ENGG 2440 离散数学

①②③④⑤⑥⑦⑧⑨⑩

binary search	二分搜索
induction hypothesis	归纳假设
intuitively	直观地
index - indices	下标
subscripts	下标
closed-form formula	闭式解，解析解（可以用有限次运算的初等函数表达，与数值解相对）
consecutive	连续的
Tower of Hanoi	汉诺塔
peg	柱子
homogeneous solution	齐次解，通解
particular solution	特解
iff	if and only if 的缩写
cardinality	势
pairwise	成对的
constituent	组分
convention	习俗，惯例
constraint	约束
configuration	配置，组态
as an aside	作为旁白
legitimate	合理的，合法的
mutual	相互的
acquaintance	熟人
corollary	推论
bijection	双射
Isomorphism	同构
isomorphic	同构的
dodecahedron	十二面体
labyrinth	迷宫
Articulation Vertex	割点
ambiguity	歧义

1. Introduction

2. Mathematical Induction (MI)

> Lecture 1：数学归纳法

注意，只讨论如何使用，不考虑为什么可以用。

proposition

predicate

> 谓词
>
> A predicate is a proposition whose truth depends on one or more variables.

例：

$P(n)$: "$n\ is\ a\ perfect\ square$"

> $P(4)$ is true, but $P(5)$ is false.

$P(n)$: "$1+2+\cdots+n=\frac{n(n+1)}{2}$"

Principle of MI (Ordinary induction)

> Let $P(n)$ be a predicate. The induction priciple involves two steps:
>
> $1.(Base\ Case):show\ that\ P(base)\ is\ true.$
>
> $2.(Inductive\ Step):\ show\ that\ P(n)\ is\ true \Rightarrow P(n+1)\ is\ true\ for\ all\ n \geq base.$
>
> Then $P(n)\ is\ true\ for\ all\ n \geq base.$

注意，$Base\ Case$ 中 $base$ 的取值要根据题意. 如果 n 从 10 开始，$base$ 就取 10.

Strong Mathematical Induction

> Let $P(n)$ be a predicate. The induction priciple involves two steps:
>
> $1.(Base\ Case):show\ that\ P(1)\ is\ true.$
>
> $2.(Inductive\ Step):\ show\ that\ P(1),\ \cdots,\ P(n)\ is\ true \Rightarrow P(n+1)\ is\ true.$

Fibonacci 数

> $x^2-x-1=0$ 的两解：$x_1=\frac{1+\sqrt{5}}{2},\ x_2=\frac{1-\sqrt{5}}{2}.$
>
> 所以，要注意到这两解满足该方程，可能有助于解题。

3. Summation

> Lecture 2：求和

1. Distributive Law

Let c be a constant. Then,

$$\sum_{k\in\kappa} ca_k = c\sum_{k\in\kappa} a_k$$

2. Associative Law

$$\sum_{k\in\kappa}(a_k + b_k) = \sum_{k\in\kappa} a_k + \sum_{k\in\kappa} b_k$$

3. Close form formula

$$\sum_{k=1}^{n}(a_k - a_{k-1}) = a_n - a_0$$

4. Perturbation Method

> 摄动方法

$$S_n = \sum_{k=1}^{n} a_k$$

rewrite it in two ways:

$$a_1 + \sum_{k=2}^{n+1} a_k = S_{n+1} = \sum_{k=1}^{n} a_k + a_{n+1}$$

5. Geomatric Sum

> 等比数列求和

Consider the sum

$$S_n = \sum_{k=1}^{n} x^k$$

$$x^1 + \sum_{k=2}^{n+1} x^k = S_{n+1} = \sum_{k=1}^{n} x^k + x^{n+1}$$

$$\sum_{k=2}^{n+1} x^k = x^2 + x^3 + \cdots + x^{n+1} = x(x + x^2 + \cdots + x^n) = xS_n$$

$$x + xS_n = S_n + x^{n+1}$$

If $x \neq 1$, then

$$S_n = \frac{x(1 - x^n)}{1 - x}$$

> 注意使用条件（$x \neq 1$）.
>
> $x = 1$ 时，求和结果为 n .

6. Quadratic Series

> 二次数列

Consider the sum

$$S_n = \sum_{k=1}^{n} k^2$$

先直接对这个 Quadratic Series 的求和应用 Perturbation Method

$$1^2 + \sum_{k=2}^{n+1} k^2 = S_{n+1} = S_n + (n+1)^2$$

$$\sum_{k=2}^{n+1} k^2 = \sum_{j=1}^{n} (j+1)^2 = S_n + 2\sum_{j=1}^{n} j + n$$

We obtain the formula

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2}$$

which is not the sum we want.

> 对幂级数使用摄动，sum 会被消掉，但是会得到低一次幂的级数求和。

这种降幂性质 suggests that we should apply the perturbation method 在更高一级的幂级数求和上，也就是说， to the sum

$$C_n = \sum_{k=1}^{n} k^3$$

$$1^2 + \sum_{k=2}^{n+1} k^3 = C_{n+1} = C_n + (n+1)^3$$

$$\sum_{k=2}^{n+1} k^3 = \sum_{j=1}^{n} (j+1)^3 = C_n + 3S_n + \frac{3n(n+1)}{2} + n$$

消去 C_n 得到

$$S_n = \frac{n(n+1)(2n+1)}{6}$$

7. 公式表

$$Geometric\ Series$$

$$\sum_{k=0}^{n} ar^k = \frac{a(r^{n+1}-1)}{r-1},\ r \neq 1$$

> 注意项数。数列一共有几项， r 的次数就是几.

$$Euler's\ Trick$$

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$Quadratic\ Series$$

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$Cubic\ Series$$

$$\sum_{k=1}^{n} k^3 = \frac{n^2(n+1)^2}{4}$$

8. Guess-and-Verify Method

除了摄动，我们也可以先猜解，再用 induction 验证.

例：Quadratic Series

Since the largest term in S_n is n^2 and there are n terms in S_n, we have $S_n \leq n^3$.

Suppose $S_n = a + bn + cn^2 + dn^3$

代入 $S_1,\ S_2,\ S_3,\ S_4$ 得 $a = 0,\ b = \frac{1}{6},\ c = \frac{1}{2},\ d = \frac{1}{3}$.

$S_n = \frac{1}{6}n + \frac{1}{2}n^2 + \frac{1}{3}n^3$

> This is still a guess, because we obtain a, b, c, d by considering S_1, S_2, S_3, S_4, and it is not clear whether these values will work for $n \geq 5$.

Verify by induction

The base case $n = 1$ obviously holds. We then compute

$S_{n+1} = S_n + (n+1)^2 = \frac{1}{6}n + \frac{1}{2}n^2 + \frac{1}{3}n^3 + (n+1)^2 = \frac{1}{6}(n+1) + \frac{1}{2}(n+1)^2 + \frac{1}{3}(n+1)^3$

9. Multiple Summation

> 多重求和

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

$$r_j = a_{j1} + a_{j2} + \cdots + a_{jn} = \sum_{k=1}^{n} a_{jk}$$

$$S = \sum_{j=1}^{m} r_j = \sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}$$

练习1.

$$\begin{aligned} T &= \sum_{j=1}^{n} \sum_{k=1}^{j-1} (j-k) \\ &= \sum_{j=1}^{n} (\sum_{k=1}^{j-1} j - \sum_{k=1}^{j-1} k) \\ &= \sum_{j=1}^{n} (j(j-1) - \frac{j(j-1)}{2}) \\ &= \frac{1}{2} (\sum_{j=1}^{n} j^2 - \sum_{j=1}^{n} j) \\ &= \frac{1}{2} [\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}] \\ &= \frac{n(n+1)(n-1)}{6} \end{aligned}$$

Interchanging Order of Summation

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

$$r_j = a_{j1} + a_{j2} + \cdots + a_{jn} = \sum_{k=1}^{n} a_{jk} \quad S = \sum_{j=1}^{m} r_j = \sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}$$

$$c_k = a_{1k} + a_{2k} + \cdots + a_{mk} = \sum_{j=1}^{m} a_{jk} \, S = \sum_{k=1}^{n} c_k = \sum_{k=1}^{n} \sum_{j=1}^{m} a_{jk}$$

$$\sum_{j=1}^{m}\sum_{k=1}^{n}a_{jk}=\sum_{k=1}^{n}\sum_{j=1}^{m}a_{jk}$$

$$\begin{bmatrix}\mathbf{a_{11}} & \mathbf{a_{12}} & \cdots & \mathbf{a_{1n}}\\ a_{21} & \mathbf{a_{22}} & \cdots & \mathbf{a_{2n}}\\ \vdots & \vdots & \ddots & \vdots\\ a_{n1} & a_{n2} & \cdots & \mathbf{a_{nn}}\end{bmatrix}$$

$$\begin{aligned}S&=\sum_{j=1}^{n}\sum_{k=j}^{n}a_{jk}\\&=\sum_{k=1}^{n}\sum_{j=1}^{k}a_{jk}\end{aligned}$$

例：$S=\sum\limits_{j=1}^{n}\sum\limits_{k=1}^{j}\frac{1}{k}$

> n-th harmonic number. n 阶调和数 $H_n=\sum\limits_{k=1}^{n}\frac{1}{k}$
>
> This sum has no known closed-form formula.

$$\begin{bmatrix}1 & & & & \\ 1 & \frac{1}{2} & & & \\ 1 & \frac{1}{2} & \frac{1}{3} & & \\ & & & \ddots & \\ 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n}\end{bmatrix}$$

$$\begin{aligned}S&=\sum_{j=1}^{n}\sum_{k=1}^{j}\frac{1}{k}\\&=\sum_{k=1}^{n}\sum_{j=k}^{n}\frac{1}{k}\\&=\sum_{k=1}^{n}\frac{n-k+1}{k}\\&=\sum_{k=1}^{n}\frac{n}{k}-\sum_{k=1}^{n}1+\sum_{k=1}^{n}\frac{1}{k}\\&=(n+1)H_n-n\end{aligned}$$

floor function

> 下取整函数，高斯函数
>
> The symbol $\lfloor x\rfloor$ is usually read as "the floor of x".

Definition: Given a real number x, define

$\lfloor x\rfloor$= greatest integer $\leq x$.

性质：

If x is an integer, then $x=\lfloor x\rfloor$.

For any real number x, $\lfloor x\rfloor\leq x<\lfloor x\rfloor+1$.

例：$S_n=\sum\limits_{k=1}^{n}\lfloor\sqrt{k}\rfloor$

where, for simplicity, we assume that n is a perfect square; i.e., $n=a^2$ for some integer $a\geq 1$.

$$
\begin{aligned}
S_n &= \sum_{k=1}^{n} \lfloor \sqrt{k} \rfloor \\
&= \sum_{k=1}^{n} \sum_{j=1}^{\lfloor \sqrt{k} \rfloor} 1 \\
&= \sum_{k=1}^{a^2} \sum_{j=1}^{\lfloor \sqrt{k} \rfloor} 1 \\
&= \sum_{j=1}^{a} \sum_{k=j^2}^{a^2} 1 \\
&= \sum_{j=1}^{a} (a^2 - j^2 + 1) \\
&= \sum_{j=1}^{a} a^2 - \sum_{j=1}^{a} j^2 + \sum_{j=1}^{a} 1 \\
&= a^3 - \frac{a(a+1)(2a+1)}{6} + a
\end{aligned}
$$

So far our discussion focuses on the case where n is a perfect square. Now, let us consider the case where n is not a perfect square. In this case, there exists an integer $a \geq 1$ such that $a^2 < n < (a+1)^2$. We break the sum as follows:

$$
S_n = \sum_{k=1}^{n} \lfloor \sqrt{k} \rfloor = \sum_{k=1}^{a^2} \lfloor \sqrt{k} \rfloor + \sum_{k=a^2+1}^{n} \lfloor \sqrt{k} \rfloor = a^3 - \frac{a(a+1)(2a+1)}{6} + a + a(n - a^2)
$$

法二：

$$
\begin{aligned}
S_n &= \sum_{j=1}^{a} \sum_{k=j^2}^{n} 1 \\
&= \sum_{j=1}^{a} (n - j^2 + 1) \\
&= a(n+1) - \frac{a(a+1)(2a+1)}{6}
\end{aligned}
$$

综上，$\sum_{k=1}^{n} \lfloor \sqrt{k} \rfloor = \lfloor \sqrt{n} \rfloor (n+1) - \frac{\lfloor \sqrt{n} \rfloor (\lfloor \sqrt{n} \rfloor + 1)(2\lfloor \sqrt{n} \rfloor + 1)}{6}$

> 对任意正整数 n 成立.

$k \setminus j$	1	2	3	$\cdots$	$a-1$	a
1	1					
2	1					
3	1					
4	1	1				
5	1	1				
$\vdots$	$\vdots$	$\vdots$				
8	1	1				
9	1	1	1			
$\vdots$	$\vdots$	$\vdots$	$\vdots$			
$(a-1)^2$	1	1	1	$\cdots$	1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
a^2-1	1	1	1	$\cdots$	1	
a^2	1	1	1	$\cdots$	1	1

例：Let $n \geq 1$ be an integer and $x \neq 1$ be a real number. Find a closed-form expression for the following double sum:

$$
S = \sum_{j=1}^{n} \sum_{k=1}^{j} k x^j
$$

Solution:

Let $S(x) = \sum_{k=1}^{n} x^k = \frac{x(1-x^n)}{1-x}$.

$S'(x) = \sum_{k=1}^{n} kx^{k-1}$

$\sum_{k=1}^{n} kx^k = xS'(x) = \frac{x \cdot (nx^{n+1} - (n+1)x^n + 1)}{(x-1)^2}$

Interchanging the summation order yields

$S = \sum_{k=1}^{n} \sum_{j=k}^{n} kx^j.$

Notice that for any integer k satisfying $1 \leq k \leq n$ it holds that

$\sum_{j=k}^{n} x^j = \sum_{j=1}^{n} x^j - \sum_{j=1}^{k-1} x^j = \frac{x-x^{n+1}}{1-x} - \frac{x-x^k}{1-x} = \frac{x^k-x^{n+1}}{1-x}$

Equivalently,

$\sum_{j=k}^{n} x^j = \frac{x^k(1-x^{n-k+1})}{1-x} = \frac{x^k-x^{n+1}}{1-x}$

$$
\begin{aligned}
S &= \sum_{k=1}^{n} \sum_{j=k}^{n} kx^j \\
&= \sum_{k=1}^{n} k \sum_{j=k}^{n} x^j \\
&= \sum_{k=1}^{n} k \cdot \frac{x^k - x^{n+1}}{1-x} \\
&= \frac{1}{1-x} \sum_{k=1}^{n} kx^k - \frac{x^{n+1}}{1-x} \sum_{k=1}^{n} k \\
&= \frac{1}{1-x} \cdot \frac{x \cdot (nx^{n+1} - (n+1)x^n + 1)}{(x-1)^2} - \frac{x^{n+1}}{1-x} \cdot \frac{n(n+1)}{2}
\end{aligned}
$$

4. Recurrences

> 递归，Recursion

In mathematics, a recurrence relation is an equation that recursively defines a sequence, once one or more initial terms are given: each further term of the sequence is defined as a function of the preceding terms. For example, the Fibonacci recurrence is: $F(n+1) = F(n) + F(n-1),\ F(1) = F(0) = 1$

In computer science, recursion is a programming technique where a function calls itself repeatedly until some base condition is met. For example, recursion can be used to calculate a Fibonacci sequence number (inefficiently) like this:

```
int fib(int n)
{
 if (n <= 1) return 1;
 return fib(n-1) + fib(n-2);
}
```

4.1 Linear Homogeneous Recurrences

> 线性齐次递归

$$T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d)$$

> d is the *degree* of the recurrence and $a_1, \cdots, a_d$ are given constants.
>
> degree：阶数，递归关系中，当前项 $T(n)$ 与最远回溯的前项的索引差.

Example: Fibonacci 数列

$$T(n) = T(n-1) + T(n-2)$$

> $d = 2$ and $a_1 = a_2 = 1$

4.1.1 Solving Recurrences

线性齐次递归的通项解法：解特征方程.

> 接下来几节 focus on 特征方程各种情况的解（有无重根，有无复数根）对应的递归通项.

$$T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d)$$

考虑所有系数和初始值为正的情况，假设 n is divisible by d，有

$$c^{\frac{n}{d}}T(0) \le \cdots \le c^2T(n-2d) \le cT(n-d) \le T(n) \le cT(n-1) \le c^2T(n-2) \le \cdots \le c^nT(0)$$

其中 $c = a_1 + a_2 + \cdots + a_d$.

在两个指数函数中间，猜解 $T(n) = x^n$，x 为待定值.

$$x^n = a_1x^{n-1} + \cdots + a_dx^{n-d}$$

$$x^d = a_1x^{d-1} + \cdots + a_d$$

> 注意这里同除以 x^{n-d}，因为我们默认 $x \neq 0$（$T(n)$ 为常数 0 显然满足递归，而且很容易根据题意判断它是不是特解，不是研究重点. 更不用说这个特解也可以从 $r \neq 0$ 的根导出的通解求出，例如前两项均为 0 的 Fibonacci 数列，求出系数 $\theta_1 = \theta_2 = 0 \Rightarrow T(n) = 0$）

整理后得到关于 x 的 $polynomial\ equation$，也叫做递归关系的**特征方程**（characteristic equation）.

$$\boxed{x^d - a_1x^{d-1} - \cdots - a_d = 0}$$

得到 d 个特征根（可以是实数或复数，且可能有重根）

对于线性齐次递归，若 d 个根是不同的实根，最终解的形式是这 d 个根的线性组合，系数由初始值决定（有 d 个初始值，写出 d 个 d 元 1 次方程，恰好解出 d 个系数）.

> 注意：重根和复数根需要另外考虑!

Example: Fibonacci 数列

$$T(n) = T(n-1) + T(n-2)$$

> $d = 2$ and $a_1 = a_2 = 1$

特征方程：$x^2 = x + 1$

解得 $r_1 = \frac{1+\sqrt{5}}{2},\ r_2 = \frac{1-\sqrt{5}}{2}$

4.1.2 无重根

$T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d)$

$\boxed{x^d - a_1x^{d-1} - \cdots - a_d = 0}$

Theorem 1

Suppose that the roots $r_1, \cdots, r_d$ of the characteristic equation are all distinct.

$T_1(n) = r_1^n,\ T_2(n) = r_2^n,\ \cdots,\ T_d(n) = r_d^n$

all solve the recurrence. Moreover, any linear combination of $T_1(n), \cdots, T_d(n)$ also solves the recurrence. In other words, for any real numbers $\theta_1, \cdots, \theta_d$,

$T_0(n) = \theta_1 r_1^n + \theta_2 r_2^n + \cdots + \theta_d r_d^n$ solves the recurrence.

> 注意，任意系数的情况下仅满足递归关系，不代表满足了初值（初始条件/边界条件）.
>
> 这是通解，如果题目给定初值，需要代入初值来求出系数 $\theta_1, \cdots, \theta_d$（特解）.

Example

> Fibonacci 数列

$T(n) = T(n-1) + T(n-2)$

> $d = 2$ and $a_1 = a_2 = 1$

特征方程：$x^2 = x + 1$

解得 $r_1 = \frac{1+\sqrt{5}}{2},\ r_2 = \frac{1-\sqrt{5}}{2}$

通解 $T_0(n) = \theta_1\left(\frac{1+\sqrt{5}}{2}\right)^n + \theta_2\left(\frac{1-\sqrt{5}}{2}\right)^n$

代入初始条件 $T(1) = 1$，$T(2) = 2$，

$$\begin{cases} 1 = T_0(1) = \theta_1\left(\frac{1+\sqrt{5}}{2}\right) + \theta_2\left(\frac{1-\sqrt{5}}{2}\right) \\ 2 = T_0(2) = \theta_1\left(\frac{1+\sqrt{5}}{2}\right)^2 + \theta_2\left(\frac{1-\sqrt{5}}{2}\right)^2 \end{cases}$$

解得 $\theta_1 = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right),\quad \theta_2 = -\frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)$

特解 $T(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)\left(\frac{1-\sqrt{5}}{2}\right)^n$

> 这个解可能和平时看到的 Fibonacci 特解不太一样，因为这里使用 $1, 2, 3, \cdots$ 开局，如果是 $1, 1, 2, \cdots$ 开局，解相当于 $n = n - 1$，$F(n) = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$（$Binet$公式），形式更简洁.
>
> 如果把 $T(0)$ 也算进去，这里采用的是 $1, 1, 2, 3, \cdots$ 开局，美观特解采用的是 $0, 1, 1, 2, \cdots$ 开局.
>
> 可见特解取决于初始条件，不能盲目背答案，容易被坑.

4.1.3 有重根

Example: 考虑以下递归

$$\begin{cases} T(n) = 2T(n-1) - T(n-2) \quad \text{for } n \geq 2, \\ T(0) = 0,\ T(1) = 1. \end{cases}$$

特征方程：$x^2 - 2x + 1 = 0$

特征根：$r_1 = r_2 = 1$（重根）

显然 $T_0(n) = \theta_1 1^n + \theta_2 1^n$ 不是通解（虽然满足递归关系，但是无法保证求出特解，最多算通解的一个子集.真正的通解应该是所有特解的集合，可以通过待定系数法求出任意特解.）

解释一下为什么可能求不出特解：

$$\begin{cases} r_1^n \theta_1 + r_2^n \theta_2 = C_1 \\ r_1^{n+1} \theta_1 + r_2^{n+1} \theta_2 = C_2 \end{cases}$$

$r_1 = r_2$ 时，要么无解，要么无穷个解.换句话说，$r_1 = r_2$ 导致 r_1^n 和 r_2^n 线性相关（无论 n 怎么变），代入任何初值去解特征方程都是在联立两条平行线. 为了解决这个问题，我们需要找到 d 个线性无关项（当然，d 个根当中可能有部分 distinct，直接 n 次方就实现线性无关，重点在重根的处理）.

Theorem 2

假设根 r 的重数 $m \geq 1$，考虑 $\tilde{T}_1(n) = r^n, \quad \tilde{T}_2(n) = nr^n, \quad \tilde{T}_3(n) = n^2 r^n, \quad \ldots, \quad \tilde{T}_m(n) = n^{m-1} r^n.$

可证明这 m 项是线性无关的，而且都满足线性递归关系 $T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_d T(n-d)$.

> 对于重根我们可以用这种方法来找线性无关项，对于不同的根，直接 n 次方即可，已经线性无关了（事实上，如果 r 不是 $\geq k+1$ 重根，$n^k r^n$ 并不满足递归关系.）

线性无关的证明很显然，$\sum_{i=1}^{m} c_i \tilde{T}_i(n) = 0$ 当且仅当 $c_1 = c_2 = \cdots = c_m = 0$.

> 不考虑 $r = 0$ 的情况. 在 $Solving\ Recurrences$ 一节解释过，x^n 是猜的特解（或者叫特解的集合，因为代入递归关系可能可以解出多个特解，但它们既不是通解，也不一定是符合题意的特解），如果 $x = 0$，相当于直接猜特解 $T(n) = 0$，这个特解是否符合题意是很好验证的. 我们本章的重点是研究特解 $T(n) \neq 0$ 的情况.
>
> 这也提醒我们如果符合题意的特解很好猜，比如等于一个常数，那么你用普适方法绕半天计算，最后得到的结果可能可以用眼睛看出来.

满足递归关系的证明如下：

证明：若 r 是 m 重根，$f(x) = 0$ 中的 $f(x)$ 可提出因子 $(x-r)^m$. 即 $f(x) = (x-r)^m g(x)$. 这意味着 r 也是 $f^{(k)}(x) = 0$ 的根，$k \in [1, m-1]$. 回到特征方程 $\boxed{x^d - a_1 x^{d-1} - \cdots - a_d = 0}$，即 $x^n - a_1 x^{n-1} - \cdots - a_d x^{n-d} = 0$，$f(x) = x^n - a_1 x^{n-1} - \cdots - a_d x^{n-d}$

若 r 是该方程的 m 重根，则

$nr^{n-1} - a_1(n-1)r^{n-2} - \cdots - a_d(n-d)r^{n-d-1} = 0$（一阶导）

注意到如果 r 满足 $f(x) = 0,\ f'(x) = 0,\ \cdots,\ f^{(k)}(x) = 0$，那么同样满足 $[xg(x)]' = 0$，其中 $g(x) = L_1(x)f(x) + L_2(x)f'(x) + \cdots + L_{k-1}(x)f^{k-1}(x)$，其中 $L_1(x), \cdots, L_{k-1}(x)$ 是 x 的任意次方多项式.

因为 r 满足 $nx^{n-1} - a_1(n-1)x^{n-2} - \cdots - a_d(n-d)x^{n-d-1} = 0$，两边同时乘 x 求导的方程仍然满足，即 $n^2 x^{n-1} - a_1(n-1)^2 x^{n-2} - \cdots - a_d(n-d)^2 x^{n-d-1} = 0$

重复该操作，直到出现 $f^{(m-1)}(x)$，即

$$n^{m-1} x^{n-1} - a_1(n-1)^{m-1} x^{n-2} - \cdots - a_d(n-d)^{m-1} x^{n-d-1} = 0$$

然后乘 x，不用求导（再求导 r 就不满足了）

$$n^{m-1} x^n - a_1(n-1)^{m-1} x^{n-1} - \cdots - a_d(n-d)^{m-1} x^{n-d} = 0$$

r 满足这个方程意味着 $\tilde{T}_m(n) = n^{m-1} r^n$ 满足递归关系. 对于

$\tilde{T}_k(n) = n^{k-1} r^n,\ k \in [1, m]$

满足递归关系的证明，都可以用同样的方法，只要把乘 x 再求导的操作控制在恰好出现 $f^{(k-1)}(x)$.

Example：r 的重数是 4，证明 n^3r^n 也满足递归关系 $T(n)=a_1T(n-1)+a_2T(n-2)+\cdots+a_dT(n-d)$.

特征方程 $f(x)=x^n-a_1x^{n-1}-\cdots-a_dx^{n-d}=0$

$f(x)=(x-r)^4g(x),\ f(r)=f'(r)=f''(r)=f'''(r)=0$

$f'(x)=nx^{n-1}-a_1(n-1)x^{n-2}-\cdots-a_d(n-d)x^{n-d-1}$

乘 x 再求导，

$f'(x)+xf''(x)=n^2x^{n-1}-a_1(n-1)^2x^{n-2}-\cdots-a_d(n-d)^2x^{n-d-1}$

> r 仍然是这个等于 0 的解.

乘 x 再求导，

$f'(x)+3xf''(x)+x^2f'''(x)=n^3x^{n-1}-a_1(n-1)^3x^{n-2}-\cdots-a_d(n-d)^3x^{n-d-1}$

> r 仍然是这个等于 0 的解，注意出现了 $f'''(x)$，不能继续求导.

乘 x，

$x[f'(x)+3xf''(x)+x^2f'''(x)]=n^3x^n-a_1(n-1)^3x^{n-1}-\cdots-a_d(n-d)^3x^{n-d}$

> r 仍然是这个等于 0 的解.

$n^3r^n-a_1(n-1)^3r^{n-1}-\cdots-a_d(n-d)^3r^{n-d}=0$

说明 n^3r^n 也满足递归关系 $T(n)=a_1T(n-1)+a_2T(n-2)+\cdots+a_dT(n-d)$.

4.1.4 统一的通解

> 考虑重根和非重根同时存在的情况下，通解的写法.

对于线性递归 $T(n)=a_1T(n-1)+a_2T(n-2)+\cdots+a_dT(n-d)$，

Suppose 特征方程 $\boxed{x^d-a_1x^{d-1}-\cdots-a_d=0}$ has t distinct roots $r_1,\cdots r_t$, where the root r_k has multiplicity $m_k\geq 1$ for $k=1,\cdots,t$.

> d 阶多项式方程有 d 个根（包括重根）$\Rightarrow m_1+m_2+\cdots+m_t=d$.

Any linear combination of 下列各项 is a solution to the recurrence：

$$
\begin{aligned}
&T_{1,1}(n)=r_1^n,\quad T_{1,2}(n)=nr_1^n,\quad \cdots,\quad T_{1,m_1}(n)=n^{m_1-1}r_1^n,\\
&T_{2,1}(n)=r_2^n,\quad T_{2,2}(n)=nr_2^n,\quad \cdots,\quad T_{2,m_2}(n)=n^{m_2-1}r_2^n,\\
&\qquad\qquad\qquad\vdots\\
&T_{t,1}(n)=r_t^n,\quad T_{t,2}(n)=nr_t^n,\quad \cdots,\quad T_{t,m_t}(n)=n^{m_t-1}r_t^n.
\end{aligned}
$$

> 一共 d 项，构成一个基本解系.

for any real numbers $\theta_{1,1},\ \cdots,\ \theta_{t,m_{t'}}$

通解 $T_0(n)=\sum\limits_{j=1}^{t}\sum\limits_{k=1}^{m_j}\theta_{j,k}T_{j,k}(n)$

> 根据 d 个初始值用待定系数法求出特解.

Example: 考虑以下递归

$$
\begin{cases}T(n)=2T(n-1)-T(n-2) & \text{for } n\geq 2,\\ T(0)=0,\ T(1)=1.\end{cases}
$$

特征方程：$x^2-2x+1=0$

特征根：$r_1=r_2=1$（重根）

显然 $T_0(n)=\theta_1 1^n+\theta_2 1^n$ 不是通解（虽然满足递归关系，但是无法保证求出特解，最多算通解的一个子集.真正的通解应该是所有特解的集合，可以通过待定系数法求出任意特解.）

通过刚才的分析，$r_1 = 1,\ m_1 = 2.$

$T_{1,1}(n) = r_1^n = 1, \quad T_{1,2}(n) = nr_1^n = n$

通解 $T_0(n) = \theta_{1,1}T_{1,1}(n) + \theta_{1,2}T_{1,2}(n) = \theta_{1,1} + \theta_{1,2}n$

代入 initial conditions，

$\begin{cases} 0 = T_0(0) = \theta_{1,1} \\ 1 = T_0(1) = \theta_{1,1} + \theta_{1,2} \end{cases}$

解得 $T(n) = n.$

4.2 Inhomogeneous Recurrences

> 非齐次

$T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d) + F(n)$

Example

> 汉诺塔：三根柱，把 n disks 从一根柱子移到另一根柱子，大的不能放小的上面.
>
> 先把 $n-1$ 块移到另一根，把最大的移到目的地，再把 $n-1$ 块移到目的地.

$T(n) = 2T(n-1) + 1 \quad for\ n \geq 1$

> Here $d = 1,\ a_1 = 2,\ F(n) = 1$

4.2.1 Solving Linear Inhomogeneous Recurrences

线性非齐次递归 $T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d) + F(n)$ 的通解等于

对应的线性齐次递归 $T(n) = a_1T(n-1) + a_2T(n-2) + \cdots + a_dT(n-d)$ 的通解加上该非齐次递归的一个特解.

Theorem 1

$T_0(n) = T_h(n) + T_p(n)$

证明：

$T_h(n) = a_1T_h(n-1) + a_2T_h(n-2) + \cdots + a_dT_h(n-d)$

$T_p(n) = a_1T_p(n-1) + a_2T_p(n-2) + \cdots + a_dT_p(n-d) + F(n)$

$(T_h(n) + T_p(n)) = a_1(T_h(n-1) + T_p(n-1)) + a_2(T_h(n-2) + T_p(n-2)) + \cdots + a_d(T_h(n-d) + T_p(n-d)) + F(n)$

齐次递归的通解如前几节所示. 非齐次的特解求法如下：

In many cases, the particular solution will be of the same function class as the inhomogeneous term $F(n)$.

> 特解的形式一般和非齐次项 $F(n)$ 类似. 例如，$F(n)$ 是 n 的 t 阶多项式，则 $T_p(n)$ will also be a polynomial in n whose degree is at least t.

Example

> 汉诺塔：三根柱，把 n disks 从一根柱子移到另一根柱子，大的不能放小的上面.
>
> 先把 $n-1$ 块移到另一根，把最大的移到目的地，再把 $n-1$ 块移到目的地.

$T(n) = 2T(n-1) + 1 \quad for\ n \geq 1$

> Here $d = 1,\ a_1 = 2,\ F(n) = 1$

$F(n) = 1$ is a polynomial of degree 0. Try $T_p(n) = x$, x 是待定常数.

$x = 2x + 1 \Rightarrow x = -1.$

$T_p(n) = -1$ is a particular solution to $T(n) = 2T(n-1) + 1.$

易得通解 $T_h(n) = \theta 2^n$.

$T_0(n) = \theta 2^n - 1$.

代入 $T(1) = 1 \Rightarrow T(n) = 2^n - 1$.

求解非齐次递归的流程总结

1.解对应的齐次递归，得到通解 $T_h(n)$

2.找到该非齐次递归的一个特解 $T_p(n)$（不一定要符合初值，只要满足递归关系）

3.$T_0(n) = T_h(n) + T_p(n)$，根据初始条件用待定系数法求出系数，即得到符合题意的解.

特解求法-进阶

找特解的本质是找出一个齐次解组合不出来的解，也就是找一个和齐次递归基本解系的所有元素线性独立的解，确保特解在齐次解的基础上引入新的线性独立项（否则相当于没找）.

Theorem 2

$T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_d T(n-d) + F(n)$

Suppose that the inhomogeneous term $F(n)$ takes the form

$F(n) = (b_0 + b_1 n + b_2 n^2 + \cdots + b_t n^t) s^n$

for some real numbers $b_0, b_1, \ldots, b_t$ and s. If s is *not* a root of the characteristic equation associated with the linear homogeneous recurrence $T(n) = a_1 T(n-1) + a_2 T(n-2) + \cdots + a_d T(n-d)$, then there is a particular solution

$T_p(n) = (x_0 + x_1 n + x_2 n^2 + \cdots + x_t n^t) s^n$,

for some real numbers $x_0, x_1, \ldots, x_t$.

On the other hand, if s is a root of multiplicity $m \geq 1$, then there is a particular solution

$T_p(n) = n^m (x_0 + x_1 n + x_2 n^2 + \cdots + x_t n^t) s^n$,

for some real numbers $x_0, x_1, \ldots, x_t$.

> 所以应该先找通解再找特解，因为找特解的时候可能要 check s 是否等于找通解时顺带求出来的特征根.
>
> n^m 是为了保证引入的特解和齐次解线性独立而不重叠（齐次解最大到 $n^{m-1} r^n$，而特解乘以这个因子后最小都有 $n^m s^n$）.

5. Asymptotics

> 渐进分析

5.1 Orders of Growth

> 增长阶

目的：估计数值函数的增长速率，比较不同算法的可扩展性.

5.2 渐进符号

5.2.1 Big-O

表示函数增长的上界.

$$f(x) = O(g(x)) \quad if \quad \exists\, c > 0,\ x_0 \geq 0 \quad such\ that \quad \forall x \geq x_0, \quad f(x) \leq cg(x)$$

> 虽然一个函数可以有多个 Big-O，但通常选择**最紧的上界**，以提供更精确的增长描述.
>
> 注意：c 大于 0.

5.2.2 Big-Ω

表示函数增长的下界.

$$f(x) = \Omega(g(x)) \quad if \quad \exists\, c > 0,\ x_0 \geq 0 \quad such\ that \quad \forall x \geq x_0, \quad f(x) \geq cg(x)$$

Theorem 1

$f(x) = O(g(x))$ is the same as $g(x) = \Omega(f(x))$.

$$\begin{aligned} f(x) = O(g(x)) &\Rightarrow \exists\, c > 0,\ x_0 \geq 0 \quad such\ that \quad \forall x \geq x_0, \quad f(x) \leq cg(x) \\ &\Leftrightarrow \exists\, c > 0,\ x_0 \geq 0 \quad such\ that \quad \forall x \geq x_0, \quad g(x) \geq \frac{1}{c} f(x) \end{aligned}$$

By taking $c' = \frac{1}{c} > 0,\ x_0' = x_0 \geq 0$, we have $g(x) = \Omega(f(x))$.

5.2.3 Θ

同时满足 O 和 Ω 的条件，表示函数增长的确切阶.

$$f(x) = \Theta(g(x)) \quad if \quad f(x) = O(g(x)) \quad and \quad g(x) = O(f(x)).$$

> $f(x) = \Theta(g(x)) \nRightarrow f(x) = g(x)$. It just means that
> $\exists\, c_1, c_2 > 0,\ x_0 \geq 0 \quad such\ that \quad \forall x \geq x_0, \quad c_1 g(x) \leq f(x) \leq c_2 g(x)$
>
> 只能说明 $f(x)$ 和 $g(x)$ 同阶.

5.2.4 Small-o

表示严格上界（严格更高阶），比 Big-O 更严格的界定.

$$f(x) = o(g(x)) \quad if \quad \lim_{x \to \infty} \frac{f(x)}{g(x)} = 0.$$

等价于

$$f(x) = o(g(x)) \quad if \quad \forall\, c > 0,\ \exists\, x_0 > 0\ such\ that\ 0 \leq f(x) < cg(x)\ for\ all\ x \geq x_0.$$

> 注意同阶不足以满足任意 c，$g(x)$ 要比 $f(x)$ 高阶才满足.

$f(x) = o(g(x)) \Rightarrow f(x) = O(g(x))$
$f(x) = O(g(x)) \nRightarrow f(x) = o(g(x))$

5.2.5 Small-ω

表示严格下界（严格更低阶），比 Big-Ω 更严格的界定.

$f(x) = \omega(g(x)) \quad if \quad \lim_{x \to \infty} \frac{g(x)}{f(x)} = 0.$

等价于

$f(x) = \omega(g(x)) \quad if \quad \forall\, c > 0,\, \exists\, x_0 > 0\ such\ that\ 0 \leq cg(x) < f(x)\ for\ all\ x \geq x_0.$

> 注意同阶不足以满足任意 c，$g(x)$ 要比 $f(x)$ 低阶才满足.

$f(x) = \omega(g(x)) \Rightarrow f(x) = \Omega(g(x))$
$f(x) = \Omega(g(x)) \nRightarrow f(x) = \omega(g(x))$

5.3 Properties for Asymptotic Analysis

> 渐进分析的性质

- 传递性（Transitivity）

> If $f(n) = \Pi(g(n))$ and $g(n) = \Pi(h(n))$, then $f(n) = \Pi(h(n))$, where $\Pi = O,\ o,\ \Omega,\ \omega,\ or\ \Theta$.

- 加法法则（Rule of sums）

> $f(n) + g(n) = \Pi(max\{f(n), g(n)\})$, where $\Pi = O, \Omega$ or Θ.

- 乘法法则（Rule of products）

> If $f_1(n) = \Pi(g_1(n))$ and $f_2(n) = \Pi(g_2(n))$, then $f_1(n)f_2(n) = \Pi(g_1(n)\ g_2(n))$, where $\Pi = O,\ o,\ \Omega,\ \omega$, or Θ.

- 转置对称性（Transpose symmetry）

> $f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$
>
> $f(n) = o(g(n))$ iff $g(n) = \omega(f(n))$

- 自反性（Reflexivity）

> $f(n) = \Pi(f(n))$, where $\Pi = O,\ \Omega$, or Θ.

- 对称性（Symmetry）

> $f(n) = \Theta(g(n))$ iff $g(n) = \Theta(f(n))$

5.3.1 多项式时间复杂度

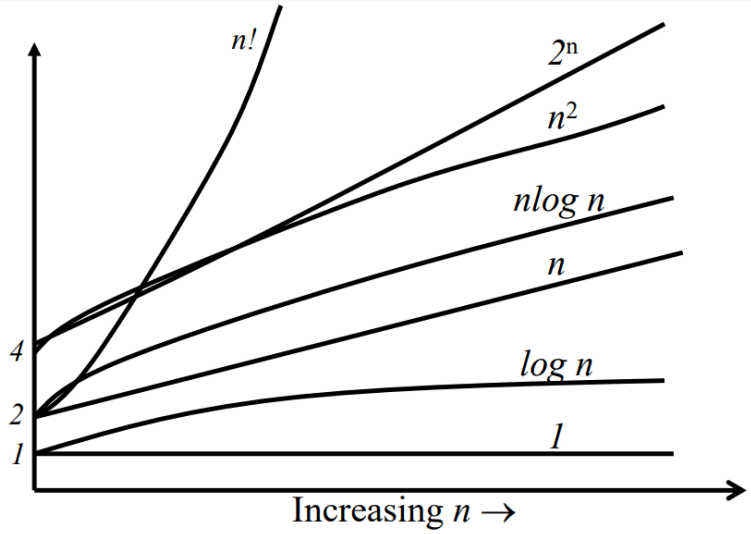
> Polynomial-time complexity

- 定义为 $O(p(n))$，其中 $p(n)$ 是关于输入规模 n 的多项式函数

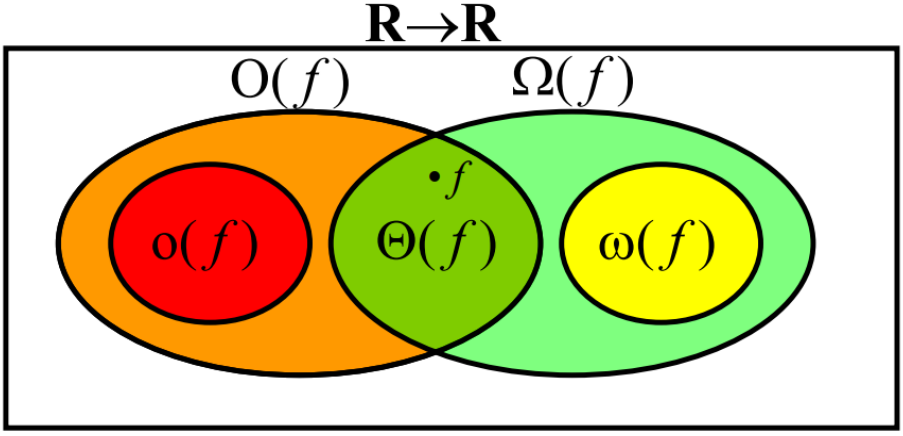
5.3.2 常见函数增长率比较

> 从低到高

- 常数函数：$O(1)$
- 对数函数：$O(log\ n)$
- 线性函数：$O(n)$
- 线性对数函数：$O(n\ log\ n)$
- 多项式函数：$O(n^k)$
- 指数函数：$O(2^n)$
- 阶乘函数：$O(n!)$



5.4 Relations Between Order-of-Growth Sets



6. Set Theory and Counting Principle

6.1 Set Theory

6.1.1 集合的定义

A set S is a collection of elements, in which each element appears only once.

> 元素不重复

Examples:

$S = \{1, 2, 3, 4, 5\}$ set of first 5 positive integers

$S = \{0, 1, 2, \cdots\}$ set of all non-negative integers

$S = \{a, b, c, \cdots, z\}$ set of all lower-case alphabets

$S = \{00, 01, 10, 11\}$ set of all binary strings of length 2

$S = \{1, 1, 4, 5, 1, 4\}$ is not a set, as the elements 1 and 4 are repeated.

6.1.2 势的定义

Given a set S, the number of elements in S is denoted by $|S|$. The quantity $|S|$ is also referred to as the cardinality of S.

6.1.3 空集，子集

A set with no element in it is called an empty set and is denoted by $\emptyset$. Naturally, the cardinality of an empty set is 0.

Given two sets S and T, we say that T is a subset of S, denoted by $T \subseteq S$, if every element in T is also in S. It follows that if $T \subseteq S$, then $|T| \leq |S|$.

6.1.4 交集，并集

Let S_1 and S_2 be two given sets.

① The union of S_1 and S_2 denoted by $S_1 \cup S_2$, is the set containing all elements from both S_1 and S_2.

② The intersection of S_1 and S_2, denoted by $S_1 \cap S_2$, is the set containing all elements that are common to both S_1 and S_2.

> 注意并集操作时，重复元素只能保留一个.

6.1.5 集合划分

Let S be a set and $m \geq 1$ be an integer. A partition of S into m parts is a collection of m subsets of S, denoted by $S_1, \cdots, S_m$, with the following properties:

① (Exhaustion) $S = S_1 \cup S_2 \cup \cdots \cup S_m$.

② (Non-Overlapping) For $i \neq j$, we have $S_i \cap S_j = \emptyset$.

> 注意，这里的 $subset$ 不要求非空. 一个集合可以被它自己和一个空集划分成 2 个部分.

6.2 Counting Principles

6.2.1 加法原理

The **addition principle** states that if $S_1, \cdots, S_m$ form a partition of S, then

$$|S| = |S_1| + |S_2| + \cdots + |S_m|$$

6.2.2 乘法原理

The **multiplication principle** states that if each element in S can be generated by performing an ordered sequence of actions, say, $A_1, A_2, \cdots, A_N$, and action A_i has p_i choices, where $i = 1, \cdots, N$, then the cardinality of S can be computed by

$$|S| = p_1 \times p_2 \times \cdots \times p_N$$

6.2.3 Permutation

Consider a set of n elements, say, $S_0 = \{1, \cdots, n\}$. We call S_0 the ground set. Also, let $r \geq 1$ be integer. An **r-permutation** of the n-element ground set S_0 is an ordered selection of r elements from S_0.

> 注意：一个 r 排列只是 $\frac{n!}{(n-r)!}$ 种排列方式中的一个.

Let S be the set of all different r-permutations of the n-element ground set S_0. We are interested in determining $|S|$ for general values of n and r. Let $P(n, r)$ denotes this number. Observe that an r-permutation of the ground set $\{1, \cdots, n\}$ can be generated by performing the following ordered sequence of actions, where we also record the number of choices for each action:

action	number of choices
A_1: pick the 1st element	n
A_2: pick the 2nd element	$n-1$
$\vdots$	$\vdots$
A_r: pick the rth element	$n-r+1$

Thus, by multiplication principle, we have

$P(n,r) = n(n-1)\cdots(n-r+1) = \frac{n!}{(n-r)!}$

6.2.4 补集

Let S be a set and $A \subseteq S$ be a subset of S. The complement of A in S, denoted by $\overline{A}$, is the set that contains all the elements in S but not in A.

By definition, A and $\overline{A}$ form a partition of S. Hence, by the addition principle, we have

$|S| = |A| + |\overline{A}|$

6.2.5 减法原理

The **subtraction principle** is simply the inverse of the above relationship; i.e.,

$|A| = |S| - |\overline{A}|$

6.2.6 除法原理

The **division principle** states that if S is partitioned into k equal-sized parts, then

$k = \frac{|S|}{number\ of\ elements\ in\ each\ part}$

6.2.7 抽屉原理

> Pigeonhole Principle

$\lceil x \rceil = least\ integer \geq x$

$\lfloor x \rfloor = greatest\ integer \leq x$

Suppose that n objects are placed into k boxes. Then, at least one box has at least $\lceil \frac{n}{k} \rceil$ objects.

Proof: We prove the proposition by contradiction. Suppose that the conclusion is false. Then, every box has at most $\lceil \frac{n}{k} \rceil - 1$ objects. Since there are k boxes altogether, we have

total number of objects $\leq k \times (\lceil \frac{n}{k} \rceil - 1)$

Now, for any real number x, it can be easily verified that $\lceil x \rceil < x + 1$. It follows that

total number of objects $< k \times \frac{n}{k} = n$

which contradicts the fact that the total number of objects is n.

6.2.8 Combination

Consider the ground set $S_0 = \{1, \cdots, n\}$ of n elements. Let $r \geq 1$ be integer. An r-combination of the n-element ground set S_0 is an unordered selection of r elements from S_0. Let S be the set of all different r-combinations of the n-element ground set S_0.

We are interested in determining $|S|$ for general values of n and r. Let $C(n, r) = \binom{n}{r}$ denote this number.

The definition implies that

$$\binom{n}{0} = 1 \quad \binom{n}{1} = n \quad \binom{n}{n} = 1$$

By convention, we set

$$\binom{n}{r} = 0 \quad \text{whenever } r > n.$$

For general values of n and r, we can determine $\binom{n}{r}$ by relating r-combinations to r-permutations. Indeed, observe that each r-permutation of S_0 can be generated via the following ordered sequence of actions:

A_1: pick r elements from S_0.

A_2: order the r elements chosen from A_1 to form the desired r-permutation.

Note that A_1 has $\binom{n}{r}$ choices, while A_2 has $r!$ choices. Hence, by the multiplication principle,

$$P(n, r) = \binom{n}{r} \times r!$$

Since $P(n,r) = \frac{n!}{(n-r)!}$, it follows that

$\binom{n}{r} = \frac{n!}{r!(n-r)!}$

*6.3 可数集

> Countable Sets. Lecture 10 补充内容

Countable set:

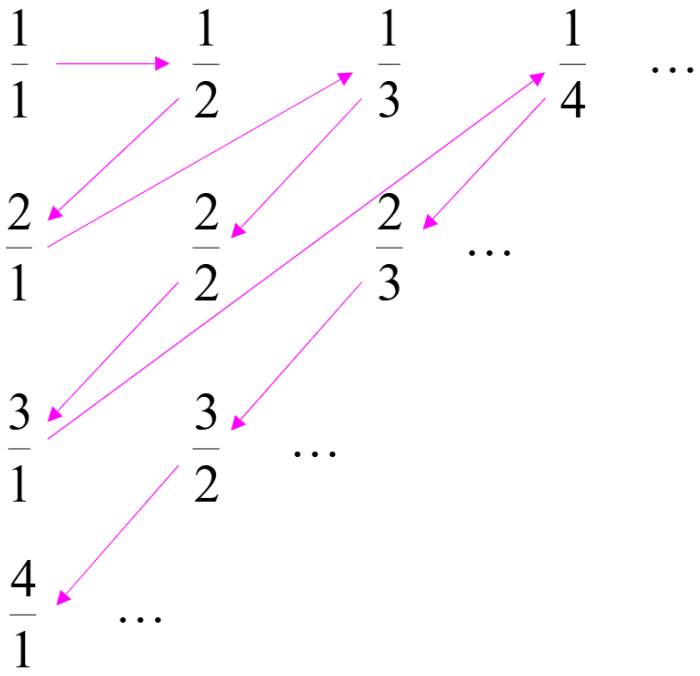
Any finite set or

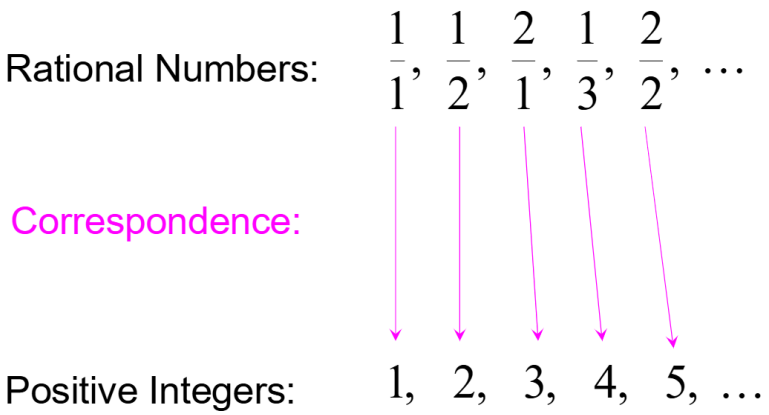
Any countably infinite set:

There is a one to one correspondence between elements of the set and Natural numbers

> 集合的元素能和自然数集建立一一对应关系.

有理数集是可数的:





*6.4 不可数集

> Uncountable Sets

A set is uncountable if it is not countable.

Theorem:

Let S be an infinite countable set

The powerset 2^S of S is uncountable

> 幂集（Powerset）是一个集合的所有子集组成的集合.

Proof:

Since S is countable, we can write

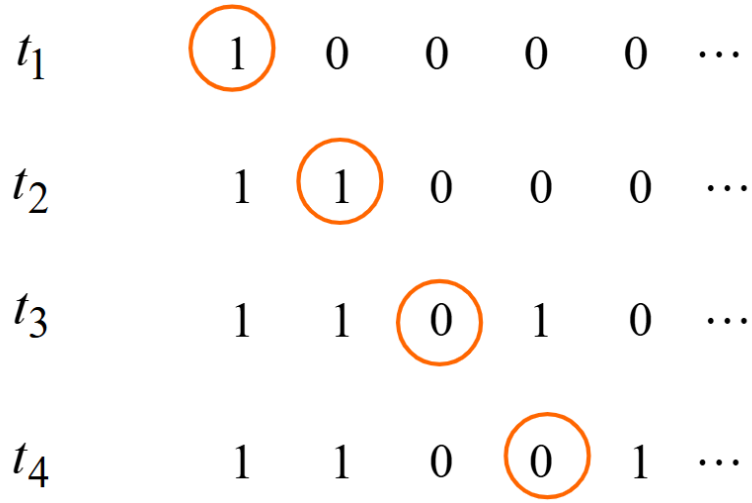
$S = \{s_1, s_2, s_3, \ldots\}$

We encode each element of the power set with a binary string of 0's and 1's

Powerset element	Encoding				
	s_1	s_2	s_3	s_4	$\cdots$
$\{s_1\}$	1	0	0	0	$\cdots$
$\{s_2, s_3\}$	0	1	1	0	$\cdots$
$\{s_1, s_3, s_4\}$	1	0	1	1	$\cdots$

反证法：假设 the powerset is countable.

> 康托对角线法



把对角线 bits 取补码，构造子集 $t_{new} = 0011\ldots$

若 powerset 可数，则 t_{new} 是 powerset 的某个元素 t_i.

但根据构造，t_{new} 的第 i 位与 t_i 的第 i 位相反，必须同时为 0 和 1，矛盾.

因此无穷可数集的 powerset 不可数.

7. Binomial Coefficients

Recall the quantity $\binom{n}{r}$, which is known as the *binomial coefficient*, arises when we count the number of different unordered selections of r elements from a n-element ground set. As it turns out, the binomial coefficients satisfy many useful identities. These identities can be proven either by invoking the formula for $\binom{n}{r}$ or by using counting arguments. The former gives rise to *algebraic proofs*, while the latter to *combinatorial proofs*.

7.1 Binomial Identities

> 二项式恒等式

Proposition 1

For any integers $n, r \geq 0$ with $0 \leq r \leq n$,

① $\binom{n}{r} = \binom{n}{n-r}$

Algebraic Proof

This can be done by simply invoking the formula

$\binom{n}{r} = \frac{n!}{r!(n-r)!}$

> 缺点在于没有解释这个恒等式的意义.

Combinatorial Proof

The idea of a combinatorial proof is that both sides of the identity ① are supposed to be **two different ways of solving a counting problem**, and our task is to explain

(i) what is the counting problem and

(ii) how the two sides of the identity are solving the counting problem.

> 组合证明的核心：证明恒等式两边是解决同一计数问题的不同方法.

To proceed, let us define the counting problem for ①. It is simply counting the number of different unordered selections of r elements from an n-element ground set. This number is, by definition, given by the LHS of ①. Now, observe that to select r elements from an n-element ground set is the same as to select which $n-r$ elements to be *excluded*. The number of ways to perform the latter is precisely $\binom{n}{n-r}$. It follows that ① holds.

> n 选 r 等价于 n 选 $n-r$（选了扔掉，剩下 r 个）

Theorem 1

> Pascal's Identity

For any integers $n, r \geq 0$ with $1 \leq r \leq n-1$,

② $\quad \binom{n}{r} = \binom{n-1}{r} + \binom{n-1}{r-1}$

Algebraic Proof

Using the formulae for $\binom{n}{r}$ and $n!$, the RHS equals

$$\begin{aligned}\frac{(n-1)!}{r!(n-r-1)!} + \frac{(n-1)!}{(r-1)!(n-r)!} &= \frac{(n-1)!}{(r-1)!(n-r-1)!}\left(\frac{1}{r} + \frac{1}{n-r}\right) \\ &= \frac{(n-1)!}{(r-1)!(n-r-1)!} \times \frac{n}{r(n-r)} \\ &= \binom{n}{r}.\end{aligned}$$

Combinatorial Proof

Again, the LHS of ② suggests that the counting problem is simply to count the number of different unordered selections of r elements from an n-element ground set. To facilitate our proof, let S be the set of all unordered selections of r elements from the n-element ground set $\{1,\ 2,\ \ldots,\ n\}$. By definition, we have

$$|S| = \binom{n}{r}.$$

What would be another way to determine $|S|$? The RHS of the identity ② involves the sum of two terms, which suggests that the addition principle is at work. Following this line of thought, let us partition S into two parts S_1 and S_2, where

$$|S| = |S_1| + |S_2|.$$

Now, observe that an alternative description of S_1 is that it contains all unordered selections of r elements from $\{2, \ldots, n\}$ (since "1" is not allowed to appear). Thus, by definition,

$$|S_1| = \tbinom{n-1}{r}.$$

For S_2, since "1" appears as one of the r elements in the selection, we can only select $r-1$ more elements from $\{2, 3, \ldots, n\}$. There are $\binom{n-1}{r-1}$ ways to do this. Thus,

$$|S_2| = \tbinom{n-1}{r-1}.$$

Putting everything together yields the identity ②.

> 某个元素有选，和某个元素没选.

Theorem 2

> Binomial Theorem，二项式定理

Let $n \geq 1$ be an integer. For any real numbers x, y,

③ $\quad (x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$

The Binomial Theorem generalizes the well-known expansion $(x+y)^2 = x^2 + 2xy + y^2$. As before, it can be proven by both algebraic and combinatorial means.

Algebraic Proof

We proceed by induction on $n \geq 1$. For the base case, the RHS of ③ equals

$$\sum_{k=0}^{1} \binom{1}{k} x^k y^{1-k} = x + y$$

which is the same as the LHS of ③.

For the inductive step,

$$\begin{aligned}
(x+y)^{n+1} = (x+y)(x+y)^n &= (x+y)\sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k} \\
&= \sum_{k=0}^{n} \binom{n}{k} x^{k+1} y^{n-k} + \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k+1} \\
&= \sum_{j=1}^{n+1} \binom{n}{j-1} x^j y^{n-j+1} + \sum_{j=0}^{n} \binom{n}{j} x^j y^{n-j+1} \\
&\text{(第一项 } j = k+1, \text{ 第二项 } j = k\text{)} \\
&= \sum_{j=1}^{n} \left(\binom{n}{j-1} + \binom{n}{j} \right) x^j y^{n-j+1} + \binom{n}{n} x^{n+1} + \binom{n}{0} y^{n+1} \\
&= \sum_{j=1}^{n} \binom{n+1}{j} x^j y^{n-j+1} + \binom{n+1}{n+1} x^{n+1} + \binom{n+1}{0} y^{n+1} \\
&\text{(第一项使用了 } Pascal's\ Identity\text{)} \\
&= \sum_{j=0}^{n+1} \binom{n+1}{j} x^j y^{n-j+1}.
\end{aligned}$$

Combinatorial Proof

To gain some insights into the structure of the expansion of $(x+y)^n$, let us first consider a simple case where $n = 3$. The LHS of ③ can be written as

$$(x+y)(x+y)(x+y).$$

Observe that the expansion of the above expression can be done by taking one of x or y from each $(x+y)$ term. For instance, if we take the bolded terms below

$$(\mathbf{x}+y)(x+\mathbf{y})(x+\mathbf{y})$$

and gather them, we get the term xy^2. Of course, we get the same term if we gather the bolded terms below

$$(x+\mathbf{y})(x+\mathbf{y})(\mathbf{x}+y).$$

Thus, the terms in the expansion of

$$(x+y)(x+y)(x+y)$$

can be obtained as follows:

choice	term
$(\mathbf{x}+y)(\mathbf{x}+y)(\mathbf{x}+y)$	x^3
$(\mathbf{x}+y)(\mathbf{x}+y)(x+\mathbf{y})$	x^2y
$(\mathbf{x}+y)(x+\mathbf{y})(\mathbf{x}+y)$	x^2y
$(x+\mathbf{y})(\mathbf{x}+y)(\mathbf{x}+y)$	x^2y
$(\mathbf{x}+y)(x+\mathbf{y})(x+\mathbf{y})$	xy^2
$(x+\mathbf{y})(\mathbf{x}+y)(x+\mathbf{y})$	xy^2
$(x+\mathbf{y})(x+\mathbf{y})(\mathbf{x}+y)$	xy^2
$(x+\mathbf{y})(x+\mathbf{y})(x+\mathbf{y})$	y^3

$(x+y)^3$ is simply equal to the sum of the terms in the second column of the above table.

Motivated by the above observation, let us now tackle the general case. Consider the term $x^k y^{n-k}$ in the expansion of $(x+y)^n$, where $0 \leq k \leq n$. Such a term can be obtained as follows: Out of the n $(x+y)$ terms, we need take an x from k of them and a y from the remaining $n-k$ of them. By definition, there are $\binom{n}{k}$ ways to perform this task. Hence, the term

$\binom{n}{k} x^k y^{n-k}$

appears in the expansion of $(x+y)^n$. By summing the above over $k = 0,\ 1,\ \ldots,\ n$, we obtain the desired identity ③.

Interestingly, the Binomial Theorem can generate many new identities by substituting various values of x and y. For instance, by setting $x = y = 1$, the identity ③ gives

④ $\quad 2^n = \sum_{k=0}^{n} \binom{n}{k}$

Combinatorial Proof

Now, to generate a subset of $S_0 = \{1,\ \ldots,\ n\}$ with k elements, we need to select k elements in S_0 to form the subset. Thus, there are $\binom{n}{k}$ different subsets of S_0 with k elements. Since a subset of S_0 can have $k = 0,\ 1,\ \ldots,\ n$ elements, by the addition principle, the number of different subsets of S_0 is given by the RHS of ④.

On the other hand, a subset of S_0 can be generated via the following ordered sequence of actions:

action	number of choices
A_1: decide whether "1" belongs to the subset	2
A_2: decide whether "2" belongs to the subset	2
$\vdots$	$\vdots$
A_n: decide whether "n" belongs to the subset	2

Hence, by the multiplication principle, there are 2^n different subsets of S_0, which is the LHS of ④.

Theorem 3

> Multinomial Theorem

Let $n \geq 1$ be an integer. For any real numbers $x_1, x_2, \ldots, x_k$,

$$(x_1 + x_2 + \cdots + x_k)^n = \sum_{\substack{n_1, n_2, \ldots, n_k \\ n_1+n_2+\cdots+n_k=n}} \frac{n!}{n_1! \times n_2! \times \cdots \times n_k!} x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k}.$$

As an application of the Multinomial Theorem, we can see that the coefficient of $x_1^2 x_2 x_3$ in the expansion $(x_1 + x_2 + x_3)^4$ is

$\frac{4!}{2!1!1!} = 12.$

This, of course, can be verified by directly expanding $(x_1 + x_2 + x_3)^4$.

7.2 Bookeeper Theorem

Now, let us count the number of different sequences that can be obtained by permuting the letters **BOOKEEPER**. Note that there are 9 letters in the word **BOOKEEPER**, yet not all $9!$ permutations give rise to a different sequence, as some of the letters are repeated.

To tackle this problem, we first observe that each sequence is completely determined by the locations of the distinct symbols. For example, the sequence **OBOKEPEER** can be specified by noting that

symbol	locations
B	2
E	5, 7, 8
K	4
O	1, 3
P	6
R	9

The number of different sequences that can be obtained by permuting the letters **BOOKEEPER** is

$$\frac{9!}{1!3!1!2!1!1!}.$$

8. Elements of Discrete Probability

8.1 Basic Definition

Def 1 概率空间

Definition 1： A probability space is a pair $(S,\ p(\cdot))$, where

- S is the set of all possible outcomes of the random experiment and is called the sample space,
- $p(\cdot)$ is a function that assigns a number to each outcome in S such that

  ① $p(s) \geq 0$ for each outcome s in S and

  ② the sum of $p(s)$ over all outcomes s in S equals 1; i.e., $\sum_{s \in S} p(s) = 1$.

> The function $p(\cdot)$ is called the probability function and the requirement ② is to normalize the numbers assigned by $p(\cdot)$.

Example 1

Consider flipping a coin once. The set of all possible outcomes is usually taken as $S = \{head,\ tail\}$.

As for the probability function, there is some flexibility in assigning numbers to the above two outcomes. For instance, if the coin is fair, one would set

$p(head) = 0.5, \quad p(tail) = 0.5.$

On the other hand, the assignment

$p(head) = 0.6, \quad p(tail) = 0.4.$

indicates that the coin is biased. Note that both of the above assignments satisfy ②.

Example 2

Let S be a finite set of possible outcomes of a random experiment. We say that every outcome in S is equally likely if $p(s) = \frac{1}{|S|}$. Clearly, such an assignment satisfies $p(s) \geq 0$ for each outcome s in S. Moreover, we have

$\sum_{s \in S} p(s) = \sum_{s \in S} \frac{1}{|S|} = \frac{1}{|S|} \sum_{s \in S} 1 = \frac{1}{|S|} \times |S| = 1.$

$\sum_{s \in S} \frac{1}{|S|} = \frac{1}{|S|} \sum_{s \in S} 1$

> This is due to the fact that each summand is the same and is equal to $\frac{1}{|S|}$, which means that it can be taken out of the sum by distributive law.

$\frac{1}{|S|} \sum_{s \in S} 1 = \frac{1}{|S|} \times |S|$

> To get this equality, recall that $\sum_{s \in S} 1$ is interpreted as the number of terms in the sum. Since there is one term for each outcome in S, we conclude that $\sum_{s \in S} 1 = |S|$.

As an aside, in order to know the probability assigned to each outcome, we need to know $|S|$. This is related to counting.

Example 3

Consider two rolls of a 6-sided die. Then, the set of all possible outcomes are given by

$S = \{(1,1),\ (1,2),\ (1,3),\ \ldots,\ (1,6),\ \ldots,\ (6,1),\ (6,2),\ \ldots,\ (6,6)\},$

where (i, j) means that the first roll gives the value i and the second roll gives the value j. Suppose that every outcome in S is equally likely. Then, the probability of the outcome (i, j) is given by $p(i, j) = \frac{1}{|S|}$, where $i = 1,\ 2,\ \ldots,\ 6$ and $j = 1,\ 2,\ \ldots,\ 6$. Now, using the multiplication principle, it can be readily seen that $|S| = 6 \times 6 = 36$.

Oftentimes we are interested in the probability of not just a single outcome but a collection of outcomes. For instance, when rolling a 6-sided die twice, we may be interested in the probability that both rolls are even. In this case, the outcomes of interest are

$T = \{(2,2),\ (2,4),\ (2,6),\ (4,2),\ (4,4),\ (4,6),\ (6,2),\ (6,4),\ (6,6)\}$

This motivates the following definition.

Def 2 事件概率

Definition 2: Let $(S,\ p(\cdot))$ be a probability space. An event T is simply a subset of S (i.e., $T \subseteq S$).

The probability of an event T is given by

$p(T) = \sum_{s \in T} p(s).$

As a quick illustration, consider the setting in Example 3. Then, the set
$T = \{(2,2),\ (2,4),\ (2,6),\ (4,2),\ (4,4),\ (4,6),\ (6,2),\ (6,4),\ (6,6)\}$ is an event. Its probability is given by

$$p(T) = \sum_{s \in T} p(s) = \frac{1}{|S|} \sum_{s \in T} 1 = \frac{|T|}{|S|} = \frac{9}{36} = \frac{1}{4}.$$

Note how the cardinality of T comes into play in the calculation of $p(T)$.

练习

Consider two rolls of 6-sided die. What is the possibility that the sum of two rolls are 7?

$(1,6),\ (2,5),\ (3,4),\ (4,3),\ (5,2),\ (6,1)$

$\frac{6}{36} = \frac{1}{6}$

Consider two rolls of 6-sided die. We are told that the outcome of at least one die is 5. What is the possibility that the sum of two rolls are 10?

$(5,1),\ (5,2),\ (5,3),\ (5,4),\ (5,5),\ (5,6),\ (1,5),\ (2,5),\ (3,5),\ (4,5),\ (6,5)$

$\frac{1}{11}$

An urn contains 6 white and 9 black balls. If 4 balls are to be randomly selected without replacement, what is the probability that the first 2 selected are white and the last 2 black?

$(6/15)(5/14)(9/13)(8/12) = 6/91$

法二：

$$\frac{\frac{1}{6} \times \binom{6}{2}\binom{9}{2}}{\binom{15}{4}}$$

> $\frac{1}{6}$：白白黑黑，黑黑白白，白黑黑白，黑白白黑，白黑白黑，黑白黑白.
>
> 不要用这种方法，太 tricky.

8.2 Further Illustrations

Example 4

> Poker

A hand of 5 cards is drawn from a standard 52-card deck. Suppose that every hand of 5 cards is equally likely. What is the probability of getting a hand with all distinct face values?

Define the event of interest T and determine its probability.

In this problem, we are interested in outcomes in which all 5 cards have distinct face values. Hence, we can simply let T be the set of all possible hands of 5 cards whose face values are distinct. To determine its probability, we need to compute $|T|$. Observe that each element in T can be generated via the following ordered sequence of actions:

action	number of choices
A_1: choose 5 distinct values to appear in the hand	$\binom{13}{5}$
A_2: choose one suit for each of the 5 distinct values	4^5

By the multiplication principle, we have

$|T| = \binom{13}{5} \times 4^5$.

It follows that the desired probability is given by

$$p(T) = \sum_{s \in T} p(s) = \frac{1}{|S|} \sum_{s \in T} 1 = \frac{|T|}{|S|} = \frac{\binom{13}{5} \times 4^5}{\binom{52}{5}} \approx 0.507.$$

法二：条件概率

$p(T) = 1 \times (1 - \frac{3}{51}) \times (1 - \frac{6}{50}) \times (1 - \frac{9}{49}) \times (1 - \frac{12}{48}) \approx 0.507.$

Consider distributing 5 cards (one hand) each to two persons from a deck of 52 playing cards. What is the number of distributions that both persons receive a 'Four of a Kind,' i.e., the hand of 5 cards contains all the 4 cards of the same face value, e.g., it contains all '3's from Heart (♡), Spade (♠), Club (♣), Diamond (♢)?

> 一副牌 52 张，2 人各摸 5 张，同时摸到炸的概率.

The sample space is the ways to distribute the cards, S. The cardinality of S can be calculated by

$$|S| = \binom{52}{5} \times \binom{47}{5} = \frac{52!}{5!5!42!}.$$

Since each distribution is equally likely, the probability function is uniform. For each $s \in S$, $P(s) = \frac{1}{|S|}$.

The event is when both persons receive a 'Four of a Kind.' There are

$$13 \times 12 \times 44 \times 43$$

possibilities.

- 13×12: select face values,
- 44×43: select the remaining 1 card.

Thus the size of the event is calculated by

$$|T| = 13 \times 12 \times 44 \times 43.$$

The probability is

$$P(T) = \frac{|T|}{|S|} = \frac{13 \times 12 \times 44 \times 43}{\frac{52!}{5!5!42!}}.$$

Example 5

> Monty Hall Problem

You are on a TV show, and the host shows you three identical boxes—A, B, and C. One of the boxes, say, A, contains the grand prize and the rest contain nothing. Of course, this is not known to you. After you pick a box and announce your choice, the host opens one of the empty boxes that is not picked by you. She then offers you the option to change your choice. The question is: **Should you change?**

Of course, whether you change your choice or not, you cannot guarantee that you will always win the grand prize. Thus, your goal is to determine the strategy that will maximize the probability of winning the grand prize. Towards that end, we first observe that the sequence of actions in the game can be described by a 3-tuple (u, v, w), where

- $u = \text{your initial choice},$
- $v = \text{host's choice of the empty box},$
- $w = \text{your final choice}.$

For instance, if your initial choice is box A, the host opens box C, and then you decide to change, then the sequence can be described by (A, C, B). Now, let us consider the two strategies.

Strategy 1: Always change V.S. Strategy 2: Never change

Strategy 1: Always change

In this case, the sample space is given by

$$S = \{(A, B, C), (A, C, B), (B, C, A), (C, B, A)\}.$$

Following Example 4, our next step is to assign probabilities to the outcomes. Since we are only concerned with the winning probability, let us first determine which of the outcomes will result in winning.

Obviously, both (B, C, A) and (C, B, A) are winning outcomes, while both (A, B, C) and (A, C, B) are losing outcomes. In particular, you will win if your initial choice is either B or C, but you will lose if your initial choice is A.

Since the three boxes are identical, it is fair to assume that every box is equally likely to be chosen by you at the beginning; i.e., each box has probability $\frac{1}{3}$ of being chosen.

Thus, we have reached an important conclusion: If your strategy is to always change your choice after the host's announcement, then you will win with probability $\frac{1}{3} + \frac{1}{3} = \frac{2}{3}$ and will lose with probability $\frac{1}{3}$.

Strategy 2: Never change

Following similar lines of reasoning as above, the sample space in this case is given by

$$S = \{(A, B, A), (A, C, A), (B, C, B), (C, B, C)\}.$$

From the above, it is easy to see that you will win if your initial choice is A, while you will lose if your initial choice is B or C. In this case, your winning probability is only $\frac{1}{3}$.

Exercise 6

> Binomial Distribution

Suppose that we flip a coin n times, where each flip is independent of one another. Furthermore, suppose that the probability of getting a head in a flip is p (and thus the probability of getting a tail is $q = 1 - p$). What is the probability of getting exactly k heads, where $k = 0,\ 1,\ \ldots,\ n$?

Step 1: Specify the sample space.

Here, we let S be the set of all length-n sequences of H and T, where H means a head and T means a tail. For instance, when $n = 3$, we have

$$S = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}.$$

Step 2: Specify the probability function.

Since each flip is assumed to be independent of each other, the probability of a given sequence of H and T can be obtained by simply replacing H in the sequence with p (the probability of getting a head) and T with q (the probability of getting a tail). For instance, when $n = 3$, we have

$$p(HTH) = p \times q \times p = p^2 q, \quad p(THH) = q \times p \times p = q p^2,$$

etc. Clearly, such a probability assignment satisfies the non-negativity requirement. We shall verify that it also satisfies the normalization requirement shortly. Thus, the above is indeed a legitimate probability assignment.

Step 3: Define the event of interest S' and determine its probability.

The event of interest is simply the set of all length-n sequences of H and T that contain exactly k H's. For instance, when $n = 3$ and $k = 2$, we have

$$S' = \{HHT, HTH, THH\}.$$

In general, since each sequence in S' has exactly k heads and $n-k$ tails, the probability of each sequence is $p^k q^{n-k}$. Moreover, we know that $|S'| = \binom{n}{k}$, since each sequence in S' corresponds to a selection of k out of n positions to put an H. Thus, we conclude that

$$p(S') = |S'| \times p^k q^{n-k} = \binom{n}{k} p^k q^{n-k}.$$

Now we can verify that the probability assignment defined in Step 2 satisfies the normalization constraint. Indeed, the sequences in S can be partitioned into $S_0,\ S_1,\ S_2,\ \ldots,\ S_n$, where S_k is the set of sequences that have exactly k heads. By the binomial theorem, we have

$$\sum_{s \in S} p(s) = \sum_{k=0}^{n} p(S_k) = \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} = (p+q)^n = 1.$$

Exercise 7

> Birthday Paradox

There are n people in a room. We are interested in their birthdays (just month and day). For simplicity, let us assume that a year only has 365 days, and that every day is equally likely to be the birthday of a person. What is the probability that at least two people have the same birthday? Of course, we assume that $n < 365$, for otherwise the answer is trivial. Again, let us follow the steps in Example 4 to tackle this question.

Step 1: Specify the sample space.

We are interested in the possible birthdays of n people. Thus, let

$$S = \text{set of all possible sequences of } n \text{ birthdays}$$

and note that

$$|S| = 365^n$$

by the multiplication principle.

Step 2: Specify the probability function.

We assume that every outcome in S is equally likely. Thus, for each outcome s in S, we have

$$p(s) = \frac{1}{365^n}.$$

Step 3: Define the event of interest S' and determine its probability.

The event of interest is the sequences in S in which at least two birthdays are the same. Let T be such an event. Then, the desired probability is given by

$$p(T) = \frac{|T|}{|S|}.$$

To determine T, let us consider its complement $\overline{T}$ in S. By definition,

$$\begin{aligned}\overline{T} &= \text{set of all sequences of } n \text{ birthdays in which at most one birthday is the same} \\ &= \text{set of all sequences of } n \text{ birthdays in which all birthdays are distinct.}\end{aligned}$$

Note that $|\overline{T}| = P(365, n)$, as each sequence in $\overline{T}$ is an ordered selection of n birthdays out of 365 days. Thus, by the subtraction principle, we have

$$|T| = |S| - |\overline{T}| = 365^n - \frac{365!}{(365-n)!}.$$

The desired probability can then be computed from $p(T) = \frac{|T|}{|S|}$.

To appreciate the significance of this result, let us consider some concrete values of n. For $n = 23$, $p(T) > 0.5$; while for $n = 30$, $p(T) > 0.7$. Thus, the probability may not be as small as one would think even when n is small compared to 365.

> 人数较少时，有生日重合的概率可能比想象中高.

9. Introduction to Graph Theory

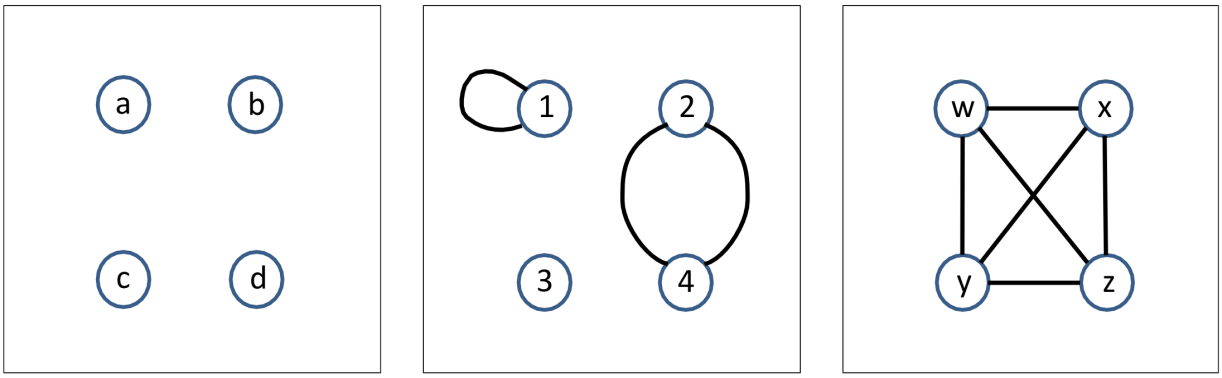
9.1 Definition

A graph consists of a nonempty set V of vertices and a set E of edges, where each edge in E connects two (may be the same) vertices in V.

Let G be a graph associated with a vertex set V and an edge set E.

We usually write $G = (V, E)$ to indicate the above relationship.

Examples



简单图

Furthermore, if each edge connects two different vertices, and no two edges connect the same pair of vertices, then the graph is a *simple graph*.

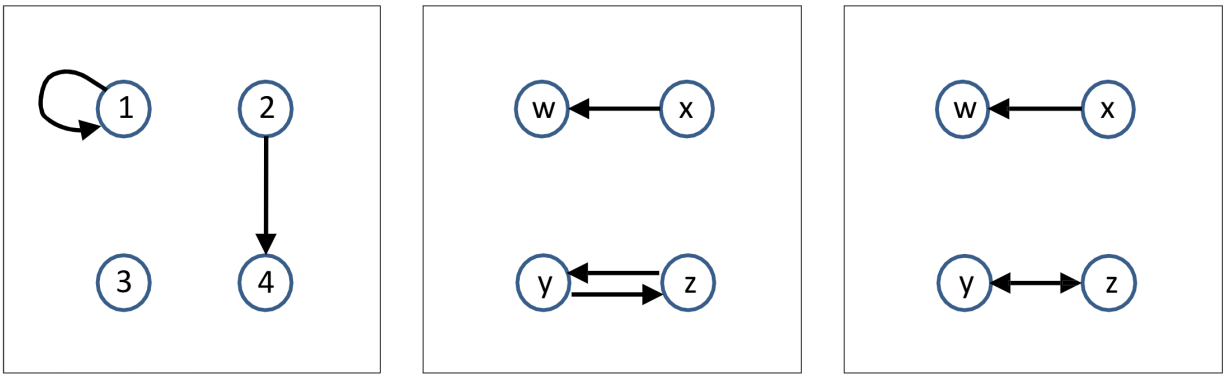
> 简单图：不含自环，不含多重（同向）边.
>
> 边集可以为空. 上图中第一幅和第三幅都是 simple graph.
>
> 对于无向简单图，两个顶点之间最多一条边；对于有向简单图，可以有双向边，既可以有一条边从 u 指向 v，另一条边从 v 指向 u.

9.2 有向图

> Directed Graph

A directed graph G consists of a nonempty set V of vertices and a set E of directed edges, where each edge is associated with an ordered pair of vertices. We write $G = (V, E)$ to denote the graph.

Examples



9.3 无向图

> Undirected Graph

Let e be an edge that connects vertices u and v. We say

(i) e is $incident\ with$ u and v
(ii) u and v are the $endpoints$ of e;
(iii) u and v are $adjacent$ (or $neighbors$)
(iv) if $u = v$, the edge e is called a $loop$.

The $degree$ of a vertex v, denoted by $deg(v)$, is the number of edges incident with v, except that a loop at v contributes **twice** to the degree of v.

Suppose we have a simple graph G with n vertices. What is the maximum number of edges G can contain, if

(i) G is an undirected graph?

> $\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{n(n-1)}{2}$

(ii) G is a directed graph?

> $2 \times \binom{n}{2} = n(n-1)$

Example 1: $R(3,3)$

> 关键词：拉姆齐数，拉姆齐定理

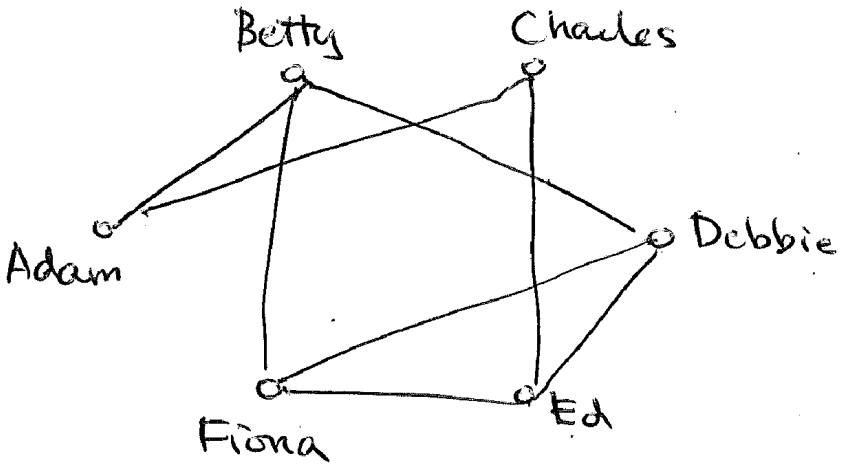
Proposition: among 6 people, there will be '3 mutual acquaintances' OR '3 mutual strangers' (both can happen at the same time).

证明：

Consider a graph $G = (V, E)$ where V is the set of people, e.g.,

$V = \{Adam,\ Betty,\ Charles,\ Debbie,\ Ed,\ Fiona\}$.

E indicates the acquaintance (e.g., friendship). E.g.,



3 mutual acquaintances $\longleftrightarrow$ triangle

3 mutual strangers $\longleftrightarrow$ 3 mutually disconnected vertices

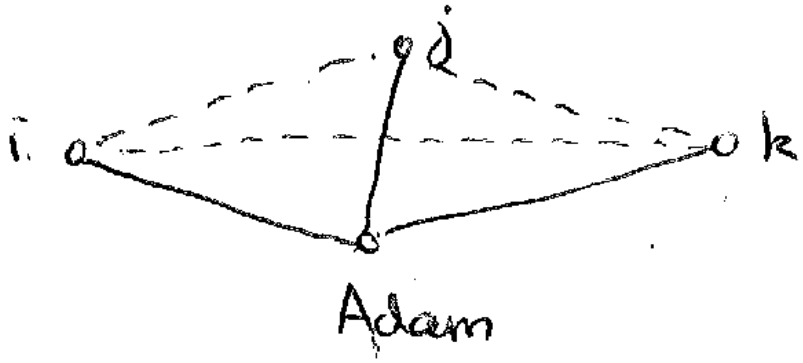
Consider a particular person, e.g., Adam, we note that

$$neighbors\ of\ Adam + non\text{-}neighbors\ of\ Adam = 5$$

As a consequence, either

$$neighbors\ of\ Adam \geq 3\ OR\ non\text{-}neighbors\ of\ Adam \geq 3$$

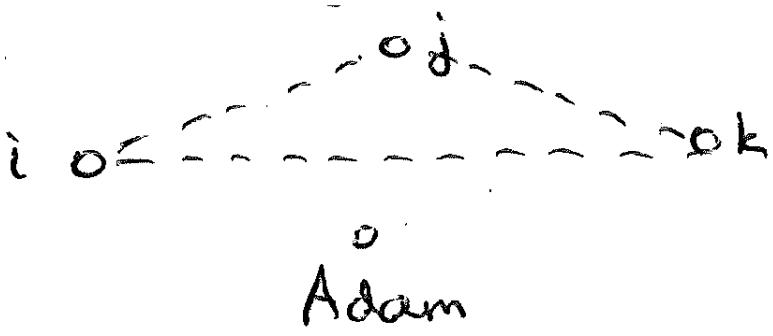
Case 1: $neighbors\ of\ Adam \geq 3$. Let i, j, k be the neighbors:



If $(i, j) \in E$ OR $(j, k) \in E$ OR $(i, k) \in E$, then we have a **triangle** formed by Adam and two members from i, j, k.

Otherwise, $(i, j) \notin E$ AND $(j, k) \notin E$ AND $(i, k) \notin E$, then i, j, k are 3 **mutual strangers**.

Case 2: $non\text{-}neighbors\ of\ Adam \geq 3$. Let i, j, k be the $non\text{-}neighbors$:



If $(i, j) \in E$ AND $(j, k) \in E$ AND $(i, k) \in E$, then we have a **triangle** formed by i, j, k.

Otherwise, we see that Adam and at least 2 members from i, j, k form a group of 3 **mutual strangers**.

Handshaking Theorem

Let $G = (V, E)$ be an undirected graph with $|E|$ edges, then

$$\sum_{v \in V} deg(v) = 2|E|$$

> 每条边贡献两次计数.

Corollary: An undirected graph has an even number of vertices of odd degree.

$$\begin{aligned} 2|E| &= \sum_{i=1}^{n} deg(v_i) \\ &= \sum_{i:\, deg(v_i)=odd} deg(v_i) + \sum_{i:\, deg(v_i)=even} deg(v_i) \end{aligned}$$

左边是偶数，右边第二项是偶数，因此 $\sum_{i:\, deg(v_i)=odd} deg(v_i)$ 是偶数.

奇数个奇数的和为奇数，偶数个奇数的和为偶数（两两配对），因此

the number of odd-degree vertices is even.

9.4 简单路径

> Simple Paths

A simple path in G is either a single vertex or an ordered list of distinct vertices $v_0 - v_1 - \ldots - v_k$.

> 无重复顶点，无重复边.

9.5 循环路径

A cycle in G is a path $v_0 - v_1 - \ldots - v_k$ such that $v_0 = v_k$ and $k \geq 3$.

> 本课程中，两个顶点来回视为 non-cycle.

The **length** of a path / cycle is the number of edges in the path.

9.6 Some Properties of Graphs

A graph is connected if every pair of vertices has a path between them.

A graph is acyclic if it does not contain cycle.

A connected, acyclic graph is called a tree.

a leaf of a tree is a vertex of degree 1.

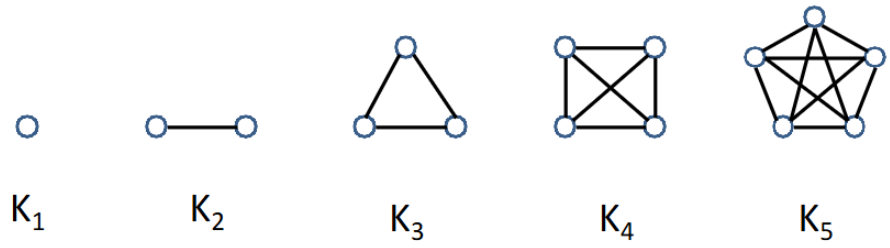
9.7 Some Special Simple Graphs

完全图

> complete graph

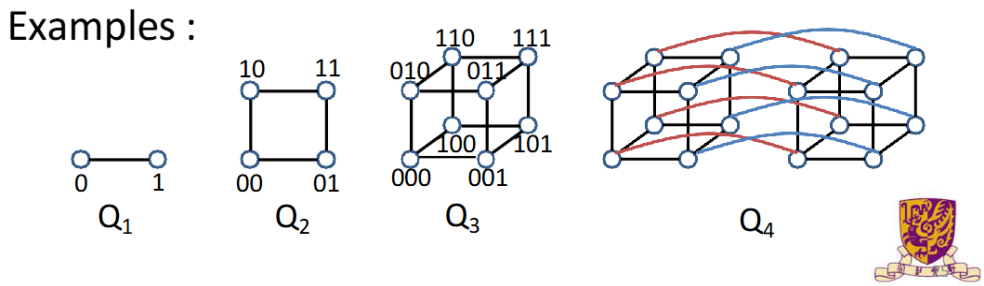
A $complete\ graph$ on n vertices, denoted by K_n, is a simple graph that contains one edge between each pair of distinct vertices.

Examples:



n 维超立方体

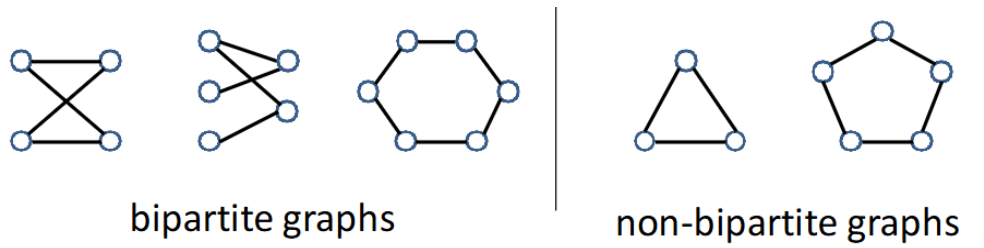
An $n-cube$, denoted by Q_n, is a graph that consists of 2^n vertices, each representing a distinct n-bit string. An edge exists between two vertices $\Leftrightarrow$ the corresponding strings differ in exactly one bit position.



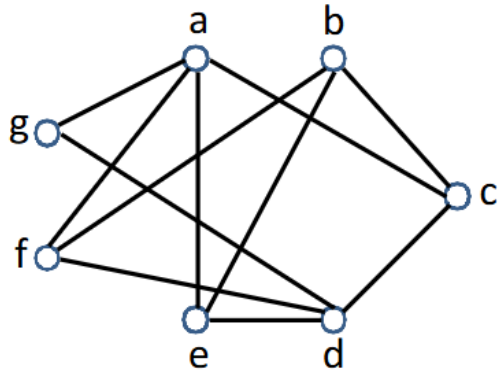
二分图

A $bipartite\ graph$ is a graph such that the vertices can be partitioned into two sets V and W, so that each edge has exactly one endpoint from V, and one endpoint from W.

Examples:



复杂的二分图：



> a b d 一组

Theorem: A simple graph is bipartite if and only if it is possible to assign one of two different colors to each vertex, so that no two adjacent vertices are assigned the same color.

正则图

> regular graph

正则图（Regular Graph）：每个顶点度数（Degree）相同.

补图

> Complement

给定一个简单无向图 $G = (V, E)$，其**补图（Complement Graph）**记为 G^c 或 $\overline{G}$，定义如下：

- 补图 G^c 与原图 G 有相同顶点集 V.
- 补图 G^c 的边集 E^c 包含原图 G 中不存在的边，即：
  - 如果 $u, v \in V$ 且 $u \neq v$，则 $\{u, v\} \in E^c$ 当且仅当 $\{u, v\} \notin E$.

> G^c 的边集是补全整个完全图 K_n 的边集，再去掉 G 的边集.

9.8 图同构

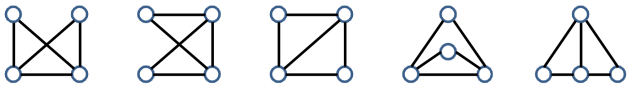
> Graph Isomorphism，指两个图的拓扑结构相同.

定义

Graphs $G = (V, E)$ and $H = (U, F)$ are *isomorphic* if we can set up a bijection $f : V \to U$ such that x and y are adjacent in $G \Leftrightarrow f(x)$ and $f(y)$ are adjacent in H.

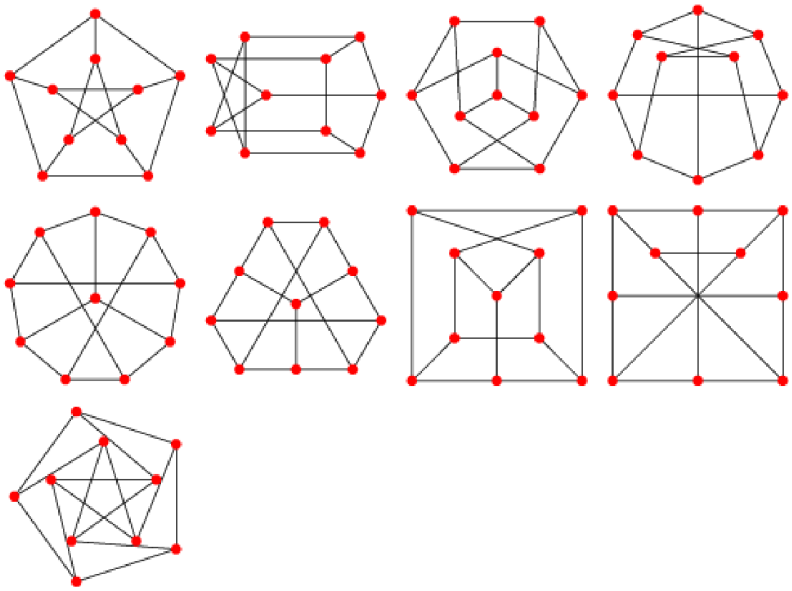
Example

Ex： The following are isomorphic to each other :



Petersen graph

The following graphs are isomorphic to each other. This graph is known as the Petersen graph:



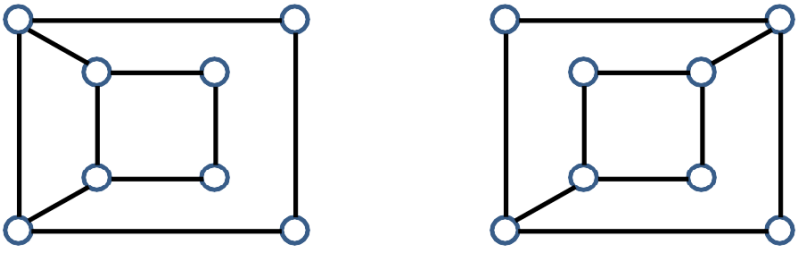
不同构判断

How to show the following are not isomorphic ?

度数分布判断

左图：$(2,2,3,3,4,4)$

右图：$(2,2,2,4,4,4)$

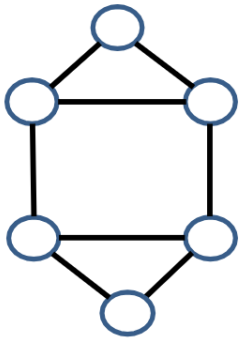


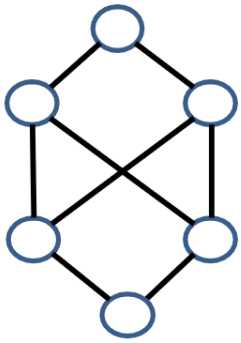
图直径（最长的最短路径）判断

设所有边长为 1，左图直径为 4，右图直径为 3.

法二：回路判断

左图四节点回路有 3 个，右图只有 2 个.





特殊类型判断

左图：非二分图

右图：二分图

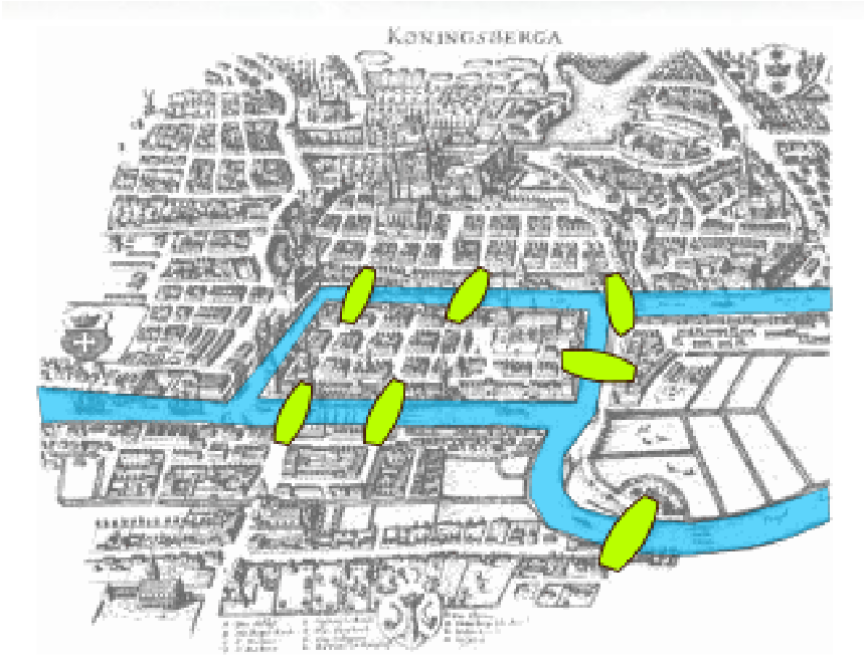
法二：回路判断

左图有三节点回路，右图没有.

9.9 Euler Paths and Circuits

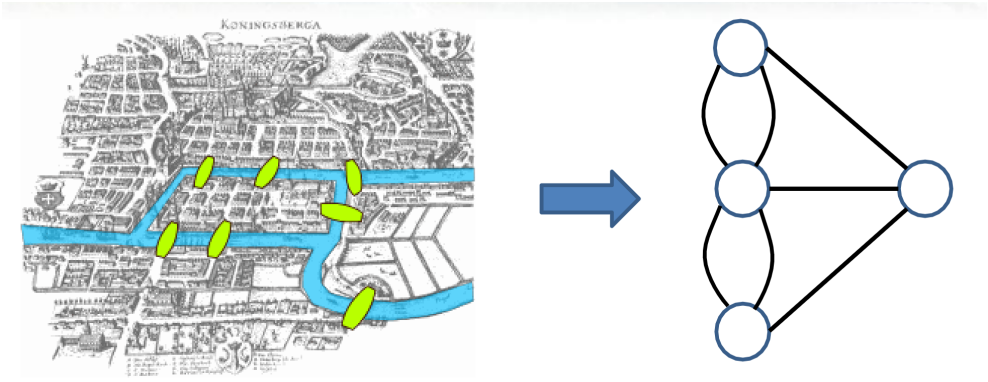
> 本节讨论七桥问题.

七桥问题



这是 18 世纪科尼斯堡的城市地图. 河流将城市分为四部分，两座岛和两块陆地. 它们由七座桥以如图所示方式连接. 问：能否从某地出发，遍历每座桥 **exactly once**，然后回到原点（一笔画问题）？

欧拉首先将问题进行数学抽象：



> 圆是陆地或岛，线是桥.

然后问题转化为：

Find a circuit that travels each edge exactly once.

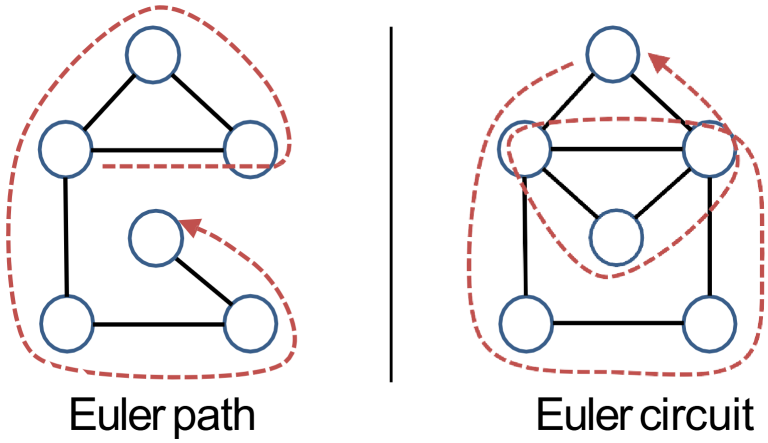
> Euler shows that there is NO such circuit.

欧拉路径与欧拉回路

> Euler Paths and Circuits

An *Euler path* in a graph is a path that contains each edge exactly once. If such a path is also a circuit, it is called an *Euler circuit*.

> 欧拉路径通俗地说就是一笔画，欧拉回路是回到原点的特殊情况.

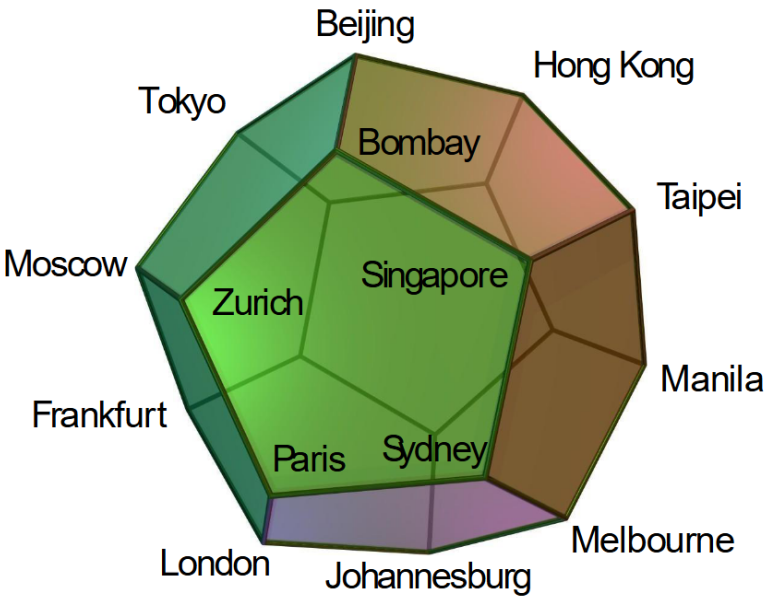


Theorem: A connected graph G has an Euler circuit $\Leftrightarrow$ each vertex of G has even degree.

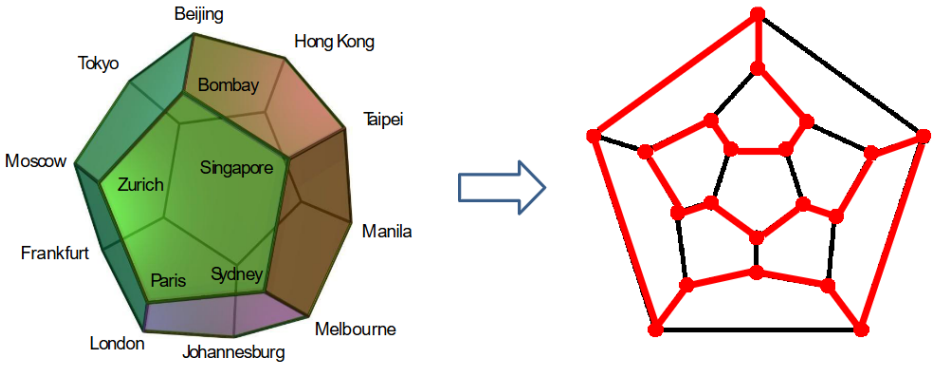
> 首先，如果一个连通图具有欧拉回路，那么可以从图中任意一个顶点出发形成一个欧拉回路.
>
> 其次，我们任选一个顶点，如果从它出发能形成欧拉回路，它的 $degree$ 必须为偶数（否则回来了还得再出去，无法以回到自身结束）.
>
> 综上，所有 $vertex$ 的 $degree$ 均为偶数.

9.10 Hamilton Paths and Circuits

> 与欧拉路径相对，汉密尔顿路径强调对 $vertex$ 而非 $edge$ 的单次遍历.



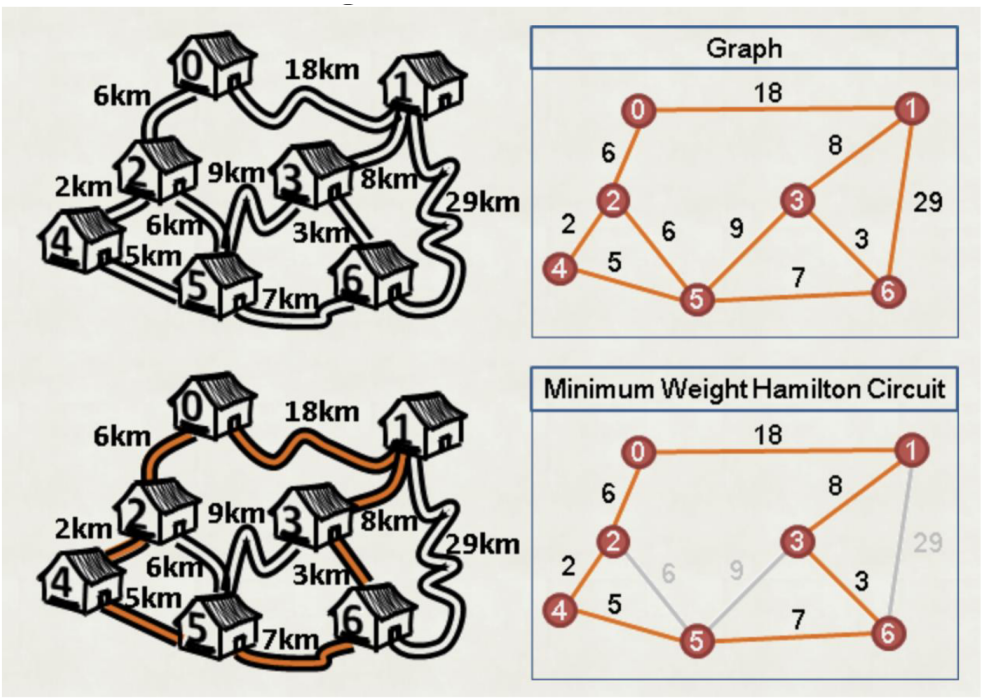
The above is a regular dodecahedron (12-faced) with each vertex labeled with the name of a city. Can we find a circuit (travelling along the edges) so that each city is visited exactly once?



The right graph is isomorphic to the dodecahedron, and it shows a possible way (in red) totravel.

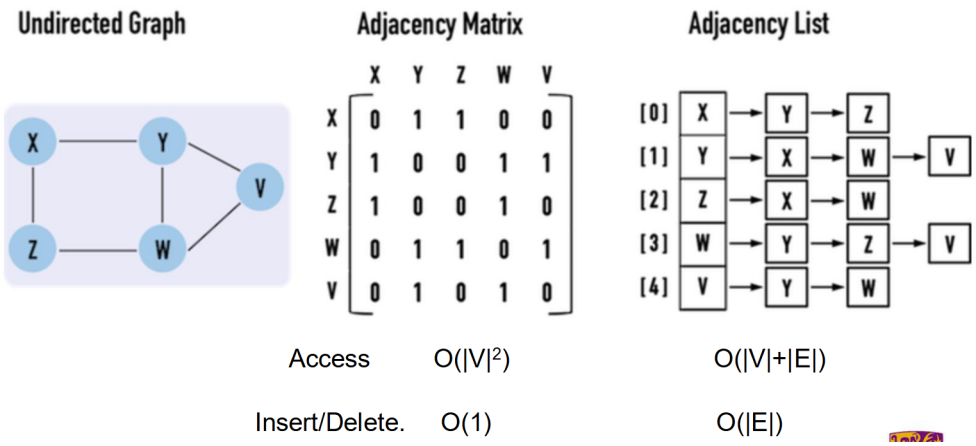
Example

> Traveling Salesman Person



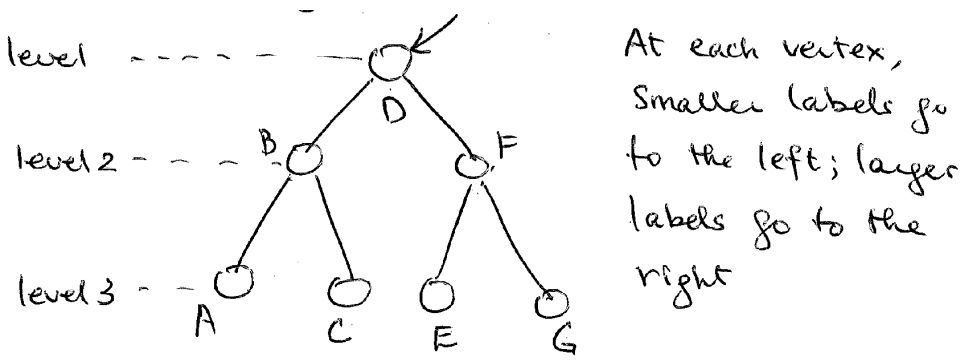
9.11 Graph Representation

> 图的几种表示法. 数据结构课也有介绍.



9.12 Tree Structure

Binary Tree



> At each level, half of the search space is eliminated.
>
> Worst case complexity to locate a file $= no.\ of\ levels = 3$
>
> Leafs of the above tree: A, C, E, G

9.13 Graph Search

A *graph search* is a systematic procedure for visiting *all vertices* of a graph.

Fundamental to many graph algorithms, e.g.,

- Dijkstra's single source shortest paths algorithm,
- Prim's minimum spanning tree algorithm,
- Topological sort,
- ...

Efficient if it visits *all vertices* and edges in *linear* time.

- Simple or complex - all in $O(|V| + |E|)$.

Two efficient methods

- Breadth-First Search (BFS) - queue FIFO
- Depth-First Search (DFS) - stack FILO

Breadth-First Search

Input:

Graph $G = (V, E)$, either directed or undirected, and *source vertex* $s \in V$.

Output:

$d[u] =$ smallest distance from s to u, for all $u \in V$.

$\pi[u] =$ the *predecessor* of u.

> $(\pi[u], u)$ is the last edge on shortest path from s to u.
>
> Set of edges $\{(\pi[u], u) : u \neq s\}$ forms a tree.

Idea: Send a wave out from s.

① First visits all vertices 1 edge from s.

② From there, visits all vertices 2 edges from s.

③ Explore all vertices at distance k from s before exploring all vertices at distance $k + 1$.

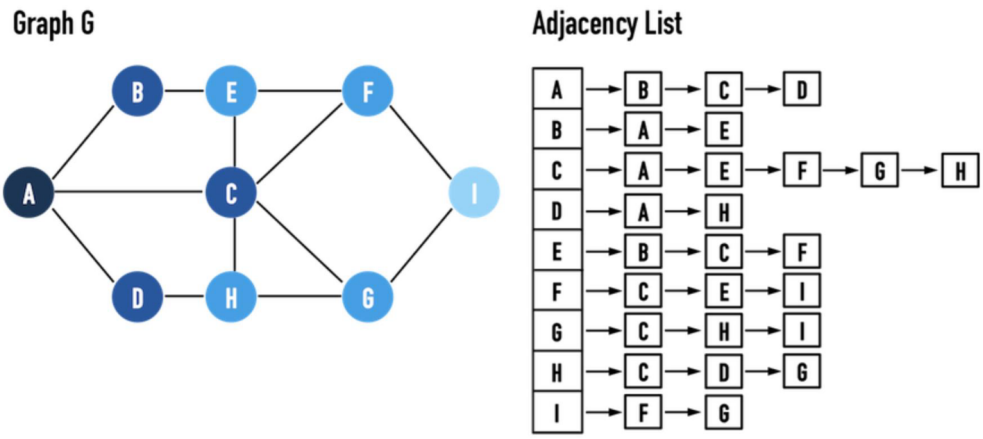
> 按照距离层层遍历. 遍历完同层节点才能遍历更深一层.

Implementation:

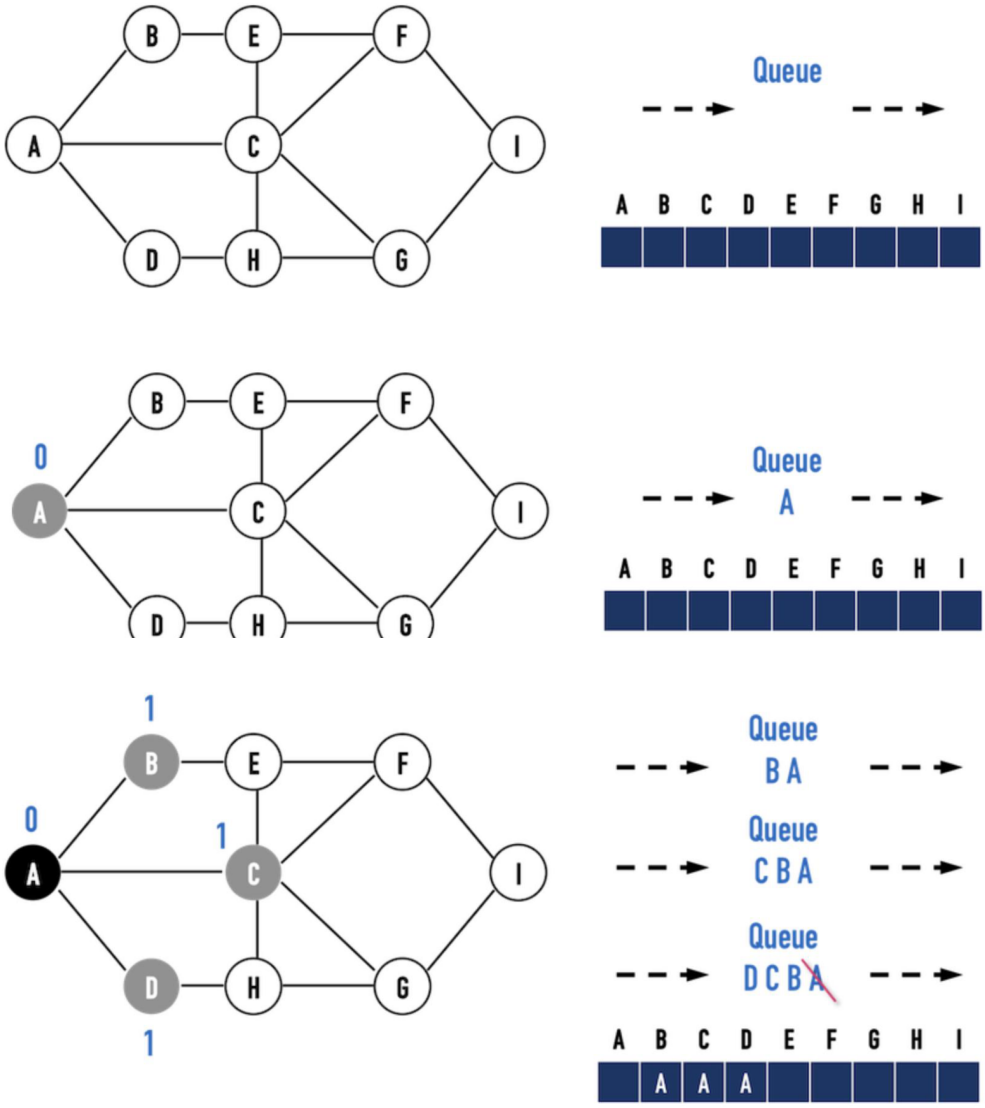
Use FIFO queue Q to maintain wavefront.

$v \in Q$ if and only if v has been discovered but not explored yet.

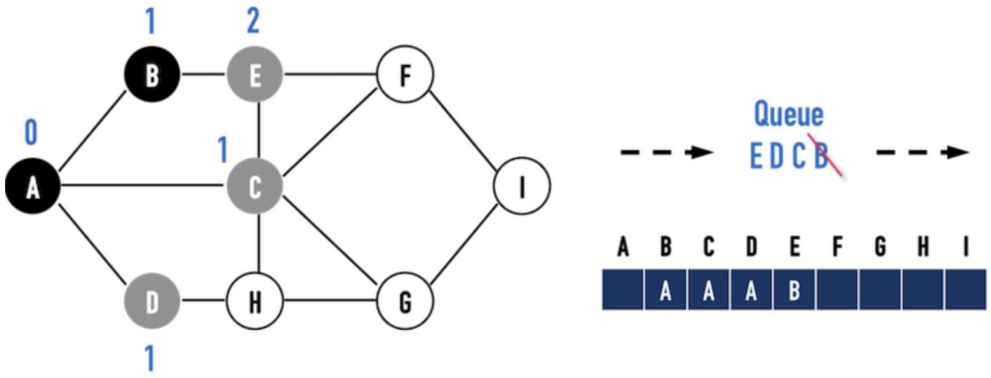
Traversal Example

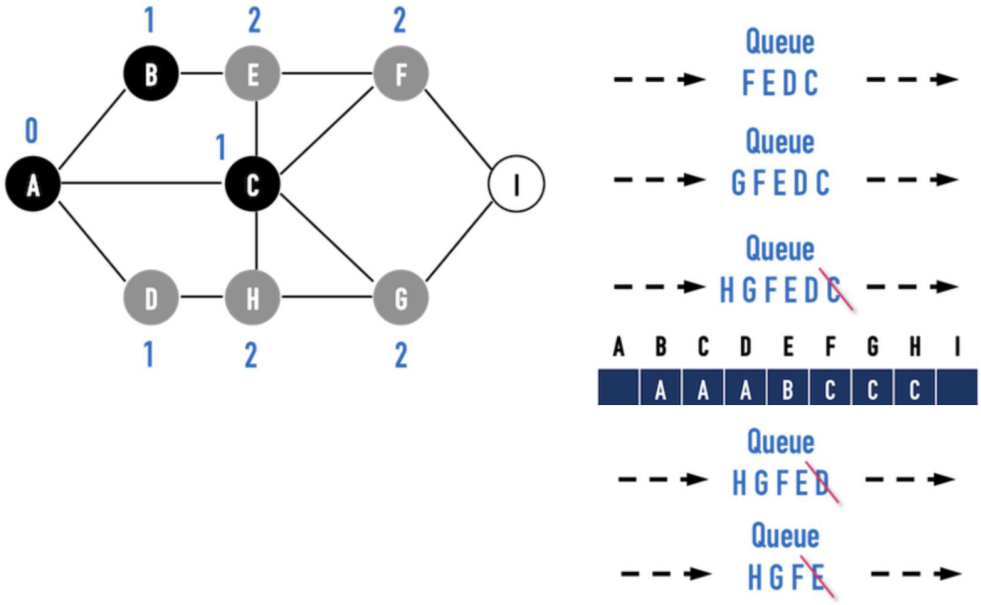


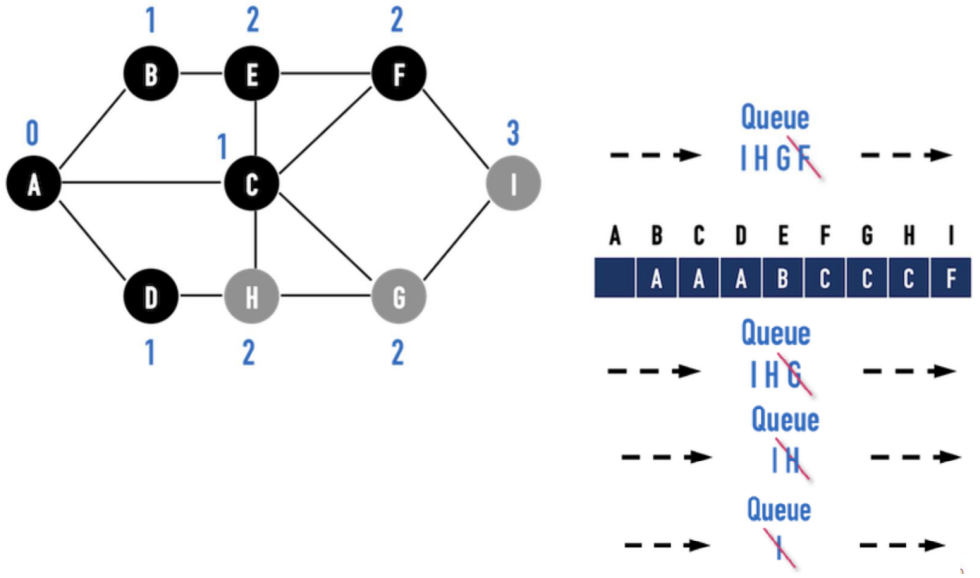
source: A

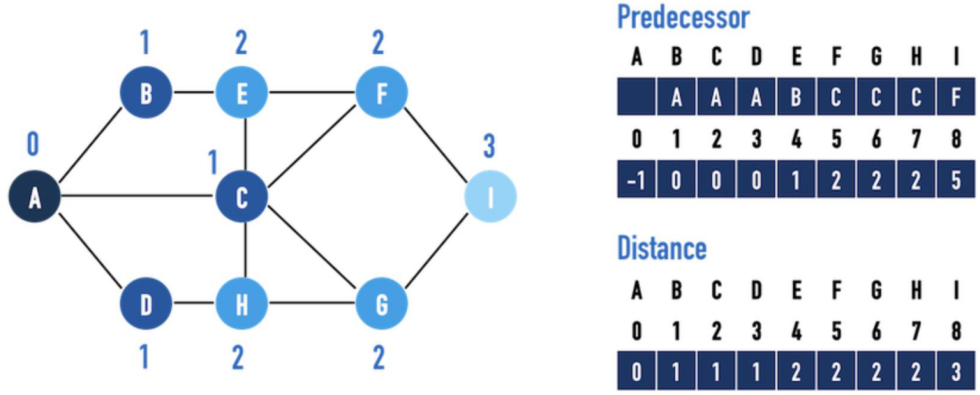


A 在探索完所有子节点并标记进它们的先驱向量后，就可以出队.









应用：李氏迷宫路由算法

> Lee's Maze Router

Lee's Maze Router 是一种经典的网格搜索算法，主要用于解决电路设计中的**布线问题**. 该算法可以在一个二维网格中，找到从起点到终点的最短路径，并且可以保证找到的路径是最优的（若存在解）. 它是基于**广度优先搜索（BFS, Breadth-First Search）**的算法.

1. **背景与应用场景**

Lee's Maze Router 最初用于**集成电路设计**中的布线问题，例如在 PCB（印刷电路板）或芯片内部连通不同节点时，找到一条没有冲突的路径. 这种问题可以抽象为一个网格图，其中：

- **起点**表示信号的起始点.
- **终点**表示信号的目标点.
- **障碍物**表示无法通过的区域.
- **路径**是需要在起点和终点之间找到的线路.

除此之外，Lee's Maze Router 也可以应用于**迷宫求解问题**或**机器人路径规划**.

2. **算法思想**

Lee's Maze Router 是基于广度优先搜索（BFS）的逐步扩展算法，核心思想是：

1. **网格建模**：
   - 将问题区域离散化成一个二维网格.
   - 每个网格单元为一个节点，网格中的障碍物标记为不可通过.
2. **波形传播**：
   - 从起点开始，向四周扩展（上下左右四个方向），逐层标记到达的网格单元的步数（或距离）.
   - 这种扩展类似于水波的传播，因此称为**波形传播**.
3. **路径回溯**：
   - 当终点被标记时，说明找到了目标点.
   - 根据标记的步数，从终点沿着最短路径回溯到起点，形成路径.
4. **处理无解情况**：
   - 如果波形扩展覆盖了整个网格仍未能到达终点，说明路径不存在.

3. **算法步骤**

输入：

- 网格 `grid`，其中标记了起点、终点和障碍物.
- 起点坐标 `start`，终点坐标 `end`.

输出：

- 如果路径存在，则返回从起点到终点的路径.
- 如果路径不存在，则返回“无解”.

具体步骤：

1. **初始化**：
   - 创建一个二维数组 `distance`，初始化为无穷大（表示未访问）.
   - 将起点的 `distance[start]` 设置为 `0`，并将起点加入一个队列 `queue`.
2. **波形传播**（BFS）：
   - 从队列中取出一个网格单元 `current`，检查其四个相邻单元：
     - 如果相邻单元是终点，则停止搜索.
     - 如果相邻单元未被访问且不是障碍物，则标记其距离为 `distance[current] + 1`，并将其加入队列.
3. **路径回溯**：
   - 如果终点被标记，则从终点开始，沿着距离逐步减少的方向回溯到起点.
   - 将路径保存为一个列表.
4. **无解情况**：

- 如果队列为空且未能到达终点，则返回“无解”.

4. **复杂度分析**

- **时间复杂度**：
  - 在最坏情况下，算法会访问网格中的每个单元一次，复杂度为 **O(V + E)**，其中 `V` 是网格中的节点数（`rows × cols`），`E` 是边数（每个节点最多有 4 条边）.
- **空间复杂度**：
  - 主要用于存储距离矩阵和 BFS 队列，复杂度为 **O(V)**.

5. **优缺点**

优点：

1. **最优性**：Lee's Maze Router 总能找到从起点到终点的最短路径（如果存在）.
2. **简单性**：算法基于 BFS，容易实现和理解.
3. **鲁棒性**：适用于任意形状的网格和障碍物分布.

缺点：

1. **效率问题**：对于大规模网格，波形传播可能非常慢，尤其是密集的障碍物区域.
2. **路径选择**：如果存在多条等长路径，算法无法根据其他准则（如转弯次数）选择最优路径.
3. **内存占用**：需要额外存储距离矩阵，内存消耗较大.

Depth-First Search

Input: $G = (V, E)$, directed or undirected.

Output: $2\ timestamps$ on each vertex

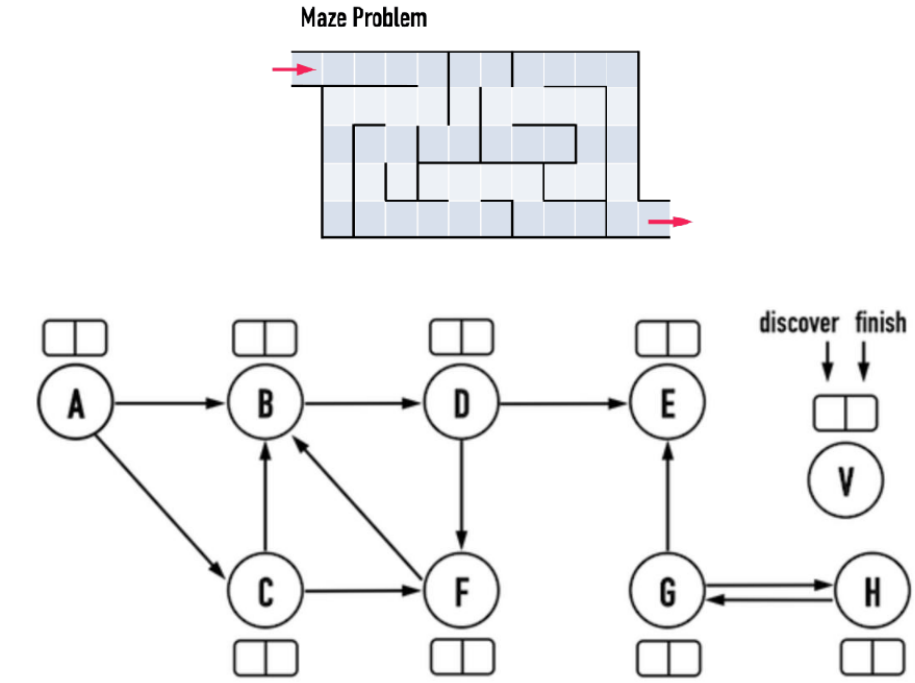
$d[v] = discovery\ time.$

$f[v] = finishing\ time.$

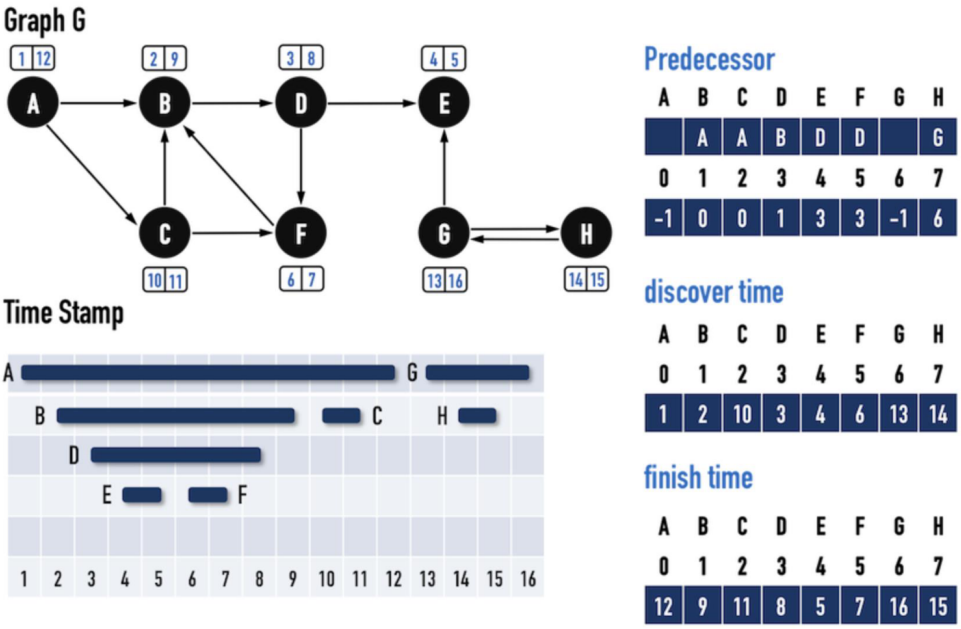
Idea: like $wandering\ in\ a\ labyrinth$ with a string and can of paint. As soon as we discover a vertex, explore from it.

> Unlike BFS, which puts a vertex on a queue so that we explore from it later.

Traversal Example



source: multiple (此处应该按字母表升序，从 A 开始，然后是 G)



BFS + DFS 应用：苏库普迷宫路由算法

> Soukup's Maze Router

Soukup's Maze Router 是一种改进的网格路由算法，它是在 **Lee's Maze Router** 基础上优化而来的，用于解决网格中的最短路径搜索问题，尤其是在电路设计中的布线问题. Soukup 的主要目标是通过减少搜索的路径数量，提高效率. 但找到的路径可能不是全局最优（采用了局部贪心，可能不是全局最短路径）.

1. **背景与应用场景**

Soukup's Maze Router 同样适用于 **二维网格布线问题**（如印刷电路板或芯片布线），其主要优化方向是避免 Lee's Maze Router 中的**盲目波形扩展**问题.

在 Lee's Maze Router 中，波形扩展是向四个方向（上下左右）同时进行的，这种方法虽然简单，但可能会在网格中产生不必要的搜索，尤其是当起点和终点相隔较远时。这会导致算法效率较低，特别是在存在较多障碍物的情况下.

Soukup 的改进通过引入一种**方向性优先的搜索策略**，减少了不必要的扩展，从而提高了算法效率.

2. **Soukup 算法的核心思想**
3. **方向优先**：在波形扩展时，优先朝着**更接近终点的方向**扩展，而不是像 Lee's 算法那样均匀地向四个方向扩展.
    - 例如，如果终点在右下方，则优先向右和下扩展.
4. **避免不必要的搜索**：通过引导搜索朝着目标方向前进，算法减少了与目标无关的区域的搜索，从而提高了效率.

3. **算法步骤**

输入：

- 一个二维网格 `grid`，其中：
    - `0` 表示空白区域.
    - `1` 表示障碍物.
- 起点 `start` 和终点 `end`.

输出：

- 从起点到终点的最短路径（如果存在）.
- 如果路径不存在，则返回"无解".

算法具体步骤：

1. **初始化**：
    - 创建一个队列 `queue`，并将起点 `start` 加入队列.
    - 创建一个二维数组 `distance`，初始化为无穷大，用于记录各网格单元到起点的距离.
    - 将 `distance[start]` 设置为 0.
2. **方向优先的波形传播**：
    - 设定四个搜索方向（上下左右），并根据终点的位置重新排序，使得优先搜索更接近终点的方向（例如，如果终点在右下方，则优先搜索右和下）.
    - 从队列中取出一个节点 `current`，尝试向四个方向扩展：
        - 如果相邻单元是空白区域且未访问过，则更新其距离为 `distance[current] + 1`，并将其加入队列.

        - 如果相邻单元是终点，则停止搜索.
3. **路径回溯**：
    - 如果终点被访问，则从终点开始，沿着距离逐步减少的方向回溯到起点，得到路径.
4. **无解情况**：
    - 如果终点没有被访问，则返回“无解”.

4. **复杂度分析**

时间复杂度：

- 与 Lee’s Maze Router 类似，最坏情况下仍需要遍历整个网格（左上到右下），时间复杂度为 **O(V + E)**，其中 `V` 是节点数（`rows × cols`），`E` 是边数（每个节点最多有 4 条边）.

空间复杂度：

- 存储距离矩阵和队列的空间复杂度为 **O(V)**.

不过，由于 Soukup 算法优先搜索更接近终点的方向，因此在实际运行中，搜索的区域往往比 Lee’s 算法小，效率更高.

5. **优缺点**

优点：

1. **效率更高**：通过方向性优先搜索，减少了无关区域的搜索，提高了搜索速度.
2. **简单性**：相比于更复杂的启发式算法（如 A* 算法），Soukup 的实现仍然相对简单.

缺点：

1. **没有全局视野**：Soukup 的方向性优先是基于终点的直线距离，但并不考虑障碍物的分布，因此在某些情况下可能仍然需要大量回溯.
2. **适用性有限**：虽然在稠密障碍物的网格中有一定提升，但在非常复杂的网格中效率提升有限.
3. **无法保证全局最优解**：方向性优先的策略可以减少不必要的搜索，但它忽略了全局最优路径可能需要绕远路的情况（即路径长但可能权重低；也可能方向性优先但路上的障碍物导致绕得更远）.

9.14 强连通图

A graph is said to be **strongly connected** if every vertex is reachable from every other vertex. The **strongly connected components** of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected.

强连通：任意两点之间都可以双向到达的**有向图**.

> 针对有向图定义的概念，对无向图没有意义（无向图所有边都是双向）.

强连通分量

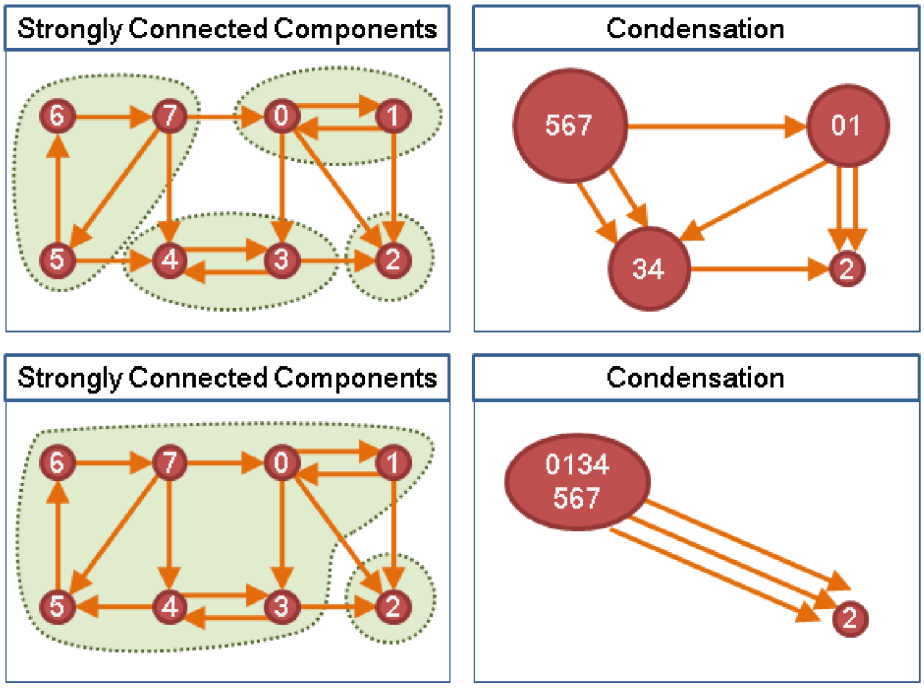
> Strongly Connected Component, SCC

对于任意一个有向图（即使它不是完全强连通的），可以将其划分为若干个子图，每个子图都是强连通的. 这些强连通的子图，称为强连通分量.

存在这类情况：一个有向图有多个部分（可能“孤立”或“联系不紧密”），每个部分内部可能强连通，但它们之间不是强连通.

强连通分量的性质：

- 强连通分量互不相交，即任何两个强连通分量之间没有重叠顶点.
- 每个强连通分量内部是强连通的，但分量之间可能没有路径相连.
- 强连通分量可以将整个图分解成若干个不相交的子图.



9.15 割点

> Articulation Vertex

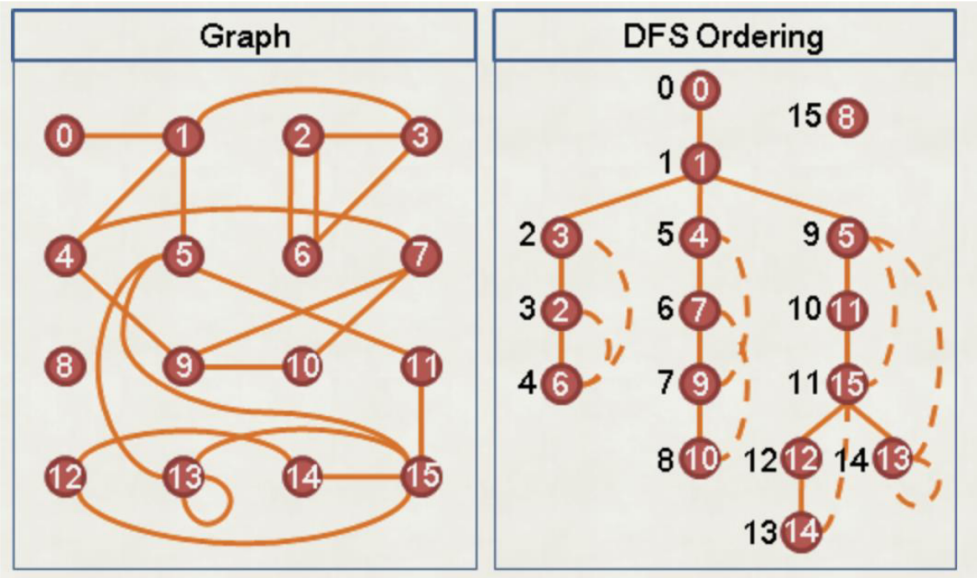
给定无向连通图 $G=(V,E)$，如果删除某个顶点 $v\in V$ 以及与该顶点相连的所有边后，图的其余部分不再连通（即图的连通分量数增加），那么顶点 v 就是一个 Articulation Vertex.

> 如果一个图是连通的单个循环（如环形结构），那么它没有割点

寻找方式：DFS

根节点：如果根节点有两个或更多子树，它是割点.

非根节点：若自己的某个子节点无法通过其他路径回到它的祖先节点（即子节点的回退时间大于等于当前节点的访问时间），当前节点是割点.



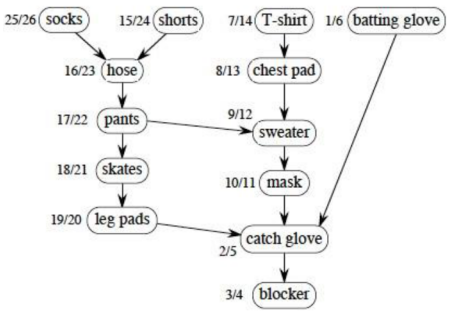
9.16 拓扑排序

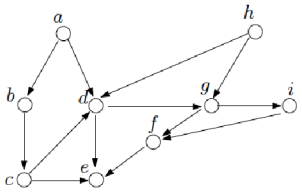
> Topological Sort
>
> 了解定义即可.

定义

拓扑排序（Topological Sort）是一种针对**有向无环图（Directed Acyclic Graph, DAG）**的顶点排序方法. 它将图中的所有顶点排列成一个线性序列，满足以下条件：

如果图中存在一条从顶点 `u` 到顶点 `v` 的有向边（`u → v`），那么在拓扑排序中，顶点 `u` 必须排在顶点 `v` 的前面.



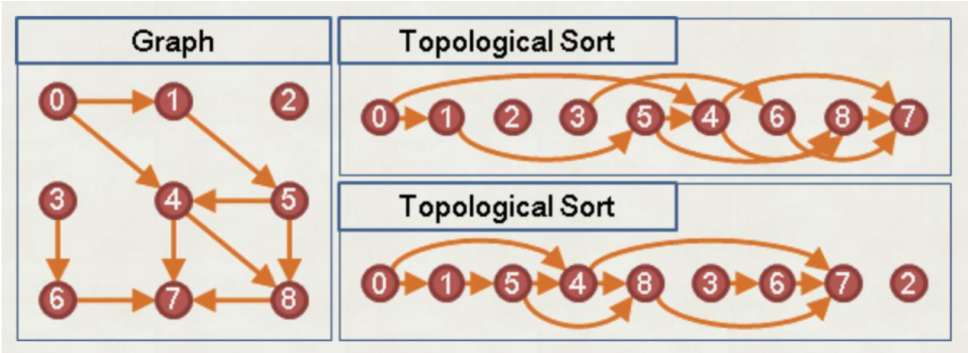


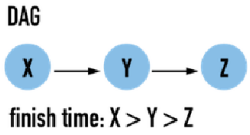
The following are two possible topological orders:

- $h, a, b, c, d, g, i, f, e$.
- $a, h, b, c, d, g, i, f, e$.

An ordering that is not a topological order:

- $a, h, d, b, c, g, i, f, e$ (because of edge (c, d)).





> 注意箭头方向，总是从左往右

特点

1. 拓扑排序只能用于**有向无环图（DAG）**，如果图中存在环，无法生成.
2. 一个有向无环图可能有多个不同的拓扑排序结果.

应用

1. **任务调度**：当任务之间存在前置依赖（例如任务 `A` 必须在任务 `B` 之前完成），拓扑排序可以确定任务的执行顺序.
2. **编译器依赖**：在编译代码时，拓扑排序可以用来解析模块或类之间的依赖关系.
3. **课程表问题**：如果课程之间有先修课程的约束，可以用拓扑排序来决定课程的学习顺序.
4. **电路设计**：在设计电路时，拓扑排序可以帮助分析信号的传播顺序.

9.17 Shortest Path Problem

给定图，起点和终点，可能有多条路径.

how to find the shortest path on a large graph?

9.18 Dijkstra's Algorithm

> 广度优先引入贪心思想的加权变体.

Initialize:

(A) maintain a table of cost $c(v)$, where starting vertex has cost $c(v_0) = 0$, others have cost $c(v_i) = \infty,\ i \neq 0$;

(B) maintain a set of visited vertices $S = \emptyset$.

Iteration:

Choose the unvisited vertex with minimum cost - denote it by v_{min} and set $S = S \cup \{v_{min}\}$.

> 注意这里是按未探索节点的距离决定下一个探索目标，小的优先（贪心）

For every neighbors of v_{min}, set

$$c(v_i) = min\{c(v_i),\ c(v_{min}) + w(v_{min}, v_i)\},\ \forall\ v_i \in N_{v_{min}}$$

> 更小就更新，不更小就跳过.
>
> 这一步就是松弛操作，下面会提到.

End when all vertices are visited, i.e. $S = V$.

Return:

a table of cost $c(v)$ with the shortest path distance.

> 有时候还会保存一个前驱向量，通过该向量可找到最短距离对应的路径.

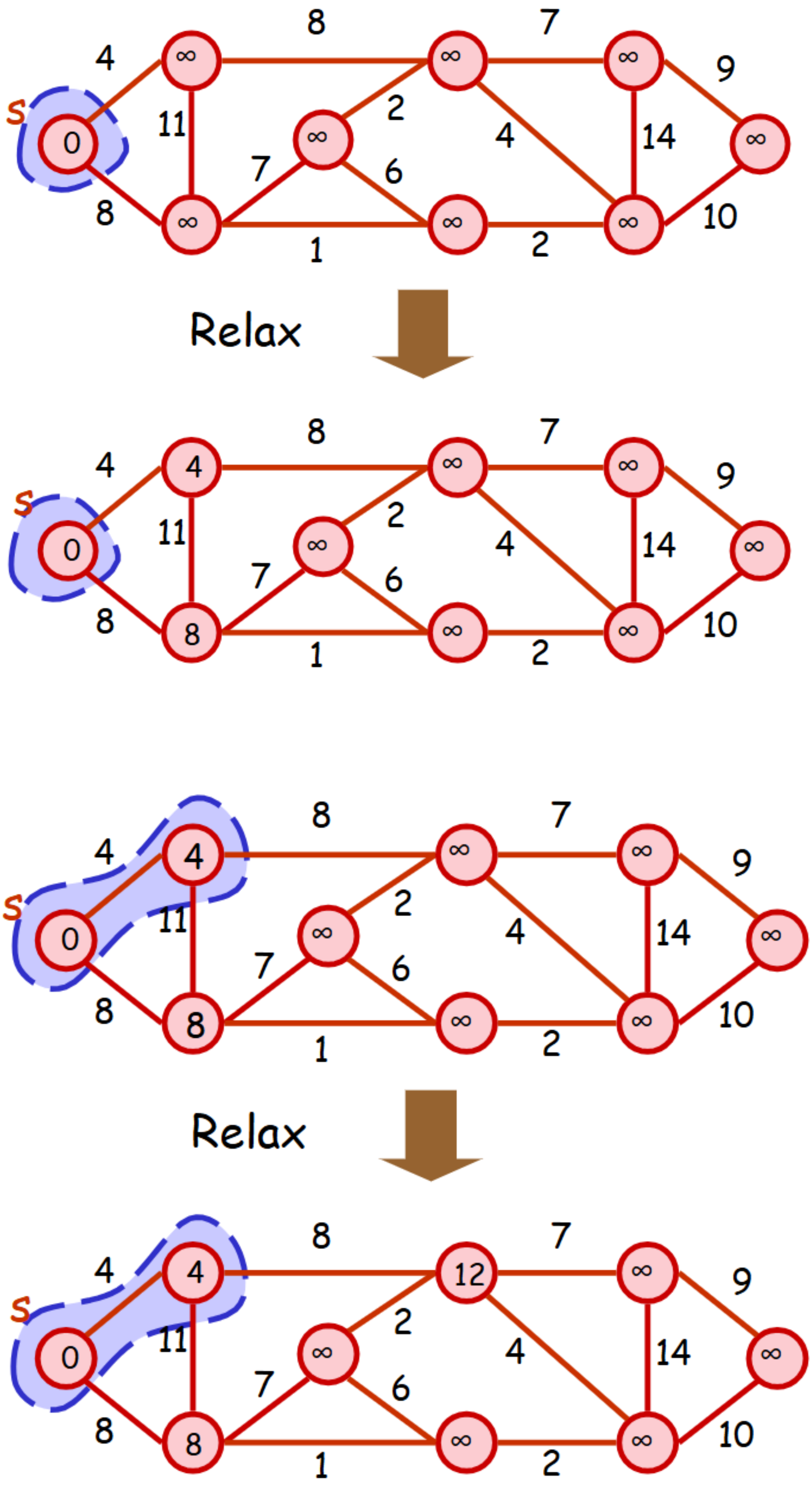
9.19 Relax

> 松弛操作.

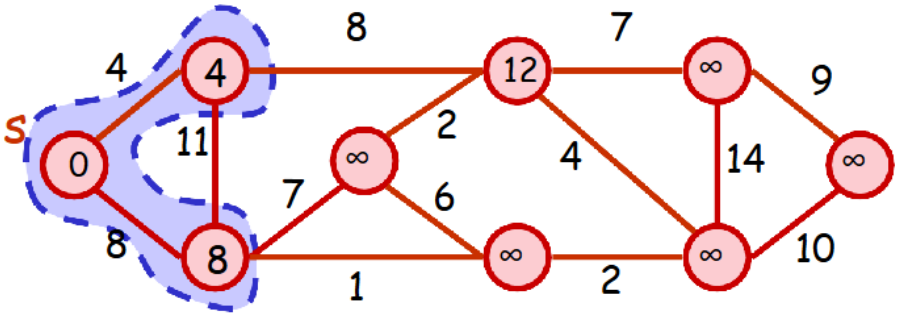
Relax is a common operation that is used in the three algorithms.

松弛的基本思想是：如果经过某条边可以找到更短路径，就更新路径估计值.

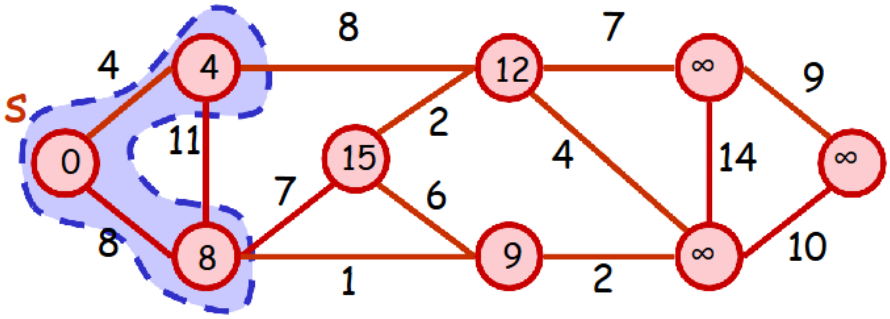
Example

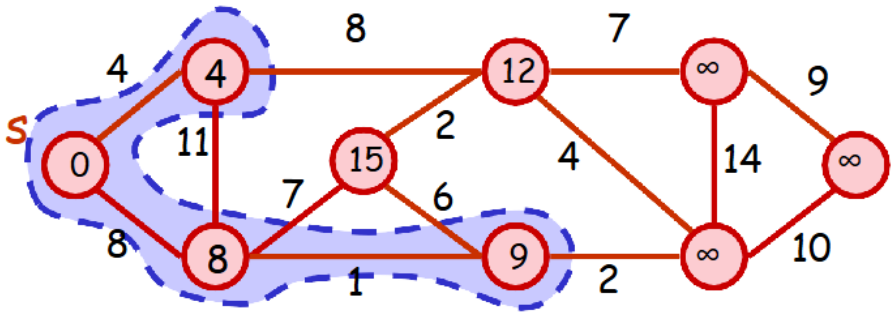


s
Relax
s
s
Relax
s

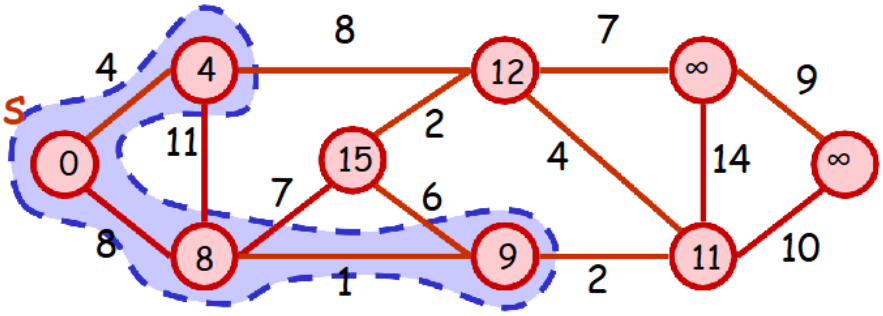

S
0
4
4
8
8
11
7
∞
2
6
12
∞
1
2
4
∞
14
∞
7
9
∞
10

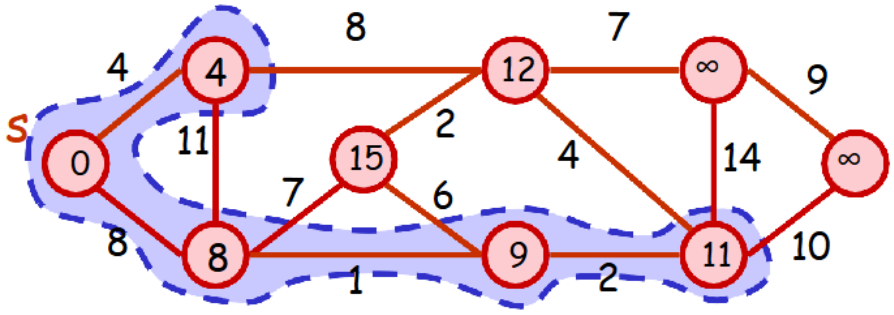
Relax


S
0
4
4
8
8
11
7
15
2
6
12
9
1
2
4
∞
14
∞
7
9
∞
10

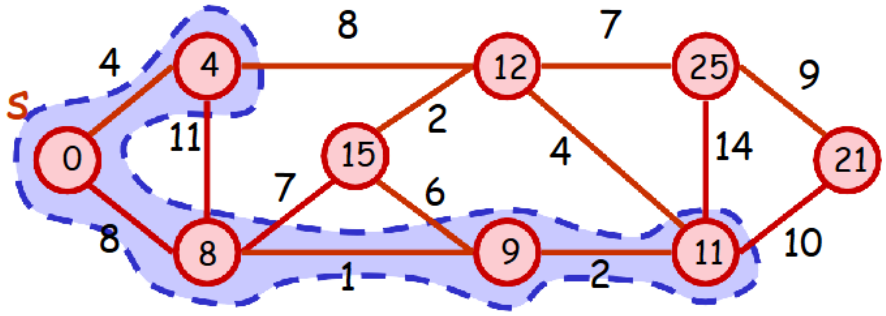

S
0
4
4
8
8
11
7
15
2
6
12
9
1
2
4
∞
14
∞
7
9
∞
10

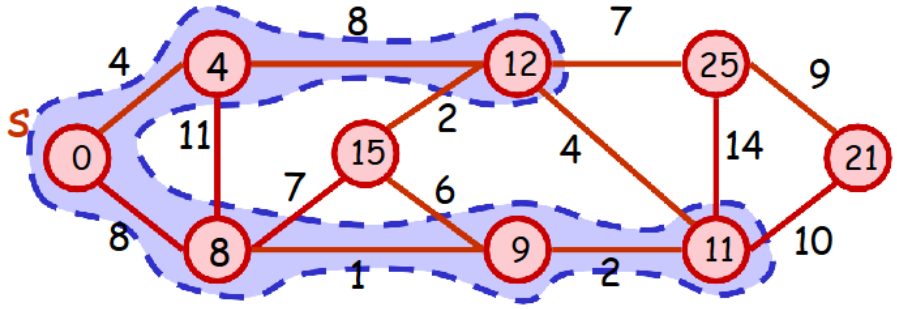
Relax


S
0
4
4
8
8
11
7
15
2
6
12
9
1
2
4
∞
14
11
7
9
∞
10

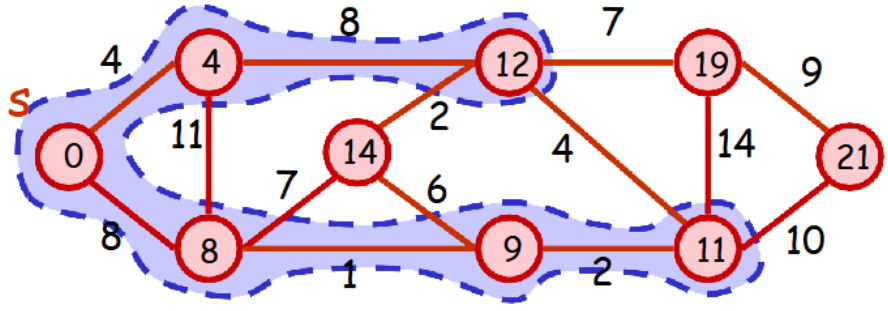

S
0
4
4
8
12
∞
15
8
9
11
∞
8
7
4
11
2
7
6
4
14
9
10
1
2

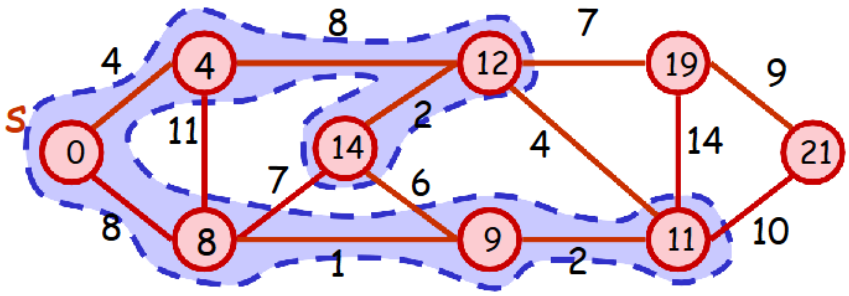
Relax


S
0
4
4
8
12
25
15
8
9
11
21
8
7
4
11
2
7
6
4
14
9
10
1
2

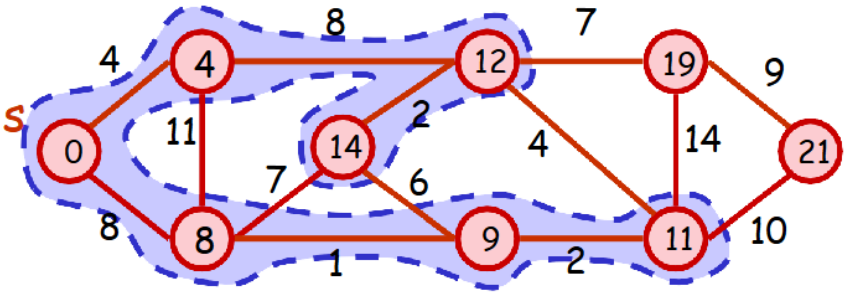

S
0
4
4
8
12
25
15
8
9
11
21
8
7
4
11
2
7
6
4
14
9
10
1
2

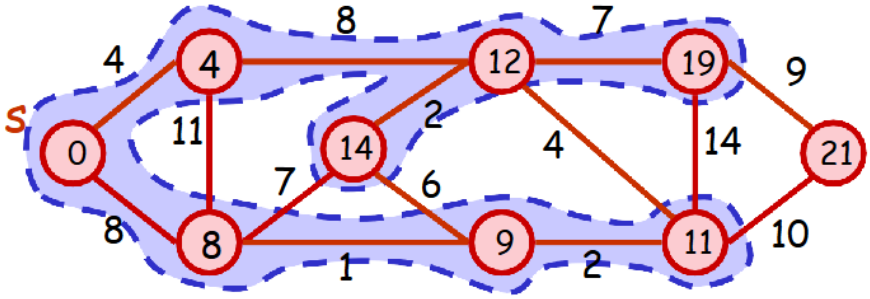
Relax


S
0
4
4
8
12
19
14
8
9
11
21
8
7
4
11
2
7
6
4
14
9
10
1
2

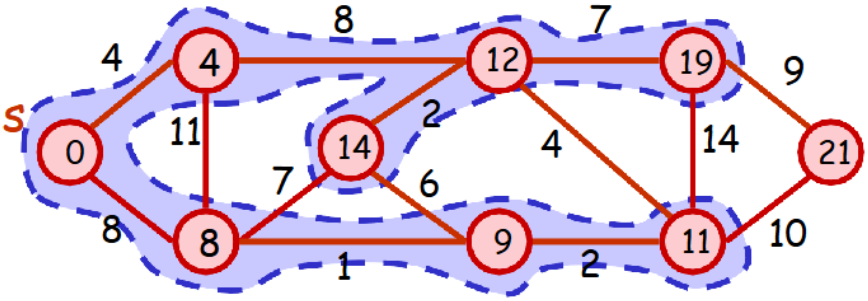

s
0
4
4
8
12
19
21
14
8
9
11
11
7
2
4
14
7
6
8
1
2
10
9

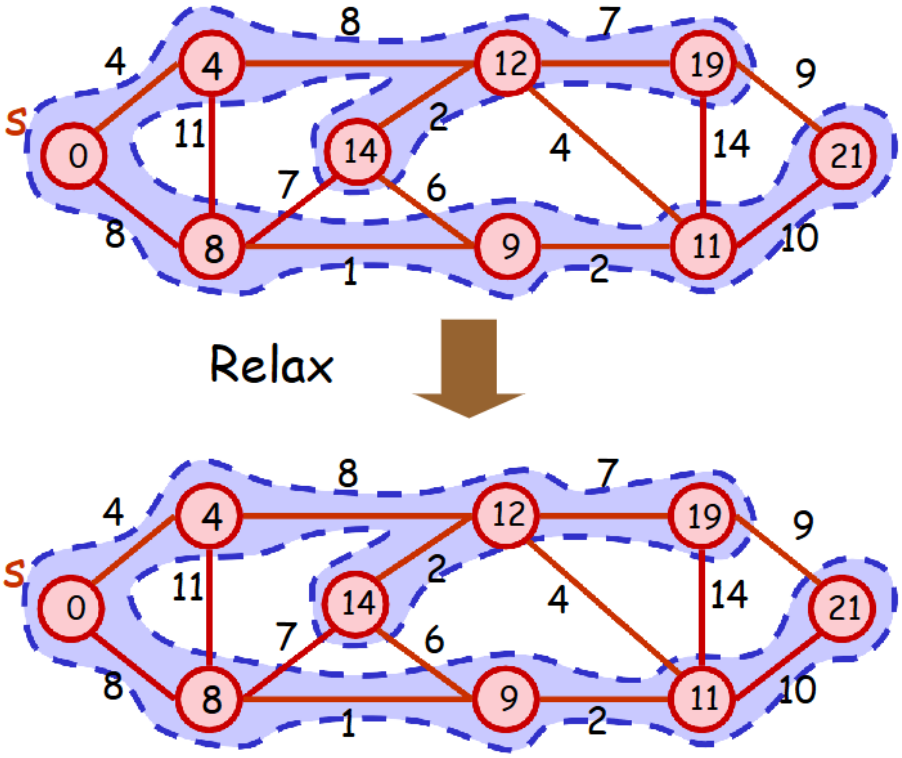
Relax


s
0
4
4
8
12
19
21
14
8
9
11
11
7
2
4
14
7
6
8
1
2
10
9


s
0
4
4
8
12
19
21
14
8
9
11
11
7
2
4
14
7
6
8
1
2
10
9

Relax


s
0
4
4
8
12
19
21
14
8
9
11
11
7
2
4
14
7
6
8
1
2
10
9



9.20 Minimum Spanning Trees

To interconnect a set of n nodes, we can use an arrangement of $n-1$ wires. (many possible arrangements)

Which arrangement uses the least length of wires?

> 找到能将图中所有节点连在一棵树上，又能保证权重和最小的树.

定义

Given an *undirected* graph $G=(V,E)$ with weights on the edges, a *minimum spanning tree* (MST) of G is *an acyclic subset* $T \subseteq E$ that connects all nodes in V and whose total weight

$$w(T)=\sum_{(u,v)\in T} w(u,v)$$

is *minimum*.

特性

$|T|=|V|-1$

> n nodes, $n-1$ wires.

MST might not be unique.

> 可能有多解. 无论是 Prim 算法还是 Kruskal 算法，都有可能碰到多个最小值的情况，但基本算法只能处理其中一种.

9.21 Prim's Algorithm

> 只能处理连通图. 如果遇到非连通图，Prim 算法只能生成一个连通分量的生成树，无法覆盖整个图.
>
> 了解算法原理即可，具体代码实现在数据结构课详细学习.

切分定理

在一个连通的加权无向图中，任意将顶点集合划分为两个不相交的部分（称为一个切分）. 如果一条边的权重是跨越切分的所有边中最小的，那么这条边必然属于某个最小生成树.

> 很好理解：切分的两部分之间必然要有边来连接；而对于最小生成树，自然由所有可以连接这两部分的边当中权重最小的一条来贡献（如果有多条，就有多解）.

Prim 算法

Prim's Algorithm（普里姆算法）是一种经典的**最小生成树（MST）**算法，用于从一个加权无向图中找到一棵最小生成树. 最小生成树是连接图中所有顶点的一个子图，它满足：

1. 是一棵树（无环）.
2. 包括图中的所有顶点.
3. 边的权重和最小.

Prim 算法以"逐步扩展树"的方式，通过贪心策略，每次选择权重最小的边，将新的顶点加入生成树，从而构造出最小生成树.

适用场景

- **输入**：一个连通的加权无向图.
- **输出**：一棵最小生成树，包括所有顶点，且边的权重和最小.
- **适用性**：
  - 适用于网络设计问题，例如电网设计、光纤铺设、路由网络等.
  - 图必须是连通的. 如果图不连通，Prim 算法只能生成一个连通分量的生成树.

算法思想

Prim 算法从一个起始顶点开始，逐步选择权重最小的边，将未访问的顶点加入生成树. 其核心是通过贪心策略逐步扩展生成树，确保每一步的选择都局部最优，从而保证全局最优.

> 根据切分定理，切分的两个部分一定要有边来连接，最优选择是能连接两部分的最小权重的那条.
>
> Prim 每次选边都是在运用切分定理，保证全局最优.

算法步骤

假设图中有 `V` 个顶点和 `E` 条边：

初始化：

1. 选择**任意**一个顶点作为起点，将其加入生成树.

2. 创建一个数组 `minCost[]`，记录每个顶点到生成树的最小权重边，初始值为 ∞（无穷大）。对起点顶点设置为 0.
3. 创建一个布尔数组 `inMST[]`，记录每个顶点是否已包含在生成树中，初始值为 `False`.

主循环：

1. 从未加入生成树的顶点中，选择 `minCost[]` 值最小的顶点 `u`.
2. 将顶点 `u` 加入生成树，并标记 `inMST[u] = True`.
3. 遍历顶点 `u` 的所有邻接顶点 `v`：
    - 如果顶点 `v` 未加入生成树，并且边 `u → v` 的权重小于 `minCost[v]`，更新 `minCost[v]` 为边 `u → v` 的权重.

    > 这里也是松弛操作.

4. 重复上述步骤，直到所有顶点都被加入生成树.

输出：

- 最小生成树的边集合，以及总权重.

算法复杂度

1. **时间复杂度**：
    - 如果使用邻接矩阵表示图，逐顶点查找最小边的时间复杂度为 **$O(V^2)$**.
    - 如果使用邻接表表示图，结合**优先队列**（最小堆），每次查找和更新最小边的复杂度为 **$O(\log V)$**，总复杂度为 **$O((V + E) \log V)$**.
2. **空间复杂度**：
    - 需要存储 `minCost[]`、`inMST[]`、`parent[]` 等数组，空间复杂度为 **$O(V)$**.

Prim 算法的优缺点

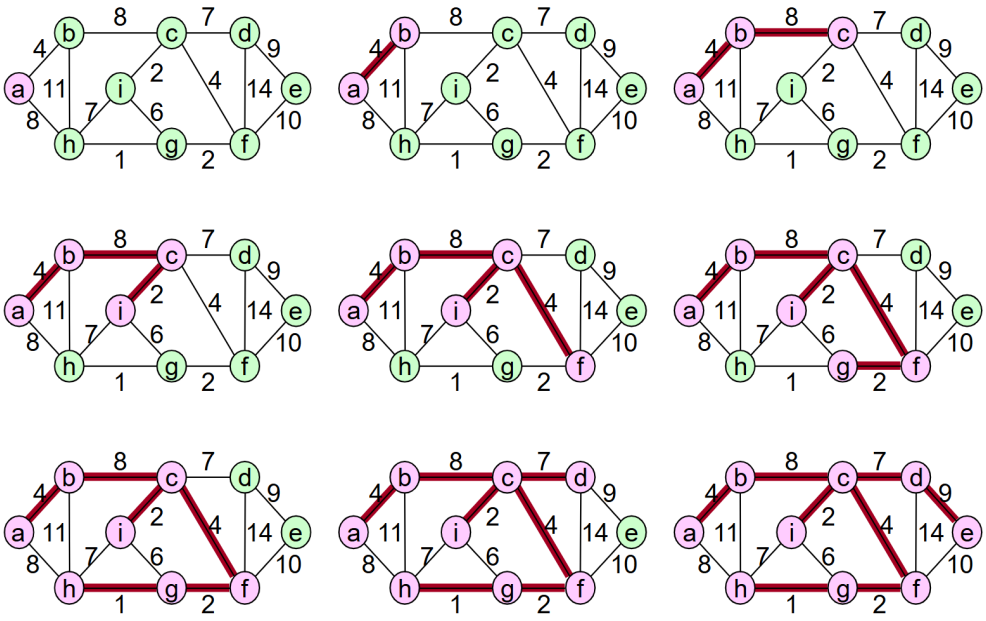
优点：

1. 简单易实现，适合稠密图（边多的图）.
2. 可以与优先队列结合，提升稀疏图（边少的图）上的效率.

缺点：

1. 对于稀疏图，使用邻接矩阵实现的效率较低（复杂度为 $O(V^2)$）.
2. 如果图不连通，Prim 算法只能生成一个连通分量的生成树，无法覆盖整个图.

Example



> 注意第三步选了 c 而没有选 h，错过了一个多解.

9.22 Kruskal's Algorithm

> 可以处理非连通图.

Kruskal 算法是一种经典的贪心算法，用于解决**最小生成树（MST, Minimum Spanning Tree）**问题. 它的核心思想是按边权从小到大排序，逐步将边加入生成树，同时避免形成环. 最终，Kruskal 算法可以生成图的最小生成树（或森林）.

算法思想

Kruskal 算法的工作方式是基于**边的选择**，而不是像 Prim 算法那样基于顶点的扩展。它的步骤如下：

1. **排序**：将所有边按权重从小到大排序.
2. **逐步加入边**：
   - 从权重最小的边开始，依次尝试将边加入生成树.
   - 每次加入一条边时，检查是否会形成环. 如果形成环，则跳过该边.
3. **停止条件**：
   - 当生成树包含 `v - 1` 条边（`v` 为顶点数）时，算法结束，此时已经构造出了最小生成树.

Kruskal 算法的关键点

1. **边的排序**：
   - 按权重从小到大对所有边进行排序，这是 Kruskal 算法的第一步，也是主导其时间复杂度的关键操作.
2. **环的检测**：
   - 使用**并查集（Union-Find 数据结构）**来高效判断加入一条边后是否会形成环.

- 并查集维护每个顶点所属的连通分量，通过判断两条边的两个顶点是否已经在同一连通分量中，来决定是否会形成环.

3. **贪心策略**：
   - 每次都选择当前权重最小的边，确保每一步的选择都是局部最优，最终生成的树是全局最优的（贪心算法的特性）.

算法步骤

假设图有 `V` 个顶点和 `E` 条边，算法的详细步骤如下：

1. **初始化**：
   - 将所有边按权重从小到大排序.
   - 初始化并查集，每个顶点自成一个连通分量.
2. **从最小边开始处理**：
   - 遍历排序后的边集合，对于每条边 `(u, v)`：
     - 使用并查集判断 `u` 和 `v` 是否在同一个连通分量中：
       - 如果不在同一连通分量，则将该边加入生成树，并合并 `u` 和 `v` 所在的连通分量.
       - 如果在同一连通分量，则跳过该边（否则会形成环）.
3. **结束条件**：
   - 当生成树包含了 `V - 1` 条边时，停止（最小生成树的定义是连通且包含 `V - 1` 条边）.
4. **输出结果**：
   - 返回生成树的边集合及其总权重.

复杂度分析

1. **边排序**：
   - 对所有 `E` 条边按权重排序，时间复杂度为 `O(E log E)`.
2. **并查集操作**：
   - 对每条边执行 `Find` 和 `Union` 操作，使用路径压缩和按秩合并的并查集实现方式，时间复杂度接近 `O(α(E))`，其中 `α` 是反阿克曼函数，在实际中几乎是常数.
3. **总复杂度**：
   - 排序主导整个算法，因此时间复杂度为：
     - **O(E log E)**，其中 `E` 是边数.

Kruskal 算法的优缺点

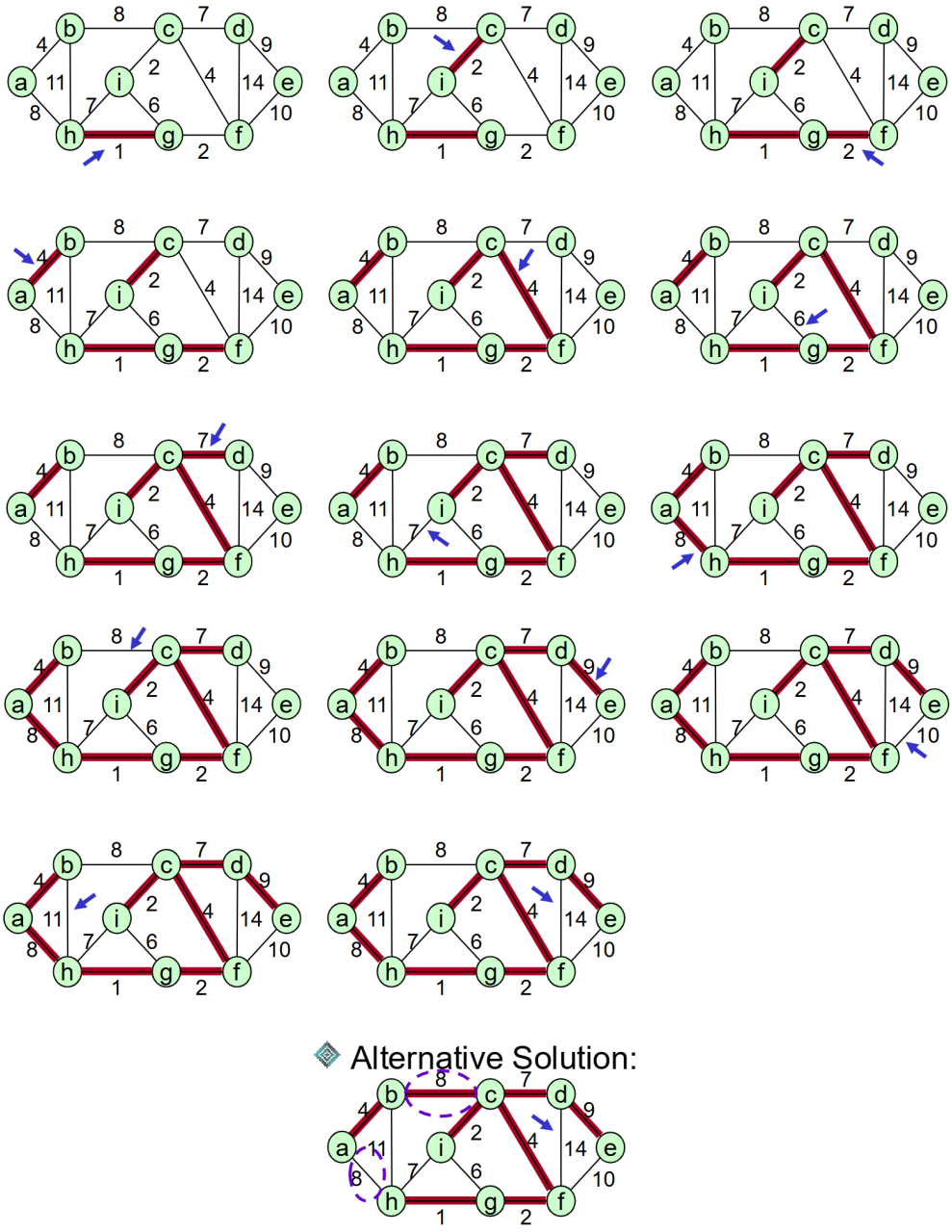
优点

1. **适用于任意图**：
   - Kruskal 算法可以处理连通图和非连通图。如果图是非连通的，它会生成一个**最小生成森林**（每个连通分量的最小生成树）。
2. **稀疏图更高效**：
   - 对于稀疏图（边数远小于顶点数的平方），Kruskal 算法的效率较高，因为它的复杂度主要依赖于边的数量。
3. **实现简单**：
   - Kruskal 算法相对直观，基于边的操作，适合边数较少的图.

缺点

1. **排序成本高**：
    - 初始的边排序操作占据主要时间复杂度，适合边数较少的图.
2. **依赖并查集**：
    - Kruskal 算法需要高效的并查集实现来检测环，否则算法效率会降低.
3. **不适合稠密图**：
    - 对于稠密图（边数接近顶点数的平方），边的数量较多，排序和并查集操作的成本较高，Prim 算法可能更适合.

Example



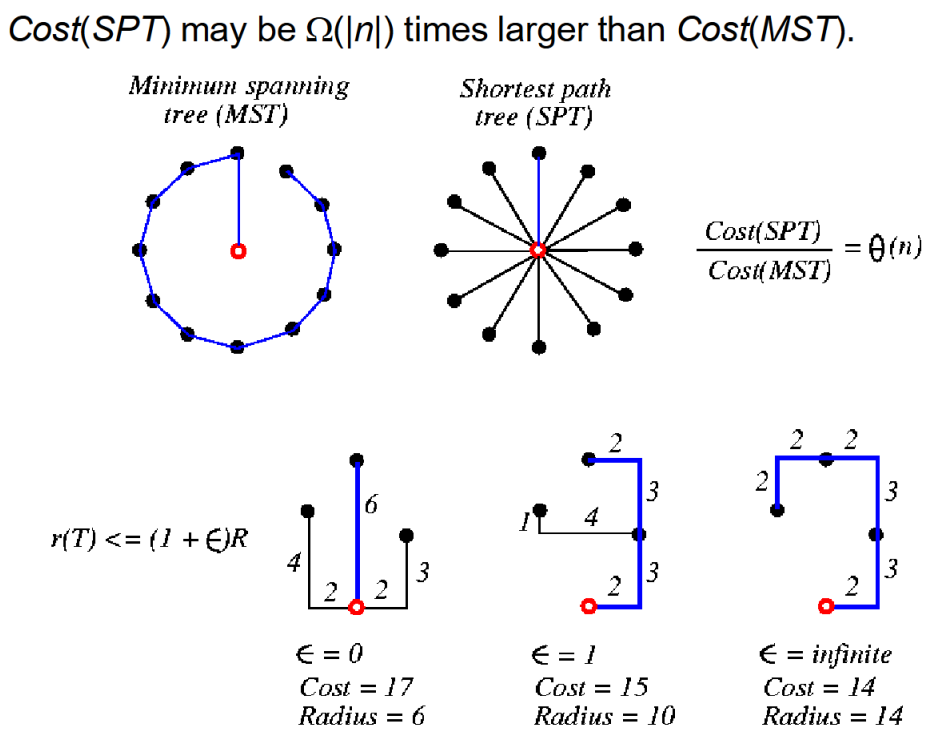
Alternative Solution:

> 注意第 9、10 两步先选了 (a, h) 而非 (b, c)，错过了一个多解.

9.23 Prim 与 Kruskal 算法比较

特性	Prim 算法	Kruskal 算法
工作方式	从一个起点开始，逐步扩展生成树.	按权重排序边，逐步加入生成树.
数据结构	使用优先队列或数组记录最小权重边.	使用并查集检测环.
适用场景	稠密图（边多的图）.	稀疏图（边少的图）.
时间复杂度	$O(V^2)$（邻接矩阵）或 $O((V + E) \log V)$（邻接表 + 堆）.	$O(E \log E)$（边排序）+ $O(E \log V)$（并查集）.
实现复杂度	相对简单.	较复杂，需要实现并查集.

9.24 MST 和 SPT 比较



R 是最优半径.

当 $\epsilon = 0$ 时，完全偏向 SPT，优先最小化根节点到其他点的路径长度.

> 半径最小，成本较高.

当 $\epsilon = 1$ 时，在 SPT 和 MST 中找到一个平衡点.

当 $\epsilon = \infty$ 时，完全偏向 MST，优先最小化总边权.

> 半径最大，总成本最小.

10. Advanced Graph Theory

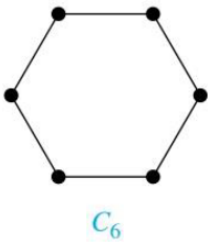
10.1 二分图进阶

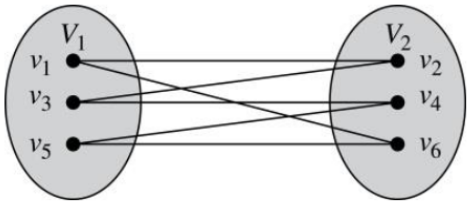
定义

A simple graph G is bipartite if V can be partitioned into two disjoint subsets V_1 and V_2, such that every edge connects a vertex in V_1 and a vertex in V_2. In other words, there are no edges which connect two vertices in V_1 or in V_2.

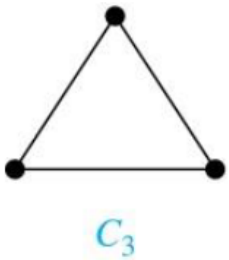
Example

Show that C_6 is bipartite.





Solution: We can partition the vertex set into $V_1 = \{v_1,\ v_3,\ v_5\}$ and $V_2 = \{v_2,\ v_4,\ v_6\}$ so that every edge of C_6 connects a vertex in V_1 and V_2.



Show that C_3 is not bipartite.

Solution: If we divide the vertex set of C_3 into two nonempty sets, one of the two must contain two vertices. But in C_3 every vertex is connected to every other vertex. Therefore, the two vertices in the same partition are connected. Hence, C_3 is not bipartite.

二分图匹配

Bipartite graphs are used to model applications that involve matching the elements of one set to elements in another, for example:

Job assignments - vertices represent the jobs and the employees, edges link employees with those jobs they have been trained to do. A common goal is to match jobs to employees so that the most jobs are done.

Marriage - vertices represent the men and the women and edges link a man and a woman if they are an acceptable spouse. We may wish to find the largest number of possible marriages.

于是自然引出一个问题：我们应该如何匹配，才能在每个点最多匹配一次的前提下，找到最多组匹配?

最大基数匹配

> Maximum Cardinality Matching

Given an undirected $G = (V, E), M \subseteq E$ is a *matching* if and only if at most one edge of M is incident on $v, \forall\ v \in V$.

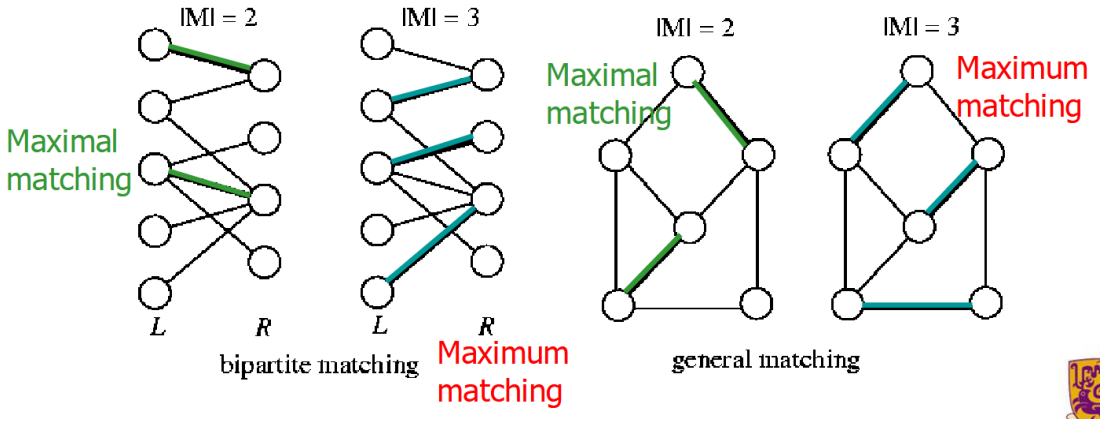
> 匹配集中的边两两不相交（没有共享顶点）

M is **maximum matching** if and only if $|M| \geq |M'| \forall$ matching M'.

> 最大匹配即最大基数匹配

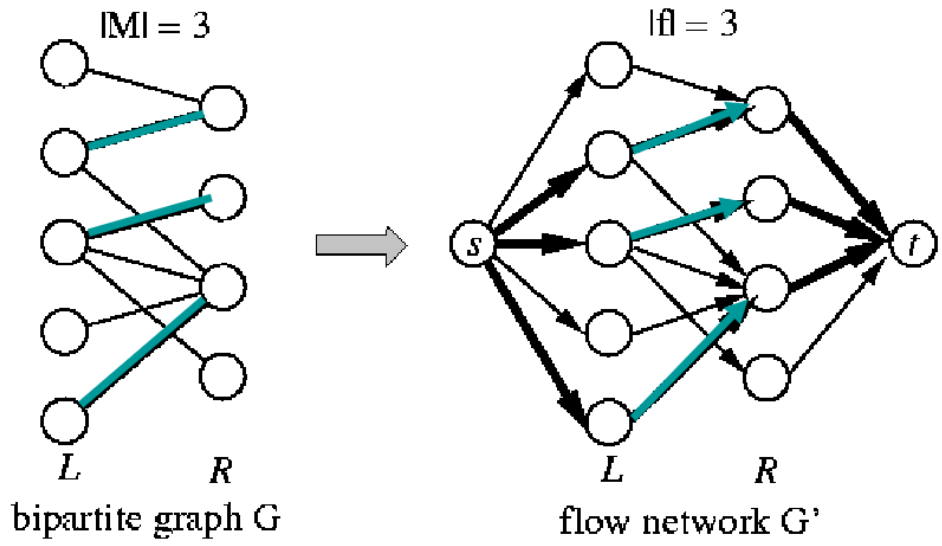
极大匹配

Maximal Matching 并不追求匹配边的数量最多，而是保证匹配无法再扩展，即不能再向匹配中加入任何边而仍符合匹配的定义.



> 注意，这里并不局限于二分图，而是任意的无向图.

最大基数二分匹配



Given a bipartite graph $G = (V, E)$, $V = L \cup R$, construct a unit-capacity flow network $G' = (V', E')$:

$V' = V \cup \{s, t\}$

$E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, (u, v) \in E\} \cup \{(v, t) : v \in R\}$

The cardinality of a maximum matching in G = the value of a maximum flow in G' (i.e., $|M| = |f|$).

Time complexity: $O(VE)$ by the Ford-Fulkerson algorithm.

Value of maximum flow $\leq min(L, R) = O(V)$

10.2 Network Flow

> 网络流：directed $G = (V, E)$

定义

capacity $c(u, v)$: $c(u, v) > 0,\ \forall\ (u, v) \in E;\ c(u, v) = 0,\ \forall\ (u, v) \notin E.$

> 容量：边能承载的最大流量.

Exactly one node with no incoming (outgoing) edges, called the **source** s (**sink** t).

> 源点 s 和汇点 t，即流量的起点和终点.

Flow $f : V \times V \to R$ that satisfies

- Capacity constraint: $f(u, v) \leq c(u, v),\ \forall\ u, v \in V.$
- Skew symmetry: $f(u, v) = -f(v, u).$
- Flow conservation: $\sum_{v \in V} f(u, v) = 0,\ \forall\ u \in V - \{s, t\}.$

> 除了源点和汇点，所有点流量守恒（流入等于流出，净流出为 0）

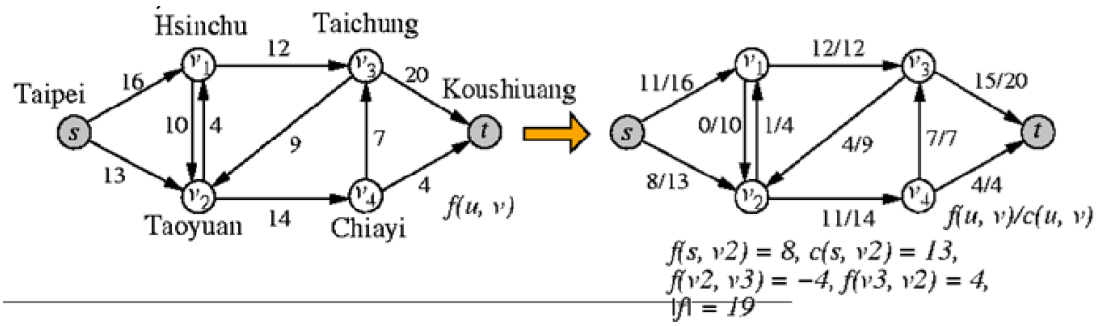
Value of a flow f. $|f| = \sum_{v \in V} f(s, v) = \sum_{v \in V} f(v, t)$, where $f(u, v)$ is the net flow from u to v.

> 即源点流出/汇点汇入的总流量.

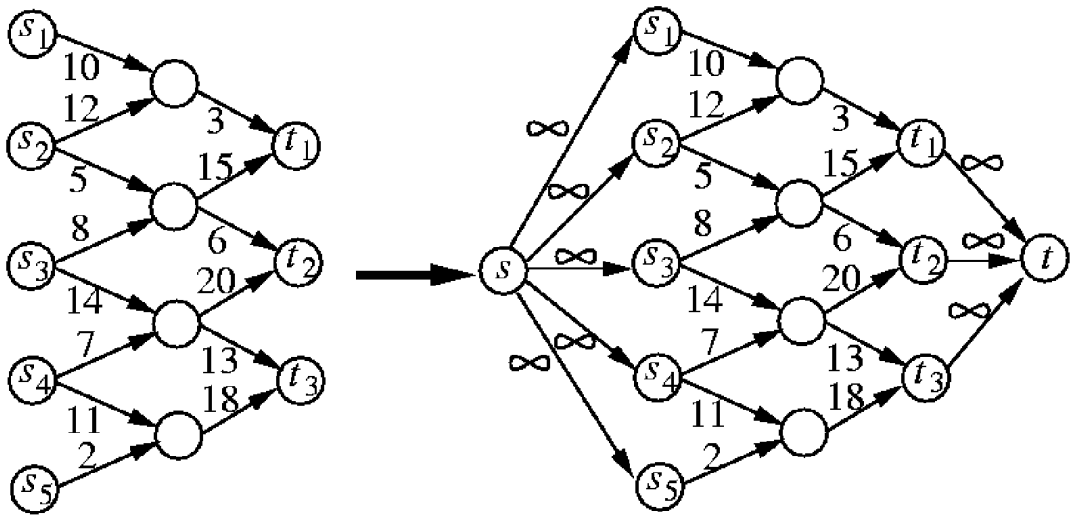
最大流问题

> The maximum flow problem

Given a flow network G with source s and sink t, find a flow of maximum value from s to t.



首先，多源多汇最大流问题可以简化为单源单汇：



Given a flow network with sources $S = \{s_1,\ s_2,\ \ldots,\ s_m\}$ and sinks $T = \{t_1,\ t_2,\ \ldots,\ t_n\}$, introduce a **supersource** s and edges $(s, s_i),\ i = 1,\ 2,\ \ldots,\ m$, with capacity $c(s, s_i) = \infty$ and a **supersink** t and edges $(t_i, t),\ i = 1,\ 2,\ \ldots,\ n$, with capacity $c(t_i, t) = \infty$.

然后，引入一些流的定义和性质

If $X, Y \subseteq V$, $f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$.

给定 $G = (V, E)$，f: flow in G

① $\forall\ X \subseteq V,\ f(X, X) = 0$.

② $\forall\ X, Y \subseteq V,\ f(X, Y) = -f(Y, X)$.

③ $\forall\ X, Y, Z \subseteq V$ with $X \cap Y = \emptyset$,

- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ and

- $f(Z,\ X \cup Y) = f(Z, X) + f(Z, Y)$.

Value of flow = net flow into the sink:

$|f| = f(V, t)$.

$$
\begin{aligned}
|f| &= f(s, V) \\
&= f(V, V) - f(V - s, V) \\
&= f(V, V - s) \\
&= f(V, t) + f(V, V - s - t) \\
&= f(V, t)
\end{aligned}
$$

> 运用了流量守恒.

Cancellating flow going in opposite direction:

Basic Ford-Fulkerson Method

Ford-Fulkerson-Method(G, s, t)

1. Initialize flow f to 0
2. while there exists an augmenting path p do
3. Augment flow f along p
4. return f

Augmenting path: A path from s to t along which we can push more flow.

> 增广路径.

Need to construct a *residual network* to find an augmenting path.

> 残余网络.

残余容量

Residual capacity of edge (u, v), $c_f(u, v)$: Amount of *additional* net flow that can be pushed from u to v before exceeding $c(u, v)$:

$$c_f(u, v) = c(u, v) - f(u, v)$$

$G_f = (V, E_f)$: residual network of $G = (V, E)$ induced by f, where $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$.

> 已经填满的就不用引入了.

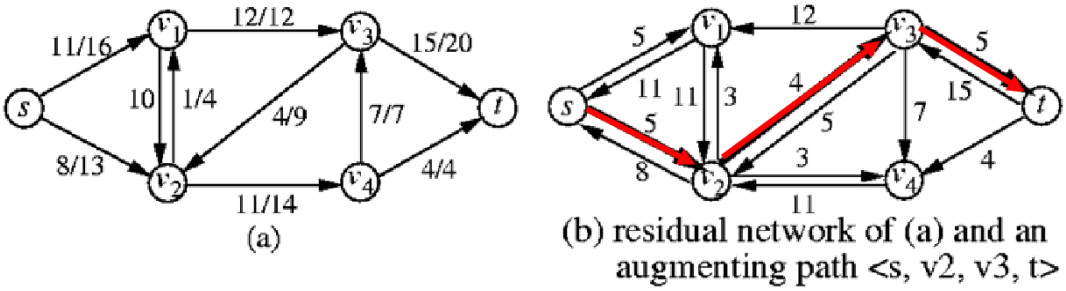
The residual network contains residual edges that can admit a positive net flow $(|E_f| \leq 2|E|)$

> 残余网络中，每条原始边 (u,v) 可能分解成正向边和反向边.
>
> 正向边：表示当前剩余的最大容量，$c_f(u,v)=c(u,v)-f(u,v)$
>
> 反向边：表示可以回退的最大流量，$c_f(v,u)=f(u,v)$

Let f and f' be flows in G and G_f, respectively. The **flow sum** $f+f': V\times V\to R$:

$$(f+f')(u,v)=f(u,v)+f'(u,v)$$

is a flow in G with value $|f+f'|=|f|+|f'|$.



An **augmenting path** p is a simple path from s to t in the residual network G_f.

- $(u,v)\in E$ on p in the forward direction (a forward edge), $f(u,v)<c(u,v)$.
- $(u,v)\in E$ on p in the reverse direction (a backward edge), $f(u,v)>0$

> 反向边：之前流过一些，现在可以回退.

Residual capacity of p, $c_f(p)$: Maximum amount of net flow that can be pushed along the augmenting path p, i.e.,

$c_f(p)=min\{c_f(u,v):(u,v)\ is\ in\ p\}$.

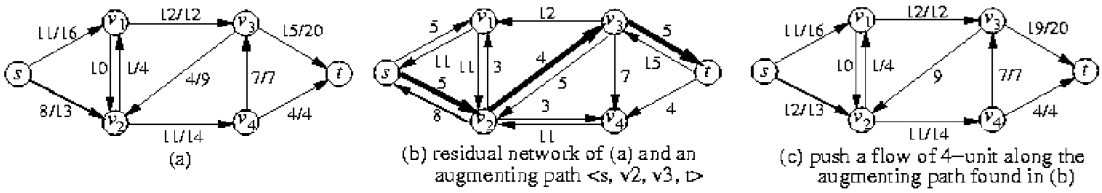
> 增广路径的容量取决于最小的一条，即最细的水管.

Let p be an augmenting path in G_f. Define $f_p: V\times V\to R$ by

$$f_p(u,v)=\begin{cases}c_f(p), & \text{if (u,v) is on p,}\\ -c_f(p), & \text{if (v,u) is on p,}\\ 0, & otherwise.\end{cases}$$

> 翻译：如果增广路径使用了反向边，就要回退流量. 如果使用正向边就增加流量.

Then, f_p is a flow in G_f with value $|f_p|=c_f(p)>0$.



流网络的割

A cut (S,T) of flow network $G=(V,E)$ is a partition of V into S and $T=V-S$ such that $s\in S$ and $t\in T$.

> 此外，割 (S,T) 将 V 分割成两个不相交子集，即 $S\cup T=V$ 且 $S\cap T=\emptyset$.

Capacity of a cut: $c(S,T)=\sum_{(u,v)\in E,\ u\in S,\ v\in T}c(u,v)$

> 只考虑 S 到 T（正向边）容量. 不考虑反向边.

$f(S,T)=|f|\leq c(S,T)$, where $f(S,T)$ is net flow across the cut (S,T).

> $f(S,T)$ 是实际净流量.
>
> 割的容量表示割能阻断的流量上限（即从 S 流向 T 的最大可能流量）.
>
> 不同的割都对流量上限有约束作用. 实际流量要满足所有割的限制，因此有了最大流最小割定理.

最大流最小割定理

> Max-flow min-cut theorem:

最大流量（从源点 s 到汇点 t）等于网络中任意割的最小容量.

The following conditions are equivalent

① f is a max-flow.

② G_f contains no augmenting path.

③ $|f|=c(S,T)$ for some cut (S,T).

10.3 Perfect Graph

一个图 G 是完美图，如果对于 G 的每个子图（包括 G 本身），都满足：

$$\chi(H)=\omega(H)$$

其中：

$\chi(H)$：子图 H 的着色数. 即用最少的颜色对 H 的顶点进行着色，使得相邻顶点的颜色不同.

$\omega(H)$：子图 H 的团数. 即 H 的最大团（clique）的顶点数.

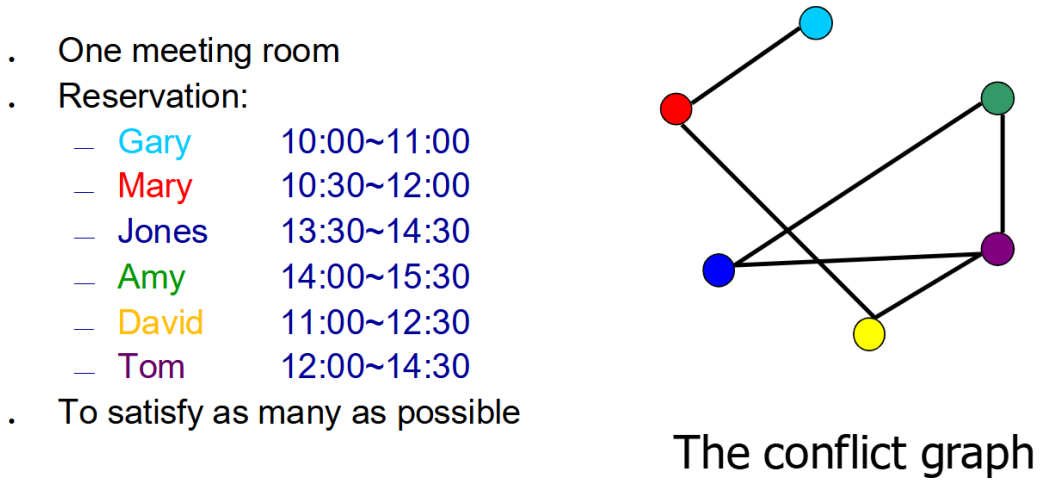
> 团：在无向图 $G=(V,E)$ 中，团是一个顶点的子集 $C\subseteq V$，使得 C 中的任意两个顶点之间都有边相连.

10.4 Independent Set Problem

定义

- **问题描述**：
  给定一个无向图 $G = (V, E)$ 和一个正整数 K，问图 G 中是否存在一个独立集 $V' \subseteq V$，满足：
  - **独立性**：V' 中的任何两个顶点之间都没有边（即 V' 是无冲突的顶点集）.
  - **大小限制**：$|V'| \geq K$，即独立集的大小至少是 K.
- **独立集（Independent Set）**：
  独立集是图中一组相互不连接（没有边相连）的顶点.
- **目标**：
  找到满足条件的独立集，或者最大化独立集的大小.

独立集与冲突图



图中给出的冲突图（Conflict Graph）是一个具体的例子：

- 每个节点表示一次预约（比如 Gary 的预约时间为 10:00~11:00）.
- 如果两个预约时间有重叠，则两个节点之间有一条边（表示冲突）.

冲突的意义

- 两个预约之间有边说明它们冲突，不能同时安排在同一个会议室中.
- 找一个独立集，就是要找一组互不冲突的预约（即这些预约之间没有边连接）.

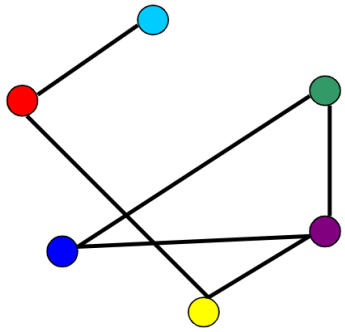
问题目标

- 找到一个**独立集**.
- 使独立集的大小尽可能大，从而最大化可满足的预约数量.

. An independent set

. Another independent set

. A maximum independent set



11. 考前复习

注意点

注意分类讨论.

计数原理

审题：the set of all passwords of length 4 over the symbol set S such that all symbols in a password are digits and such that **adjacent symbols are distinct**.

> 相邻的不同 $\neq$ 所有都不同，不能连续出现重复元素，但可以跳着出现.