习题一

① 习题一

0.1	$f(n)$	$g(n)$
(a)	$n-100$	$n-200$

$$f=\Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(b)	$n^{1/2}$	$n^{2/3}$

$$f = O(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(c)	$100n + \log n$	$n + (\log n)^2$

$$f = \Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(d)	$n\log n$	$10n\log 10n$

$$g(n) = 10n(\log n + \log 10) = 10n\log n + 10\log 10\, n$$

$$f = \Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(e)	$\log 2n$	$\log 3n$

$$f(n) = \log n + \log 2$$

$$g(n) = \log n + \log 3$$

$$f = \Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(f)	$10\log n$	$\log(n)^2$

$$g(n) = 2\log n$$

$$f = \Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(g)	$n^{1.01}$	$n\log^2 n$

$$\frac{f(n)}{g(n)} = \frac{n^{1.01}}{n\log^2 n} = \frac{n^{0.01}}{\log^2 n}$$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{n^{0.01}}{\log^2 n} = \lim_{n\to\infty}\frac{0.01n^{-0.99}}{2\log n\cdot\frac{\ln 2}{n}} = \lim_{n\to\infty}\frac{0.01n^{0.01}}{2\ln 2\log n} = \lim_{n\to\infty}\frac{10^{-4}n^{0.01}}{2\ln^2 2} = \infty$$

$$f = \Omega(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(h)	$n^2/\log n$	$n(\log n)^2$

$$\frac{f(n)}{g(n)} = \frac{n^2/\log n}{n(\log n)^2} = \frac{n}{(\log n)^3}$$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{n}{(\log n)^3} = \lim_{n\to\infty}\frac{1}{3(\log n)^2\frac{\ln 2}{n}} = \lim_{n\to\infty}\frac{n}{3\ln 2\,(\log n)^2}$$

$$= \lim_{n\to\infty}\frac{n}{6\ln^2 2\log n} = \lim_{n\to\infty}\frac{n}{6\ln^3 2} = \infty$$

$$f = \Omega(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(i)	$n^{0.1}$	$(\log n)^{10}$

$$\frac{f(n)}{g(n)} = \frac{n^{0.1}}{(\log n)^{10}}$$

$$\lim_{n\to\infty}\frac{f(n)}{g(n)} = \lim_{n\to\infty}\frac{n^{0.1}}{(\log n)^{10}} = \lim_{n\to\infty}\frac{0.1n^{-0.9}}{10(\log n)^{9}\frac{\ln 2}{n}} = \lim_{n\to\infty}\frac{0.1n^{0.1}}{10\ln 2\,(\log n)^{9}} = \infty$$

$$f = \Omega(g)$$

1 习题一

0.1	$f(n)$	$g(n)$
(j)	$(\log n)^{\log n}$	$n/\log n$

$$x = \log n$$

$$f(x) = x^x \qquad g(x) = 2^x/x$$

$$\frac{f(x)}{g(x)} = \frac{x^{x+1}}{2^x} = \left(\frac{x}{2}\right)^x x$$

$$f = \Omega(g)$$

1 习题一

0.1	$f(n)$	$g(n)$
(k)	$\sqrt{n}$	$(\log n)^3$

$$\frac{f(n)}{g(n)} = \frac{n^{1/2}}{(\log n)^3}$$

$$f = \Omega(g)$$

1 习题一

0.1	$f(n)$	$g(n)$
(l)	$n^{1/2}$	$5^{\log_2 n}$

$$g(n) > 2^{\log_2 n} = n$$

$$f = O(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(m)	$n2^n$	3^n

$$\frac{f(n)}{g(n)} = \frac{n2^n}{3^n} = n\left(\frac{2}{3}\right)^n$$

$$f = O(g)$$

1 习题一

0.1	$f(n)$	$g(n)$
(n)	2^n	2^{n+1}

$$f = \Theta(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(o)	$n!$	2^n

$$f = \Omega(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(p)	$(\log n)^{\log n}$	$2^{(\log_2 n)^2}$

$$g(n) = 2^{\log_2 n \cdot \log_2 n} = n^{\log n}$$

$$f = O(g)$$

① 习题一

0.1	$f(n)$	$g(n)$
(q)	$\sum_{i=1}^{n} i^k$	n^{k+1}

Faulhaber公式

$$\sum_{k=1}^{n} k^p = \frac{n^{p+1}}{p+1} + \frac{n^p}{2} + \sum_{k=2}^{p} \frac{B_k p!\, n^{p-k+1}}{k!\,(p-k+1)!}$$

$f = \Theta(g)$

① 习题一

0.2 c is a positive real number, $g(n) = 1 + c + c^2 + \cdots + c^n$

(a)

$$c < 1, g(n) = \frac{1 - c^{n+1}}{1 - c}$$

$$\frac{g(n)}{1} \leq \frac{1}{1 - c} \qquad \frac{g(n)}{1} \geq 1$$

$$g(n) = \Theta(1)$$

① 习题一

0.2 c is a positive real number, $g(n) = 1 + c + c^2 + \cdots + c^n$

(b)

$c = 1, g(n) = n + 1$

$g(n) = \Theta(n)$

① 习题一

0.2 c is a positive real number, $g(n) = 1 + c + c^2 + \cdots + c^n$

(c)

$$c > 1, g(n) = \frac{c^{n+1} - 1}{c - 1}$$

$$\frac{g(n)}{c^n} = \frac{c - c^{-n}}{c - 1} \le \frac{c}{c - 1} \qquad \frac{g(n)}{c^n} = \frac{c - c^{-n}}{c - 1} \ge 1$$

$$g(n) = \Theta(c^n)$$

① 习题一

1.14 find an efficient way to compute $F_n \bmod p$

$$F_n = F_{n-1} + F_{n-2}$$

$$\begin{pmatrix} F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} F_{n-1} & F_{n-2} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2}$$

用矩阵快速幂求$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2}$，计算中间结果时$mod\ p$

1.14 find an efficient way to compute $F_n \ mod \ p$

```
int p;
int temp[2][2];
void multiply(int a[][2], int b[][2]){
    memset(temp,0,sizeof(temp));
    for(int i=0; i<2; i++)
        for(int j=0; j<2; j++)
            for(int k=0; k<2; k++)
                temp[i][j]=(temp[i][j]+a[i][k]*b[k][j])%p;
    for(int i=0; i<2; i++)
        for(int j=0; j<2; j++)
            a[i][j]=temp[i][j];
}
```

```
int res[2][2];
void qpow_m(int a[][2],int n){
    memset(res,0,sizeof(res));
    for(int i=0;i<2;i++)
        res[i][i]=1;
    while(n){
        if(n&1)
            multiply(res,a,N);
        multiply(a,a,N);//a=a*a
        n>>=1;
    }
}
```

1.14 find an efficient way to compute $F_n \ mod \ p$

$$F_n = F_{n-1} + F_{n-2}$$

$$\begin{pmatrix} F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} F_{n-1} & F_{n-2} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2}$$

用矩阵快速幂求$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-2}$，计算中间结果时$mod \ p$

最后求$res_{11} + res_{21} \ mod \ p$

中间所得结果均小于p，每一步运算的复杂度为$O(\log^2 p)$，
总复杂度为$O(\log n \log^2 p)$

① 习题一

1.20

(a) find the inverse of $20\ mod\ 79$

$(20,79)=1$，79为质数

由费马小定理得，$20^{78}\equiv 1\ (mod\ 79)$

$$
\begin{aligned}
&20^{-1}\equiv 20^{77}\equiv 20\times(400)^{38}\\
&\equiv 20\times 5^{38}\equiv 20\times 25\times 125^{12}\\
&\equiv 500\times 46^{12}\equiv 26\times 62^{6}\\
&\equiv 26\times 52^{3}\equiv 8\times 26^{4}\\
&\equiv 8\times 44^{2}\equiv 8\times 40\\
&\equiv 4\ (mod\ 79)
\end{aligned}
$$

① 习题一

1.20

(b) find the inverse of $3 \ mod \ 62$

$(3,62)=1$

由欧拉定理得，$3^{\varphi(62)} \equiv 1 \ (mod \ 62)$

$$3^{-1} \equiv 3^{\varphi(62)-1} \equiv 3^{29}$$
$$\equiv 3 \times 19^{7}$$
$$\equiv 3 \times 19 \times 51^{3}$$
$$\equiv 57 \times 51 \times 59$$
$$\equiv (-5) \times 51 \times (-3)$$
$$\equiv 21 \ (mod \ 62)$$

① 习题一

1.20

(c) find the inverse of $21\ mod\ 91$

$$(21,91) = 7 \neq 1$$

$21^{-1}\ (mod\ 91)$不存在

1.20

(d) find the inverse of $5\ mod\ 23$

$$(5,23) = 1$$

设$5^{-1} \equiv x\ (mod\ 23) \Rightarrow 5x \equiv 1\ (mod\ 23) \Rightarrow 5x + 23y = 1$

由扩展欧几里得算法，得

$5x_1 + 23y_1 = 1$	$(x_5, y_5) = (0,1)$
$\Rightarrow 3x_2 + 5y_2 = 1$	$\Rightarrow (x_4, y_4) = (1,0)$
$\Rightarrow 2x_3 + 3y_3 = 1$	$\Rightarrow (x_3, y_3) = (-1,1)$
$\Rightarrow x_4 + 2y_4 = 1$	$\Rightarrow (x_2, y_2) = (2,-1)$
$\Rightarrow 0x_5 + y_5 = 1$	$\Rightarrow (x_1, y_1) = (-9,2)$

$$5^{-1} \equiv -9 \equiv 14\ (mod\ 23)$$

① 习题一

1.31

(a) If N is an n-bit number, how many bits long is N!

$N!$的位数为$\log N! = \Theta(N\log N)$

思路一

$$N! \le N^N, \log N! = O(\log N^N) = O(N\log N)$$

$$N! \ge \frac{N}{2}\cdot\left(\frac{N}{2}+1\right)\cdots N \ge \frac{N^{\frac{N}{2}}}{2}, \log N!$$

$$= \Omega\left(\log\frac{N^{\frac{N}{2}}}{2}\right) = \Omega\left(\frac{N}{2}\log\frac{N}{2}\right) = \Omega(N\log N)$$

1.31

(a) If N is an n-bit number, how many bits long is N!

$N!$的位数为$\log N! = \Theta(N\log N)$

思路二

$$\log N! = \sum_{i=1}^{N} \log i = \sum_{i=1}^{\log N - 1} \left(2^{i+1} - 2^{i}\right) i + \left(N - 2^{\log N}\right) \log N$$
$$= (\log N - 2)2^{\log N} + 2 + N\log N - \log N\, 2^{\log N}$$
$$= N\log N - 2^{\log N + 1} + 2 = \Theta(N\log N)$$

1.31

(a) If N is an n-bit number, how many bits long is N!

$N!$的位数为$\log N! = \Theta(N\log N)$

思路三

由斯特林公式，得$N! = \sqrt{2\pi N}\left(\frac{N}{e}\right)^N$

$$\log N! = \log\sqrt{2\pi N}\left(\frac{N}{e}\right)^N$$

$$= \log\sqrt{2\pi} + \frac{1}{2}\log N + N\log N - N\log e$$

$$= \Theta(N\log N)$$

① 习题一

1.31

(b) Give an algorithm to compute N! and analyze its running time

直接从1连续乘到N

考虑第i步，即$(i-1)! \cdot i$

$(i-1)!$为$\Theta\big((i-1)\log(i-1)\big) = \Theta(i\log i)$位，$i$为$\log i$位，则第$i$步的复杂度为$\mathrm{O}(i\log^2 i)$

整个算法的复杂度为

$$\mathrm{O}\left(\sum_{i=1}^{N} i\log^2 i\right) = \mathrm{O}\left(\log^2 N\sum_{i=1}^{N} i\right) = \mathrm{O}(N^2\log^2 N)$$

1.35

(a) If p is prime, then we know every number $1 \leq x < p$ is invertible modulo p. Which of these numbers are their own inverse?

$$x^{-1} \equiv x\ (mod\ p) \Rightarrow x^2 \equiv 1\ (mod\ p) \Rightarrow x^2 = kp + 1$$

若$k = 0$，$x^2 = 1$，$x = 1$

若$k \neq 0$，$kp = x^2 - 1 = (x-1)(x+1)$，则$p|(x-1)$或$p|(x+1)$

若$p|(x-1)$，$0 \leq x - 1 \leq p - 2$，不成立

若$p|(x+1)$，$2 \leq x + 1 \leq p$，$x = p - 1$

1.35

(b) By pairing up multiplicative inverses, show that $(p-1)! \equiv -1 \ (mod\ p)$ for prime p.

$(p-1)! = 1 \times 2 \times \cdots \times (p-1)$，$p$为素数，则$1,2,\cdots,p-1$共有偶数个数

考察$x \in \{2,3,\cdots,p-2\}$，则由(a)可知$x^{-1} \not\equiv x\ (mod\ p)$，则$\exists y \in \{2,3,\cdots,p-2\}$且$y \neq x$，$x^{-1} \equiv y\ (mod\ p)$，同理$y^{-1} \equiv x\ (mod\ p)$，$xy \equiv 1\ (mod\ p)$

则$2,3,\cdots,p-2$这$p-3$个偶数可两两配对组成$\frac{p-3}{2}$个互逆对

$(p-1)! \equiv 1 \times (p-1) \equiv -1\ (mod\ p)$

1.35

(c) Show that if N is not prime, then $(N - 1)! \not\equiv -1 \ (mod\ N)$

思路一：反证

假设存在合数N且$(N-1)! \equiv -1 \ (mod\ N)$
则$(N-1)! \times (N-1) \equiv 1 (mod\ N)$，$(N-1)!^{-1}$为$N-1$，
即$(N-1)!\, mod\ N$的逆元存在，则$gcd((N-1)!, N) = 1$，
与N为合数矛盾
得证若N为合数则$(N-1)! \not\equiv -1 (mod\ N)$

1.35

(c) Show that if N is not prime, then $(N - 1)! \not\equiv -1 \ (mod\ N)$

思路二：讨论N

若$N = ab$，且$1 < a < b$，则$(N-1)! = 1 \times \cdots \times a \times \cdots \times b \times \cdots \times (N-1)$，$(\mathrm{N}-1)! \equiv 0 \ (mod\ N)$

若$N = k^2$，当$k > 2$时，$N > 2k$，则$(N-1)! = 1 \times \cdots \times k \times \cdots \times 2k \times \cdots \times (N-1)$，$(\mathrm{N}-1)! \equiv 0 \ (mod\ N)$

若$N = k^2$且$k \leq 2$，则$N = 1$或4

$N = 1$时，$(N-1)! \equiv 1 \equiv 0 \ (mod\ 1)$

$N = 4$时，$(N-1)! \equiv 3! \equiv 6 \equiv 2 \not\equiv -1 \ (mod\ 4)$

1.35

(d) Unlike Fermat’s Little theorem, Wilson’s theorem is an if-and-only-if condition for primality. Why can’t we immediately base a primality test on this rule?

用费马小定理检验素数时需计算 a^{N-1} ，可用快速幂算法，复杂度为$\mathrm{O}(\log N)$；用Wilson定理则需要计算$(N-1)!$，复杂度为$\mathrm{O}(N)$

习题三

4.11

Give an algorithm that takes as input a directed graph with positive edge lengths, and returns the length of the shortest cycle in the graph (if the graph is acyclic, it should say so). Your algorithm should take time at most $O(|V|^3)$.

思路一：

$|V|$次Dijkstra算法，对每一个节点求出其到其他节点的最短路大小

对每一条边$e: u \to v$，包含点u, v的最短环长度为$dist(u,v) + dist(v,u)$

取所有最短环长度中的最小值，即得到图中最短环长度，不存在环时得到∞

思路二：

将$|V|$次Dijkstra算法更改为一次Floyd算法得到任意两点的最短路大小

② 习题三

4.11

Give an algorithm that takes as input a directed graph with positive edge lengths, and returns the length of the shortest cycle in the graph (if the graph is acyclic, it should say so). Your algorithm should take time at most $O(|V|^3)$.

思路三：Floyd求最短环，松弛到节点k时，用节点序号最大为k的最小环更新ans

```
int ans=inf;
for(int k=1; k<=n; k++){
    for(int i=1; i<k; i++)
        for(int j=i+1; j<k; j++)
            ans=min(ans,f[i][j]+graph[j][k]+graph[k][i]);
    for(int i=1; i<=n; i++)
        for(int j=1; j<=n; j++)
            if(f[i][k]+f[k][j]<=f[i][j])
                f[i][j]=f[i][k]+f[k][j];
    }
```

4.12

Give an undirected graph $G = (V, E)$ whose edge lengths > 0 and an edge $e \in E$. Compute the length of the shortest cycle containing edge e in $O(|V|^2)$.

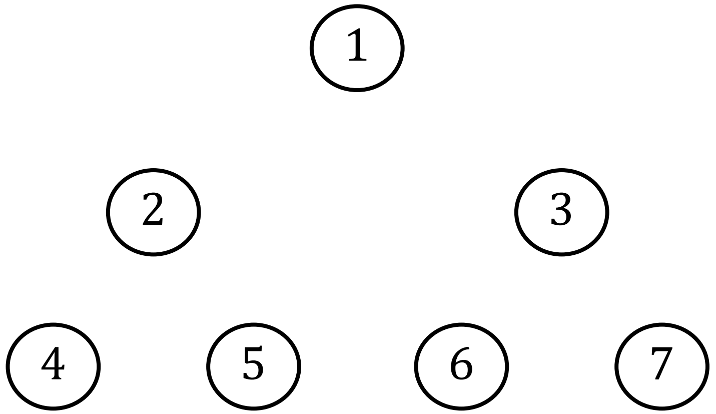
设$e=(u,v)$，从G中删除e，然后求出u到v的最短路$dist(u,v)$，再加上e的长度l_e得到$dist(u,v)+l_e$，即为G中包含e的最短环长度。

若$dist(u,v)+l_e=\infty$则说明G中不存在包含e的环

② 习题三

4.16

(a) Consider the node at position j of the array. Show that its parent is at position $\lfloor j/2 \rfloor$ and its children are at $2j$ and $2j + 1$ (if these numbers are $\leq n$)



设下标为j的节点为第m层第n个节点，j, m, n均从1开始，则前$m-1$层共有$2^{m-1}-1$个节点，$j = 2^{m-1}+n-1$

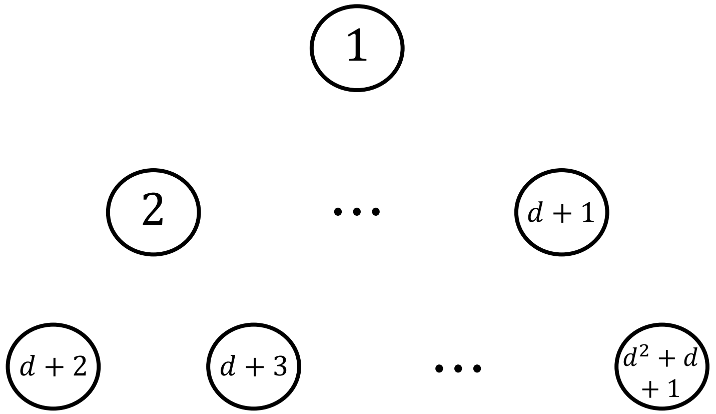
则其父节点为第$m-1$层第$\left\lceil \frac{n+1}{2} \right\rceil$个节点，则父节点下标为$2^{m-2}+\left\lfloor \frac{n+1}{2} \right\rfloor - 1 = 2^{m-2}+\left\lfloor \frac{n-1}{2} \right\rfloor = \left\lfloor \frac{2^{m-1}+n-1}{2} \right\rfloor = \left\lfloor \frac{j}{2} \right\rfloor$

其子节点下标k则满足$\left\lfloor \frac{k}{2} \right\rfloor = j$，则$k = 2j$或$2j+1$

② 习题三

4.16

(b) What the corresponding indices when a complete d-ary tree is stored in an array?



设下标为j的节点为第m层第n个节点，j,m,n均从1开始，则前$m-1$层共有$\frac{d^{m-1}-1}{d-1}$个节点，$j=\frac{d^{m-1}-1}{d-1}+n$

$$d^{m-1}=(j-n)(d-1)+1$$

则其父节点为第$m-1$层第$\left\lceil\frac{n+d-1}{d}\right\rceil$个节点，则父节点下标为$\frac{d^{m-2}-1}{d-1}+\left\lfloor\frac{n+d-1}{d}\right\rfloor=\left\lfloor\frac{d^{m-1}-d}{(d-1)d}+\frac{n+d-1}{d}\right\rfloor=\left\lfloor\frac{(j-n)(d-1)+1-d}{(d-1)d}+\frac{n+d-1}{d}\right\rfloor=\left\lfloor\frac{j-n-1+n+d-1}{d}\right\rfloor=\left\lfloor\frac{j+d-2}{d}\right\rfloor$

其子节点下标k则满足$\left\lfloor\frac{k+d-2}{d}\right\rfloor=j$，则$k=dj-d+2,dj-d+1,\cdots,dj+1$

② 习题三

4.16

$|h|$, which returns the number of elements currently in the array;

```
function makeheap(S)
h = empty array of size |S|
for x ∈ S:
    h(|h| + 1) = x
for i = |S| downto 1:
    siftdown(h, h(i), i)
return h
```

将S中的元素以对应的key值为基准构造一个小根堆，根节点下标为1

```
procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x
```

若当前节点的key值比其$minchild$节点小，则将该节点下移

```
function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \le j \le \min\{|h|, 2i+1\}\}$
```

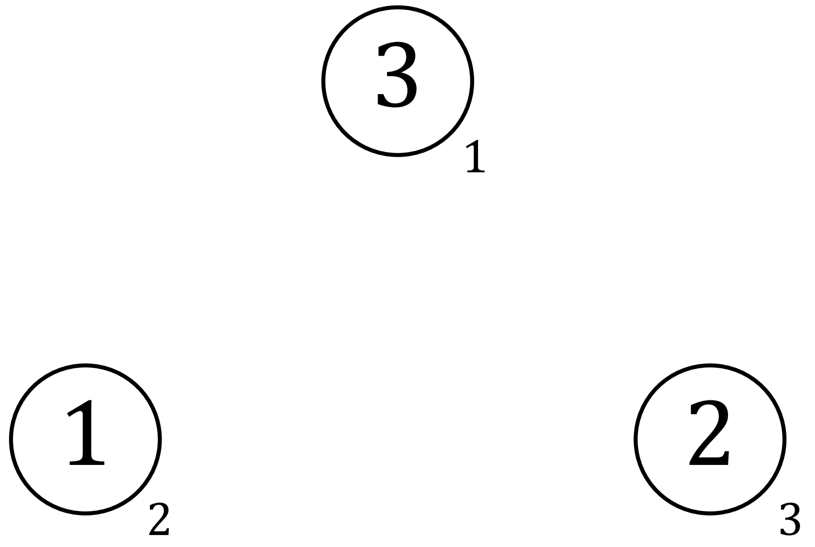
得到2个子节点中key值更小的一个

② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

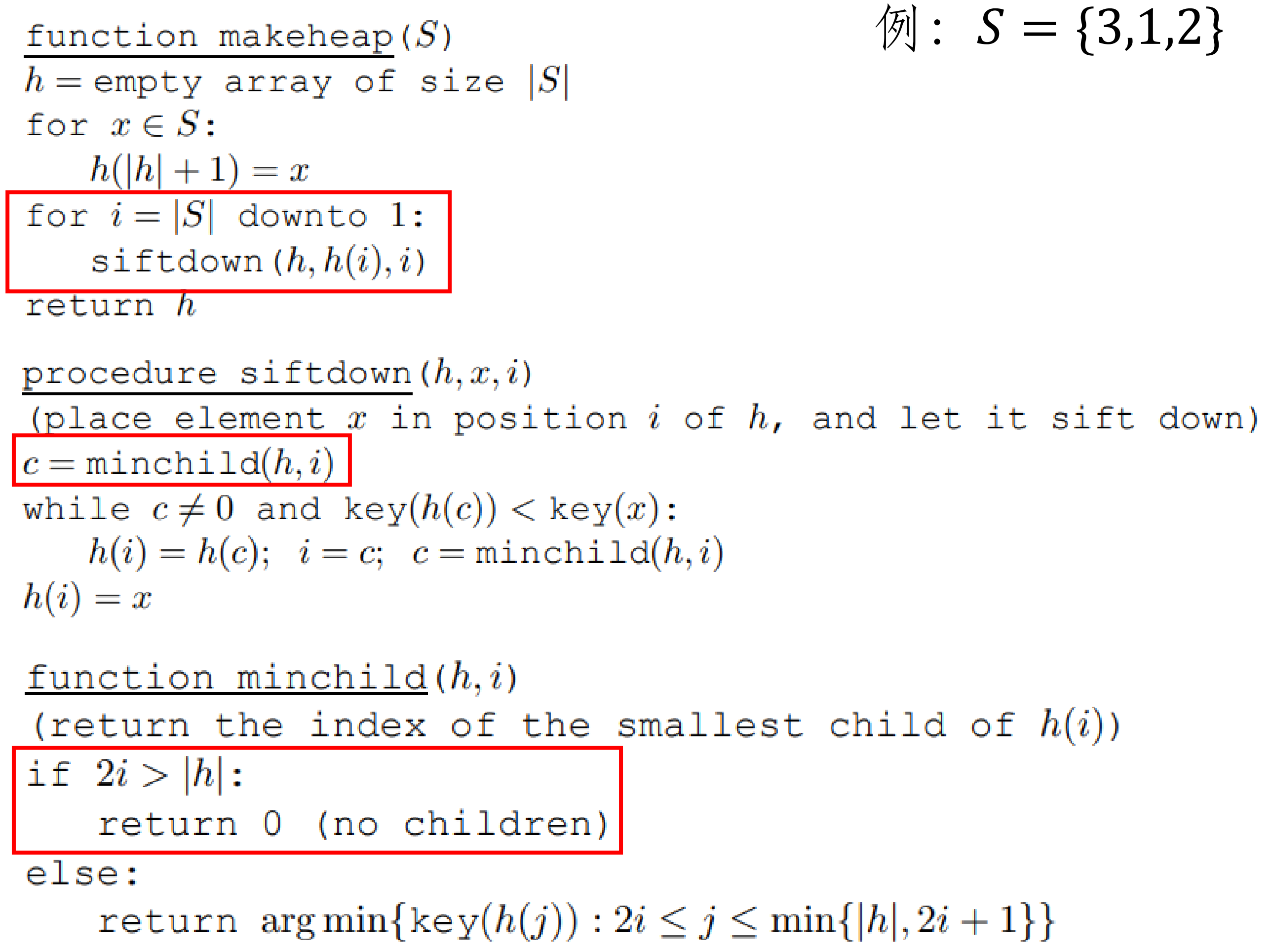
```
function makeheap($S$)
$h =$ empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$

procedure siftdown($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it sift down)
$c =$ minchild($h, i$)
while $c \neq 0$ and key($h(c)$) < key($x$):
    $h(i) = h(c)$;  $i = c$;  $c =$ minchild($h, i$)
$h(i) = x$

function minchild($h, i$)
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{$key($h(j)$) $: 2i \le j \le \min\{|h|, 2i+1\}\}$
```



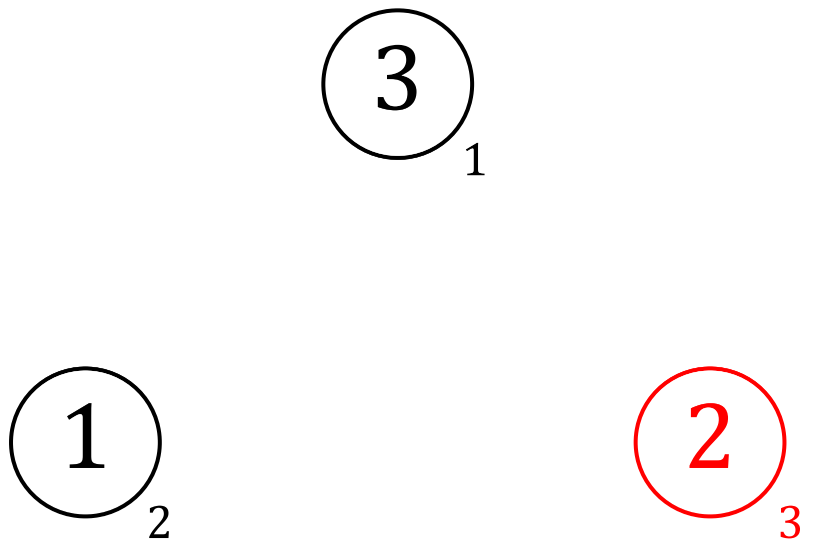
② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S=\{3,1,2\}$



```
function makeheap($S$)
$h$ = empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$

procedure siftdown($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it sift down)
$c$ = minchild($h, i$)
while $c \neq 0$ and key($h(c)$) < key($x$):
    $h(i) = h(c)$;  $i = c$;  $c$ = minchild($h, i$)
$h(i) = x$

function minchild($h, i$)
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{$key$(h(j)) : 2i \le j \le \min\{|h|, 2i+1\}\}$
```



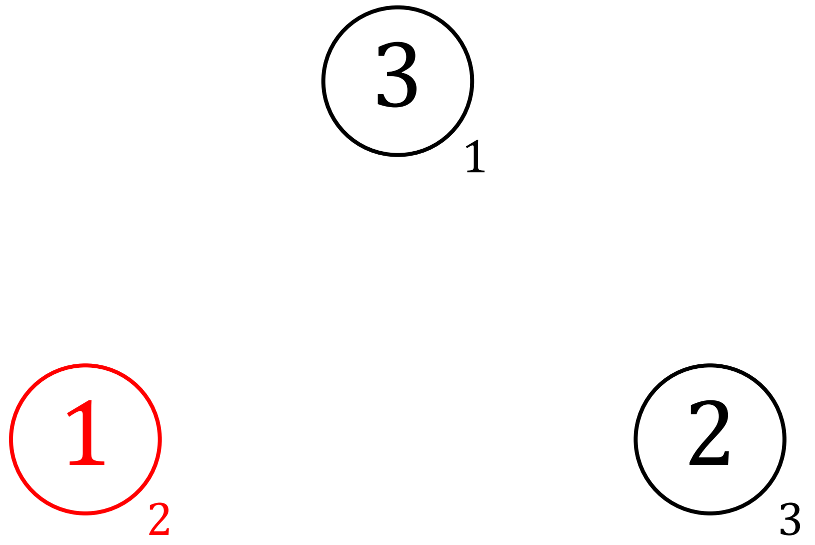
$x=2$
$i=3$
$c=0$

② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

```
function makeheap(S)
h = empty array of size |S|
for x ∈ S:
    h(|h| + 1) = x
for i = |S| downto 1:
    siftdown(h, h(i), i)
return h

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0  (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



$x = 1$
$i = 2$
$c = 0$

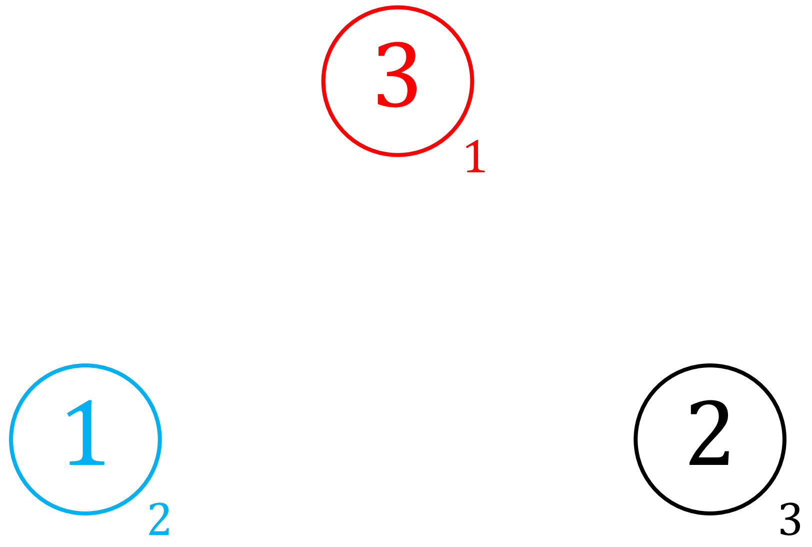
② 习题三

4.16

$|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

```
function makeheap(S)
h = empty array of size |S|
for x ∈ S:
    h(|h| + 1) = x
for i = |S| downto 1:
    siftdown(h, h(i), i)
return h

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```

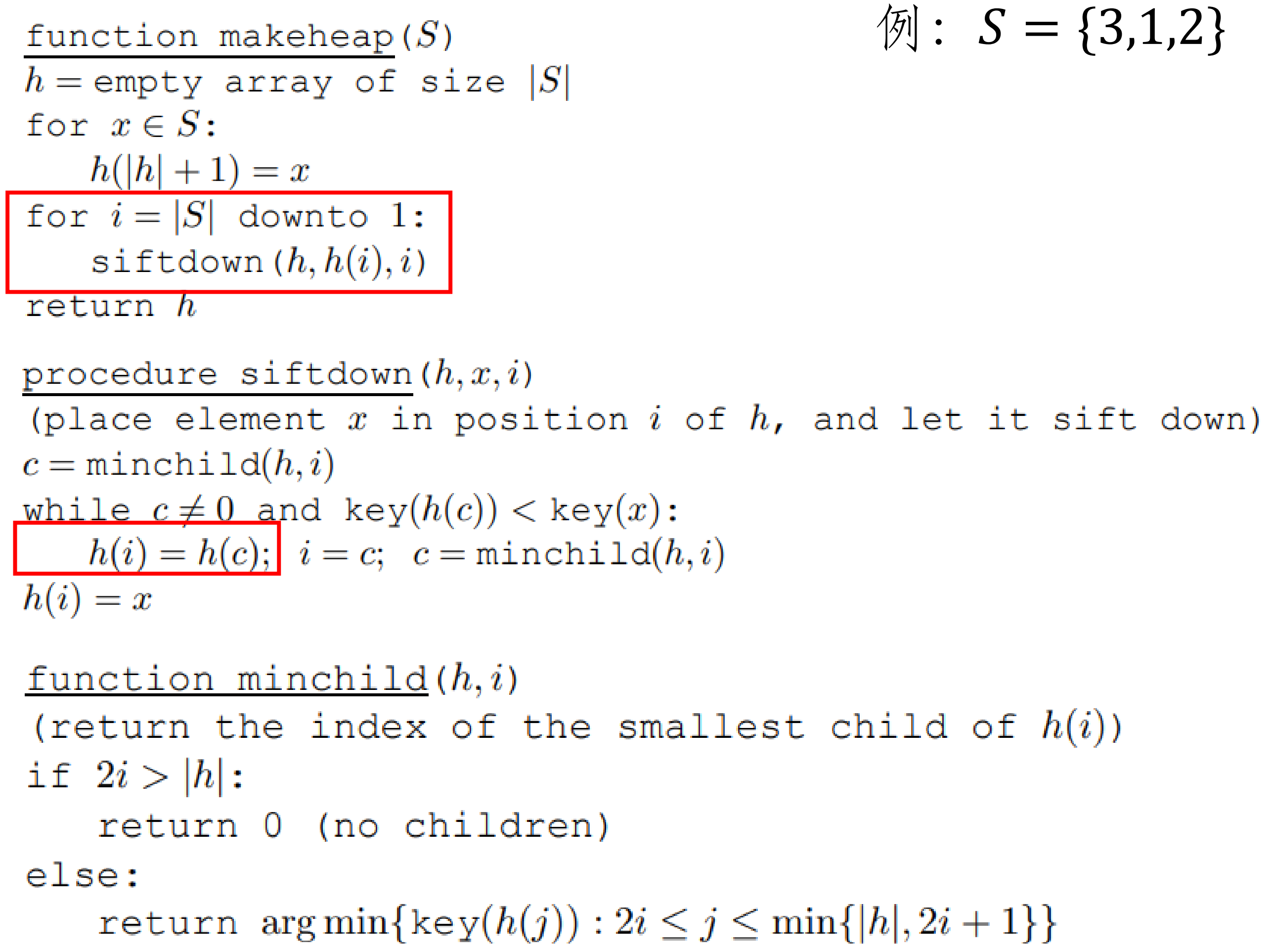


$x = 3$
$i = 1$
$c = 2$

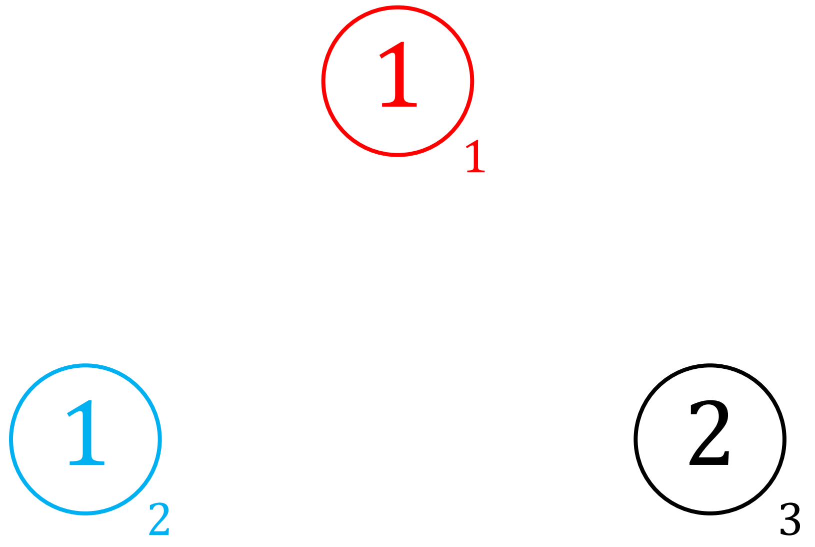
② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$



```
function makeheap($S$)
$h =$ empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$

procedure siftdown($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it sift down)
$c =$ minchild($h, i$)
while $c \neq 0$ and key($h(c)$) $<$ key($x$):
    $h(i) = h(c)$;  $i = c$;  $c =$ minchild($h, i$)
$h(i) = x$

function minchild($h, i$)
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \le j \le \min\{|h|, 2i+1\}\}$
```



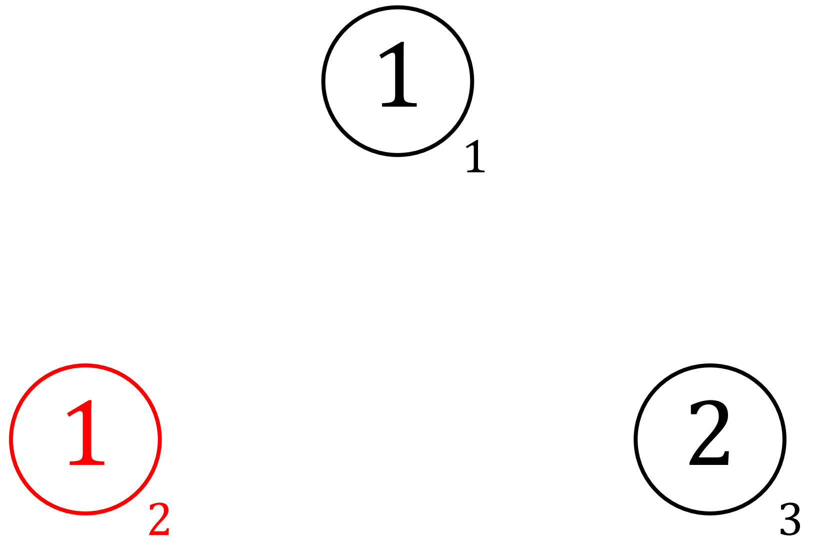
$x = 3$
$i = 1$
$c = 2$

2 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

```
function makeheap($S$)
$h$ = empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown$(h, h(i), i)$
return $h$

procedure siftdown$(h, x, i)$
(place element $x$ in position $i$ of $h$, and let it sift down)
$c$ = minchild$(h, i)$
while $c \neq 0$ and key$(h(c))$ < key$(x)$:
    $h(i) = h(c)$; $i = c$;  $c$ = minchild$(h, i)$
$h(i) = x$

function minchild$(h, i)$
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{$key$(h(j)) : 2i \leq j \leq \min\{|h|, 2i+1\}\}$
```



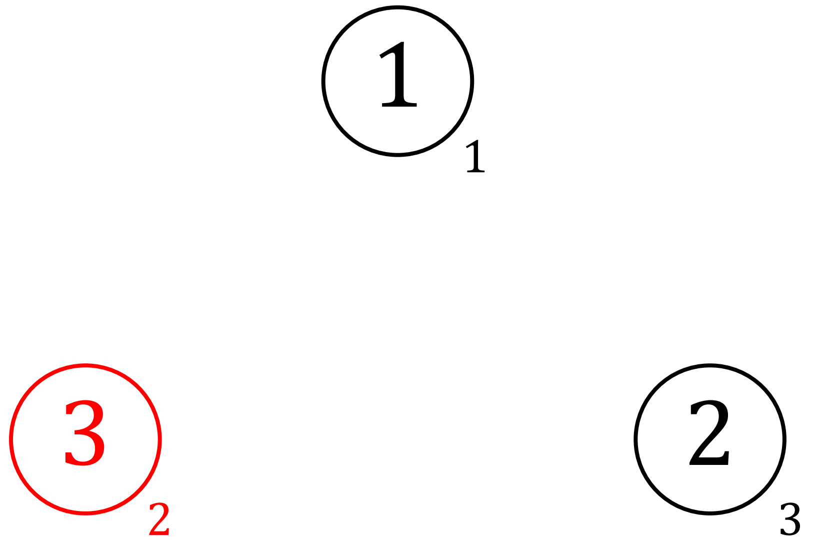
$x = 3$
$i = 2$
$c = 0$

② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

```
function makeheap($S$)
$h$ = empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$

procedure siftdown($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it sift down)
$c$ = minchild($h, i$)
while $c \neq 0$ and key($h(c)$) < key($x$):
    $h(i) = h(c)$;  $i = c$;  $c$ = minchild($h, i$)
$h(i) = x$

function minchild($h, i$)
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \le j \le \min\{|h|, 2i+1\}\}$
```



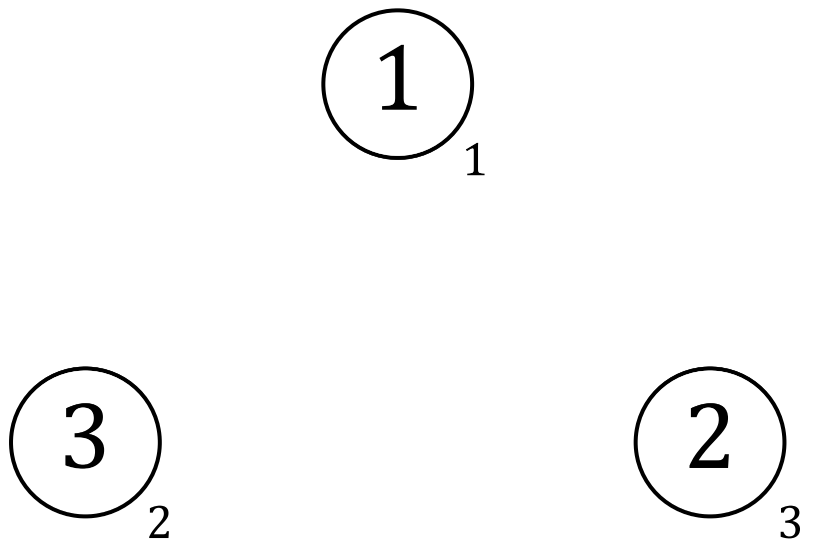
$x = 3$
$i = 2$
$c = 0$

② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2\}$

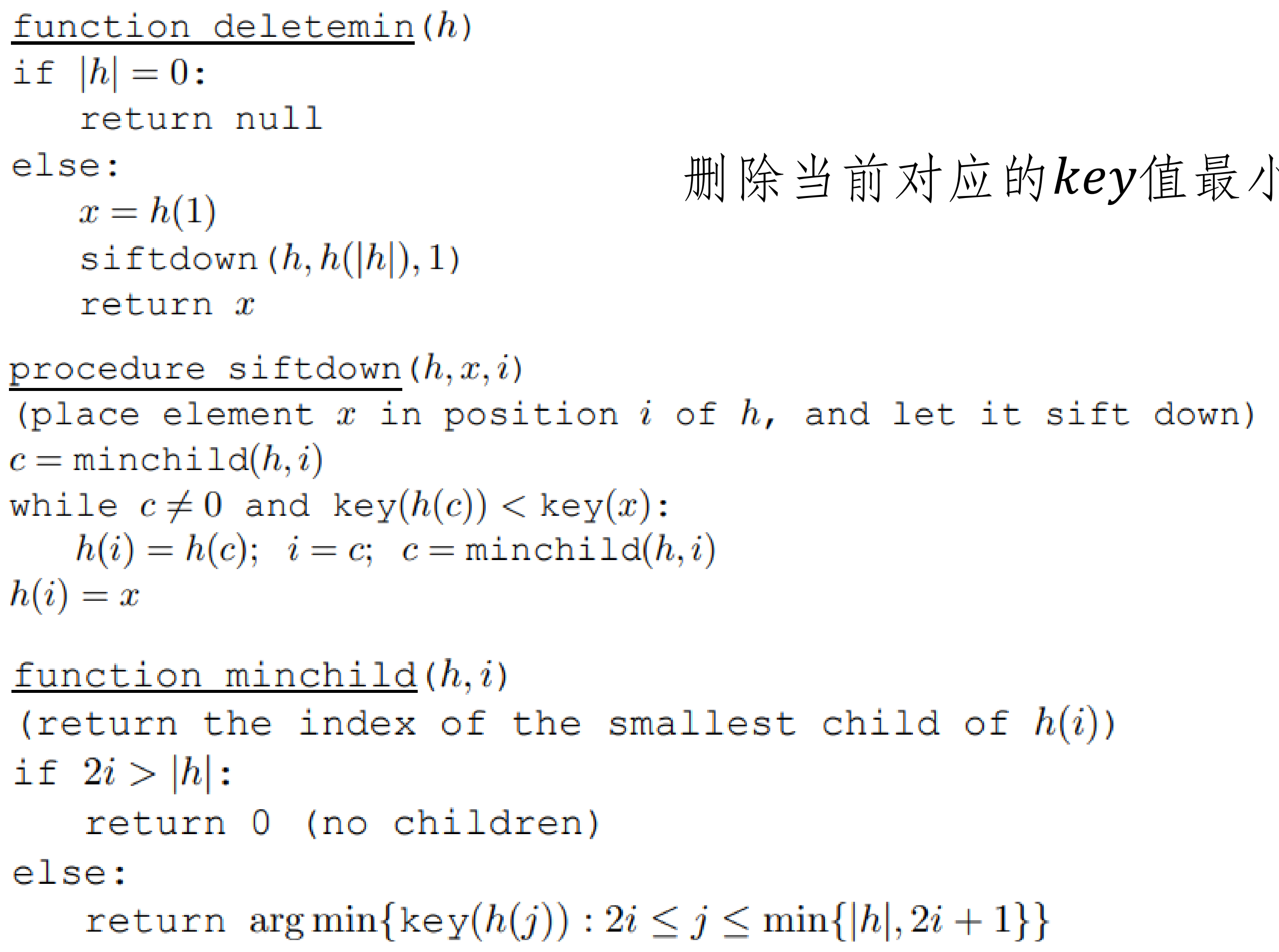
```
function makeheap($S$)
$h =$ empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
$\boxed{\text{return } h}$

procedure siftdown($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it sift down)
$c =$ minchild($h, i$)
while $c \neq 0$ and key($h(c)$) $<$ key($x$):
    $h(i) = h(c)$;  $i = c$;  $c =$ minchild($h, i$)
$h(i) = x$

function minchild($h, i$)
(return the index of the smallest child of $h(i)$)
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \leq j \leq \min\{|h|, 2i+1\}\}$
```



② 习题三

4.16

$|h|$, which returns the number of elements currently in the array;



```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    siftdown(h, h(|h|), 1)
    return x

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```

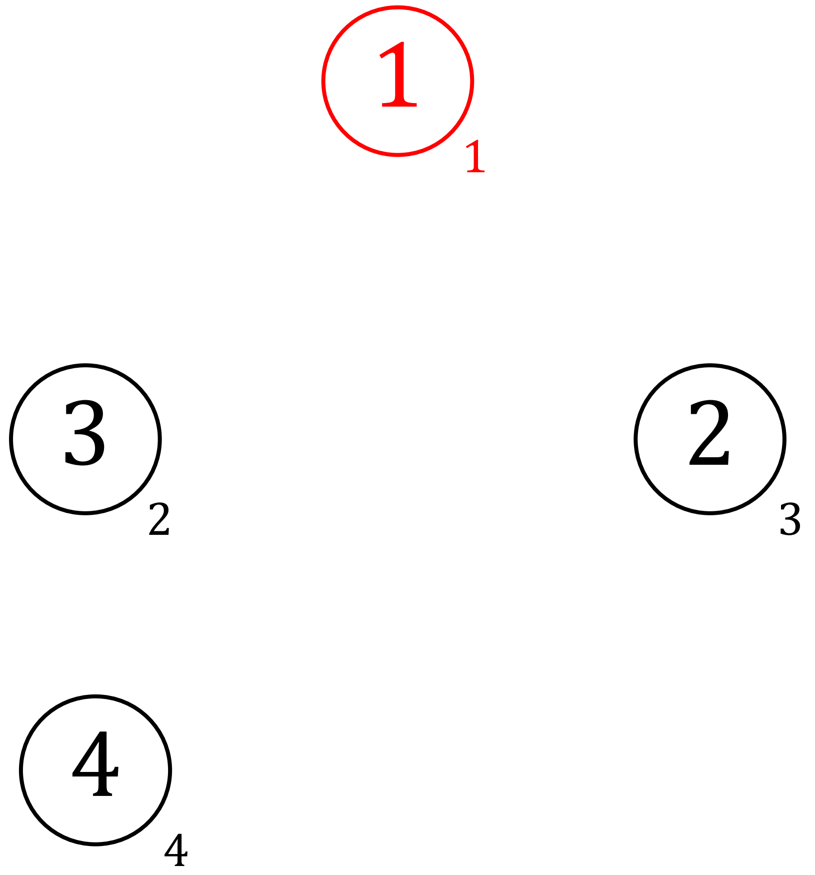
删除当前对应的*key*值最小的元素，即堆顶元素，并维护堆

2 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

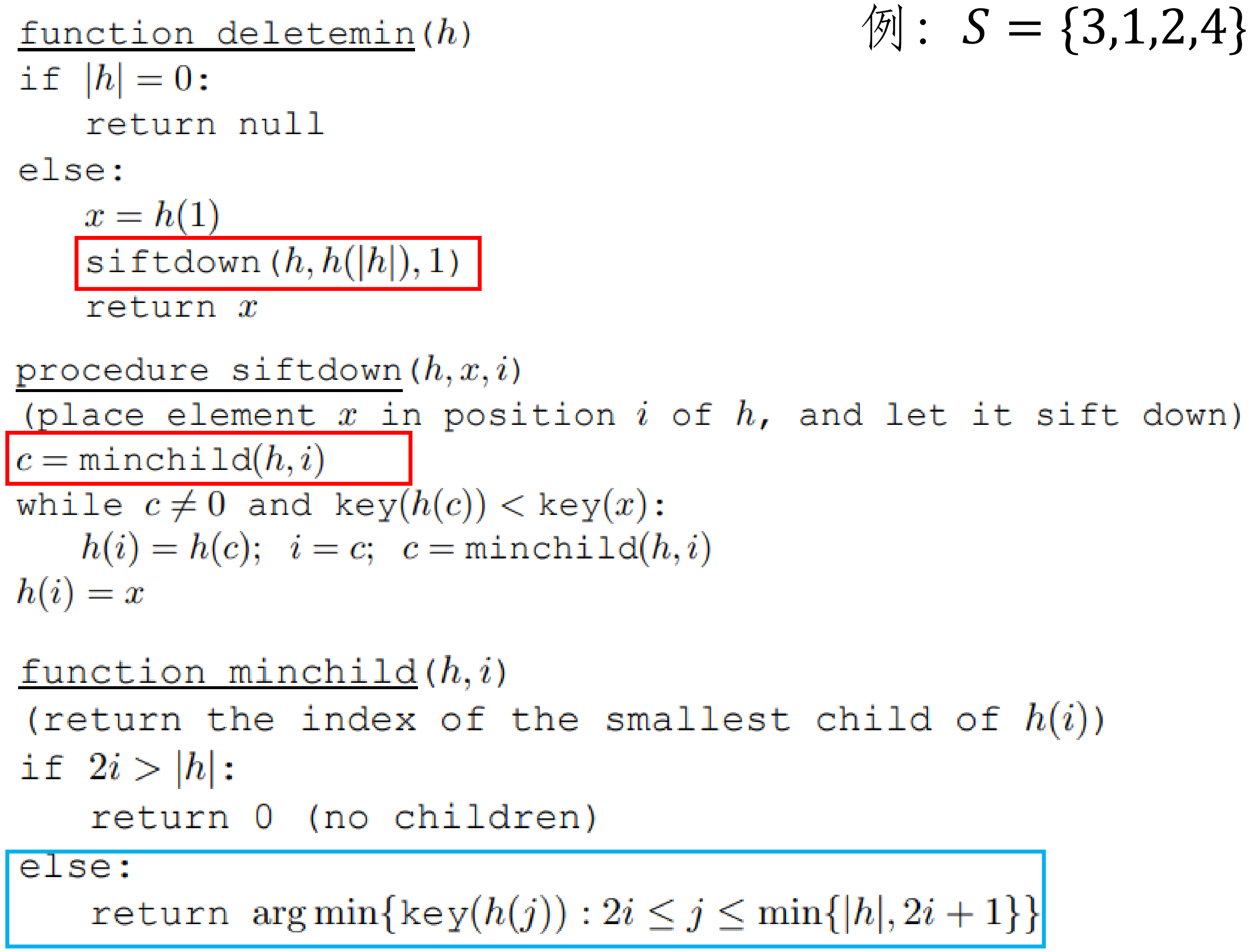
```
function deletemin(h)
if |h| = 0:
    return null
else:
    $\boxed{x = h(1)}$
    siftdown(h, h(|h|), 1)
    return $x$

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
$c = $ minchild($h, i$)
while $c \neq 0$ and key($h(c)$) < key($x$):
    $h(i) = h(c)$;  $i = c$;  $c = $ minchild($h, i$)
$h(i) = x$

function minchild(h, i)
(return the index of the smallest child of h(i))
if $2i > |h|$:
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \le j \le \min\{|h|, 2i+1\}\}$
```



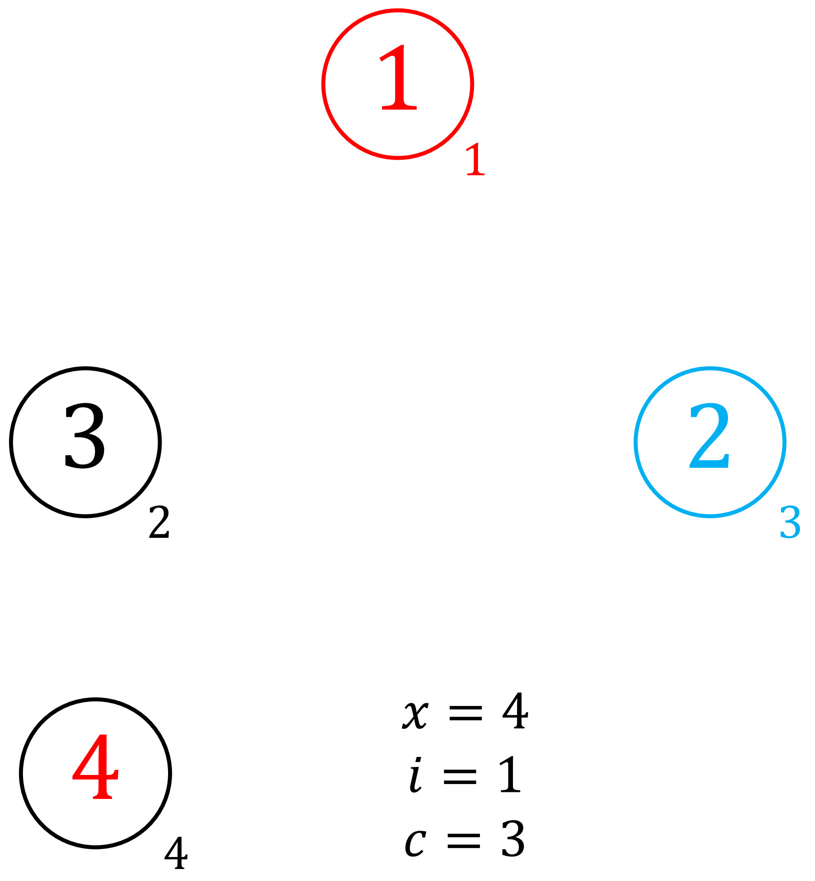
② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例：$S = \{3,1,2,4\}$



```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    siftdown(h, h(|h|), 1)
    return x

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



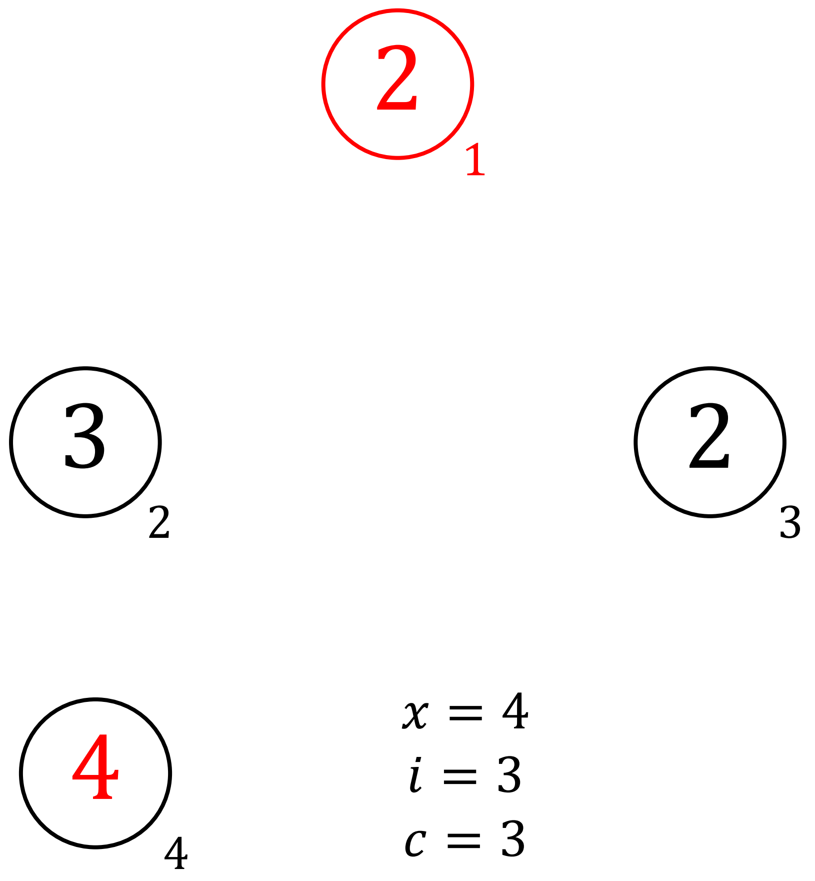
$x = 4$
$i = 1$
$c = 3$

② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    siftdown(h, h(|h|), 1)
    return x

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



$x = 4$
$i = 3$
$c = 3$

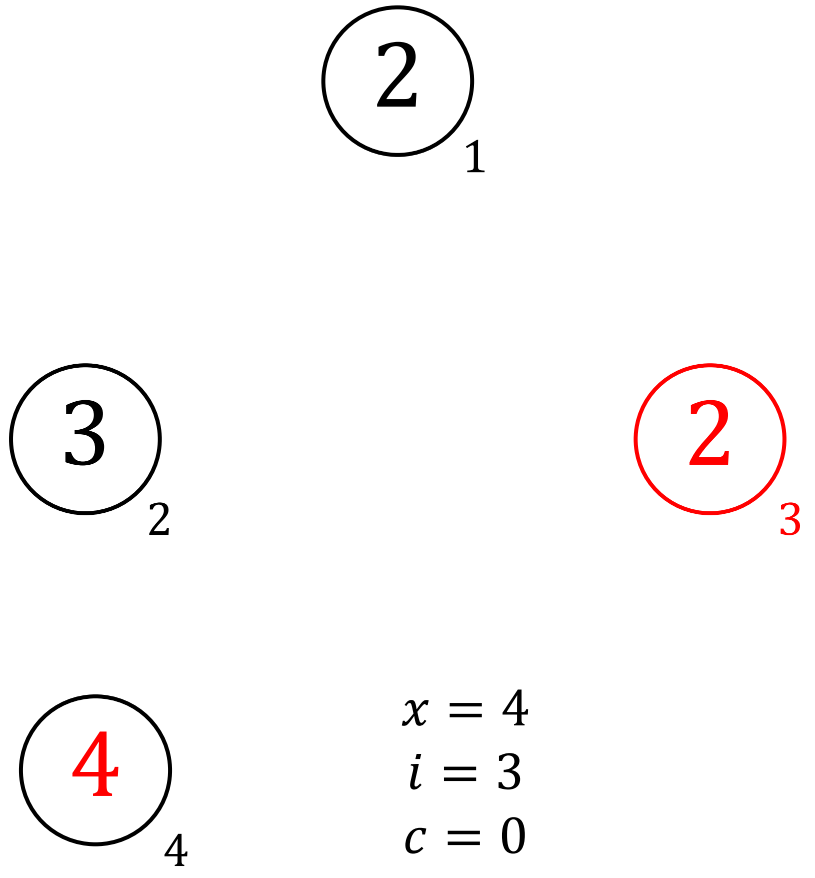
② 习题三

4.16

$|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

```
function deletemin(h)
if $|h| = 0$:
    return null
else:
    $x = h(1)$
    $\boxed{\text{siftdown}(h, h(|h|), 1)}$
    return $x$

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
$c = \text{minchild}(h, i)$
while $c \neq 0$ and $\text{key}(h(c)) < \text{key}(x)$:
    $h(i) = h(c)$;  $i = c$;  $\boxed{c = \text{minchild}(h, i)}$
$h(i) = x$

function minchild(h, i)
(return the index of the smallest child of h(i))
if $2i > |h|$:
    return 0  (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : 2i \leq j \leq \min\{|h|, 2i+1\}\}$
```



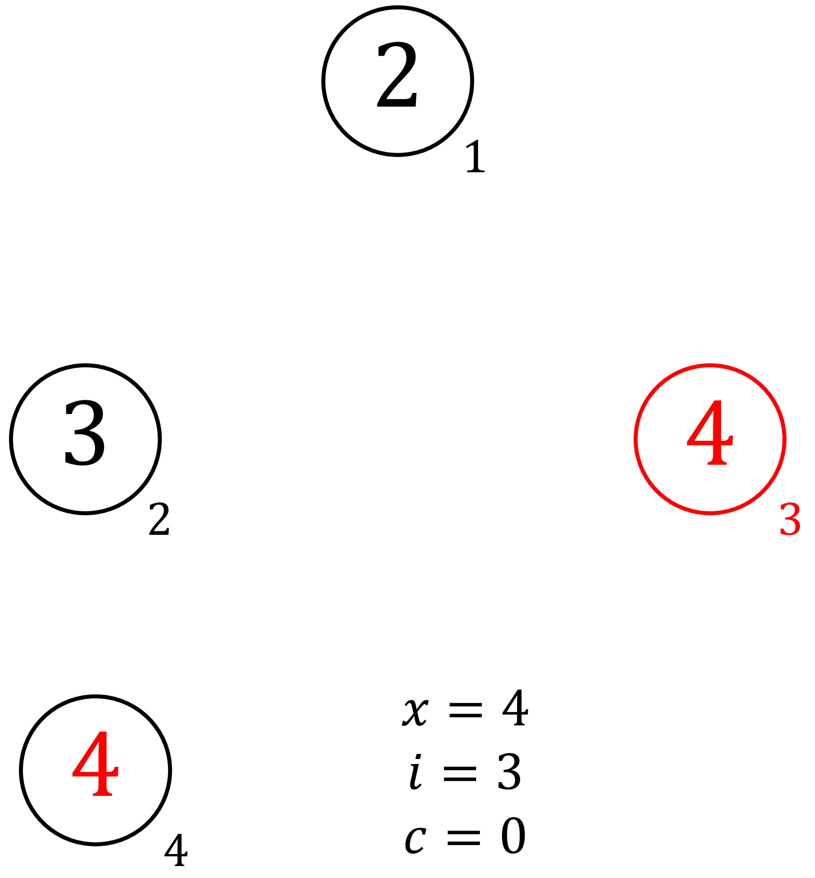
$x = 4$
$i = 3$
$c = 0$

2 习题三

4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    $\boxed{\text{siftdown}(h, h(|h|), 1)}$
    return x

procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
$\boxed{h(i) = x}$

function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



$x = 4$
$i = 3$
$c = 0$

② 习题三

4.16

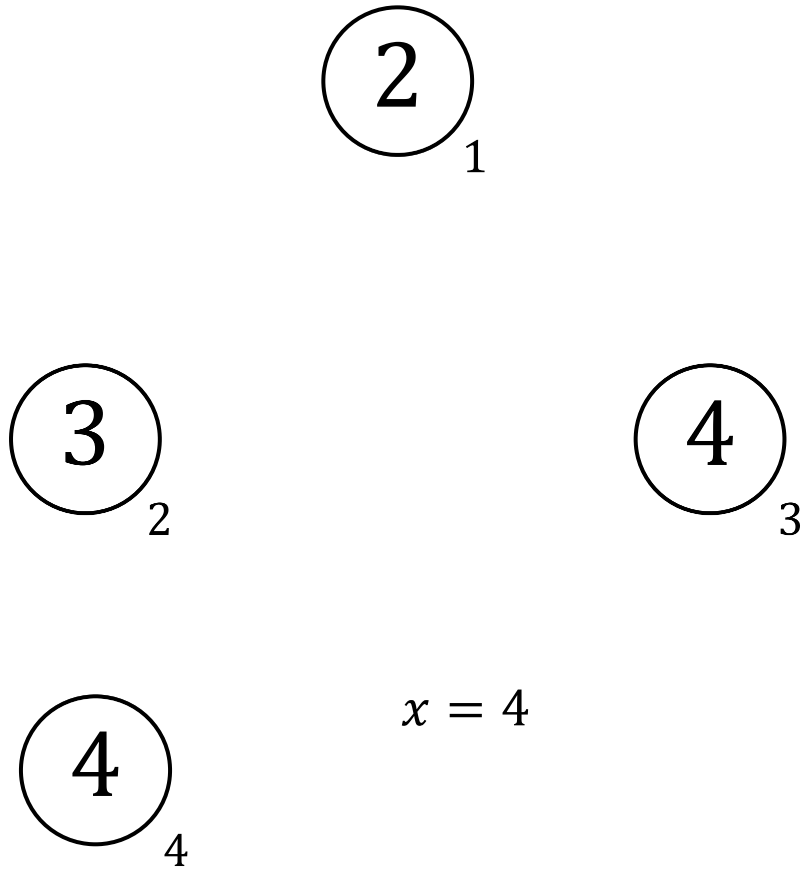
$|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    siftdown(h, h(|h|), 1)
    return x
```

```
procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x
```

```
function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



$x = 4$

② 习题三

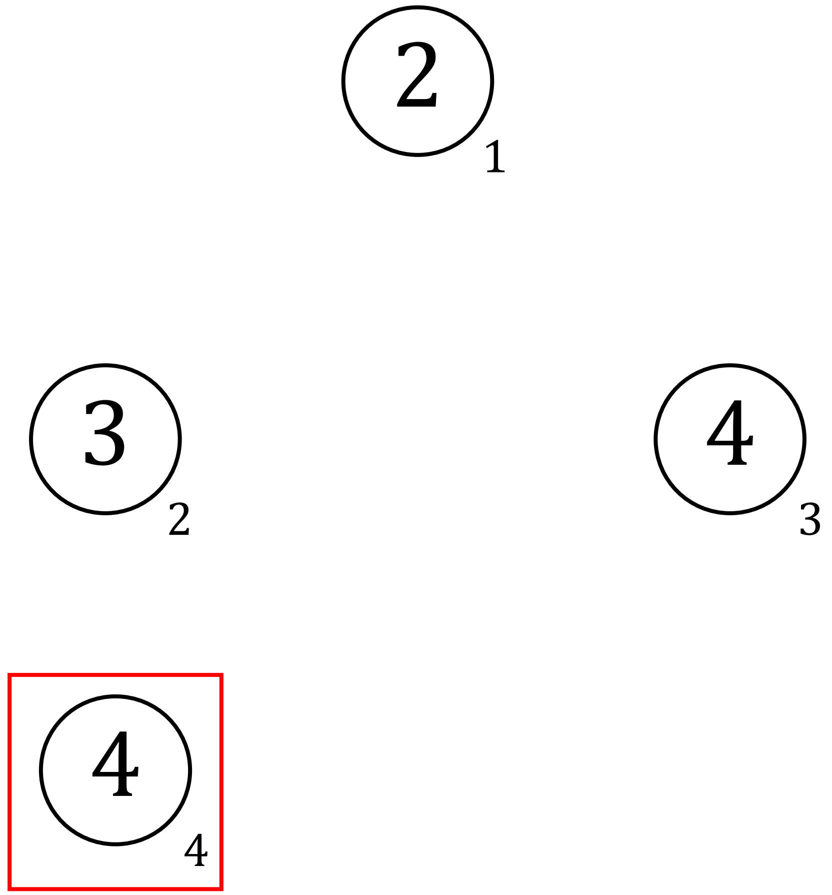
4.16 $|h|$, which returns the number of elements currently in the array;

例： $S = \{3,1,2,4\}$

```
function deletemin(h)
if |h| = 0:
    return null
else:
    x = h(1)
    siftdown(h, h(|h|), 1)
    return x
```

```
procedure siftdown(h, x, i)
(place element x in position i of h, and let it sift down)
c = minchild(h, i)
while c ≠ 0 and key(h(c)) < key(x):
    h(i) = h(c);  i = c;  c = minchild(h, i)
h(i) = x
```

```
function minchild(h, i)
(return the index of the smallest child of h(i))
if 2i > |h|:
    return 0 (no children)
else:
    return arg min{key(h(j)) : 2i ≤ j ≤ min{|h|, 2i + 1}}
```



② 习题三

4.16 $|h|$, which returns the number of elements currently in the array;
h^{-1}, which returns the position of an element within the array

```
procedure insert($h, x$)
bubbleup($h, x, |h| + 1$)
```

向堆中加入新元素，将新元素添加到堆底再上移到合适位置

```
procedure decreasekey($h, x$)
bubbleup($h, x, h^{-1}(x)$)
```

减小了某个元素对应的***key***值后，需判断该元素是否需要上移

```
procedure bubbleup($h, x, i$)
(place element $x$ in position $i$ of $h$, and let it bubble up)
$\boxed{p = \lceil i/2 \rceil}$
while $i \neq 1$ and key($h(p)$) > key($x$):
    $h(i) = h(p)$;  $i = p$;  $\boxed{p = \lceil i/2 \rceil}$
$h(i) = x$
```

若当前节点的***key***值比其父节点小，则将该节点上移

4.16

(c) Show that the *makeheap* procedure takes $O(n)$ time when called on a set of n elements. What is the worst-case input?

```
function makeheap($S$)
$h$ = empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$
```

makeheap:
将队列先全部扔进堆里
从叶子节点开始维护堆
保证所有节点元素的*key*值都比子节点元素小
否则，将节点元素与其*minchild*节点元素交换，不断下移直到满足小根堆

4.16

(c) Show that the *makeheap* procedure takes $O(n)$ time when called on a set of n elements. What is the worst-case input?

```
function makeheap($S$)
$h$ = empty array of size $|S|$
for $x \in S$:
    $h(|h|+1) = x$
for $i = |S|$ downto 1:
    siftdown($h, h(i), i$)
return $h$
```

考察第i个节点，最多移到堆底，交换次数为$\log n - \log i$，复杂度为$O(\log\frac{n}{i})$；总复杂度即为

$$O\left(\sum_{i=1}^{n}\log\frac{n}{i}\right) = O\left(\log\frac{n^n}{n!}\right)$$

由斯特林公式得$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$，则有

$$O\left(\log\frac{n^n}{n!}\right) = O\left(\log\frac{n^n}{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}\right) = O\left(\log\frac{e^n}{\sqrt{2\pi n}}\right) = O(n)$$

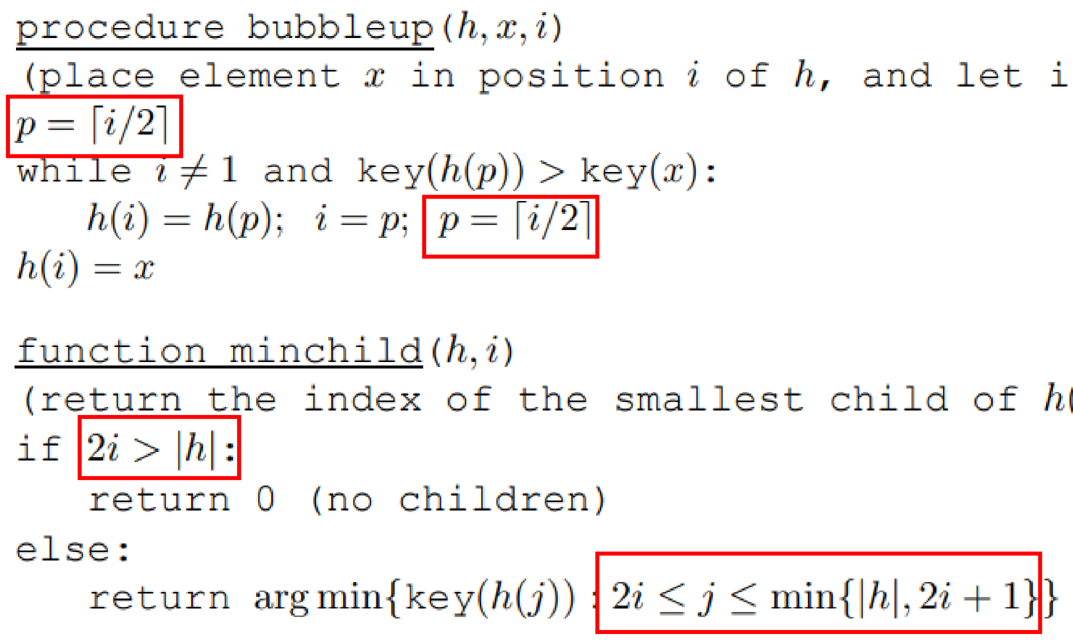
构造最坏解——保证每个子树的根节点都是子树里最大的，按倒序$n, n-1, \cdots, 2, 1$输入

② 习题三

4.16

(d) What needs to be changed to adapt this pseudocode to d-ary heaps

从当前节点下标计算父、子节点下标的过程需要修改



```
procedure bubbleup(h, x, i)
(place element x in position i of h, and let it bubble up)
$\boxed{p = \lceil i/2 \rceil}$
while $i \neq 1$ and key($h(p)$) > key($x$):
    $h(i) = h(p)$;  $i = p$;  $\boxed{p = \lceil i/2 \rceil}$
$h(i) = x$

function minchild(h, i)
(return the index of the smallest child of $h(i)$)
if $\boxed{2i > |h|:}$
    return 0 (no children)
else:
    return $\arg\min\{\text{key}(h(j)) : \boxed{2i \le j \le \min\{|h|, 2i+1\}}\}$
```

$$p = \left\lfloor \frac{i+d-2}{d} \right\rfloor$$

$$di - d + 2 > |h|$$

$$di - d + 2 \le j \le \min\{|h|, di + 1\}$$

5.4

Show that if an undirected graph with n vertices has k connected components, then it has at least $n - k$ edges.

引理：对连通图$G = (V, E)$，满足$|E| \geq |V| - 1$

设图$G = (V, E)$的k个连通分量分别为$G_1 = (V_1, E_1), G_2 = (V_2, E_2), \dots, G_k = (V_k, E_k)$
则有

$$|V| = n = \sum_{i=1}^{k} |V_i|$$

$$|E| = \sum_{i=1}^{k} |E_i|$$

5.4

Show that if an undirected graph with n vertices has k connected components, then it has at least $n - k$ edges.

引理：对连通图$G = (V, E)$，满足$|E| \geq |V| - 1$

设图$G = (V, E)$的k个连通分量分别为$G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_k = (V_k, E_k)$
由引理可知

$$|E_k| \geq |V_k| - 1$$

综上可得

$$|E| = \sum_{i=1}^{k} |E_i| \geq \sum_{i=1}^{k} (|V_i| - 1) = \sum_{i=1}^{k} |V_i| - k = n - k$$

② 习题三

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

归纳奠基：当$|V| = 2$时，结论显然成立
假设当$|V| = k$时，结论成立，即对节点数为k的无向连通图G，当各边边权不同时，该图存在唯一的最小生成树。
当$|V| = k + 1$时，选出G的子图$G' = (V', E')$，其中$|V'| = k$，$E' = \{e = (u, v) | e \in E \ and\ u, v \in V'\}$。显然$G'$为无向连通图，且各边边权不同。由归纳假设可知，$G'$存在唯一的最小生成树$T'$。
考察V中剩余的点v ($\{v\} = V \backslash V'$)，由于G为连通图，v与V'存在连边，设为$E_{vV'}$。$E_{vV'}$中各边边权不同，一定存在边权最小的边，设为e_{min}。$T' \cup e_{min}$构成G的一棵生成树。

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

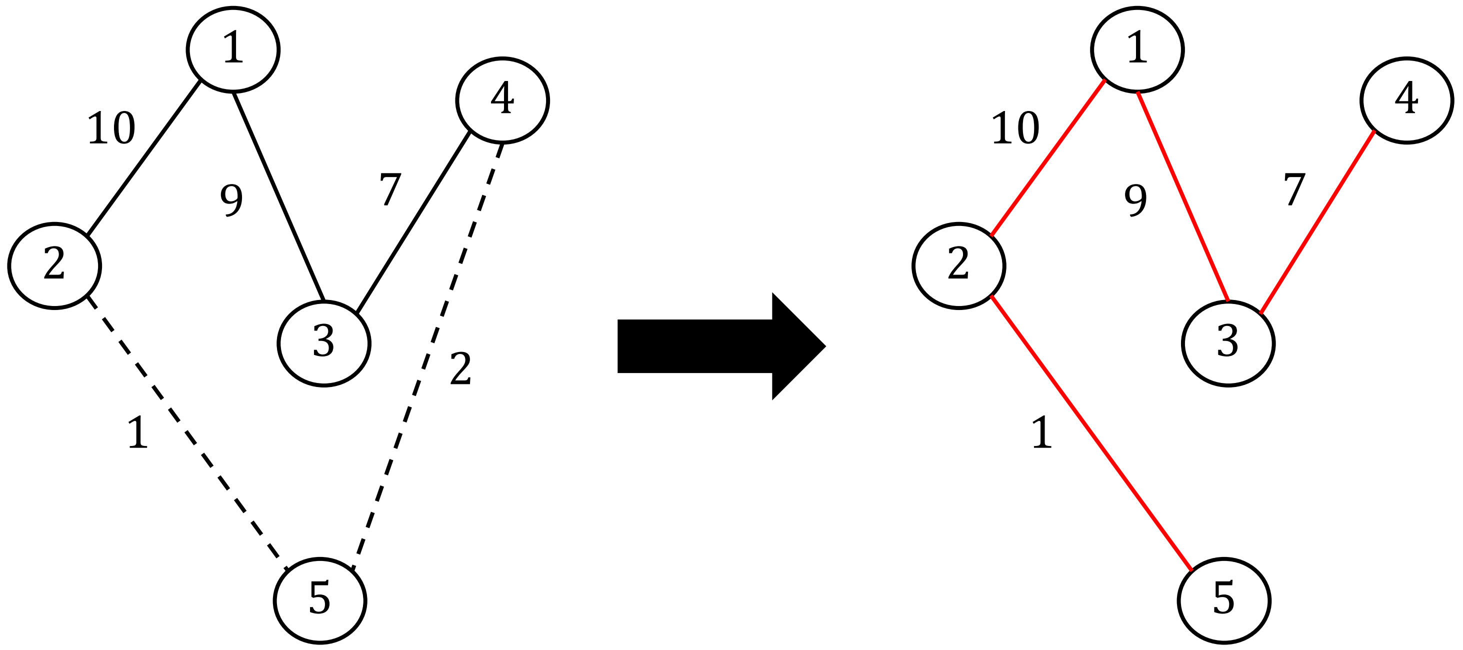
归纳奠基：当$|V| = 2$时，结论显然成立
假设当$|V| = k$时，结论成立，即对节点数为k的无向连通图G，当各边边权不同时，该图存在唯一的最小生成树。
当$|V| = k + 1$时，选出G的子图$G' = (V', E')$，其中$|V'| = k$，$E' = \{e = (u, v) | e \in E \ and \ u, v \in V'\}$。显然$G'$为无向连通图，且各边边权不同。由归纳假设可知，$G'$存在唯一的最小生成树$T'$。
考察V中剩余的点v ($\{v\} = V \backslash V'$)，由于G为连通图，v与V'存在连边，设为$E_{vV'}$。$E_{vV'}$中各边边权不同，一定存在边权最小的边，设为e_{min}。$T' \cup e_{min}$构成G的一棵生成树。
但$T' \cup e_{min}$并不一定是G的最小生成树。

② 习题三

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

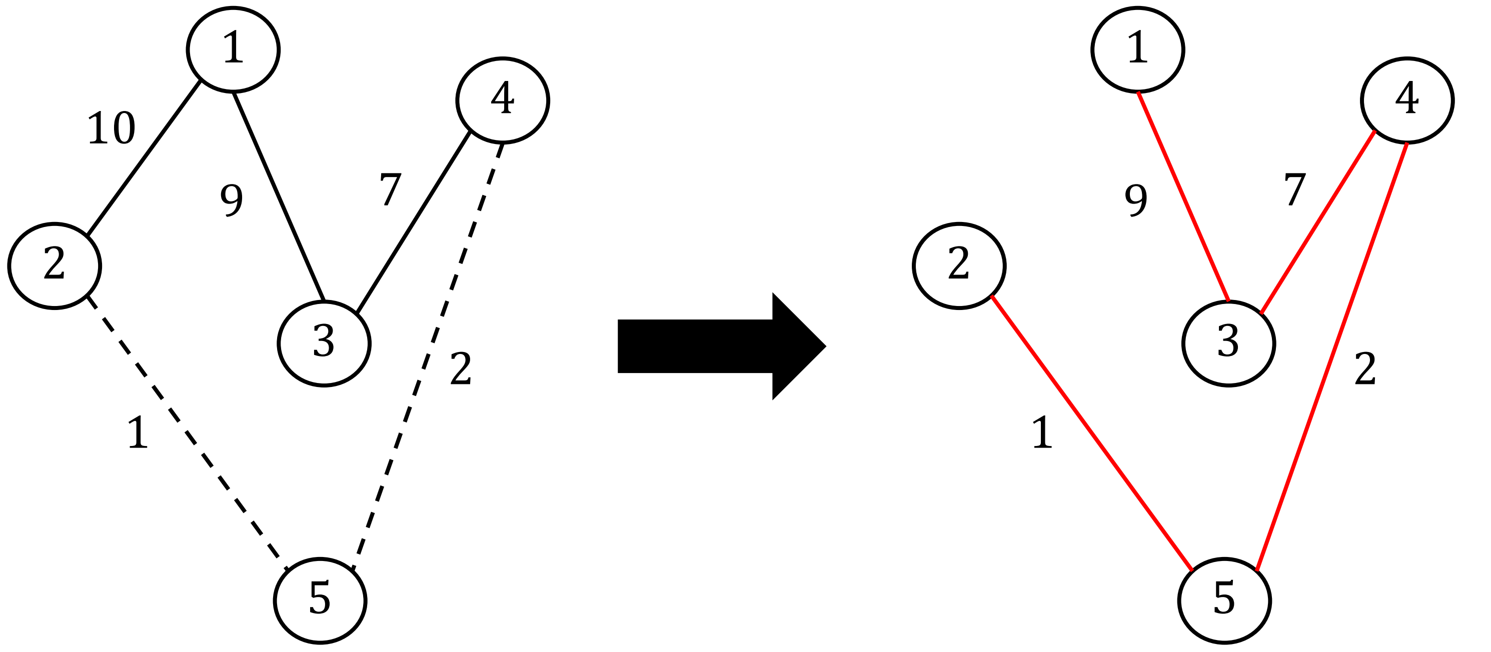


② 习题三

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法



5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

令$T = T' \cup e_{min}$，将$E_{vV'} \setminus \{e_{min}\}$中每条边依次加入$T$。每加入一条边，$T$中出现一个环，去掉环上权重最大的边。最终得到了$G$的一棵最小生成树。

② 习题三

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

令$T = T' \cup e_{min}$，将$E_{vV'} \backslash \{e_{min}\}$中每条边依次加入$T$。每加入一条边，$T$中出现一个环，去掉环上权重最大的边。最终得到了$G$的一棵最小生成树。
尚未说明最小生成树的唯一性——是否存在另一种G'的选择方式，最终得到和T不同但权重相等的最小生成树?

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法

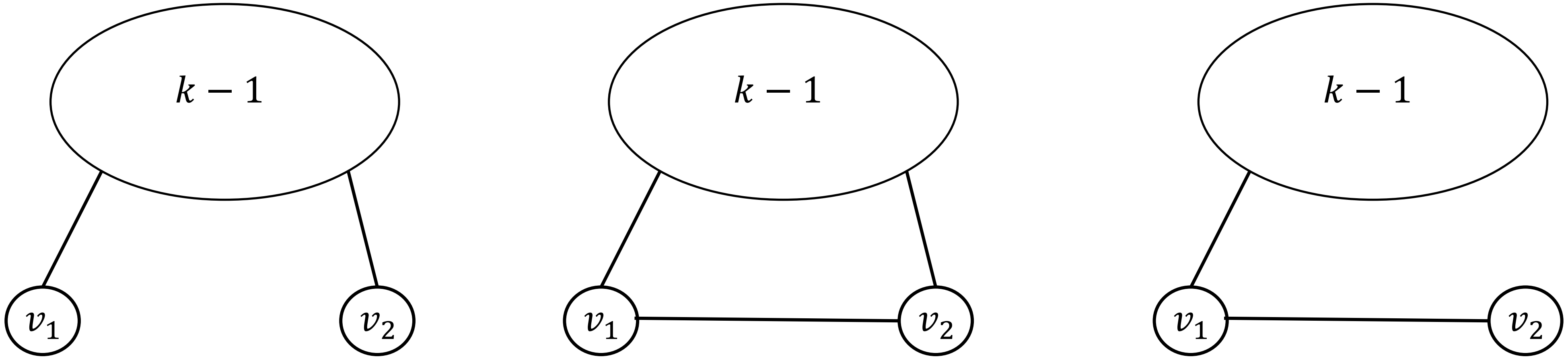
归纳奠基：当$|V| = 2$时，结论显然成立
假设当$|V| \leq k$时，结论成立，即对无向连通图G，当G各边边权不同且节点数小于等于k时，G存在唯一的最小生成树。
当$|V| = k + 1$时
1、证明从给定$|V'| = k$到$|V| = k + 1$，存在一棵最小生成树
2、证明任意两种不同的k点子图选法，$G_1 = (V_1, E_1), G_2 = (V_2, E_2)$，$|V_1| = |V_2| = k$，最终得到的最小生成树相同
　　考察V_1和V_2的$k - 1$个公共点以及各自剩余节点v_1、v_2之间的连通性

② 习题三

5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路一：归纳法



5.6

Let $G = (V, E)$ be an undirected graph. Prove that if all its edge weights are distinct, then it has a unique minimum spanning tree.

思路二：反证法

假设G的最小生成树不唯一，设T_1和T_2分别为G的最小生成树，T_1和T_2权重相等且所有边边权各不相同。
考察T_2中不属于T_1的边集中边权最小的边e_{min}：
将e_{min}加入T_1，必然形成一个环C。
若$\exists e_1 \in C$，$e_1 > e_{min}$，则去掉e_1可构成比T_1权重更小的生成树，矛盾；
若$\forall e_1 \in C$，$e_1 < e_{min}$，则$\exists e_2 \in C$，$e_2 \notin T_2$（否则T_2中有环）且$e_2 < e_{min}$。因此在T_2中插入e_2、去掉e_{min}，可构成比T_2权重更小的生成树，矛盾。

5.10

Let T be an MST of graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

假设$T \cap H$不被包含于H的任意一棵最小生成树，即对H的一棵最小生成树T'，存在边$e \in T \cap H$，$e \notin T'$。
将e插入T'，构成环C，考察C中其他边与e的边权大小
若$\exists e' \in C \backslash \{e\}$，$e'$边权$> e$，则$H$存在一棵权重更小的生成树，矛盾；
若$\forall e' \in C \backslash \{e\}$，$e'$边权$< e$，显然$\exists e' \in C$，$e' \notin T$，此时$G$存在一个权重更小的生成树，矛盾；
若$\exists e' \in C \backslash \{e\}$，$e'$边权$= e$，替换$e'$为$e$，得到$H$的另一棵最小生成树$T''$且$e \in T''$。
对$\forall e \in T \cap H$，存在这样的e'，满足将e'替换为e后，e属于H的一棵最小生成树。

5.10

Let T be an MST of graph G. Given a connected subgraph H of G, show that $T \cap H$ is contained in some MST of H.

假设$T \cap H$不被包含于H的任意一棵最小生成树，即对H的一棵最小生成树T'，存在边$e \in T \cap H$，$e \notin T'$。
将e插入T'，构成环c，考察c中其他边与e的边权大小
若$\exists e' \in c$，e'边权$> e$，则H存在一棵权重更小的生成树，矛盾；
若$\forall e' \in c$，e'边权$< e$ ，显然$\exists e' \in c$，$e' \notin T$，此时G存在一个权重更小的生成树，矛盾；
若存在$e' = e$，则替换e'为e，得到H的另一棵最小生成树T''，使$e \in T''$。
将T''替换为T'，若$\exists e'' \in T \cap H$，$e'' \notin T$‘，重复上述推导，最终可替换得到H的最小生成树$T^{(k)}$满足$T \cap H \subseteq T^{(k)}$。