in some place or when they have access to a secure channel. A source state is the information that Alice wants to communicate to Bob. When Alice wants to communicate the source $s \in \mathcal{S}$ to Bob, she uses the authentication rule e_k to construct the authenticator $a = e_K(s)$. The message m is formed by concatenating s and a, i.e., $m = (s, a)$, The message is then sent over the channel. When Bob receives m, he verifies that $a = e_K(s)$ to authenticate the source state s. If $a \neq e_K(s)$, then Bob is able to detect that an attack has taken place.

An authentication code can be represented by the $|\mathcal{K}| \times |\mathcal{S}|$ the authentication matrix in which the rows are indexed by the keys, the columns are indexed by source states, the the entry in row K and column s of the matrix is $e_K(s)$.

When Oscar performs an attack, his goal is to have his bogus message $m' = (s', a')$ accepted as authentic by Bob. So if K is the secret key (that is not known to Oscar), then Oscar is hoping that $a' = e_K(s')$.

The strength of an authentication code is measured by the deception probabilities P_0 and P_1, which represent the probability that Oscar can deceive Bob by impersonation and substitution, respectively. In computing the deception probabilities, we always assume that Oscar is using an optimal strategy. When Alice and Bob use an authentication code, they want P_0 and P_1 to be small and they also want $|\mathcal{K}|$ to be small so that they don't need large storage for the keys.

1.3.1 A construction from orthogonal arrays

Suppose that B is an OA(m, n) on symbol set $\{1, \ldots, n\}$. We define $\mathcal{S} = \{1, \ldots m\}$, $\mathcal{A} = \{1, \ldots, n\}$ and $\mathcal{K} = \{1, \ldots, n^2\}$. The rows of B are indexed

by $\mathcal{K}$ and the columns are indexed by $\mathcal{S}$. For $1 \leq K \leq n^2$, the authentication rule e_K is defined as

$$e_K(s) = B(K, s)$$

for $1 \leq s \leq m$. In this way, the orthogonal array B is used as the authentication matrix of the code.

In computing the deception probabilities of this authentication code, we assume that the authentication matrix is known to Oscar. The only information that Oscar does not know is that which key Alice and Bob have chosen.

Suppose that Oscar places a message $m = (s, a)$ into the channel. Then m is accepted as authentic if and only if $e_K(s) = a$, i.e., $B(K, s) = a$. Here K is a random row of the orthogonal array B, and the value K is know by Alice and Bob but not Oscar.

Let $\mathcal{L}(s, a) = \{L : B(L, s) = a\}$. Then $|\mathcal{L}(s, a)| = n$. Oscar's deception will succeed if and only if $K \in \mathcal{L}(s, a)$. Since $|\mathcal{K}| = n^2$, we have

$$P_0 = \frac{|\mathcal{L}(s, a)|}{|\mathcal{K}|} = \frac{1}{n}.$$

To consider P_1, we suppose that Oscar sees a valid message $m = (s, a)$, and he replaces it with a bogus message $m' = (s', a')$, where $s \neq s'$. Since Oscar has seen the message m, he knows that $K \in \mathcal{L}(s, a)$. In other words, Oscar knows that the number of possible keys is n (not n^2). Oscar's deception will succeed if and only if $K \in \mathcal{L}(s', a')$, i.e., $K \in \mathcal{L}(s, a) \cap \mathcal{L}(s', a')$. Since B is an OA(m, n), we have $|\mathcal{L}(s, a) \cap \mathcal{L}(s', a')| = 1$. So the success probability of this substitution attack is

$$P_1 = \frac{|\mathcal{L}(s, a) \cap \mathcal{L}(s', a')|}{|\mathcal{L}(s, a)|} = \frac{1}{n}.$$

Theorem 1.3.2 *Suppose there is an OA(m, n). Then there is an authentication code for m source states, having n authenticators and n^2 keys, in which $P_0 = P_1 = \frac{1}{n}$.*

Remark.

- For authentication codes with n authenticators, the best we can do is $P_0 = P_1 = \frac{1}{n}$.

- When constructing an authentication code using an OA(m, n), the parameter n relates to the security of the code, while the parameter m determines the number of source states. In order for an OA(m, n) to exist, $m \leq n + 1$.

- The above authentication code is one-time code: a key should be used to authenticate only one source state.

Example. Suppose that Bob is Alice's stockbroker and Alice owns 100 shares of Acme stock. For $0 \leq i \leq 99$, we will let source state i correspond to the order "sell $i + 1$ shares"; and for $100 \leq i \leq 199$, we will let source state i correspond to the order "buy $i - 99$" shares. Thus we desire a code with at least 200 source states, so we need an OA(m, n) with $m \geq 200$. Now suppose that Alice and Bob want a security level of 1/1000, i.e., they want a code with $P_0 \leq 1/1000$, $P_1 \leq 1/1000$. This means that they will use an OA(m, n) with $n \geq 1000$.

To implement that, we can choose a smallest prime exceeding 1000, for example, $n = 1009$. Then they construct an OA$(200, 1009)$ and use it as the authentication code.

Theorem 1.3.3 *Suppose there exists an authentication code for msource states and having n authenticators, in which $P_0 = P_1 = 1/n$. Then,*

1. *$|\mathcal{K}| \geq n^2$, and equality occurs if and only if the authentication matrix is an orthogonal array OA(m, n) and the authentication rules are used with equal probability.*

2. *$|\mathcal{K}| \geq m(n-1)+1$, and equality occurs if and only if the authentication matrix is a 2-(m, n, λ)-OA, where $\lambda = (m(n-1)+1)/n^2$, and the authentication rules are used with equal probability.*

Proof. Here we only give the proof of 1. For the proof of part 2, see [4].

Suppose $P_0 = P_1 = 1/n$. For $s, s' \in \mathcal{S}, s \neq s'$ and $a, a' \in \mathcal{A}$, we have $|\{K \in \mathcal{K} : e_K(s) = a, e_K(s') = a'\}| \geq 1$ (Otherwise $P_1 > 1/n$). So we have $|\mathcal{K}| \geq n^2$. The equality occurs if and only if $|\{K \in \mathcal{K} : e_K(s) = a, e_K(s') = a'\}| = 1$ and therefore the matrix is an OA(m, n). □

For more information about combinatorial authentication codes, see [4].

1.4 Secrete sharing

In a bank, there is a vault which must be opened every day. There are 3 senior tellers in the bank. The bank wants a system that any two senior tellers can gain access to the vault, but no individual teller can do so. Another real story is that in Russia in the early 1990s, the control of nuclear weapons depended upon two of ten three parties: the President, the Defense Minister and the Defense Minister.

Definition 1.4.1 *Let t, w be positive integers, $t \leq w$. A perfect (t, w)-threshold scheme is a method of sharing a key K among a set of w participants denoted by $\mathcal{P} = \{P_1, \ldots, P_w\}$, in such a way that any t participants can compute the value of K, but no group of $t-1$ participants can do so.*

We will study the unconditional security of secret sharing schemes, i.e., we do not limit the amount of computation that can be performed by any subset of participants.

We need to setup someone to choose the value of K and distribute some information (called shares) to participants. We call this person the dealer. The shares should be distributed secretly, so no participants knows the share given to another participant.

At a later time, a subset of participants $B \subset \mathcal{P}$ pool their shares in an attempt to compute the secret K. If $|B| \geq t$, then they should be able to compute the value of K as a function of the shares they jointly hold; if $|B| < t$, then should not be able to get any information about K. The above examples are some $(2, 3)$-threshold scheme.

1.4.1 Threshold schemes from OAs

We can construct a (t, w)-threshold scheme from a t-$(v, w+1, 1)$-OA as follows. Suppose that an orthogonal array, A, is defined on symbol set X of order v, the columns are labeled $1, 2, \ldots, w+1$, and the rows are labeled $1, 2, \ldots, v^t$. The scheme will use X as $\mathcal{K}$ and $\mathcal{S}$, so it accommodates v possible secrets. Associate the first w columns of the array with the w participants and the last column with the secret. For every $K \in X$, define $R_K = \{r : A(r, w+1) = K\}$. So R_K is the set of rows of A having the element K in the last column. When the dealer D wants to share the secret $K \in X$, he chooses a random row $r \in R_K$. Then D gives the share $A(r, i)$ to participant P_i for $1 \leq i \leq w$.

Suppose that t participants, say $P_{i_1}, \ldots, P_{i_t}$, wish to determine the secret. The orthogonal array A is known to all the participants (it is public). Let s_j be P_{i_j}s share, $1 \leq j \leq t$. Because A is a t-$(v, w+1, 1)$-OA, there is a unique row r such that $A(r, i_j) = s_j, 1 \leq j \leq t$. So they can find out that $K = A(r, w+1)$.

Now suppose that $t-1$ participants $P_{i_1}, \ldots, P_{i_{t-1}}$ want to find out the key. Then for any value $s \in X$ in the last column, there is a unique row r determined by $s_1, \ldots s_{t-1}, s$. That means that any value $s \in X$ might be the key. So they cannot find out the value of K.

So we have the following theorem.

Theorem 1.4.2 *Suppose there is a t-$(v, w+1, 1)$-OA. Then there exists a perfect (t, w)-threshold scheme with $|\mathcal{S}| = |\mathcal{K}| = v$.*

1.4.2 Shamir's threshold scheme

Shamir used polynomial method to construct threshold schemes. For simplicity, we just suppose that the keys and shares are from $\mathbb{Z}_p$ for some prime $p > w$. In fact, this method can also be used for the keys and shares from a finite field $\mathbb{F}_q$.

In the scheme, the dealer D will do the following.

1. D chooses w distinct, non-zero elements of $\mathbb{Z}_p$, denoted x_i, $1 \leq i \leq w$. For $1 \leq k \leq w$, D gives the value x_i to P_i. The values x_i are public.

2. Suppose D wants to share a key $K \in \mathbb{Z}_p$. D secretly chooses (independently at random) $t-1$ elements of $\mathbb{Z}_p$, which are denoted $a_1, \ldots, a_{t-1}$.

3. For $1 \leq i \leq w$, D computes $y_i = A(x_i)$, where

$$A(x) = K + \sum_{j=1}^{t-1} a_j x^j \bmod p.$$

4. For $1 \leq i \leq w$, D gives the share y_i to P_i.

Now suppose that t participants $P_{i_1}, \ldots, P_{i_t}$ want to reconstruct the key K. They know that $y_{i_j} = A(x_{i_j})$, but they do not know the polynomial $A(x)$. They can find out the polynomial (and thus find out the key) as follows. They

know that the polynomial is of degree at most $t-1$. So the polynomial is of the form:

$$A(x) = a_0 + a_1 x + \cdots + a_{t-1}x^{t-1},$$

where the coefficients $a_0, \ldots, a_{t-1}$ are unknown elements of $\mathbb{Z}_p$. From their shares, they obtain t linear equations:

$$a_0 + a_1 x_{i_j} + \cdots + a_{t-1}x_{i_j}^{t-1} = y_{i_j} \bmod p.$$

Since these linear equations are linearly independent (note that x_i are distinct non-zero elements), they can get a unique solution.

On the other hand, for $t-1$ participants, they only can get $t-1$ linear equations with t unknowns. So a_0 can be any element in $\mathbb{Z}_p$. They cannot get any information about the key.

By Lagrange interpolation formula for polynomials, we have an explicit formula for $A(x)$:

$$A(x) = \sum_{j=1}^{t} \left(y_{i_j} \prod_{1 \le k \le t, k \ne j} \frac{x - x_{i_k}}{x_{i_j} - x_{i_k}} \right) \bmod p.$$

Let $x = 0$, we can compute the key:

$$K = \sum_{j=1}^{t} \left(y_{i_j} \prod_{1 \le k \le t, k \ne j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}} \right) \bmod p.$$

Suppose we define

$$b_j = \prod_{1 \le k \le t, k \ne j} \frac{x_{l_k}}{x_{i_k} - x_{i_j}} \bmod p,$$

$1 \le j \le t$. Note that the b_j's can be precomputed, if desired, and their values are not secret. We can use the following formula to compute the key:

$$K = \sum_{j=1}^{t} b_j y_{i_j} \bmod p.$$

Example. Suppose that $p = 17, t = 3$, and $w = 5$; and the public x-coordinates are $x_i = i, 1 \le i \le 5$. Suppose that P_1, P_3, P_5 pool their shares,

which are respectively $8, 10$ and 11. So they have the following equations in mod17:

$$\begin{aligned} a_0 + a_1 + a_2 &= 8 \\ a_0 + 3a_1 + 9a_2 &= 10 \\ a_0 + 5a_1 + 8a_2 &= 11 \end{aligned}$$

This linear system has a unique solution in $\mathbb{Z}_{17}$: $a_0 = 13, a_1 = 10$, and $a_2 = 2$. The key is 13.

Or using the formula we have

$$\begin{aligned} b_1 &= \frac{x_3 x_5}{(x_3 - x_1)(x_5 - x_1)} \bmod 17 \\ &= 3 \times 5 \times (-2)^{-1} \times (-4)^{-1} \bmod 17 \\ &= 3 \times 5 \times 8 \times 4 \bmod 17 \\ &= 480 \bmod 17 \\ &= 4. \end{aligned}$$

Similarly, $b_2 = 3$ and $b_3 = 11$. So

$$K = 4 \times 8 + 3 \times 10 + 11 \times 11 \bmod 17 = 13.$$

1.4.3 Ramp scheme

In a (t, w)-threshold scheme, any t subset of participants is called authorized subsets while any subset with $t-1$ participants is called unauthorized subset. The threshold schemes can be a little generalized as ramp schemes as follows.

A $(c, r, b)-$*ramp scheme* is a secret sharing scheme, where $c < r < b$, in which the authorized subsets are all the subsets of $\mathcal{P}$ with cardinality r and the unauthorized subsets are all the subsets of $\mathcal{P}$ with cardinality c. When $c = r - 1$, the ramp scheme becomes a threshold scheme.

We can obtain a ramp scheme from an orthogonal array as follows. Suppose there is an r-$(q, b + r - c, 1)$-OA which is published so that every user can look at it. The secret information K is a $r - c$ tuple from $GF(q)$. The Dealer D chooses secretly a random row in the OA such that the last $r - c$ columns of that row is K. It is easy to know from the property of OA that there are q^c such rows. D then gives each of b users one value of the first b

columns of that row. Since the OA is of index 1, any r of these values decide a row of the OA uniquely. Thus r users can get K by putting together their shares. However, from any c values, the users cannot obtain any information about K, since these c values together with last $r - c$ columns of any row in OA decide one row. That means the key might be any of the $r - c$ tuple.

When $c < r - 1$, a set of d participants can get some information about the key, but not exactly the key K.

1.4.4 General secret sharing

In previous subsections, the size of authorized subset of participants is fixed. A more general situation is to specify exactly which subsets of participants are authorized subsets. Let Γ be a set of subsets of $\mathcal{P}$ such that the subsets in Γ are those subsets of participants that should be able to compute the key (the authorized subsets). Γ is called an *access structure*.

In a (t, w)-threshold scheme, the access structure is:

$$\Gamma = \{B \subseteq \mathcal{P} : |B| \geq t\}.$$

Definition 1.4.3 *A* perfect secret sharing scheme *realizing the access structure* Γ *is a method of sharing a key* K *among a set of* w *participants (denoted by* $\mathcal{P}$*), in such a way that the following two properties are satisfied:*

1. *If an authorized subset of participants* $B \in \Gamma$ *pool their shares, then they can determine the value of* K*.*

2. *If an unauthorized subset of participants* B *(*$B \notin \Gamma$*), pool their shares, then they can determine nothing about the value of* K*.*

The threshold schemes are perfect secret sharing schemes, but the ramp scheme is not perfect.

The access structure of a secret sharing scheme should have the *monotone property:*

if $B \in \Gamma$ and $B \subseteq C \subseteq \mathcal{P}$, then $C \in \Gamma$.

If Γ is an access structure, then $B \in \Gamma$ is a *minimal authorized subset* if $A \notin \Gamma$ whenever $A \subseteq B, A \neq B$. Usually the set of minimal authorized subsets of Γ is denoted Γ_0 and is called the *basis* of Γ. In the (t, w)-threshold scheme, the Γ_0 contains all the t-subsets of $\mathcal{P}$.

Example. Suppose $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ and $\Gamma_0 = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}, \{P_2, P_3\}\}$. Then

$$\Gamma = \Gamma_0 \cup \{\{P_1, P_2, P_3\}, \{P_2, P_3, P_4\}, \{P_1, P_2, P_3, P_4\}\}.$$

One realizing method can be the follows. The Dealer D uses $\mathbb{Z}_m$ for some large integer m as the space of key and shares. D chooses a random key K and independent random numbers a_1, a_2, b_1, b_2 and c_1 from $\mathbb{Z}_m$. Then D distributes shares as below.

- P_1 received $(y_1^1, y_1^2) = (a_1, b_1)$.
- P_2 received $(y_2^1, y_2^2) = (a_2, c_1)$.
- P_3 received $(y_3^1, y_3^2) = (b_2, K - c_1)$.
- P_4 received $(y_4^1, y_4^2) = (K - a_1 - a_2, K - b_1 - b_2)$.

The computations for subsets in Γ_0 are as follows.

- For $\{P_1, P_2, P_4\}$,

$$K = y_1^1 + y_2^1 + y_4^1 = a_1 + a_2 + (K - a_1 - a_2) \bmod m$$

- For $\{P_1, P_3, P_4\}$,

$$K = y_1^2 + y_3^1 + y_4^2 = b_1 + b_2 + (K - b_1 - b_2) \bmod m$$

- For $\{P_2, P_3\}$,

$$K = y_2^2 + y_3^2 = c_1 + (K - c_1) \bmod m$$

The maximal unauthorized subsets are $\{P_1, P_2\}, \{P_1, P_3\}, \{P_1, P_4\}, \{P_2, P_4\}, \{P_3, P_4\}$. It is easy to check that any unauthorized subset cannot get any information about the key K. For example, $\{P_1, P_2\}$ only have four random numbers a_1, a_2, b_1, c_1. $\{P_1, P_3\}$ only have values $a_1, b_1, b_2, K - c_1$. They can not get any information about K. Other cases can be checked in a similar way.

There are different methods used to construct general secret sharing schemes. A brief description can be found in [5]. Details may be found in the references of [5].

1.5 Broadcast encryption

Suppose a network consists of a *trusted authority* (TA) and a set of n users $\mathcal{U}$. A set $\mathcal{P} = \{P \subseteq \mathcal{U} : P \text{ is a privileged subset}\}$ and a set $\mathcal{F} = \{F \subseteq \mathcal{U} : F \text{ is a forbidden subset}\}$. Any information transmitted by the TA will be received by every user, so the network is a broadcast channel.

In a set up stage, the TA generates and distributes secret information u_i to each user $i \in \mathcal{U}$ off-band. At a later time, the TA will want to broadcast a message m_P to a privileged subset $P \in \mathcal{P}$. The particular privileged subset P is, in general, not known ahead of time. The broadcast is a ciphertext b_P. Once b_P is broadcast, each user $i \in P$ should be able to decrypt it and obtain m_P. On the other hand, no forbidden set $F \in \mathcal{F}$ disjoint from P should be able to compute any information about the message.

It is desirable to find a scheme which minimized the amount of secret information that needs to be stored by each user and maximized the size of broadcast information.

We discuss the security in terms of a single broadcast. We give an informal definition about the broadcast encryption scheme as follows.

A $(\mathcal{P}, \mathcal{F})$-One-Time Broadcast Encryption Scheme (OTBES) satisfying the following conditions:

1. Without knowing the broadcast, no subset of users has any information about m_P even given all the secret information u_i.

2. The message for a privileged user is uniquely determined by the broadcast and the user's information.

3. After receiving the broadcast, no forbidden subset F disjoint from P has any information on m_P.

When the forbidden set contains all the subsets of w users, the scheme will denote as $(\mathcal{P}, w)$-OTBES.

A brief method is that the TA gives each user a different secret key. When TA wants to broadcast a message, he uses each key of the privileged key to encrypt a copy. But this method is not efficient. TA wants to encrypt less copies.

1.5.1 Designs

Now we consider some methods to construct OTBESs. We need the following definition.

Definition 1.5.1 *An (n, β, r, λ)-design is a pair $(X, \mathcal{B})$ such that the following properties are satisfied:*

1. *$|X| = n$.*
2. *$\mathcal{B}$ is a set of β subset of X called blocks (the block can be of different sizes).*
3. *every point of X occurs in exactly r blocks.*
4. *every pair of points occurs in at most λ blocks.*

Usually, $X = \{0, 1, \ldots n-1\}$, $\mathcal{B} = \{B_1, B_2 \ldots B_\beta\}$.
Example. A $(6, 10, 5, 2)$-design: $X = \{0, 1, 2, 3, 4, 5\}$ and the blocks are:

$$012 \;\; 013 \;\; 024 \;\; 035 \;\; 045 \;\; 125 \; 134 \;\; 145 \;\; 234 \;\; 235$$

An incidence matrix of a design is a 0-1 $\beta \times n$ matrix $(a_{i,j})$ indexed by the points and blocks such that

$$a_{i,j} = \begin{cases} 1 & \text{if } i \in B_j \\ 0 & \text{otherwise} \end{cases}$$

A design and its incidence matrix are one-to-one and thus are equivalent. The incidence matrix of the above $(6, 10, 5, 2)$-design is:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Suppose $\mathcal{D} = (X, \mathcal{B})$ is an (n, β, r, λ)-design. Then the dual of $\mathcal{D}$ is $\mathcal{D}^* = (\mathcal{B}, X)$, where $B \in \mathcal{B}$ is contained in $x \in X$ if and only if x is contained in B in $\mathcal{D}$.

Furthermore, if every block in $\mathcal{D}$ has the same size k and any two blocks have at most μ common points, then $\mathcal{D}^*$ is a (β, n, k, μ)-design.

It is easy to see that the incidence matrix of $\mathcal{D}^*$ is the transpose of the incidence matrix of $\mathcal{D}$.

An (n, β, r, λ)-design is called a w-threshold design provided that, for all $F \subseteq X$ such that $|F| \leq w$, we have

$$|\{B \in \mathcal{B} : |F \cap B| \geq 2\}| \leq r - 1.$$

Lemma 1.5.2 *An (n, β, r, λ)-design is a w-threshold design if $r > \lambda\binom{w}{2}$.*

Proof. Suppose $F \subseteq X, |F| \leq w$. Since any pair of points occurs in at most λ blocks, we have $|\{B \in \mathcal{B} : |F \cap B| \geq 2\}| \leq \lambda\binom{w}{2} < r$. So the lemma follows from the definition of w-threshold design. □

We can use an (n, β, r, λ)-w-threshold design to construct an OTBES, where the number of users is n, the privilege set can be any subset of the users, and the forbidden subset will be all the subsets of size w of the users. We briefly outline the construction as follows.

In the scheme, each user i is assigned an $x_i \in X$. For each block B, first the TA chooses $|B| + 1$ secrete values s_B and $s_{B,x}$ for $x \in B$. The value s_B is given to every user in B and the value $s_{B,x}$ is given to each user in $B \backslash \{x\}$. Hence each user of B obtains $|B|$ values.

TA also uses an (r, β) Shamir threshold scheme, define over $\mathbb{F}_q$ for some prime power $q \geq \beta + 1$.

Let $P \subseteq \mathcal{U}$. The TA can broadcast a message $m_P \in \mathbb{F}_q$ to P using the following algorithm:

1. For each $B \in \mathcal{B}$, the TA computes a share $y_B \in \mathbb{F}_q$ corresponding to the secret m_P using the threshold scheme.

2. For each $B \in \mathcal{B}$, TA computes

$$k_B = s_B + \sum_{\{x \in B : x \notin P\}} s_{B,x}.$$

3. For each $B \in \mathcal{B}$, the TA computes

$$b_B = y_B + k_B.$$

4. The broadcast is

$$b_P = (b_B : B \in \mathcal{B}).$$

Each user $x \in P$, x can computes

$$y_B = b_B - k_B.$$

for all $B \in \mathcal{B}$ containing x. So x can compute r shares $\{y_B : x \in B\}$. Thus x can compute the secrete message m_P from the threshold scheme. But for any forbidden subset F, $|F| \leq w, F \cap P = \emptyset$, define

$$A_F = \{B \in \mathcal{B} : |F \cap B| \geq 2\}.$$

The coalition F can compute K_B, and hence y_B, for every $B \in A_F$. However, they can obtain no information about the shares $y_B, B \notin A_F$. Since $|A_F| \leq r - 1$, F has no information about the value of m_P.

Suppose x is contained in $B_1, B_2, \cdots, B_r$. Then x needs to store $\sum_{i=1}^{r} |B_i|$ secret values. Since every pair of points occurs in at most λ blocks, we have $\sum_{i=1}^{r}(|B_i| - 1) \leq \lambda(n-1)$. Hence $\sum_{i=1}^{r} |B_i| \leq r + \lambda(n-1)$. Therefore a user needs to store at most $r + \lambda(n-1)$ values. The broadcast contains β values. So we have proved the following theorem, where $2^{\mathcal{U}}$ means all the possible subsets of $\mathcal{U}$.

Theorem 1.5.3 *Suppose there exists an (n, β, r, λ)-w-threshold design. Then for any prime power $q \geq \beta + 1$, there exists a $(2^{\mathcal{U}}, w)$-OTBES for a set $\mathcal{U}$ of n users, having message set $\mathbb{F}_q$ such that each user needs to store at most $r + \lambda(n-1)$ values in $\mathbb{F}_q$ and the broadcast contains β values in $\mathbb{F}_q$.*

1.5.2 A construction from OAs

There are several methods to construct the required designs. Here we introduce a construction from orthogonal arrays.

Suppose that $A = (a_{i,j})$ is a t-(v, k, λ)-OA. For any two rows r_1 and r_2 of A, define

$$\mu(r_1, r_2) = |\{j : a_{r_1,j} = a_{r_2,j}\}|.$$

Then define

$$\mu(A) = \max\{\mu(r_1, r_2) : r_1 \neq r_2\}.$$

Theorem 1.5.4 *If A is a t-(v, k, λ)-OA, then there exists a $(\lambda v^t, kv, k, \mu(A))$-design.*

Proof. Suppose A has entries from the set $\{1, \ldots, v\}$. Define $X = \{(x, y) : 1 \leq x \leq k, 1 \leq y \leq v\}$. For every row $(y_1, y_2, \ldots, y_k)$ of A, construct a block $\{(y_i, i) : 1 \leq i \leq k\}$. Let $\mathcal{B}$ consist of the λv^t blocks thus obtained. Any two distinct blocks in $\mathcal{B}$ have at most $\mu(A)$ points in common. This implies that the dual design, $(\mathcal{B}, X)$, is a $(\lambda v^t, kv, k, \mu(A))$-design. □

Now we use an example to explain the construction. First we use an OA to construct a design which is expressed as an incidence matrix. Then we obtain a required design using the dual design (according to the transpose of the matrix).

The following is a 2-$(3, 3, 1)$-OA:

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \\ 2 & 2 & 1 \\ 0 & 0 & 2 \\ 1 & 0 & 1 \\ 2 & 1 & 2 \\ 0 & 2 & 0 \end{pmatrix}$$

It is easy to check that A is an OA and $\mu(A) = 1$. From this OA, we can construct a design with incidence matrix:

	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	$(3,0)$	$(3,1)$	$(3,2)$
B_1	1	0	0	0	1	0	0	1	0
B_2	0	1	0	0	0	1	0	0	1
B_3	0	0	1	1	0	0	1	0	0
B_4	0	1	0	0	1	0	1	0	0
B_5	0	0	1	0	0	1	0	1	0
B_6	1	0	0	1	0	0	0	0	1
B_7	0	1	0	1	0	0	0	1	0
B_8	0	0	1	0	1	0	0	0	1
B_9	1	0	0	0	0	1	1	0	0

The dual design is a $(9, 9, 3, 1)$-design.

If A is an t-$(v, k, 1)$-OA, then clearly $\mu(A) = t - 1$. From Corollary 1.2.7, we know that for any prime power q, there exists a t-$(q, q, 1)$-OA. So we have the following from Theorem 1.5.4.

Theorem 1.5.5 *For any prime power q and for any positive integer $t < q$, there exists a $(q^t, q^2, q, t-1)$-design.*

Example. Suppose that p is a prime. Let t, n, w be positive integers such that $p^n > (t-1)\binom{w}{2}$. Then there is a $(2^{\mathcal{U}}, w)$-OTBES for a set $\mathcal{U}$ of p^{nt} users, having message set $\mathbb{F}_{p^{2n+1}}$ such that each user needs to store at most $p^n + (t-1)(p^{nt} - 1)$ values in $\mathbb{F}_{p^{2n+1}}$. The broadcast contains p^{2n} values in $\mathbb{F}_{p^{2n+1}}$.

1.6 Key predistribution schemes

We will use the same model as we did in the previous section. In our model, there is a TA (trusted authority) and a set of users $\mathcal{U} = \{1, 2, \ldots, n\}$. We also assume that the network is a broadcast channel and any information transmitted by the TA will be received by every user.

In a *key predistribution scheme*, the TA generates and distributes secret information to each user. The information given to user i is denoted by u_i and must be distributed "off-band" in a secure manner. This secret information will enable various privileged subsets to compute keys.

We will use $\mathcal{P}$ to denote the collection of all privileged subsets to which the TA is distributing keys. $\mathcal{F}$ will used to denote the collection of all possible coalitions, called forbidden subsets, against which each key is to remain secure.

Let $\mathcal{K}$ be a finite set of keys. There will be a key $k_P \in \mathcal{K}$ for every privileged subset $P \in \mathcal{P}$.

Once the secret information is distributed, each user i in a privileged subset $P \in \mathcal{P}$ should be able to compute the key k_P associated with P. But no forbidden set $F \in \mathcal{F}$ disjoint from P should be able to compute any information about k_P.

We will use $(\mathcal{P}, \mathcal{F})$-KPS to denote such a key predistribution scheme. If $\mathcal{P}$ consists of all t-subsets of $\mathcal{U}$, and $\mathcal{F}$ consists of all w-subsets of $\mathcal{U}$, then the scheme will be denoted (t, w)-KPS. The efficiency of a KPS is measured by the amount of secret information that is distributed to each user.

1.6.1 Key distribution patterns

One method of constructing KPS uses a *key distribution pattern*. We now give a description of the KDP (key distribution pattern).

Let $\mathcal{B} = \{B_1, B_2, \ldots, B_\beta\}$ be a set of subset of $\mathcal{U}$ called blocks. We say that $(\mathcal{U}, \mathcal{B})$ is a $(\mathcal{P}, \mathcal{F})$-Key Distribution Pattern (or $(\mathcal{P}, \mathcal{F})$-KDP) if the following condition is satisfied for all $P \in \mathcal{P}$ and $F \in \mathcal{F}$ such that $P \cap F = \emptyset$;

$$\{B_j : P \subseteq B_j \text{ and } F \cap B_j = \emptyset\} \neq \emptyset.$$

The KDP $(\mathcal{U}, \mathcal{B})$ is public knowledge.

For each user $i \in \mathcal{U}$ define:

$$r_i = |\{B_j : i \in B_j\}|.$$

A construction of KPS using KDP is as follows.

Theorem 1.6.1 *Suppose $(\mathcal{U}, \mathcal{B})$ is a $(\mathcal{P}, \mathcal{F})$-KDP. Let q be any prime power. Then there exists a $(\mathcal{P}, \mathcal{F})$-KPS with $\mathcal{K} = \mathbb{F}_q$ and each user having r_i values of $\mathbb{F}_q$.*

Proof. For $1 \leq j \leq \beta$, the TA chooses a random value $s_j \in \mathbb{F}_q$ and gives s_j to every user in B_j. Thus user i receives r_i elements of $\mathbb{F}_q$ as his or her secret information.

The key k_P for a privileged set $P \in \mathcal{P}$ is defined to be

$$k_P = \sum_{\{j : P \subseteq B_j\}} s_j.$$

So each member of P can compute k_P. However, if $F \in \mathcal{F}$ and $F \cap P = \emptyset$, then there is at least one block B_j such that $P \subseteq B_j$ and $F \cap B_j = \emptyset$. F does not know the value of s_j, and hence has no information about k_P. □

1.6.2 A construction from OAs

Suppose $A = (a_{j,i})$ is a t-(v, n, λ)-OA, where $t \geq 3$ with entries from the set $Y = \{0, 1, \ldots, v-1\}$. Let $\beta = \lambda v^t$, the number of rows of A. Label the rows $1, \ldots, \beta$, label the columns $1, \ldots, n$, and define $\mathcal{U} = \{1, \ldots, n\}$.

Let $Z \subseteq Y$ with $|Z| = z$, where $1 \leq z \leq v-1$. For $1 \leq j \leq \beta$, define

$$B_j = \{i : a_{j,i} \notin Z\},$$

and define

$$\mathcal{B} = \{B_j : 1 \le j \le \beta\}.$$

Let $2 \le s \le t-1$, and denote $w = t - s$. We can use $(\mathcal{U}, \mathcal{B})$ as a (s, w)-KDP. To see that, suppose P is an s-subset and F is a w-subset such that $P \cap F = \emptyset$. Since $s + w = t$, the t-tuples with s elements in $Y \backslash Z$ and w elements in Z appears in some row, say jth row, of A. That means $P \subseteq B_j$ and $F \cap B_j = \emptyset$.

Next we observe that any symbol in Y appears exactly β times in each column in A. Therefore

$$r_i = \frac{(v-z)\beta}{v}$$

for $1 \le i \le n$. So we have the following theorem which comes from Theorem 1.6.1

Theorem 1.6.2 *Suppose there is a t-(v, n, λ)-OA with $t \ge 3$. Suppose that $1 \le z \le v-1$ and $2 \le s \le t-1$. Then for any prime power q, there exists a $(s, t-s)$-KPS for a set of n users, having key set $\mathbb{F}_q$ and each user having $\frac{(v-z)\beta}{v}$ values of $\mathbb{F}_q$.*

We now use an example to explain the construction. The example is not practical useful, but gives an explanation of how to use an OA to construct a KPD.

We have a 3-$(3, 4, 1)$-OA as follows. We display it as its transpose version to save spaces.

```
0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
0 1 2 1 2 0 2 0 1 1 2 0 2 0 1 0 1 2 2 0 1 0 1 2 1 2 0
0 1 2 2 0 1 1 2 0 1 2 0 0 1 2 2 0 1 2 0 1 1 2 0 0 1 2
```

Let $Y = \{0, 1, 2\}$, $Z = \{2\}$ and $\mathcal{U} = \{1, 2, 3, 4\}$. Then the 27 blocks are:

$(1,2,3,4), (1,2,3,4), (4), (2,3,4), (1,3,4), (1,2,4), (1,3,4), (2,3,4), (1,2,4),$
$(1,2,3,4), (3,4), (1,2,4), (1,3,4), (1,2,3,4), (2,4), (2,3,4), (1,2,3,4), (1,4),$
$(3), (1,2,3), (1,2), (1,2,3), (2,3), (1), (1,2,3), (1,3), (2)$

It is easy to check that it is a $(2, 1)$-KDP.

The corresponding $(2, 1)$-KPS is as follows. TA chooses 27 random values $s_1, s_2, \ldots, s_{27}$ which are associated to the 27 blocks. For a user, TA

gives s_i to him if he is in ith block. For example, the user 1 will obtain $s_1, s_2, s_5, s_6, s_7, s_9, s_{10}, s_{12}, s_{13}, s_{14}, s_{17}, s_{18}, s_{20}, s_{21}, s_{22}, s_{24}, s_{25}, s_{26}$.

For a privileged subset of users, say $P = \{1, 2\}$, the key is

$$k_P = s_1 + s_2 + s_6 + s_9 + s_{10} + s_{12} + s_{14} + s_{17} + s_{20} + s_{21} + s_{22} + s_{25}.$$

Since both user 1 and user 2 have all of these values, they can compute k_P. But user 3 does not have s_6, s_9, s_{12}, s_{21}, he cannot get any information about k_P. Similarly, user 4 cannot get any information about k_P as well.

1.7 Codes

Definition 1.7.1 *A code is a pair $(\mathcal{Q}, \mathcal{C})$ such that the following properties are satisfied.*

1. *$\mathcal{Q}$ is a set of elements called* symbols.
2. *$\mathcal{C}$ is a set of n-tuples of symbols called codewords (i.e., $\mathcal{C} \subseteq \mathcal{Q}^N$), where $n \geq 1$ is an integer.*

If $\mathcal{Q} = \mathbb{F}_2$, then a code $(\mathcal{Q}, \mathcal{C})$ is called a binary code.

The concept of "distance" is important to the study of codes. We give several relevant definitions now.

Definition 1.7.2 *Let $(\mathcal{Q}, \mathcal{C})$ be a code, where $\mathcal{C} \subseteq \mathcal{Q}^n$. For $\mathbf{x}, \mathbf{y} \in \mathcal{Q}^n$, define the Hamming distance between $\mathbf{x}$ and $\mathbf{y}$ to be*

$$d(\mathbf{x}, \mathbf{y}) = |\{i : x_i \neq y_i\}|,$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$.

The distance of the code $(\mathcal{Q}, \mathcal{C})$, denoted $d(\mathcal{C})$, is the smallest positive integer d such that $d(\mathbf{x}, \mathbf{y}) \geq d$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}$.

$(\mathcal{Q}, \mathcal{C})$ is an (n, M, d, q)-code if the following properties are satisfied:

1. *$|\mathcal{Q}| = q$,*
2. *$\mathcal{C} \subseteq \mathcal{Q}^n$,*
3. *$|\mathcal{C}| = M$, and*

4. $d(\mathcal{C}) \geq d$.

There are some basic properties of Hamming distance that are easy to verify. We list as a lemma below.

Lemma 1.7.3 *For all* $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{Q}^n$*, the following hold:*

1. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$,
2. $d(\mathbf{x}, \mathbf{y}) = 0$ *if and only if* $\mathbf{x} = \mathbf{y}$.
3. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z})$ *(this is known as the triangle inequality).*

1.7.1 Linear codes

Linear codes are one of the important codes studied in coding theory.

Definition 1.7.4 *A code* $(\mathcal{Q}, \mathcal{C})$ *is a linear code of dimension* m *if* $\mathcal{Q} = \mathbb{F}_q$ *for some prime power* q *and* $\mathcal{C}$ *is an* m*-dimensional subspace of the vector space* $(\mathbb{F}_q)^n$*. The* dual code *of a linear code* $(\mathcal{Q}, \mathcal{C})$ *is the code* $(\mathcal{Q}, \mathcal{C}^\perp)$*, where*

$$\mathcal{C}^\perp = \{\mathbf{y} \in (\mathbb{F}_q)^n : \mathbf{x} \cdot \mathbf{y} = 0 \text{ for all } \mathbf{x} \in \mathcal{C}\}.$$

The $(\mathcal{Q}, \mathcal{C}^\perp)$ *is a linear code of dimension* $n - \dim \mathcal{C}$.

In the above definition, $\mathbf{x} \cdot \mathbf{y}$ denote the inner product over $\mathbb{F}_q$ of the two vectors $\mathbf{x}$ and $\mathbf{y}$. The subspaces $\mathcal{C}$ and $\mathcal{C}^\perp$ are called orthogonal complements of each other.

Suppose that $\mathbf{x} \in (\mathbb{F}_q)^n$. Define the weight of $\mathbf{x}$ to be

$$\mathtt{wt}(\mathbf{x}) = |\{i : x_i \neq 0\}|,$$

where $\mathbf{x} = (x_1, \ldots, x_n)$.

Lemma 1.7.5 *Suppose* $(\mathbb{F}_q, \mathcal{C})$ *is a linear code, where* $\mathcal{C} \subseteq (\mathbb{F}_q)^n$*. Then*

$$d(\mathcal{C}) = \min\{\mathtt{wt}(\mathbf{x}) : \mathbf{x} \in \mathcal{C}, \mathbf{x} \neq (0, \ldots, 0)\}.$$

Proof. Denote $w = \min\{\mathtt{wt}(\mathbf{x}) : \mathbf{x} \in \mathcal{C}, \mathbf{x} \neq (0, \ldots, 0)\}$. Let $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ be two codewords such that $d(\mathbf{x}, \mathbf{y}) = d(\mathcal{C})$. The vector $\mathbf{x} - \mathbf{y} \in \mathcal{C}$ because $\mathcal{C}$ is a linear code. We have $\mathtt{wt}(\mathbf{x} - \mathbf{y}) = d(\mathbf{x}, \mathbf{y}) = d(\mathcal{C})$. Therefore $w \leq d(\mathcal{C})$.

On the other hand, let $\mathbf{x} \in \mathcal{C}$ be a codeword such that $\mathtt{wt}(\mathbf{x}) = w$. The vector $(0, \ldots, 0) \in \mathcal{C}$ because $\mathcal{C}$ is linear. Then $d(\mathbf{x}, (0, \ldots, 0)) = \mathtt{wt}(\mathbf{x}) = w$. Therefore $w \geq d(\mathcal{C})$. ∎

1.7.2 Orthogonal arrays and codes

Theorem 1.7.6 *Suppose that $\mathcal{C} \subseteq (\mathbb{F}_q)^n$ is a linear code of dimension m. Then $(\mathbb{F}_q, \mathcal{C})$ is an (n, q^m, d, q)-code if and only if $\mathcal{C}^\perp$ is a linear $(d-1)$-(q, n, λ)-OA, where $\lambda = q^{n-m-d+1}$.*

Proof. Suppose that $(\mathbb{F}_q, \mathcal{C})$ is a linear (n, q^m, d, q)-doce. Then $\mathcal{C}^\perp$ is a subspace having dimension $n - m$; we will show that it is an orthogonal array. Let D be a basis for $\mathcal{C}^\perp$, and write the vectors in D as an $n-m$ by n matrix. We will prove that D satisfies the conditions of Theorem 1.2.5, and hence it will follow that $\mathcal{C}^\perp$ is an orthogonal array with the stated parameters.

Suppose that there exist $e \leq d-1$ columns of D that are linearly dependent, and therefore there exists a dependence relation of the form

$$\sum_{i=1}^{e} \alpha_i \mathbf{c}_{i_j} = (0, \ldots, 0)^T,$$

where $\mathbf{c}_1, \ldots, \mathbf{c}_n$ are the columns of D. Define a vector $\mathbf{x} = (x_1, \ldots, x_n)$ as follows:

$$x_h = \begin{cases} \alpha_h & \text{if } h = i_j \text{ for some } j \\ 0 & \text{otherwise.} \end{cases}$$

then $\mathbf{x} \cdot \mathbf{r} = 0$ for every row $\mathbf{r}$ of D and hence $\mathbf{x} \in \mathcal{C}$. However, $\mathtt{wt}(\mathbf{x}) = e < d(\mathcal{C})$, which contradicts Lemma 1.7.5.

Conversely, suppose that $\mathcal{C}^\perp$ is a linear $(d-1)$-(q, n, λ)-OA, there $\lambda = q^{n-m-d+1}$. This implies that $\mathcal{C}^\perp$ has dimension $n - m$, and hence $\mathcal{C}$ has dimension m. Let D be a basis for $\mathcal{C}^\perp$; then D has $n - m$ rows when it is written as an array.

We will prove that the minimum distance of $\mathcal{C}$ is at least d. If not, then there exists a vector $\mathbf{x} \in \mathcal{C}$ such that $0 \leq \mathtt{wt}(\mathbf{x}) \leq d-1$. Suppose that the nonzero entries of $\mathbf{x}$ occur in coordinates $i_1, \ldots, i_e$, where $e = \mathtt{wt}(\mathbf{x})$. Then $\mathbf{x} \cdot \mathbf{y} = 0$ for every row $\mathbf{y} \in D$. When $\mathcal{C}^\perp$ is view as an orthogonal array, it follows that

$$\sum_{j=1}^{e} x_{i_j} y_{i_j} = 0$$

for every row $\mathbf{y}$. In other words, in every row of $\mathcal{C}^\perp$, the entries in columns $i_1, \ldots i_e$ satisfy a linear dependence relation. This means that it is impossible that every e-tuple of symbols occurs in a row of $\mathcal{C}^\perp$ within the e columns under consideration. Therefore $\mathcal{C}^\perp$ is not a $(d-1)$-$(q, n\lambda)$-OA, which is a contradiction. This proves that the minimum distance of $\mathcal{C}$ is at least d. ∎

Example. Previously we have an example of a 3-$(5, 5, 1)$-OA which has the basis: $(1, 1, 1, 1, 1), (0, 1, 2, 3, 4)$, and $(0, 1, 4, 4, 1)$. Using linear algebra, we can obtain a basis for the orthogonal complement, for example: $(4, 3, 2, 1, 0)$ and $(2, 3, 4, 0, 1)$. The code generated by these two vectors is a $(5, 25, 4, 5)$-code.

Theorem 1.7.7 ((Singleton Bound)) *Suppose that $\mathcal{C}$ is an (n, M, d, q)-code. Then $M \leq q^{n-d+1}$.*

Proof. Suppose that $M > q^{n-d+1}$. Then, by the pigeonhole principle, there exist two codewords $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ such that $x_i = y_i$ for all i such that $1 \leq i \leq n - d + 1$. Then $d(\mathbf{x}, \mathbf{y}) \leq n - (n - d + 1) = d - 1$. ∎

Corollary 1.7.8 *Suppose that $\mathcal{C}$ is an (n, M, d, q)-code. Then $d \leq n + 1 - \log_q M$.*

Orthogonal arrays with $\lambda = 1$ turn out to be equivalent to codes that meet the Singleton Bound with equality.

Theorem 1.7.9 *An (n, M, d, q)-code in which $M = q^{n-d+1}$ is equivalent to a t-$(q, n, 1)$-OA in which $t = n - d + 1$.*

Proof. Suppose that (X, D) is any t-$(q, n, 1)$-OA. Construct a code $(X, \mathcal{C})$ by taking the q^t rows of D to be the codewords in $\mathcal{C}$. We will prove that $(X, \mathcal{C})$ is an $(n, q^t, n - t + 1, q)$-code, as follows. Suppose that $d(\mathcal{C}) \leq n - t$. Then there exit two codewords $\mathbf{x}, \mathbf{y} \in \mathcal{C}$ such that the entries of $\mathbf{x}$ and $\mathbf{y}$ are the same in at least t columns. Within these t columns, the corresponding rows of D are identical, which contradicts the assumption that $\lambda = 1$ in the orthogonal array (X, D).

Conversely, suppose that $(X, \mathcal{C})$ is an (n, M, d, q)-code in which $M = q^{n-d+1}$. Construct an $M \times n$ array, D, by taking the codewords in $\mathcal{C}$ to be the rows of D. Consider the restriction of D to any subset of $n-d+1$ columns. The $q^{n-1-d}(n - d + 1)$-tuples obtained from the rows of D in this restriction must all be different. Since there are q^{n-d+1} different $(n - d + 1)$-tuples, it follows that every possible $(n - d + 1)$-tuple occurs in exactly one row of D in this restriction. Because this property holds for all possible subsets of $n - d + 1$ columns of D, it follows that D is an $(n - d + 1)$-$(q, n, 1)$-OA. ∎

A code in which the Singleton Bound is met with equality is called a maximum distance separable code (or MDS code). The above theorem establishes that MDS codes are equivalent to orthogonal arrays with $\lambda = 1$.

Corollary 1.7.10 *Let $t \geq 2$ be an integer, and let q be a prime power. Then there exists an MDS $(q, q^t, q - t + 1, q)$-code.*

Proof. Because we already know that there exists a t-$(q, q, 1)$-OA for $t \geq 2$ and q a prime power (see Corollary 1.2.7). □

The codes obtained in above corollary are commonly known as Reed-Solomon codes.

Chapter 2

Basics for Graph Theory

In this chapter, we give basic definitions and notations of graph theory.

2.1 Definitions and Models

Definition 2.1.1 *A graph $G = (V, E)$ is a mathematical structure consisting of two finite sets V and E. The elements of V are called vertices (or nodes), the the elements of E are called edges. Each edge has a set of one or two vertices associated to it, which are called its endpoints.*

An edge is said to join its endpoints. A vertex joined by an edge to a vertex v is said to be a neighbor of v.

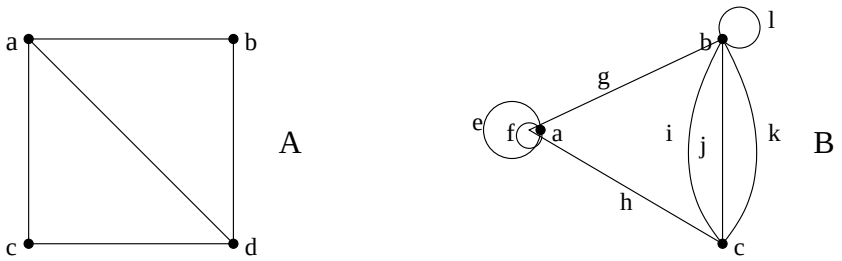


Figure 2.1: Example of graph A and graph B

Example. The Figure 2.1 displays two graphs A and B. Their vertex and edge sets are described as follows.

$V_A = \{a, b, c, d\}$ and $E_A = \{ab, ac, ad, cd, bd\}$

$V_B = \{a, b, c\}$ and $E_B = \{e, f, g, h, i, j, k, l\}$

A proper edge is an edge that joints two different points and a loop is an edge that joints a single endpoint to itself. A multi-edge is a collection of tow or more edges having identical endpoints.

A *simple graph* has neither loops nor multi-edges. Graph A in Figure 2.1 is simple. For a simple graph, the edges can be described as their endpoints.

Definition 2.1.2 *A directed edge (or arc) is an edge, one of whose endpoints is designated as the tail, and whose other endpoint is designated as the head. An arc is said to be directed from its tail to its head. A directed graph (or digraph) is a graph each of whose edges is directed.*

The *underlying graph* of a directed graph G is the graph that results from removing all the designations of head and tail from the directed edges of G.

Example. The digraph D in Figure 2.2 has the graph G as its underlying graph. Graph D has arcs: uv, wu, wv, and vw.

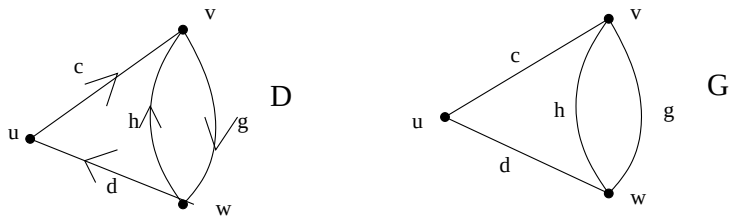


Figure 2.2: A digraph and its underlying graph

2.1.1 Representations of graphs and digraphs

A vertex joined by an edge to a vertex v is said to be a neighbor of v. The neighborhood of a vertex v in a graph G, denoted $N(v)$, is the set of all the neighbors of v. A simple graph can be represented by an adjacency table with a row for each vertex, containing the list of neighbors of that vertex.

For example, the graph A in Figure 2.1 can be represented as:

$$\begin{array}{lllll} a: & b & c & d \\ b: & a & d \\ c: & a & d \\ d: & a & b & c \end{array}$$

For a general graph, we can represent it by a two-row incidence table whose columns are indexed by the edge. The entries in the column corresponding to edge e are the endpoints of e.

For example, the graph B in Figure 2.1 can be represented as follows.

$$V = \{a, b, c\}, \quad E = \{e, f, g, h, i, j, k, l\}$$

edge	e	f	g	h	i	j	k	l
endpts	a	a	a	a	b	b	b	b
	a	a	b	c	c	c	c	b

For a digraph, we can also use incidence table by specifying the tail and head. As an example, the following is a presentation of graph D in Figure 2.2

edge	c	d	h	g
tail	u	w	w	v
head	v	u	v	w

In computer implementation, the above representation might cause problems. So the following method can be used. For the same graph D, the incidence table is as follows, where a superscript "h" is used to indicate the head vertex.

edge	c	d	h	g
endpts	u	w	v^h	v
	v^h	u^h	w	w^h

Definition 2.1.3 *Adjacent vertices are two vertices that are joint by an edge. Adjacent edges are tow edges that have an endpoint in common.*

Matrix representations are also used for graphs. For a simple graph G, we can use the adjacency matrix A_G to represent G. The rows and columns of the adjacency matrix are both indexed by identical ordering of V_G, such that

$$A_G[u, v] = \begin{cases} 1 & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

The following the the adjacency matrix of graph A in Figure 2.1.

$$A_A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

The adjacency matrix of a simple digraph G is similar, where

$$A_G[u,v] = \begin{cases} 1 & \text{if there is an edge from } u \text{ to } v \\ 0 & \text{otherwise} \end{cases}$$

The adjacency matrix of graph D in Figure 2.2 is as follows.

$$A_D = \begin{array}{c} \\ u \\ v \\ w \end{array} \begin{array}{c} \begin{array}{ccc} u & v & w \end{array} \\ \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

Another matrix representation of graph G is its incidence matrix I_G whose rows and columns are indexed by some orderings of V_G and E_G, such that

$$I_G[v,e] = \begin{cases} 0 & \text{if } v \text{ is not an endpoint of } e \\ 1 & \text{if } v \text{ is an endpoint of } e \\ 2 & \text{if } e \text{ is a self-loop at } v \end{cases}$$

The incidence matrix of graph B in Figure 2.1 is

$$I_B = \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{cccccccc} e & f & g & h & i & j & k & l \end{array} \\ \begin{pmatrix} 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

The incidence matrix I_G of digraph G is similar, where

$$I_G[v,e] = \begin{cases} 0 & \text{if } v \text{ is not an endpoint of } e \\ 1 & \text{if } v \text{ is the head of } e \\ -1 & \text{if } v \text{ is the tail of } e \\ 2 & \text{if } e \text{ is a self-loop at } v \end{cases}$$

Example. The incidence of the digraph in Figure 2.3 is

$$I = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{ccccc} e & f & g & h & i \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & -1 \\ -1 & 2 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 1 \end{pmatrix} \end{array}$$

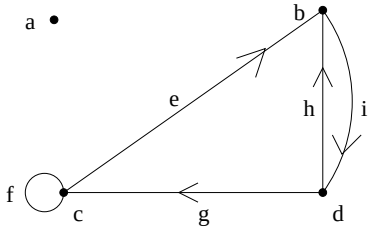


Figure 2.3: A digraph and its incidence matrix

2.1.2 Parameters

Definition 2.1.4 *The degree (or valence) of a vertex v in a graph G, denoted $deg(v)$, is the number of proper edges incident on v, plus twice the number of self-loops.*

The smallest and largest degree in a graph G are denoted δ_{min} and δ_{max}. Some people use δ instead of δ_{min} and Δ instead of δ_{max}

Definition 2.1.5 *The degree sequence of a graph G is the sequence formed by arranging the vertex degrees of G in non-increasing order.*

Example. The degree sequence of graph in Figure 2.4 is: $\langle 4, 4, 3, 1, 0 \rangle$.

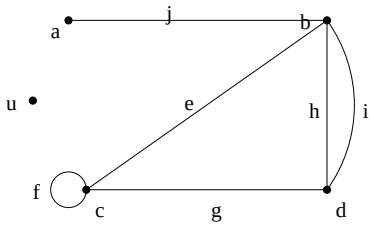


Figure 2.4: A graph and its degree sequence

Two different graphs may have same degree sequences (see Figure 2.5 as an example).

Theorem 2.1.6 (Euler's Degree-sum Theorem) *The sum of the degree of the vertices of a graph is twice the number of edges.*

Proof. Each edge contributes two to the degree sum. □

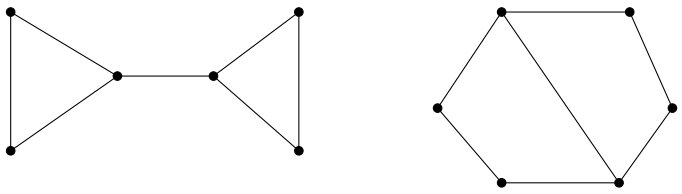


Figure 2.5: Two graphs with same degree sequence

Corollary 2.1.7 *In a graph, there is an even number of vertices having odd degree.*

Proof. Consider separately, the sum of the degree that are odd and the sum of those that are even. The sum of the even degree is even. But the sum of all the degrees is even by Theorem 2.1.6. So the number of vertices having odd degree must be even. □

Theorem 2.1.8 *Suppose that* $\langle d_1, d_2, \ldots, d_n \rangle$ *is a sequence of nonnegative integers whose sum is even. Then there exists a graph with vertices* $v_1, v_2, \ldots, v_n$ *such that* $dev(v_i) = d_i$*, for* $i = 1, \ldots, n$*.*

Proof. Start with n isolated vertices $v_1, v_2, \ldots, v_n$. For each i, if d_i is even, draw $d_i/2$ self-loops on vertex v_i, and if d_i is odd, draw $(d_i - 1)/2$ self-loops. By above Corollary, there is an even number of odd d_i's. Thus the construction can be completed by grouping the vertices associated with the odd terms into pairs and then joining each pair by a single edge. □

Example. Construct a graph with degree sequence $\langle 5, 4, 3, 2, 1, 1 \rangle$.

A sequence $\langle d_1, d_2, \ldots, d_n \rangle$ is said to be *graphic* if there is a permutation of it that is the degree sequence of some simple graph. Such a simple graph is said to realize the sequence.

Theorem 2.1.9 *Let* $\langle d_1, d_2, \ldots, d_n \rangle$ *be a graphic sequence, with* $d_1 \geq d_2 \geq \ldots \geq d_n$*. Then there is a simple graph with vertex set* $\{v_1, v_2, \ldots, v_n\}$ *satisfying* $deg(v_i) = d_i$*,* $1 \leq i \leq n$*, such that* v_1 *is adjacent to vertices* $v_2, \ldots, v_{d_1+1}$*.*

Proof. Since $\langle d_1, d_2, \ldots, d_n \rangle$ is a graphic sequence, there exist simple graphs of n vertices such that $deg(v_i) = d_i, i = 1, \ldots, n$. Let G be one of these graphs

for which $r = |N_G(v_1) \cap \{v_2, \ldots, v_{d_1+1}\}|$ is maximum. If $r = d_1$, then the conclusion follows. If $r < d_1$, then there is a vertex $v_t, t > d_1+1$ such that v_1 is adjacent to v_t because $deg(v_1) = d_1$ and there is a vertex $v_s, 2 \leq s \leq d_1 + 1$, such that v_1 is not adjacent to v_s. Since $deg(v_s > deg(v_t)$, there exists a vertex v_k such that v_k is adjacent to v_s bu not to v_t. Let G' be the graph obtained from G by replacing the edges v_1v_t and v_sv_k. with the edges v_1v_s and v_tv_k. Then the degree are all preserved and $v_s \in N_{G'}(v_1) \cap \{v_2, \ldots v_{d_1+1}\}$. That is $|N_{G'}(v_1) \cap \{v_2, \ldots v_{d_1+1}\}| = r + 1$, which contradicts the choice of G. □

Corollary 2.1.10 *A sequence $\langle d_1, d_2, \ldots, d_n\rangle$ of nonnegative integers such that $d_1 \geq d_2 \geq \ldots, \geq d_n$ is graphic if and only if the sequence $\langle d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2} \ldots, d_n\rangle$ is graphic.*

From the above corollary, we have the following algorithm to decide if a non-increasing sequence is graphic. The algorithm uses a recursive method.

GraphicSequence($\langle d_1, d_2, \ldots, d_n\rangle$)

Input: a non-increasing sequence $\langle d_1, d_2, \ldots, d_n\rangle$.
Output: TRUE if the sequence is graphic; FALSE if it is not.
  if $d_1 = 0$
    return TRUE
  else
    if $d_n < 0$
      return FALSE
    else
      let $\langle a_1, a_2, \ldots, a_{n-1}\rangle$ be a non-increasing permutation of $\langle d_2 - 1, \ldots, d_{d_1+1} - 1, d_{d_1+2} \ldots, d_n\rangle$.
      return GraphicSequence($\langle a_1, a_2, \ldots, a_{n-1}\rangle$)

Definition 2.1.11 *The indegree of a vertex v in a digraph is the number of arcs directed to v and the outdegree of v is the number of arcs directed from v. Each self-loop at v counts one toward the indegree of v and one toward the outdegree.*

Theorem 2.1.12 *In a digraph, the sum of the indegrees and the sum of outdegrees both equal to the number of edges.*

Proof. Each directed edge e contributes one to the indegree at $head(e)$ and one to the outdegree at $tail(e)$. ∎

2.1.3 Some families of graphs

In this subsection, we give some common families of graphs.

Complete graphs

A complete graph is a simple graph such that every pair of vertices is joint by an edge. Any complete graph on n vertices is denoted K_n.

Example. The complete graph K_n for $n = 1, 2, 3, 4, 5$ are displaied in Figure 2.6.

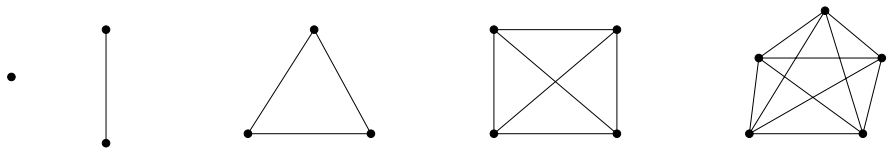


Figure 2.6: Complete graphs of small orders

Bipartite graphs

A bipartite graph G is a graph whose vertex set V can be partitioned into two subsets U and W, such that each edge of G has one endpoint in U and one endpoint in W. The pair U, W is called bipartition of G, and U and W are called the bipartition subsets.

A complete bipartite graph is a simple bipartite graph such that every vertex in one of the the bipartition subsets is joint to every vertex in the other bipartition subset. A complete bipartite graph that has m vertices in one of its bipartition subsets and n vertices in the other is denoted $K_{m,n}$.

Example. The graphs in Figure 2.7 are bipartite graphs. A, C and D are complete bipartite graphs.

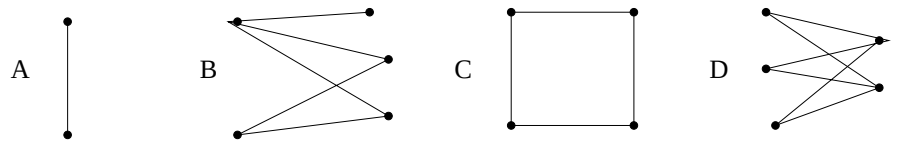


Figure 2.7: Examples of bipartite graphs

Regular graphs

A regular graph is a graph whose vertices all have equal degrees. A k-regular graph is a regular graph whose common degree is k.

Example. The Petersen graph is the 3-regular graph represented by the line drawing in Figure 2.8. Because is possesses a number of interesting graph theoretic properties, the Peterson graph is frequently used both to illustrate established theorems and to test conjectures. Another graph in Figure 2.8 is a 2-regular graph which represents the oxygen molecule O_2.

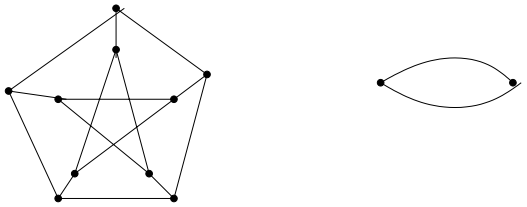


Figure 2.8: Petersen graph

Path graphs and cycle graphs

A path graph P is a simple graph with $|V_P| = |E_P| + 1$ that can be drawn so that all of its vertices and edges lie on a single straight line. A path graph with n vertices and $n - 1$ edges is denoted P_n.

A cycle graph is a single vertex with a self-loop or a simple graph C with $|V_C| = |E_C|$ that can be drawn so that its vertices and edges lie on a single circle.

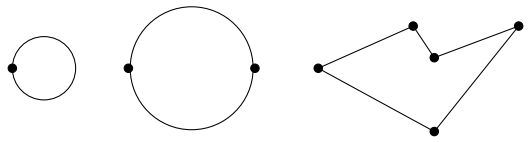


Figure 2.9: Cycle graphs

Hypercubes and circular ladders

The hypercube graph Q_n is the n-regular graph whose vertex set is the set of bit strings of length n such that there is an edge between two vertices if and only if they differ in exactly one bit.

The circular ladder graph CL_n is visualized as two concentric n-cycles in which each of the$n$ pairs of corresponding vertices is joint by an edge.

A hypercube Q_3 and a circular ladder CL_4 are displayed in Figure 2.10

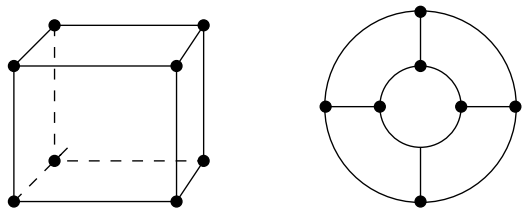


Figure 2.10: Hypercube and circular ladder

Circulant graphs

To the group of integers $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ under addition modulo n and a set $S \subseteq \mathbb{Z}_n \backslash \{0\}$, we associate the circulant graph $circ(n : S)$ whose vertex set is $\mathbb{Z}_n$, such that two vertices i and j are adjacent if and only if there is a number $s \in S$ such that $i + s = j \bmod n$ or $j + s = i \bmod n$. The elements of the set S are called connections. When $S = \{s_1, \ldots, s_r\}$ the circulant graph can be denoted as $circ(n : s_1, \ldots, s_r)$.

The Figure 2.11 displays the $circ(5 : 1, 2)$ and $circ(6 : 1, 2)$.

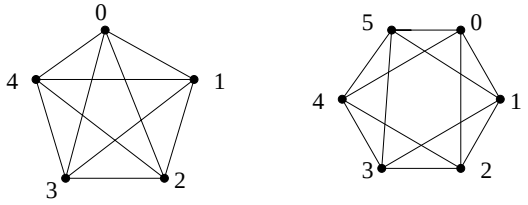


Figure 2.11: Circulant graphs

Intersection and interval graphs

A simple graph G with vertex set $V_G = \{v_1, v_2, \ldots, v_n\}$ is an intersection graph if there exists a family of sets $\mathcal{F} = \{S_1, S_2, \ldots, S_n\}$ such that vertex v_i is adjacent to v_j if and only if $i \neq j$ and $S_i \cap S_j \neq \emptyset$.

A simple graph is an interval graph if it is an intersection graph corresponding to a family of intervals on the real line.

The graph is Figure 2.12 is an interval graph for the following family of intervals:

$$a \leftrightarrow (1,3) \;\; b \leftrightarrow (2,6) \;\; c \leftrightarrow (5,8) \;\; d \leftrightarrow (4,7)$$

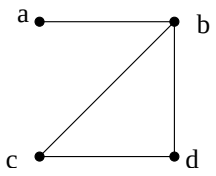


Figure 2.12: Interval graph

Line graph

The line graph of a graph G, denoted $L(G)$, has a vertex for each edge of G and two vertices in $L(G)$ are adjacent if and only if the corresponding edges in G have a vertex in common.

Line graphs are a special case of intersection graph corresponding to the endpoint sets of the edges of G.

The Figure 2.13 displays a graph G and its line graph $L(G)$.

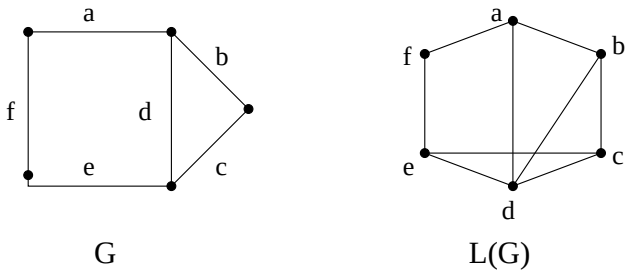


Figure 2.13: G and its line graph

2.1.4 Connectedness

Definition 2.1.13 *In a graph G, a walk from vertex v_0 to vertex v_n is an alternating sequence of vertices and edges*

$$W = \langle v_0, e_2, v_1, e_2, \ldots, v_{n-1}, e_n, v_n \rangle,$$

such that $endpts(e_i) = \{v_{i-1}v_i\}$, for $i = 1, \ldots, n$. If G is a digraph, then W is a directed walk if each edge e_i is directed from vertex v_{i-1} to vertex v_i.

In a simple graph, a walk can be presented as a vertex sequence.

$$W = \langle v_0, v_1 \ldots, v_n \rangle,$$

A walk from a vertex x to a vertex y is also called an x-y walk (an x-y directed walk). The length of a walk or directed walk is the number of edge-steps in the walk. A walk that begins and ends at the same vertex is called a closed walk. Otherwise called an open walk.

A vertex v is reachable from vertex u if there is a walk from u to v.

Definition 2.1.14 *A graph is connected if for every pair of vertices u and v, there is a walk from u to v. A digraph is connected if its underlying graph is connected. A digraph is strongly connected if every two of its vertices are mutually reachable.*

A non-connected graph can be divided into connected subgraphs called components of that graph.

Definition 2.1.15 *The distance $d(s,t)$ from a vertex s to a vertex t in a graph G is the length of a shortest s-t walk if one exists; otherwise, $d(s,t) = \infty$. For digraphs, the directed distance $\vec{d}(s,t)$ is the length of a shortest directed walk from s to t.*

A shortest walk contains no repeated vertices or edges. There are several algorithms to find out the shortest walk. A walk without repeated edges is called a trail. A path is a trail with no repeated vertices (except possibly the initial and final vertices).

If a walk W contains a nontrivial closed sub-walk W', then we can delete W' from W to obtain a reduction of W by deleting from W all of the vertices and edges of W' except the initial vertex of W' (which is also the end vertex of W').

Lemma 2.1.16 *Every open x-y walk is either an x-y path or contains a closed sub-walk.*

Proof. If the walk is not an x-y path, then the subsequence of the walk between repeated vertices defines a closed sub-walk. □

Theorem 2.1.17 *Let W be an open x-y walk. Then either W is an x-y path or there is an x-y path that is a reduced walk of W.*

Proof. If W is not a path, then we can reduce a closed sub-walk by Lemma 2.1.16. Repeat that process until there is no closed sub-walk. □

Corollary 2.1.18 *The distance from a vertex x to a reachable vertex y is always realizable by an x-y path.*

A nontrivial closed path is called a cycle. An acyclic graph is a graph having no cycles.

Definition 2.1.19 *A cycle that includes every vertex of a graph is called a hamiltonian cycle. A graph that has a hamildtonian cycle is called a hamiltonian graph.*

Theorem 2.1.20 *A graph G is bipartite if and only if it has no cycle of odd length.*

Proof. If G is bipartite, then it is easy to see that any cycle (if exists) has odd length.

Now suppose G has no cycles of odd length. Without loss of generality, we assume that G is connected. Pick a vertex u from G. Define

$$X = \{x \in V_G : d(x,u) \text{ is even}\}.\ Y = \{y \in V_G : d(x,u) \text{ is odd}\}.$$

If (X, Y) is not a bipartition of G, then there are two vertices in one of the set, say v and w, that are joint by an edge e. Let P_1 be a shortest u-v path and P_2 be a shortest u-w path. By the definition of X and Y, the length of P_1 and P_2 are both even or odd. Starting from vertex u, let x be the last vertex common to both path. Since both paths are shortest path, their u to x sections have equal length. Therefore the length of x to v in P_1 and the length of x to w in P_2 are both even or odd. Then the concatenations those two sections with edge e forms a cycle of odd length, contradicting the hypothesis. □

Definition 2.1.21 *A weighted graph is a graph in which each edge is assigned a number called its edge-weight.*

Similarly, we can define vertex-weights for a graph. These definitions can also be used for digraphs. Weighted graphs and digraphs have many applications.

2.2 Isomorphism

Deciding whether two line drawing of graphs are presenting a same graph is a difficult problem for large graphs.

Let G and H be two simple graphs. A vertex function $f : V_G \to V_H$ preserves adjacency if for every pair of adjacent vertices u and v in graph G, the vertices $f(u)$ and $f(v)$ are adjacent in graph H. Similarly, f preserves non-adjacency if $f(u)$ and $f(v)$ are non-adjacent whenever u and v are non-adjacent.

A vertex bijection $f : V_G \to V_H$ between the vertex-sets of two simple graphs G and H is structure-preserving if it preserves adjacency and non-adjacency.

Definition 2.2.1 *Two simple graph G and H are isomorphic, denoted $G \cong H$, if there exists a structure-preserving vertex bijection: $f : V_G \to V_H$. Such a function f is called an isomorphism from G to H.*

The above definition can be generalized to general graphs, i.e., a graph have multi-edges and/of loops. In that case, the function f needs to satisfy two extra conditions:

1. the number of edges (even if 0) between every pair of distinct vertices u and v in graph G equals the number of edges between their images $f(u)$ and $f(v)$ in graph H.

2. the number of self-loop at each vertex x in G equals the number of self-loops at the vertex $f(x)$ in H.

Definition 2.2.2 *Let G and H be two isomorphic graphs. A vertex bijection $f_V : V_G \to V_H$ and an edge bijection: $f_E : E_G \to E_H$ are consistent if for every edge $e \in E_G$, the function f_V maps the endpoints of e to the endpoints of edge $f_E(e)$.*

Theorem 2.2.3 *Let G and H be two graphs. Then $G \cong H$ if and only if there is a vertex bijection $f_V : V_G \to V_H$ and an edge bijection $f_E : E_G \to E_H$ that are consistent.*

Proof. If vertex bijection f_V and edge bijection f_E are consistent, then f_V is structure-preserving.

Conversely, if $G \cong H$, then by definition, there is a structure-preserving vertex bijection $f_V : V_G \to V_H$. We can construct a consistent edge bijection as follows: for each pair of distinct vertices $u, v \in V_G$, map the set of edges between u and v bijectively to the set of edges between vertices $f_V(u)$ and $f_V(v)$. This is that because these two edge set have the same number of elements. □

Definition 2.2.4 *If G and H are graphs with multi-edges, then an isomorphism from G to H is specified by giving a vertex bijection $f_V : G_V \to V_H$ and an edge bijection $f_E : E_G \to E_H$ that are consistent.*

2.2.1 Properties of isomorphic graphs

The following results provide some properties of isomorphic graphs which can be used as some preliminary criteria to check if a pair graphs are isomorphic. These properties actually apply to general graphs as well.

Theorem 2.2.5 *Let G and H be isomorphic graphs with the isomorphism f. Then*

1. *They have the same number of vertices and the same number of edges.*
2. *$deg(v) = deg(f(v))$ for any $v \in V_G$.*
3. *They have the same degree sequence.*
4. *The endpoints of edge $f(e)$ have the same degrees as the endpoints of e.*

The proof of the theorem is straightforward and omitted.

If $f = (f_V, f_E)$ is an isomorphism from G to H, then $f^{-1} = (f_V^{-1}, f_E^{-1})$ is an isomorphism from H to G. Thus, the relation $\cong$ is symmetric. Reflexivity and transitivity are also easy to prove. So the relation $\cong$ is an equivalence relation.

Each equivalence class under $\cong$ is called an isomorphism type. Isomorphic graphs are graphs of the same isomorphism type. Counting isomorphism types of graphs generally involves the algebra of permutation groups.

Example. There are four isomorphism types for simple 3-vertex graphs.

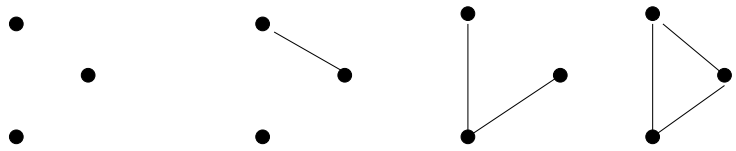


Figure 2.14: Isomorphic types for 3 vertices

The graph-isomorphism problem is to devise a practical general algorithm to decide graph isomorphism, or prove no such algorithm exists.

Although for some specific types of graphs, there are polynomial algorithms to check if two graphs are isomorphic. But in general, it is an NP problem neither known to be solvable in polynomial time nor NP-complete.

2.2.2 Automorphisms and symmetry

2.3 Trees

Definition 2.3.1 *A tree is a connected graph that has no cycles. A leave is a vertex of degree 1 in a tree.*

Some characteristics of a tree is summarized in the following theorem. We omitted the proofs.

Theorem 2.3.2 *Let T be a graph with n vertices. Then the following statements are equivalent.*

1. *T is a tree.*
2. *T contains no cycles and has $n-1$ edges.*
3. *T is connected and has $n-1$ edges.*
4. *T is connected, and every edge is a cut-edge.*
5. *Any two vertices of T are connected by exactly one path.*
6. *T contains no cycle, and for any new edge e, the graph $T+e$ has exactly one cycle.*

A directed tree is a directed graph whose underlying graph is a tree. A rooted tree is a tree with a designated vertex called the root. Each edge is considered to be directed away from the root. So a rooted tree is a directed tree such that the root has indegree 0 and all other vertices have indegree 1.

In a rooted tree, the *depth* or *level* of a vertex v is its distance from the root. The *height* of a rooted tree is the length of a longest path form the root.

If vertex v immediately precedes vertex w on the path from the root to w, then v is the *parent* of w and w is the *child* of v. Vertices having the same parent are called *siblings*.

An *internal vertex* in a rooted tree is any vertex that has at least on child. The root is internal unless the tree is trivial.

2.3.1 Shortest path trees

Definition 2.3.3 *Let v be a vertex in a connected graph G. A shortest-path tree for G from v is a rooted tree T with vertex-set V_G and root v such that the unique path in T from v to each vertex w is a shortest path in G from v to w.*

A *spanning* tree T of a graph G is a tree with vertex-set V_G. The discovery order is a listing of the vertices of G in the order in which they are added (discovered) as tree T is grown (i.e., the distance from the vertex to the start vertex). The position of a vertex in this list, starting with 0 for the start vertex, is called the *discovery number* of that vertex.

A *frontier edge* for a given tree T in a graph is a non-tree edge with one endpoint in T called its *tree endpoint*, and one endpoint not in T, its *non-tree endpoint*.

Next we introduce an algorithm for form a shortest path tree from a connected graph G. First we need a function called *nextEdge* defined as follows.

Let S be the current set of frontier edges. The function *bfs-nextEdge* is defined as follows: $bfs\text{-}nextEgde(G, S)$ selects and return as its value the frontier edge whose tree-endpoint has the smallest discovery number. number. If there is more than one such edge, then $bfs\text{-}nextEdge(G, S)$ selects the one determined by the default priority.

Breadth-first Search

Input: a connected graph G, a starting vertex $v \in V_G$.
Output: an ordered spanning tree T of G with root v.
  Initialize tree T as vertex v.
  Initialize S as the set of proper edges incident on v.
  While $S \neq \emptyset$
    Let $e = bfs\text{-}nextEdge(G, S)$.
    Let w be the non-tree endpoint of edge e.
    Add edge e and vertex w to tree T.
    $updateFrontier(G, S)$.
  Return tree T.

Theorem 2.3.4 *The breadth-first tree produced by application of above algorithm is a shortest-path tree for the input graph.*

For a weighted graph G, one interesting problem is the shortest-path problem: Let s and t be two vertices of a connected weighted graph. Find a path from s to t whose total edge-weight is minimum.

Dijkstra's algorithm is some generalization of the breadth-first search algorithm, that are used to solve the shortest-path problem.

The function $Dijkstra\text{-}nextEdge$ is defined as follows: Let S be the current set of frontier edges. $Dijkstra\text{-}nextEdge(G, S)$ selects and returns as its value the frontier edge whose non-tree endpoint is closest to the start vertex s (means the edge-weight from it to s is the minimum). If there is more than one such edge, the function selects the one determined by the default priority.

Dijkstra's Shortest Path

Input: a weighted connected graph G, a starting vertex $s \in V_G$.
Output: a shortest-path tree T with root s.
  Initialize tree T as vertex s.
  Initialize S as the set of proper edges incident on s.
  While $S \neq \emptyset$
    Let $e = Dijkstra\text{-}nextEdge(G, S)$.
    Let w be the non-tree endpoint of edge e.
    Add edge e and vertex w to tree T.
    $updateFrontier(G, S)$.
  Return tree T.

Theorem 2.3.5 *Let T_j be the Dijkstra tree after j iterations of Dijkstra's algorithm on a connected graph G, for $0 \leq j \leq |V_G| - 1$. Then for each v in T_j, the unique s-v path in T_j is a shortest s-v path.*

Proof. The assertion is trivial for T_0. By way of induction, assume for some $j, 0 \leq j \leq |V_G| - 2$, that T_j satisfies the property. Let e, with labeled endpoint x and unlabeled endpoints y be the frontier edge added to T_j in the $(j + 1)$ iteration. Since y is the only new vertex in T_{j+1}, it suffices to show that the s-y path Q in T_{j+1} is a shortest path in G.

Now let R be any s-y path in G. Suppose that edge f, with endpoints v and z is the first edge of path R that is not in T_j (so v is in T_j but z is not). By the algorithm, the weight from s to y along Q is not greater than the weight from s to z along R. Therefore the weight of Q will not be greater than the weight of R. □

2.3.2 Huffman trees and optimal prefix codes

Definition 2.3.6 *A binary tree is a tree in which each vertex has at most two children. In a binary tree, each child is designated either a left-child of a right-child. The left (right) subtree of a vertex v in a binary tree is the binary subtree spanning the left (right)-child of v and all of its descendants.*

Binary trees have a lot of applications in compute science. Here we introduce one application of encoding method.

Definition 2.3.7 *A binary code is an assignment of symbols or other meanings to a set of bitstrings. Each bitstring is referred to as a codeword. A prefix code is a binary code with the property that no codeword is an initial substring of any other codeword.*

A binary tree can be used to construct a prefix code for a given set of symbols. The method is as follows. First, draw an arbitrary binary tree whose leaves are bijectively labeled by the symbols (i.e., each leaf gets a different symbol). Next, label every edge that goes to a left-child with a 0, and label every edge that goes to a right child a 1. Then each symbol corresponds to the codeword formed by the sequence of edge labels on the path from the root to the leaf labeled by that symbol.

Example. A binary tree used to a prefix code for the alphabet set $\{a, b, c, d, e, f, g\}$ as in Figure 2.15.

letter	a	b	c	d	e	f	g
$frequency$	.2	.05	.1	.1	.25	.15	.15
codeword	000	0010	0011	0101	011	100	101

In a prefix code, if we use shorter codewords to encoder the more frequently occurring symbols, then the message will tend to require fewer bits for encoding. Therefore one measure of a code's efficiency is the average

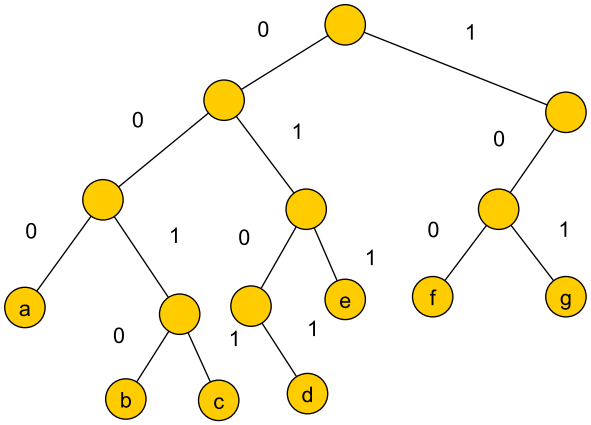


Figure 2.15: A binary tree for coding

weighted length of its codewords, where the length of each codeword is multiplied by the frequency of the symbol it encodes.

The average weighted construction of a codeword for the prefix code in Figure 2.15 is

$$3 \times .2 + 4 \times .05 + 4 \times .1 + 3 \times .25 + 3 \times .15 + 3 \times .15 = 3.25$$

For a prefix code constructed from a binary tree, the length of each codeword is simply the depth of its corresponding leaf. So we will use the following definition to discuss the efficiency of the code.

Definition 2.3.8 *Let T be a binary tree with leaves $s_1, s_2, \ldots, s_l$, such that each leave s_i is assigned a weight w_i. Then the average weighted depth of the binary tree T , denoted $wt(T)$, is given by*

$$wt(T) = \sum_{i=1}^{l} depth(s_i) \cdot w_i.$$

If the weight assigned to a leaf is the frequency of the symbol that labels that leaf, then the average weighted length of a codeword equals the average weighted depth of the binary tree.

Huffman Prefix Code

Input: a set $\{s_1, s_2, \ldots, s_l\}$ of symbols; a list $\{w_1, w_2, \ldots, w_l\}$, where w_i is the weight associated with symbol s_i.
Output: a binary tree representing a prefix code for a set of symbols whose codewords have minimum average weighted length.

Initialize F to be a forest of isolated vertices, labeled $s_1, \ldots, s_l$, with respective weights $w_1, \ldots, w_l$.
For $i = 1$ to $l - 1$
Choose from F two trees, T and T', of smallest weights in F.
Create a new binary tree whose root has T and T' as its left and right subtrees, respectively.
Label the edge to T with a 0 and the edge to T' with a 1.
Assign to the new tree the weight $w(T) + w(T')$.
Replace trees T and T' in forest F by the new tree.
Return$F$.

Using the Huffman prefix code algorithm, the code for the above example will be

letter	*a*	*b*	*c*	*d*	*e*	*f*	*g*
frequency	.2	.05	.1	.1	.25	.15	.15
codeword	00	0100	0101	011	10	110	111

The corresponding binary tree is as in Figure 2.16.

The average length of a codeword for this prefix code is

$$2 \times .2 + 4 \times .05 + 4 \times .1 + 3 \times .1 + 2 \times .25 + 3 \times .15 + 3 \times .15 = 2.7$$

Lemma 2.3.9 *If the leaves of a binary tree are assigned weights, and if each internal vertex is assigned a weight equal to the sum of its children's weights, then the tree's average weighted depth equals the sum of the weights of its internal vertices.*

This Lemma can be easily proved using a recursive method: when the tree only has one or two leaves, it is easy to verify. Now suppose the lemma is true for the left subtree and the right subtree, then it is also easy to prove that it is true for the whole tree.

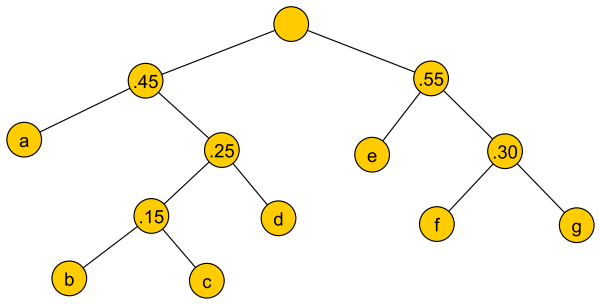


Figure 2.16: Huffman code for the example

Theorem 2.3.10 *For a given list of weights $w_1, w_2, \ldots, w_l$, a Huffman tree has the smallest possible average weighted depth among all binary trees whose leaves are assigned those weight.*

Proof. If $l = 2$, then the Huffman tree consists of a root with weight $w_1 + w_2$ and a left-child and a right-child. Any other binary tree with two leaves has at least two internal vertices, one with weight $w_1 + w_2$. So it has greater average weight depth by Lemma 2.3.9.

Now assume for $l \geq 2$ that the Huffman algorithm produces a Huffman tree of minimum average weight depth for any list of l weight. Consider the case $l+1$. Let $w_1, w_2, \ldots, w_{l+1}$ be any list of weights and w_1, w_2 are two of the smallest ones. Then the Huffman tree H that results consists of a Huffman tree H' for the weights $w_1 + w_2, w_3, \ldots, w_{l+1}$, where the leaf of weight $w_1 + w_2$ is replaced by a vertex with leaves assigned weights w_1 and w_2. By Lemma 2.3.9, $wt(H) = wt(H') + w_1 + w_2$ and H' is optimal among all binary trees whose leaves are assigned weights, $w_1 + w + 2, w_3, \ldots, w_{l+1}$ by induction.

Now suppose T^* is an optimal binary tree for the weights $w_1, w_2, \ldots, w_{l+1}$. Let x be an internal vertex with greatest depth and suppose that y and z are its left-child and right-child, respectively. We can assume that y and z have weight w_1 and w_2, since otherwise we could swap their weight with w_1 and w_2 to produce a tree with smaller average weight depth. Consider the tree T' formed by deleting the leaves y and z from T^*. Then $wt(T^*) = wt(T') + w_1 + w_2$. But T' is a binary tree with leaves of weight $w_1 + w_2, w_3, \ldots, w_{l+1}$, and hence $wt(T') \geq wt(H')$. Therefore $wt(T^*) \geq wt(H)$. □

Chapter 3

Network Flows

3.1 Flow and cuts in networks

Definition 3.1.1 *A single source-single sink network is a connected digraph that has a distinguished vertex called the source, with nonzero outdegree, and a distinguished vertex called the sink, with nonzero indegree.*

A single source-single sink network with source s and sink t is refereed as an s-t network.

A capacitated network is a connected digraph such that each arc e is assigned a nonnegative weight $cap(e)$, called the capacity of arc e.

Let v be a vertex in a digraph N. The out-set of v denoted $Out(v)$ is the set of all arcs that are directed from vertex v and the in-set of v, denoted $In(v)$, is the set of all arcs that are directed to vertex v.

For any two vertex subsets X and Y of a digraph N, let $\langle X, Y\rangle$ denote the set of all arcs in N that are directed from a vertex in X to a vertex in Y. So

$$\langle X, Y\rangle = \{e \in E_n : tail(e) \in X \text{ and } head(e) \in Y\}.$$

Example. A 5-vertex capacitated s-t network is displayed in Figure 3.1. If $X = \{x, v\}$ and $Y = \{w, t\}$, then $\langle X, Y\rangle$ contains 2 arcs and $\langle Y, X\rangle$ contains one arc.

Definition 3.1.2 *Let N be a capacitated s-t network. A feasible flow f in N is a function $f : E_N \to R^+$ that assigns a nonnegative real number $f(e)$ to each arc e such that*

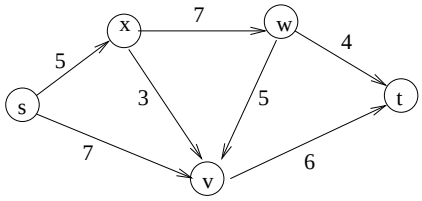


Figure 3.1: An s-t network

1. $f(e) \leq cap(e)$, *for every arc* e *in network* N.
2. $\sum_{e \in In(v)} f(e) = \sum_{e \in Out(v)} f(e)$ *for every vertex* v *in network* N, *other than source* s *and sink* t.

In the definition, the first property is called capacity constraints and the second property is called conservation constraints.
Example. Figure 3.2 displays a feasible flow for the network at Figure 3.1, where the number at the right of a label of an arc denotes the capacity while the number at right side denotes $f(e)$.

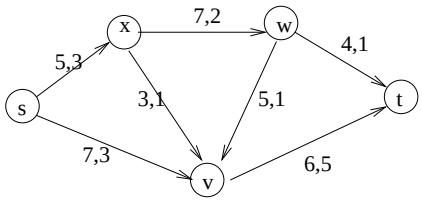


Figure 3.2: A flow for a network

Definition 3.1.3 *The value of flow* f *in a capacitated network* N, *denoted* $val(f)$, *is the net flow leaving the source* s, *i.e.,* $val(f) = \sum_{e \in Out(s)} f(e) - \sum_{e \in In(s)} f(e)$. *The maximum flow* f^* *is a flow having the maximum value.*

The value of the net flow of the graph in Figure 3.2 is 6. It is not difficult to see that $f^* = 10$.

Definition 3.1.4 *Let* N *be an* s-t *network, and let* V_s *and* V_t *form a partition of* V_N *such that source* $s \in V_s$ *and sink* $t \in V_t$. *Then the set of all arcs that are directed from a vertex in set* v_s *to a vertex in set* V_t *is called an* s-t *cut of network* N *and denoted* $\langle V_s, V_t \rangle$

Lemma 3.1.5 *Let $\langle V_s, V_t\rangle$ be an s-t cut of a network N. Then every directed s-t path in N contains at least one arc in $\langle V_s, V_t\rangle$.*

The proof is simple.

It is easy to see that $val(f)$ can be rewritten as

$$val(f) = \sum_{e\in\langle\{s\},V_N-\{s\}} f(e) - \sum_{e\in\langle V_N-\{s\},\{s\}\rangle} f(e).$$

The above concept can be generalized.

Lemma 3.1.6 *Let $\langle V_s, V_t\rangle$ be any s-t cut of an s-t network N. Then*

$$\bigcup_{v\in V_s} Out(v) = \langle V_s, V_s\rangle \cup \langle V_s, V_t\rangle \text{ and } \bigcup_{v\in V_s} In(v) = \langle V_s, V_s\rangle \cup \langle V_t, V_s\rangle$$

Proof. For any vertex $v \in V_s$, each arc directed from v is either in $\langle V_s, V_s\rangle$ or in $\langle V_s, V_t\rangle$. Similarly, each arc directed to vertex v is either in $\langle V_s, V_s\rangle$. □

Using above lemma, we can prove the following result.

Theorem 3.1.7 *Let f be a flow in an s-t network N, and let $\langle V_s, V_t\rangle$ be any s-t cut of N. Then*

$$val(f) = \sum_{e\in\langle V_s,V_t\rangle} f(e) - \sum_{e\in\langle V_t,V_s\rangle} f(e)$$

Proof. By definition, $val(f) = \sum_{e\in Out(s)} f(e) - \sum_{e\in In(s)} f(e)$, and by the conservation of flow, $\sum_{e\in Out(v)} f(e) - \sum_{e\in In(v)} f(e) = 0$ for every $v \in V_s$ other than s. Therefore, $val(f) = \sum_{v\in V_s}\left(\sum_{e\in Out(v)} f(e) - \sum_{e\in In(v)} f(e)\right) = \sum_{v\in V_s}\sum_{e\in Out(v)} f(e) - \sum_{v\in V_s}\sum_{e\in In(v)} f(e)$. By Lemma 3.1.6, we have
$\sum_{v\in V_s}\sum_{e\in Out(v)} f(e) = \sum_{e\in\langle V_s,V_s\rangle} f(e) + \sum_{e\in\langle V_s,V_t\rangle} f(e)$ and
$\sum_{v\in V_s}\sum_{e\in In(v)} f(e) = \sum_{e\in\langle V_s,V_s\rangle} f(e) + \sum_{e\in\langle V_t,V_s\rangle} f(e)$.
The conclusion follows by substitution. □

Corollary 3.1.8 *Let f be a flow in an s-t network. Then*

$$val(f) = \sum_{e\in In(t)} f(e) - \sum_{e\in Out(t)} f(e)$$

Proof. Apply the above theorem to the *s-t* cut $In(t) = \langle V_N - \{t\}, \{t\}\rangle$. □

Definition 3.1.9 *The capacity of a cut $\langle V_s, V_t\rangle$, denoted $cap\langle V_s, V_t\rangle$, is the sum of the capacities of the arcs in cut $\langle V_s, V_t\rangle$. A minimum cut of a network N is a cut with the minimum capacity.*

Example. The minimum cut of the graph in Figure 3.1 is $\langle\{s, x, v, w\}, \{t\}\rangle$, with capacity 10.

Next we will explain that the problem of finding the maximum flow in a capacitated network N and finding a minimum cut in N are closely related.

Lemma 3.1.10 *Let f be any flow in an s-t network N, and let $\langle V_s, V_t\rangle$ be any s-t cut. Then $val(f) \leq cap\langle V_s, V_t\rangle$.*

Proof. From Theorem 3.1.7, we have

$$\begin{aligned} val(f) &= \sum_{e\in\langle V_s,V_t\rangle} f(e) - \sum_{e\in\langle V_t,V_s\rangle} f(e) \\ &\leq \sum_{e\in\langle V_s,V_t\rangle} cap(e) - \sum_{e\in\langle V_t,V_s\rangle} f(e) \\ &= cap\langle V_s, V_t\rangle - \sum_{e\in\langle V_t,V_s\rangle} f(e) \\ &\leq cap\langle V_s, V_t\rangle \end{aligned}$$

□

Corollary 3.1.11 (Weak Duality for Flows) *Let f^* be a maximum flow in an s-t network N, and let K^* be a minimum s-t cut in N. Then*

$$val(f^*) \leq cap(K^*).$$

Proof. This follows directly from the above Lemma. □

Corollary 3.1.12 (Certificate of Optimality for flows) *Let f be a flow in an s-t network N and K an s-t cut, and suppose that $val(f) = cap(K)$. Then flow f is a maximum flow in network N, and cut K is a minimum cut.*

Proof. Let f^* be any feasible flow in network N. The maximality of flow f is implied by Lemma 3.1.10 that $val(f^*) \leq cap(K) = val(f)$. The minimality of cut K can be established with a similar argument. □

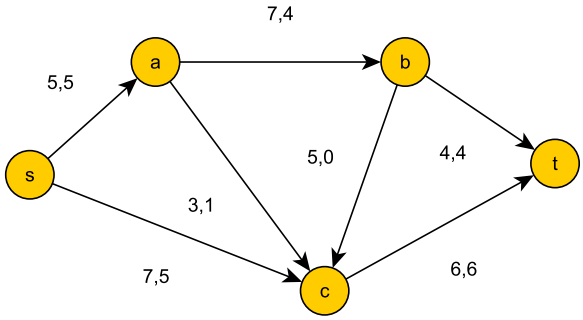


Figure 3.3: A maximum flow and minimum cut

Example. The flow for the example in Figure 3.3 has value 10 which is also the capacity of the s-t cut $\langle\{s, a, b, c\}, \{t\}\rangle$. So both the flow and the cut are optimal according to the above corollary.

Corollary 3.1.13 *Let $\langle V_s, V_t\rangle$ be an s-t cut in a network N, and suppose that f is a flow such that*

$$f(e) = \begin{cases} cap(e) & if\ e \in \langle V_s, V_t\rangle \\ 0 & if\ e \in \langle V_t, V_s\rangle \end{cases}$$

Then f is a maximum flow in N, and $\langle V_s, V_t\rangle$ is a minimum cut.

3.1.1 The maximum-flow problem

In this section we introduce the work originated by Ford and Fulkerson.

Suppose that F is a flow in a capaciated s-t network N, and suppose that there exists a directed s-t path $P = \langle s, e_1, v_1, \ldots, e_k, t\rangle$ in N, such that $f(e_i) < cap(e_i)$ for $i = 1, \ldots, k$. Then considering arc capacities only, the flow on each arc e_i can be increased by as much as $cap(e_i) - f(e_i)$. But to maintain the conservation-of-flow property at each of the vertices v_i, the increases on all the arcs of path P must be equal. Thus, if Δ_P denotes this increase, then the largest possible value for Δ_P is $\min\{cap(e_i) - f(e_i)\}$.

Example. In the example network shown on th left in Figure 3.4, the value of the current flow is 6. Consider the path $P = \langle s, x, w, t\rangle$. The flow on each

arc of the path can increase by $\Delta_P = 2$, and the resulting flow having value 8 is shown in the center. Similarly, consider the path $\langle s, v, t\rangle$ and increase $\Delta = 1$, the flow can then be increased to 9. The resulting flow is shown on the right of Figure 3.4.

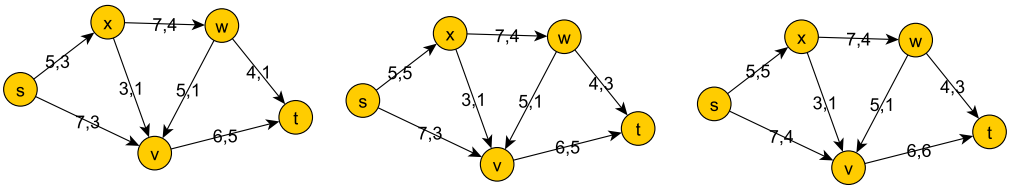


Figure 3.4: Increase flow

In this point, the flow cannot be increased any further along directed s-t path, because each such path must use either the arc directed from s to x or the arc from v to t.

However, the flow can be increased further. One method is to increase the flow on the arc from s to v by one unit, and decrease the flow on the arc from w to v by on unit.

Definition 3.1.14 *An s-t quasi-path in a network N is an alternating sequence*

$$Q = \langle s = v_0, e_1, v_1, \ldots, v_{k-1}, e_k, v_k = t\rangle$$

of vertices and arcs that forms an s-t path in the underlying graph of N. In the quasi-path, an arc e_i is called forward arc if it is directed for vertex v_{i-1} to v_i, and arc e_i is called a backward arc if it is directed from v_i to v_{i-1}.

Clearly, a quasi-path is a path if its arcs are all forward. A quasi-path is also called semipath.

Definition 3.1.15 *Let f be a flow in an s-t path network N. An f-augmenting path Q is an s-t quasi-path in N such that the flow on each forward arc can be increased, and the flow on each backward arc can be decreased.*

By the definition, for each arc e on an f-augmenting path Q:

$$\begin{array}{ll} f(e) < cap(e), & \text{if } e \text{ is a forward arc} \\ f(e) > 0, & \text{if } e \text{ is a backward arc} \end{array}$$

For each arc e on a given f-augmenting path, let Δ_e be the quantity given by

$$\Delta_e = \begin{cases} cap(e) - f(e), & \text{if } e \text{ is a forward arc} \\ f(e), & \text{if } e \text{ is a backward arc} \end{cases}$$

The quantity Δ_e is called the slack on arc e.

Conservation of flow requires that the change in flow on the arcs of an augmenting flow path be of equal magnitude. Therefore the maximum allowable change in the flow on an arc of quasi-path Q is

$$\Delta_Q = \min_{e \in Q}\{\Delta_e\}.$$

Example. For the graph in Figure 3.4, the current flow f has value 9, and the quasi-path $Q = \langle s, v, w, t\rangle$ is an f-augmenting path, with $\Delta_Q = 1$. Therefore we can change the flow so that the value will increase to 10. The resulting network is in Figure 3.3.

Lemma 3.1.16 (Flow Augmentation) *Let f be a flow in a network N, and let Q be an f-augmenting path with minimum slack Δ_Q on its arcs. Then the augmented flow*

$$f^* = \begin{cases} f(e) + \Delta_Q, & \textit{if } e \textit{ is a forward arc of } Q \\ f(e) - \Delta_Q, & \textit{if } e \textit{ is a backward arc of } Q \\ f(e), & \textit{otherwise} \end{cases}$$

is a feasible flow in network N, and $val(f^*) = val(f) + \Delta_Q$.

Proof. Clearly $0 \leq f^*(e) \leq cap(e)$, by the definition of Δ_Q. The only vertices through which the net flow may have changed are those vertices on the augmenting path Q. To verify that f^* satisfies conservation of flow, only the internal vertices of Q need to be checked.

For a given vertex v on augmenting path Q, the two arcs of Q that are incident on v are configured in one of four possible ways. In each case, the net flow into or out of the vertex does not change, so the conservation of flow property is preserved.

Finally, we need to show that the flow have increased by Δ_Q. The only arc incident on source s whose flow has changed is the first arc e_1 of augmenting path Q. If e_1 is a forward arc, then $f^*(e_1) = f(e_1) + \Delta_Q$, and if e_1 is a backward arc, then $f^*(e_1) = f(e_1) - \Delta_Q$. In either cases, $val(f^*) = \sum_{e \in Out(s)} f^*(e) - \sum_{e \in In(s)} f^*(e) = \Delta_Q + val(f)$. ∎

Corollary 3.1.17 *Let f be an integer-valued flow in a network whose arc capacities are integers. Then the flow that results from each successive flow augmentation will be integer valued.*

Theorem 3.1.18 (Characterization of Maximum Flow) *Let f be a flow in a network N. Then f is a maximum flow in network N if and only if there does not exist an f-augmenting path.*

Proof. Suppose that f is a maximum flow in network N. Then by Lemma 3.1.16, there is no f-augmenting path.

Now suppose that there does not exist any f-augmenting path in network N. Consider the collection of all quasi-paths that start at vertex s and have the following property: each forward arc on the quasi-path has positive slack and each backward arc on the quasi-path has positive flow. Let V_s be the vertex-set of these quasi-paths.

Since there is no f-augmenting path, it follows that sink $t \notin V_s$. Let $V_t = V_N - V_s$. Then $\langle V_s, V_t \rangle$ is an s-t cut of network N. By definition of V_s and V_t,

$$f(e) = \begin{cases} cap(e) & \text{if } e \in \langle V_s, V_t \rangle \\ 0 & \text{if } e \in \langle V_t, V_s \rangle \end{cases}$$

Therefore f is a maximum flow by Corollary 3.1.13. □

Theorem 3.1.19 (Max-Flow Min-Cut) *For a given network, the value of a maximum flow is equal to the capacity of a minimum cut.*

Proof. The s-t cut constructed in the proof of the above theorem has capacity equal to value of the maximum flow. □

3.1.2 Algorithms

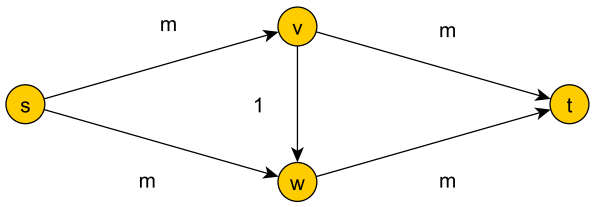
In this subsection, we introduce several algorithms for finding f-augmenting paths. The first algorithm is due to Ford and Fulkerson, that finds an f-augmenting path when one exists.

To use that algorithm, we need a method to find out augmenting paths. Ford and Fulkerson use a method similar to the breadth-first search algorithm. Recall that we define a frontier arc to be an arc e directed from a labeled endpoint to an unlabeled endpoint. But for f-augmenting path,

Outline for Maximum Flow

Input: as s-t network N.
Output: a maximum flow f^* in network N.
[Initialization]
For each arc e in network N
$f^*(e) := 0$
[Flow Augmentation]
While there exists an f^*-augmenting path in network N
Find an f^*-augmenting path Q
Let $\Delta_Q = \min_{e \in Q}\{\Delta_e\}$.
If e is a forward arc
$f^*(e) := f^*(e) + \Delta_Q$
Else
$f^*(e) := f^*(e) - \Delta_Q$
Return flow f^*.

frontier arc is allowed to be backward. Let e be a frontier arc of tree T with endpoints v and w. Then arc e is said to be usable if for the current flow f, either e is directed from vertex v to vertex w and $f(e) < cap(e)$, or e is directed from vertex w to v and $f(e) > 0$. So each time, the algorithm can add a usable arc to the tree. If the vertex t is labeled, then an augmenting path is find. But this algorithm sometimes is not efficient. For example, the following network could requires as many as $2m$ augmentations if the augmenting path is always go through the edge vw.



To avoid that situation, Edmonds and Karp gave the following algorithm

which is a slight refinement of the Ford-Fulkerson scheme. The algorithm either returns an f-augmenting path or returns a minimum cut that indicates the current flow f is a maximum flow.

Finding an Augmenting Path

Input: a flow f in an s-t network N.
Output: an f-augmenting path Q or a minimum s-t cut with capacity $val(f)$.
Initialize vertex set $V_s := \{s\}$.
Write label 0 on vertex s.
Initialize label counter $i := 1$
While vertex set V_s does not contain sink t
If there are usable arcs
Let e be a usable arc whose labeled endpoint v has the smallest possible label.
Let w be the unlabeled endpoint of arc e.
Set $backpoint(w) := v$
Write label i on vertex w.
$V_s := V_s \cup \{w\}$.
$i := i + 1$
Else
Return s-t cut $\langle V_s, V_N - V_s \rangle$.
Reconstruct the f-augmenting path Q by following backpointers, starting from sink t.
Return f-augmenting path Q.

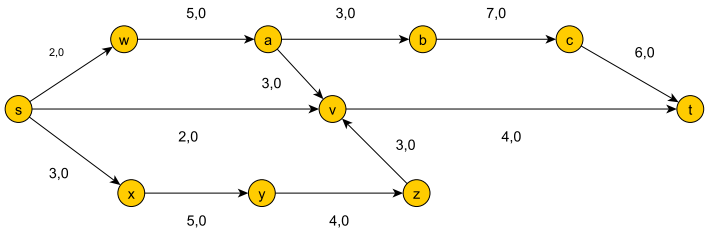
Combine the previous two algorithms, we get the following FFEK algorithm, where "FFEK" refers to Ford, Fulkerson, Edmonds and Karp.

Note: Edmonds and Karp show that the FFEK algorithm find a maximum flow in no more than $|E_N|(|V_N| + 2)/2$ augmentations. Since the computation for breadth-first search is $O(|E_N|)$, it follows that the overall computation of the FFEK algorithm is $O(|E_N|^2|V_N|)$. There are more efficient algorithms with computation of $O(|V_N||E_N|)$, which are more complicated.

Example. We use the following network to illustrate the FFEK algorithm.

FFEK - Maximum Flow

Input: an s-t network N.
Output: a maximum flow f^* in network N.
[Initialization]
For each arc e in network N
$f^*(e) := 0$
[Flow Augmentation]
Repeat
Find an f^*-augmenting path Q by "*Finding an Augmenting Path*" algorithm
Let $\Delta_Q = \min_{e \in Q}\{\Delta_e\}$.
For each arc e of augmenting path Q
If e is a forward arc
$f^*(e) := f^*(e) + \Delta_Q$
Else
$f^*(e) := f^*(e) - \Delta_Q$
Until an f^*-augmenting path cannot be found in network N.
Return flow f^*.



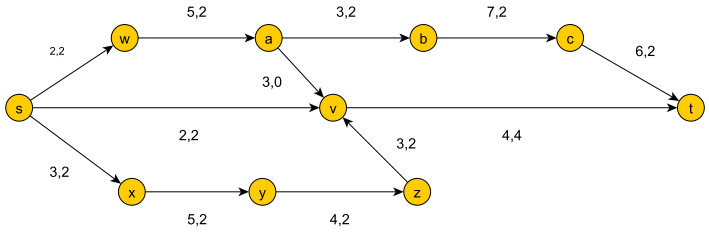
1. Find the augmenting path $Q = \{s, v, t\}$ (we omitted the edges in the path), with $\Delta_Q = 2$.

2. Find the augmenting path $Q = \{s, w, a, v, t\}$, with $\Delta_Q = 2$.

3. Find the augmenting path $Q = \{s, x, y, z, v, a, b, c, t\}$, with $\Delta_Q = 2$.

4. Final iteration find: $val(f) = 6 = cap\langle\{s, x, y, z, v\}, \{w, a, b, c, t\}\rangle$.

The result graph is as follows. The minimum cut is the set of arcs $\{sw, vt\}$.



Maximum flow problem has many applications in various fields. In this subsection, we only consider non-negative edge-weighted networks. Note that the above algorithms do not work for a network with negative weights. There are algorithms for general cases which can be used for applications requiring negative weights.

3.2 Flows and connectivity

This section introduces the Menger's Theorem about the connectivity of graphs. There are different ways to prove Menger's Theorem. the proof given here is based on network flow.

Lemma 3.2.1 *Let N be an s-t network such that $outdegree(s) > indegree(s)$, $indegree(t) > outdegree(t)$ and $outdegree(v) = indegre(v)$ for all other vertices v. Then there exists a directed s-t path in network N.*

Proof. Let W be a longest directed trail in network N that starts at s, and let z be its terminal vertex. If v is not t, then there would be an arc not in trail W that is directed from z (since $indegree(z) = outdegree(z)$). But this would contract the maximality of trial W. So v must be t. If W has a repeated vertex, then part of W determines a direct cycle which can be deleted from W. In this way we can get a directed path from s to t. □

Lemma 3.2.2 *Let N be an s-t network such that $outdegree(s) - indegree(s) = indegree(t) - outdegree(t) = m$, and $outdegree(v) = indegree(v)$ for all other vertices. Then there exist m arc-disjoint directed s-t paths in network N.*

Proof. If $m = 1$, then the conclusion follows from Lemma 3.2.1. By way of induction, assume that the assertion is true for $m = k$, and consider the case of $m = k+1$. There exists a directed s-t path P by Lemma 3.2.1. If the arcs of path P are deleted from network N, then the resulting network N' satisfies the condition of the lemma for $m = k$. By the induction hypotheses, there exist k arc-disjoint directed s-t paths in network N'. These paths together with P form the required m arc-disjointed s-t paths in network N. □

Lemma 3.2.3 *Let N be an s-t network such that $cap(e) = 1$ for every arc e. Then the value of a maximum flow in network N equals the maximum number of arc-disjoint directed s-t paths in N.*

Proof. Let f^* be a maximum flow in network N, and let r be the maximum number of arc-disjoint directed s-t paths in N. Consider the network N^*, obtained by deleting from N all arcs e such that $f^*(e) = 0$. Then $f^*(e) = 1$ for all arcs e in network N^*. So $\sum_{e \in Out(v)} f^*(e) = |Out(v)| = outdegree(v)$ and $\sum_{e \in In(v)} f^*(e) = |In(v)| = indegree(v)$. By definition of $val(f^*)$ and by the conservation-of-flow property, we have $outdegree(s) - indegree(s) = val(f^*) = indegree(t) - outdegree(t)$ and $outdegree(v) = indegree(v)$ for all vertices $v \neq s, t$. By Lemma 3.2.2, there are $val(f^*)$ arc-disjoint directed s-t paths in network N^* and in N, which implies that $val(f^*) \leq r$.

On the other hand, let $\{P_1, P_2, \ldots, P_r\}$ be a largest collection of arc-disjoint direct s-t paths in N, and consider the function $f : E_N \to R^+$ defined by

$$f(e) = \begin{cases} 1, & \text{if some path } P_i \text{ uses arc } e \\ 0, & \text{otherwise.} \end{cases}$$

Then f is a feasible flow in network N, with $val(f) = r$. It follows that $val(f^*) \geq r$. □

Let s and t be distinct vertices in a graph G. An *s-t separating edge set* in G is a set of edges whose removal destroys all s-t paths in G. Similarly, an *s-t separating arc set* in digraph D is a set of arcs whose removal destroys all directed s-t paths in D.

Lemma 3.2.4 *Let N be an s-t network such that $cap(e) = 1$ for every arc e. Then the capacity of a minimum s-t cut in network N equals the minimum number of arcs in an s-t separating arc set in N.*

Proof. Let $K^* = \langle V_s, V_t \rangle$ be a minimum s-t cut in network N, and let q be the minimum number of arcs in an s-t separating arc set in N. Since K^* is an s-t cut, it is also an s-t separating arc set. Thus $cap(K^*) \geq q$.

On the other hand, let S be an s-t separating arc set in network N containing q arcs and let R be the set of all vertices in N that are reachable from s by a directed path that contains no arc from set S. By the definition of arc set S and R, $t \notin R$. So $\langle R, V_N - R \rangle$ is an s-t cut and $\langle R, V_N - R \rangle \subseteq S$. We have

$$\begin{aligned} cap(K^*) &\leq cap\langle R, V_N - R \rangle \\ &= |\langle R, V_N - R \rangle| \\ &\leq |S| \\ &= q \end{aligned}$$

□

Theorem 3.2.5 (Arc Form of Menger's Theorem) *Let s and t be distinct vertices in a digraph D. Then the maximum number of arc-disjoint directed s-t paths in D is equal to the minimum number of arcs in an s-t separating arc set of D.*

Proof. Let N be the s-t network obtained by assigning a unit capacity to each arc of digraph D. Then the results follows from the previous lemmas and the Max-Flow Min-cut Theorem. □

Let s and t be distinct vertices of a graph G. Let $\overrightarrow{G}$ be the digraph obtained by replacing each edge of G with a pair of oppositely directed arcs having the same endpoints as the edge. Then the minimum number of edges in an s-t separating edge set of graph G is equal to the minimum number of arcs in an s-t separating arc set of digraph $\overrightarrow{G}$.

Theorem 3.2.6 (Edge Form of Menger's Theorem) *Let s and t be distinct vertices in a graph G. Then the maximum number of edge-distinct s-t paths in G equals the minimum number of edges in an s-t separating edge set of graph G.*

Proof. Let m and M be the sizes of a minimum s-t separating edge set and a maximum set of edge-disjoint s-t paths, respectively, in G. Let $\overrightarrow{m}$ and

$\overrightarrow{M}$ be the sizes of a minimum s-t separating arc set and a maximum set of arc-disjoint s-t directed paths. in $\overrightarrow{G}$.

If $\overrightarrow{F}^*$ is a maximum set of arc-disjoint s-t directed paths in $\overrightarrow{G}$, there there exists a set F of s-t paths in G with $|\overrightarrow{F}^*| \leq |F|$, which implies that $\overrightarrow{M} = |\overrightarrow{F}^*| \leq |F| \leq M$. By Theorem 3.2.5, we have $\overrightarrow{m} = \overrightarrow{M} \leq M \leq m = \overrightarrow{m}$. □

Definition 3.2.7 *The edge-connectivity $\kappa(G)$ is the size of a smallest edge-cut in graph G. The local edge-connectivity $\kappa(s,t)$ between distinct vertices s and t in a graph G is the minimum number of edges in an s-t separating edge set in G.*

The following Lemma is straightforward.

Lemma 3.2.8 *The edge-connectivity of a graph G is equal to the minimum of the local edge-connectivities, taken over all pairs of vertices s and t.:*

$$\kappa(G) = \min_{s,t \in V_G} \{\kappa(s,t)\}.$$

Let sets V_1 and V_2 be a partition of the vertices of a graph G. Then $\langle V_1, V_2 \rangle$ is called a partition-cut of G.

Lemma 3.2.9 *Let $\langle V_1, V_3 \rangle$ be a partition-cut of minimum cardinality in a graph G, and v_1 and v_2 be any vertices in V_1 and V_2 respectively. Then the edge-connectivity $\kappa(G)$ equals the local edge-connectivity $\kappa(v_1, v_2)$.*

Proof. Suppose that the minimum local edge-connectivity is achieved between vertices x and y. Then $\kappa(G) = \kappa(x,y)$. We want to prove that $\kappa(v_1, v_2) = \kappa(x,y)$.

Let $\overrightarrow{G}$ be the digraph obtained by replacing each edge of G with two oppositely directed arcs. Then $\overrightarrow{G}$ can be considered as a v_1-v_2 capacitated network $\overrightarrow{G}_{v_1,v_2}$ and as an x-y capacitated network $\overrightarrow{G}_{x,y}$, where each arc is assigned unit capacity. Let K^* be a minimum v_1-v_2 cut in network $\overrightarrow{G}_{v_1,v_2}$. Then $cap(K^*) \leq cap\langle V_1, V_2 \rangle$.

Next let f^* be a maximum flow and $\langle V_x, V_y \rangle$ a minimum x-y cut in $\overrightarrow{G}_{x,y}$, so that $cap\langle V_x, V_y \rangle = val(f^*)$. Then

$$\kappa(v_1, v_2) \quad \leq \quad cap\langle V_1, V_2 \rangle$$

$$\begin{aligned} &= |\langle V_1, V_2\rangle| \\ &\leq |\langle V_x, V_y\rangle| \\ &= cap\langle V_x, V_y\rangle \\ &= val(f^*) \\ &= \kappa(x, y) \end{aligned}$$

The conclusion follows. □

From the above lemma, the following corollary can be obtained easily.

Corollary 3.2.10 *Let s be any vertex in a graph G. Then*

$$\kappa(G) = \min_{t \in V_G - \{s\}} \{\kappa(s, t)\}$$

We then have the following algorithm to find the edge-connectivity of a graph.

Edge-Connectivity Calculation

Input: a graph G.
Output: the edge-connectivity $\kappa(G)$.

Construct digraph $\overrightarrow{G}$.
Let s be an arbitrary vertex of graph G.
Initialize edge-connectivity $\kappa := |E_G|$.
For each vertex $t \in V_G - \{s\}$
  Find out the maximum flow f^* for the s-t network $\overrightarrow{G}_{s,t}$.
  If $val(f^*) < \kappa$, then $\kappa := val(f^*)$.
Return κ.

The above algorithm requires $O(n)$ iterations of calling the maximum flow algorithm. If we use the FFEK algorithm, the the complexity will be $O(n^2|E|^2)$.

Next we consider the vertex form of Menger's Theorem for digraphs and graphs. We need the following construction.

Construction of digraph N^D from a digraph D:

Let s and t be any pair of non-adjacent vertices in a digraph D. The digraph N^D is constructed from digraph as follows:

- Each vertex $x \in V_D - \{s,t\}$ corresponds to two vertices x' and x'' in digraph N^D and an arc ditected from x' to x''.
- Each arc in digraph D that is directed from vertex s to a vertex $x \in V_D - \{s,t\}$ corresponds to an arc in digraph N^D directed from s to x'.
- Each arc in D that is directed from a vertex $x \in V_D - \{s,t\}$ to vertex t corresponds to an arc in N^D directed from x'' to t.
- Each arc in D that is directed from a vertex $x \in V_D - \{s,t\}$ to a vertex $y \in V_D - \{s,t\}$ corresponds to an arc in N^D directed from x'' to y'.

An example of N^D and and the corresponding digraph are displayed in Figure 3.5.

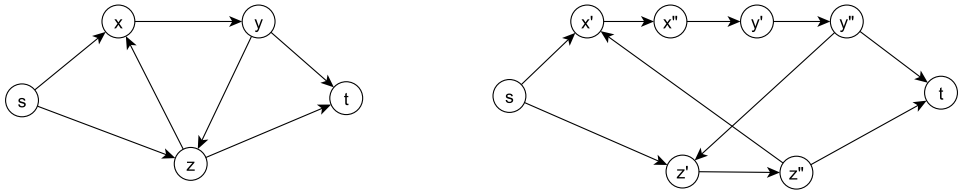


Figure 3.5: Digraph D and its N^D

Definition 3.2.11 *Let s and t be a pair of non-adjacent vertices in a graph G (or digraph D). An s-t separating vertex set in G(or D) is a set of vertices whose removal destroys all s-t paths in G (or all directed s-t paths in D).*

Two s-t paths in a digraph are internally disjoint if they have no internal vertices in common. From the definition of N^D, we have the following lemma.

Lemma 3.2.12 *Suppose a digraph N^D is constructed from a digraph D, then*

- *There is a one-to-one correspondence between directed s-t paths in D and directed s-t paths in N^D.*

- *Two directed s-t paths in D are internally disjoint if and only if their corresponding s-t directed paths in N^D are arc-disjoint.*

- *The maximum number of internally disjoint directed s-t paths in D is equal to the maximum number of arc-disjoint directed s-t paths in N^D.*

- *The minimum number of vertices in an s-t separating vertex set in digraph D is equal to the minimum number of arc-disjoint directed s-t separating arc set in digraph N^D.*

From the above lemma, we can easily obtain the following Theorems.

Theorem 3.2.13 (Vertex form of Menger for digraph) *Let s and t be a pair of non-adjacent vertices in a digraph D. Then the maximum number of internally disjoint directed s-t paths in D is equal to the minimum number of vertices in an s-t separating vertex set in D.*

Proof. The conclusion follows from above lemma and the arc from of Menger's Theorem. □

Theorem 3.2.14 (Vertex form of Menger for undirected graphs) *Let s and t be a pair of non-adjacent vertices in a graph G. Then the maximum number of internally disjoint s-t paths in G is equal to the minimum number of vertices in an s-t separating vertex set in G*

Proof. Construct a digraph $\overrightarrow{G}$ from G and apply the above Lemma and Theorem. □

Using the vertex form of Menger's Theorem, we can obtain an algorithm to determine the vertex-connectivity of a graph.

Definition 3.2.15 *Let s and t be non-adjacent vertices of a connected graph G. Then the local vertex-connectivity between s and t, denoted $\kappa_v(s,t)$, is the minimum number of vertices in an s-t separating vertex set. The vertex-connectivity κ_v of G is the minimum of the local vertex-connectivity $\kappa_v(s,t)$, taken over all pairs of non-adjacent vertices s and t.*

Vertex-connectivity calculation

Input: a graph G with $V_G = \{v_1, v_2, \ldots, v_n\}$.
Output: the vertex-connectivity $\kappa_v(G)$

Construct digraph $\overrightarrow{G}$.
Initialize vertex-connectivity $\kappa_v := |V_G|$.
Initialize index $k := 0$.
While $k \leq \kappa_v$
  $k := k + 1$
  For $j = k + 1$ to n
    If vertex v_k and v_j are not adjacent
      Construct digraph $N^{\overrightarrow{G}}$.
      Assign unit capacity to each arc in $N^{\overrightarrow{G}}$.
      Find the maximum flow f^* for v_k-v_j network $N^{\overrightarrow{G}}$.
      If $val(f^*) < \kappa_v$, then $\kappa_v := val(f^*)$
Return κ_v.

Appendix A

Finite Algebra

In this appendix, we give some basic concepts of finite algebra.

A.1 Group

Definition A.1.1 *A group is a pair $(G, \circ)$, where G is a set with $|G| > 0$ and $\circ$ is an operation on G satisfying the following group axioms.*

1. *For all $a, b \in G$, $a \circ b \in G$.*

2. *For all $a, b, c \in G$, $(a \circ b) \circ c = a \circ (b \circ c)$.*

3. *There exists an element $e \in G$, called identity element, such that $e \circ a = a \circ e = a$ for every element $a \in G$.*

4. *For each $a \in G$, there is an element $b \in G$, such that $a \circ b = b \circ a = e$. Usually we can denote b as a^{-1} and call it the inverse of a.*

In general, in a group $a \circ b$ and $b \circ a$ may not be equal. If $a \circ b = b \circ a$ for any $a, b \in G$, then the group G is called abelian. In an abelian group, we usually denote the operation as $+$ and the identity is denoted as 0 or 0_G.

In a group, the group identity is unique and the inverse of an element is unique.

$(\mathbb{Z}_n, +)$ and $(\mathbb{Z}_n, \cdot)$ are examples of finite groups, and both of them are abelian.

A.2 Ring

Definition A.2.1 *A ring is a triple $(R, +, \cdot)$, where R is a set with $|R| \geq 2$, $+$ and $\cdot$ are two operations on R satisfying the following properties.*

1. *$(R, +)$ is an abelian group with identity 0_R called additive identity.*
2. *For all $a, b \in R$, $a \cdot b \in R$.*
3. *There is a multiplicative identity $1_R \in R$ such that $1_R \cdot a = a \cdot 1_R = a$.*
4. *$(a \cdot b) \cdot c = a \cdot (b \cdot c)$*
5. *$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.*

The existence of the multiplicative identity is not required in some definition. We adapt it here to simplify the discussion.

If $a \cdot b = b \cdot a$ for all $a, b \in R$, then the ring is called a commutative ring.

$(\mathbb{Z}_n, +, \cdot)$ is an example of commutative ring. Note that in a ring, a nonzero element is not necessary to have a multiplicative inverse. For example, $(\mathbb{Z}_6, +, \cdot)$ is a commutative ring. In this ring only 1 and 5 have multiplicative inverse.

A.3 Field

Definition A.3.1 *A field is a triple $(F, +, \cdot)$, where F is a set with $|F| \geq 2$, $+$ and $\cdot$ are two operations on F satisfying the following properties.*

1. *$(F, +, \cdot)$ is a communicative ring.*
2. *For any $a \in F \backslash \{0_F\}$, there is a multiplicative inverse element in F, denoted a^{-1} such that $a \cdot a^{-1} = a^{-1} \cdot a = a$.*

All the real numbers with addition and multiplication is a (infinite) field. All the integers with addition and multiplication is not a field but a ring.

$(\mathbb{Z}_p, +, \cdot)$ is a finite field where p is a prime.

Theorem A.3.2 *There exists a finite field of order q if and only if $q = p^n$, where p is a prime and $n \geq 1$.*

The non-zero elements of $\mathbb{F}_q$ form a group under multiplication called multiplicative group of $\mathbb{F}_q$, denoted by $\mathbb{F}_q^*$.

$\mathbb{F}_q^*$ is a cyclic group of order $q-1$, meaning that there is an element $a \in \mathbb{F}_q^*$ such that $\mathbb{F}_q^* = \{a^0, a, a^2, \ldots, a^{q-1}\}$. Such an a is called a primitive element or generator of $\mathbb{F}_q$.

A.3.1 Polynomial basis representation

Suppose F is a field. Then $F[x]$ denotes all the polynomials in one variable with coefficients in F.

Definition A.3.3 *$F[x]/f(x)$ denotes the set of (equivalence classes of) polynomials in $F[x]$ of degree less than $n = \deg f(x)$. The addition and multiplication are performed modulo $f(x)$.*

Let p be a prime and let $\mathbb{Z}_p[x]$ denote the set of polynomials in one variable with coefficients in $\mathbb{Z}_p$ (Note that $\mathbb{Z}_p$ is a field). A polynomial $f(x) \in \mathbb{Z}_p[x]$ is reducible, if $f(x)$ is a constant or it can be written in form

$$f(x) = g(x)h(x)$$

with $g(x), h(x) \in \mathbb{Z}_p[x]$, and both $g(x)$ and $h(x)$ have degree at least one. We say that $f(x)$ is irreducible if $f(x)$ is not reducible.

Example. In $\mathbb{Z}_2$, $x^2 + 1$ is reducible, because $x^2 + 1 = (x + 1)(x + 1)$. But $f(x) = x^2 + x + 1$ is irreducible, because $f(x) \neq 0$ in $\mathbb{Z}_2$ which means $f(x)$ has no one factor.

Now we give two theorems about a construction of finite field without giving proofs. The proofs can be found in Algebra textbooks and out of this course's scope.

Theorem A.3.4 *If $f(x)$ is irreducible over a field F, then $F[x]/f(x)$ is a field.*

Theorem A.3.5 *If $f(x)$ is an irreducible polynomial over $\mathbb{Z}_p$ with degree n, Then $\mathbb{Z}_p[x]/f(x)$ is the field $\mathbb{F}_{p^n}$.*

Some known irreducible polynomials of $\mathbb{Z}_p[x]$, where p is a prime.

- $x^2 + 1$ for $p \equiv 3 \bmod 4$.

- $x^2 + x + 1$ for $p \equiv 2 \bmod 3$.
- $x^4 + x^3 + x^2 + x + 1$ for $p \not\equiv \pm 1 \bmod 5$ and $p \neq 5$.
- $x^3+x^2+1, x^4+x+1, x^5+x^2+1, x^6+x+1, x^7+x+1, x^8+x^6+x^5+x+1$ for $p = 2$.

Example. $x^2 + 1$ is irreducible in $\mathbb{Z}_3$. We can use it to construct a $\mathbb{F}_9$ as follows. The elements of this field are:

$$0, 1, 2, x, x + 1, x + 2, 2x, 2x + 1, 2x + 2.$$

We can use a pair of elements in $\mathbb{Z}_3$ to denote these field elements:

$$(0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2).$$

The addition table is:

$+$	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$(0,0)$	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$(0,1)$	$(0,1)$	$(0,2)$	$(0,0)$	$(1,1)$	$(1,2)$	$(1,0)$	$(2,1)$	$(2,2)$	$(2,0)$
$(0,2)$	$(0,2)$	$(0,0)$	$(0,1)$	$(1,2)$	$(1,0)$	$(1,1)$	$(2,2)$	$(2,0)$	$(2,1)$
$(1,0)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	$(0,0)$	$(0,1)$	$(0,2)$
$(1,1)$	$(1,1)$	$(1,2)$	$(1,0)$	$(2,1)$	$(2,2)$	$(2,0)$	$(0,1)$	$(0,2)$	$(0,0)$
$(1,2)$	$(1,2)$	$(1,0)$	$(1,1)$	$(2,2)$	$(2,0)$	$(2,1)$	$(0,2)$	$(0,0)$	$(0,1)$
$(2,0)$	$(2,0)$	$(2,1)$	$(2,2)$	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$
$(2,1)$	$(2,1)$	$(2,2)$	$(2,0)$	$(0,1)$	$(0,2)$	$(0,0)$	$(1,1)$	$(1,2)$	$(1,0)$
$(2,2)$	$(2,2)$	$(2,0)$	$(2,1)$	$(0,2)$	$(0,0)$	$(0,1)$	$(1,2)$	$(1,0)$	$(1,1)$

The multiplication table:

$\times$	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$	$(0,0)$
$(0,1)$	$0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$(0,2)$	$(0,0)$	$(0,2)$	$(0,1)$	$(2,0)$	$(2,2)$	$(2,1)$	$(1,0)$	$(1,2)$	$(1,1)$
$(1,0)$	$(0,0)$	$(1,0)$	$(2,0)$	$(0,2)$	$(1,2)$	$(2,2)$	$(0,1)$	$(1,1)$	$(2,1)$
$(1,1)$	$(0,0)$	$(1,1)$	$(2,2)$	$(1,2)$	$(2,0)$	$(0,1)$	$(2,1)$	$(0,2)$	$(1,0)$
$(1,2)$	$(0,0)$	$(1,2)$	$(2,1)$	$(2,2)$	$(0,1)$	$(1,0)$	$(1,1)$	$(2,0)$	$(0,2)$
$(2,0)$	$(0,0)$	$(2,0)$	$(1,0)$	$(0,1)$	$(2,1)$	$(1,1)$	$(0,2)$	$(2,2)$	$(1,2)$
$(2,1)$	$(0,0)$	$(2,1)$	$(1,2)$	$(1,1)$	$(0,2)$	$(2,0)$	$(2,2)$	$(1,0)$	$(0,1)$
$(2,2)$	$(0,0)$	$(2,2)$	$(1,1)$	$(2,1)$	$(1,0)$	$(0,2)$	$(1,2)$	$(0,1)$	$(2,0)$

A.3.2 $\mathbb{F}_{2^8}$ used in AES

Now we can construct the $\mathbb{F}_{2^8}$ (also denoted $GF(2^8)$) used in AES. The following irreducible polynomial defined on $\mathbb{Z}_2$ is used to construct the finite field:

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

Each element in $GF(2^8)$ can be expressed as a polynomial or a 8-bit binary string (vector). For example the following notations are used to denote the same element of $GF(2^8)$.

$$\begin{array}{l} x^6 + x^4 + x^2 + x + 1 \\ 01010111 \end{array}$$

Suppose there is another element $x^7 + x^4 + x$ (10010010). The addition in that field is simple.

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x^4 + x) = x^7 + x^6 + x^2 + 1.$$

Note that in binary notation, the addition is simply XOR:

$$(01010111) \oplus (10010010) = 11000101.$$

The multiplication in $GF(2^8)$ is a little complicated. But there is still an efficient way to do that. Note that

$$x^8 \equiv x^4 + x^3 + x + 1 \bmod m(x).$$

Suppose an element in $GF(2^8)$ is $P(x) = \sum_{i=0}^{7} b_i x^i$, where $b_i = 0$ or $1, i = 0, 1, \cdots, 7$. Then

$$\begin{aligned} xP(x) &= \sum_{i=0}^{7} b_i x^{i+1} \\ &\equiv \begin{cases} \sum_{i=0}^{6} b_i x^{i+1} & \text{if } b_7 = 0, \\ (\sum_{i=0}^{6} b_i x^{i+1}) + x^4 + x^3 + x + 1 & \text{if } b_7 = 1. \end{cases} \end{aligned}$$

Using the binary notation, this operation can be expressed as:

$$xP(x) = \begin{cases} b_6 b_5 b_4 b_3 b_2 b_1 b_0 0 & \text{if } b_7 = 0, \\ (b_6 b_5 b_4 b_3 b_2 b_1 b_0 0) \oplus (00011011) & \text{if } b_7 = 1. \end{cases}$$

So to compute $xP(x)$, we just need to do a shift and an XOR. Repeat i times of these operations we get $x^i P(x)$. Basically, we can use shifts and XORs to do the multiplications in $GF(2^8)$.

Appendix B

Linear Algebra

B.1 Linear equations

A linear equation in n unknowns $x_1, x_2, \ldots, x_n$ is an equation of the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b,$$

where $a_1, a_2 \cdots, a_n$ are elements in a field $\mathbb{F}$.

A system of m linear equations in n unknowns $x_1, x_2, \ldots x_n$ is a family of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m. \end{aligned}$$

Above system can be written concisely as

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, i = 1, 2, \ldots, m.$$

We wish to determine if such a system has a solution, that is to find out if there exist $x_1, x_2, \ldots, x_n \in \mathbb{F}$ which satisfy each of the equations simultaneously. We say that the system is *consistent* if it has a solution. Otherwise the system is called *inconsistent*.

The following matrix is called the coefficient matrix of the above system.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

A system of homogeneous linear equations is a system of the form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= 0 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= 0 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= 0. \end{aligned}$$

Such a system is always consistent as $x_1 = 0, \ldots, x_n = 0$ is a solution. This solution is called the *trivial solution*. Any other solution is called a *nontrivial solution*.

A homogeneous system of m linear equations in n unknowns always has a nontrivial solution if $m < n$.

B.2 Subspaces

Let $\mathbb{F}$ be a field. $\mathbb{F}^n$ will be denote the set of n-dimensional vectors with components from field $\mathbb{F}$.

Definition B.2.1 *A subset S of $\mathbb{F}^n$ is called a subspace of $\mathbb{F}^n$ if*

1. *The zero vector belongs to S; (that is, $\mathbf{0} \in S$);*
2. *If $\mathbf{u} \in S$ and $v \in S$, then $u+v \in S$; (S is said to be closed under vector addition);*
3. *If $\mathbf{u} \in S$ and $t \in \mathbb{F}$, then $t\mathbf{u} \in S$; (S is said to be closed under scalar multiplication).*

Example. Let A be an $m \times n$ matrix over $\mathbb{F}$. Then the set of vectors $\mathbf{x} \in \mathbb{F}^n$ satisfying $A\mathbf{x} = \mathbf{0}$ is a subspace of $\mathbb{F}^n$ called the null space of A (or solution space of A).

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m \in \mathbb{F}^n$. Then the set consisting of all linear combinations $x_1\mathbf{x}_1 + \cdots + x_m\mathbf{x}_m$, where $x_1, \cdots x_n \in \mathbb{F}$, is a subspace of $\mathbb{F}^n$ called the subspace spanned or generated by $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

Vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ in $\mathbb{F}^n$ are said to be linearly dependent if there exist $x_1, \ldots, x_m \in \mathbb{F}$, not all zero, such that

$$x_1\mathbf{x}_1 + \cdots + x_m\mathbf{x}_m = \mathbf{0}.$$

In other words, $\mathbf{x}_1, \ldots, \mathbf{x}_m$ in $\mathbb{F}^n$ are linearly dependent, if some $\mathbf{x}_i$ is expressible as a linear combination of the remaining vectors.

$\mathbf{x}_1, \ldots, \mathbf{x}_m$ in $\mathbb{F}^n$ are called linear independent if they are not linearly dependent.

Definition B.2.2 *Vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathbb{F}^n$ belonging to a subspace S are said to form a basis of S if*

1. *Every vector in S is a linear combination of $\mathbf{x}_1, \ldots, \mathbf{x}_m$,*
2. *$\mathbf{x}_1, \ldots, \mathbf{x}_m$ are linearly independent.*

Theorem B.2.3 *Any two bases for a subspace S must contain the same number of elements.*

The number in the above theorem is called the dimension of S and is written $\dim S$.

Bibliography

[1] C.J. Colbourn and J.H. Dinitz, The CRC handbook of combinatorial designs, CRC press, 2007.

[2] J.L. Gross and J. Yellen, Graph theory and its applications, 2nd edt. Chapman & Hall/CRC, 2006.

[3] J.H. van Lint and R.M. Wilson, A course in combinatorics, Cambridge University Press, Cambridge, 1992.

[4] D.R. Stinson, Combinatorial characterizations of authentication codes, Designs, Codes and Cryptography, 2(1992), 175-187.

[5] D.R. Stinson, Cryptography, Theory and Practice, 3rd edition, Chapman & Hall/CRC, 2006.