1.2 三个数的中位数

(1) 思路：遍历一遍三个数，同时找出最小值，最大值，和它们三个数的和，输出和减去最大值减去最小值的结果，即为三个数的中位数

伪代码：

```
输入x
min=max=sum=x
for i := 1 to 2 do
    输入x
    if x < min then
        min=x
    if x > max then
        max=x
    sum += x
return sum-min-max
```

(2) 最坏情况和平均情况下我的算法都需要进行4次比较
第二和第三个输入分别和min和max比较

(3) 最坏情况下至少进行4次比较 证明同上(2)

1.3 集合最小覆盖问题

(1) U={1,2,3,4} S1={1,2}, S2={1,3}, S3={2,4}
不妨假设同样大小的Si选择最靠前的那个
按上述算法得出的最小覆盖为{S1,S2,S3}
而真正的最小覆盖为{S2,S3}

(2) 算法：
记录所有数字在S中出现的次数
然后从S中删除 所有元素次数大于1并且集合最小的Si 并把相应元素次数减一
重复上述步骤，直到再也不能删除一个集合，此时的S为一个集合覆盖

正确性证明：
无论怎么删除都保证每个元素在覆盖中至少出现一次，即删除过的S是U的一个覆盖

(3)不能
反例：U={1,2,3,4} S1={1,2} S2={3,4} S3={1,4} S4={2,4}
按我的算法得出的最小覆盖是{S2,S3,S4}
而真正的最小覆盖是{S1,S2}

1.7多项式计算

数学归纳法：
1.n=0时 p=a[0]=a0 显然正确
2.假设n=k时正确，则当n=k+1时
由n=k时正确可得 前k次循环(HORNER(A[1...k+1],x)后

$$p = a_{k+1}x^k + a_k x^{k-1} ... + a_1$$

第k+1次循环p=p*x+A[0] 得

$$p = a_{k+1}x^{k+1} + a_k x^k ... + a_1 x + a_0$$

n=k+1时得证
3.由数学归纳法可知算法正确

1.8整数相乘

(1)数学归纳法：
1.z=0时，返回值为0 显然成立
2.假设z=k时算法正确，则当z=k+1时，返回值为

$$NT\_MULT(2\cdot y, \lfloor\frac{k+1}{2}\rfloor) + y\cdot((k+1)\ mod\ 2)$$
$$\text{因为}\lfloor\frac{k+1}{2}\rfloor \le k\text{，所以}INT\_MULT\text{函数返回值正确}$$
$$\text{如果}(k+1)mod\ 2\text{为}0\text{，则返回值为}INT\_MULT(2\cdot y, \frac{k+1}{2})\text{等于}y\cdot(k+1)\text{即}y\cdot z$$
$$\text{如果}(k+1)mod\ 2\text{为}1\text{，则返回值为}INT\_MULT(2\cdot y, \frac{k}{2}) + y\text{等于}y\cdot k + y\cdot k\text{即}y\cdot z$$
$$z = k+1\text{时得证}$$

3.由数学归纳法可知算法正确

(2)不妨设

$$z/c = p \quad z\ mod\ c = q\ \text{即}\ z = cp + q$$

数学归纳法：
1.z=0时，返回值为0 显然成立
2.假设z=k时算法正确，则当z=k+1时，返回值为

$$INT\_MULT(c\cdot y, \lfloor\frac{k+1}{c}\rfloor) + y\cdot((k+1)\ mod\ c)$$

$$\text{因为}c \ge 2\ \text{则}\lfloor\frac{k+1}{c}\rfloor \le k\text{，所以}INT\_MULT\text{函数返回值正确}$$

$$= INT\_MULT(c\cdot y, p) + y\cdot q$$

$$= cyp + yq = y(cp+q) = yz$$

z=k+1时得证
3.由数学归纳法可知算法正确

1.9

平均算法复杂度为O(n)

$$\frac{1}{4}\times 10 + \frac{1}{4}\times 20 + \frac{1}{4}\times 30 + \frac{1}{4}\times n = 15 + \frac{1}{4}n$$

1.10

(1)最坏情况是所有元素都不相同

$$\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2}(n-i-1) = \frac{n(n-1)}{2} --- O(n^2)$$

(2)设两个相等的元素在x位置和y位置 (y>x)

$$\sum_{i=0}^{x-1}\sum_{j=i+1}^{n-1} 1 + y - x = \frac{2n-3}{2}x - \frac{x^2}{2} + y$$

$$\frac{2}{n(n-1)}\sum_{x=0}^{n-2}\sum_{y=x+1}^{n-1}(\frac{2n-3}{2}x - \frac{x^2}{2} + y)$$

第二个求和符号拆解后x^2项变成最高次项n^3，第一个求和符号拆解后变成n^4，所以最终结果为$O(n^2)$

(3)

$$P(x,y)=\frac{A_k^{y-1}k^{n-1-y}}{k^n}\ (y\leq k)$$

$$n\to\infty,i=0\text{时能够找到相同的元素，所以复杂度为}\Theta(k)$$

2.2

$$2^k\leq n\leq 2^{k+1}-1,\ k\text{为自然数}$$

$$\lceil log(n+1)\rceil=k+1,\ \lfloor logn\rfloor+1=k+1$$

$$\text{所以}\lceil log(n+1)\rceil=\lfloor logn\rfloor+1$$

2.5

(1)

$$n_0+n_2-1=2\times n_2$$

$$n_0=n_2+1$$

(2)满足

$$n0+n1+n2-1=2\times n_2+n_1$$

$$n_0=n_2+1$$

2.7函数渐近增长率的基本性质

(1)

$$O:\text{任意}f(n)=O(g(n)),\ g(n)=O(h(n))$$

$$\text{则}\lim_{n\to\infty}\frac{f(n)}{g(n)}=c1<\infty,\ \lim_{n\to\infty}\frac{g(n)}{h(n)}=c2<\infty$$

$$\lim_{n\to\infty}\frac{f(n)}{h(n)}=c1\cdot c2<\infty$$

$$\text{所以}f(n)=O(h(n)),\text{即}O\text{满足传递性}$$

$$\Omega:\text{任意}f(n)=\Omega(g(n)),\ g(n)=\Omega(h(n))$$

$$\text{则}\lim_{n\to\infty}\frac{f(n)}{g(n)}=c1>0,\ \lim_{n\to\infty}\frac{g(n)}{h(n)}=c2>0$$

$$\lim_{n\to\infty}\frac{f(n)}{h(n)}=c1\cdot c2>0$$

$$\text{所以}f(n)=\Omega(h(n)),\text{即}\Omega\text{满足传递性}$$

$$\Theta:\text{任意}f(n)=\Theta(g(n)),\ g(n)=\Theta(h(n))$$

$$\text{则}\lim_{n\to\infty}\frac{f(n)}{g(n)}=c1,\ \lim_{n\to\infty}\frac{g(n)}{h(n)}=c2,\ 0\leq c1,c2\leq\infty$$

$$\lim_{n\to\infty}\frac{f(n)}{h(n)}=c1\cdot c2,\ 0\leq c1\cdot c2\leq\infty$$

所以$f(n)=\Theta(h(n))$，即Θ满足传递性

o : 任意$f(n)=o(g(n)),\ g(n)=o(h(n))$

则 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=0,\ \lim\limits_{n\to\infty}\dfrac{g(n)}{h(n)}=0$

$$\lim_{n\to\infty}\frac{f(n)}{h(n)}=0$$

所以$f(n)=o(h(n))$，即o满足传递性

ω : 任意$f(n)=\omega(g(n)),\ g(n)=\omega(h(n))$

则 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=\infty,\ \lim\limits_{n\to\infty}\dfrac{g(n)}{h(n)}=\infty$

$$\lim_{n\to\infty}\frac{f(n)}{h(n)}=\infty$$

所以$f(n)=\omega(h(n))$，即ω满足传递性

(2)

$$O:\lim_{n\to\infty}\frac{f(n)}{f(n)}=1<\infty$$

$f(n)=O(f(n))$ 即O满足自反性

$$\Omega:\lim_{n\to\infty}\frac{f(n)}{f(n)}=1>0$$

$f(n)=\Omega(f(n))$ 即Ω满足自反性

$$\Theta:\lim_{n\to\infty}\frac{f(n)}{f(n)}=1,\ 0<1<\infty$$

$f(n)=\Theta(f(n))$ 即Θ满足自反性

(3)

由$(1)(2)$知，Θ满足传递性和自反性，要证Θ是一个等价关系，只要证Θ满足对称性

对于任意$f(n)=\Theta(g(n)),\ \lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=c,\ 0<c<\infty$

$\lim\limits_{n\to\infty}\dfrac{g(n)}{f(n)}=\dfrac{1}{c},\ 0<\dfrac{1}{c}<\infty,$ 即$g(n)=\Theta(f(n)),\ \Theta$有对称性，原命题得证

(4)

$$
\begin{aligned}
&f(n)=\Theta(g(n))\\
&\Leftrightarrow \lim_{n\to\infty}\frac{f(n)}{g(n)}=c,\ 0<c<\infty\\
&\Leftrightarrow \lim_{n\to\infty}\frac{f(n)}{g(n)}=c>0\ and\ \lim_{n\to\infty}\frac{f(n)}{g(n)}=c<\infty\\
&\Leftrightarrow f(n)=O(g(n)),\ f(n)=\Omega(g(n))
\end{aligned}
$$

(5)

$$f = o(g) \Leftrightarrow lim\frac{f}{g} = c < \infty \Leftrightarrow lim\frac{g}{f} = \frac{1}{c} > 0 \Leftrightarrow g = \Omega(f)$$

$$f = o(g) \Leftrightarrow lim\frac{f}{g} = 0 \Leftrightarrow lim\frac{g}{f} = \infty \Leftrightarrow g = \omega(f)$$

(6)

$$f = o(g),\ h = \omega(g) \Rightarrow lim\frac{f}{g} = 0,\ lim\frac{h}{g} = \infty \Rightarrow f \cap h = \emptyset$$

$$f = \Theta(g),\ h = o(g) \Rightarrow lim\frac{f}{g} = c(0 < c < \infty),\ lim\frac{h}{g} = 0 \Rightarrow f \cap h = \emptyset$$

$$f = \Theta(g),\ h = \omega(g) \Rightarrow lim\frac{f}{g} = c(0 < c < \infty),\ lim\frac{h}{g} = \infty \Rightarrow f \cap h = \emptyset$$

2.8

(1)

$$logn < n < nlogn < n^2 \leq n^2 + logn < n^3 < n - n^3 + 7n^5 < 2^n$$

(2)

$$loglogn < lnn < logn < (logn)^2 < \sqrt{n} < n < nlogn < n^{1+\epsilon}$$
$$< n^2 \leq n^2 + logn < n^3 < n - n^3 + 7n^5 < 2^{n-1} \leq 2^n < e^n < n!$$

2.16

(1)

$$E = log_3 2,\ f(n) = 1 = O(n^{E - log_3 2})$$

$$T(n) = \Theta(n^{log_3 2})$$

(2)

$$P(n) = clogn + clog(\frac{n}{2}) + \ldots + clog(\frac{n}{2^k})$$
$$k = logn,\ P(n) = \frac{c(k+1)*k}{2}, T(n) = \Theta((logn)^2)$$

(3)

$$E = 0, f(n) = cn = \Omega(1)$$
$$T(n) = \Theta(n)$$

(4)

$$E = 1, f(n) = cn = \Theta(n)$$
$$T(n) = \Theta(nlogn)$$

(5)

$$P(n) = cn[logn + log(\frac{n}{2}) + \ldots + log(\frac{n}{2^k})]$$
$$k = logn,\ P(n) = \frac{cn(k+1)*k}{2},\ T(n) = \Theta(n(logn)^2)$$

(6)

$$T(n)=3T(\frac{m}{3})+nlog_3n=nT(1)+n(log_3n+log_3\frac{n}{3}+...+log_3\frac{n}{3^k})=\Theta(n(log_3n)^2),\ k=log_3n$$

(7)

$$E=1,\ f(n)=cn^2=\Omega(n)$$
$$T(n)=\Theta(n^2)$$

(8)

$$E=log_5 7,\ f(n)=n^{\frac{3}{2}}logn=\Omega(n)$$
$$T(n)=\Theta(n^{\frac{3}{2}}logn)$$

(9)

$$T(n)=T(1)+2\times(n-1)=2n-1=\Theta(n)$$

(10)

$$T(n)=T(n-1)+n^c=T(n-2)+(n-1)^c+n^c=T(1)+\sum_{i=2}^{n}i^c=\Theta(n^{c+1})$$

(11)

$$T(n)=T(n-1)+c^n=T(n-1)+c^{n-2}+c^n=T(1)+\frac{c^2(c^{n-1}-1)}{c-1}=\Theta(c^n)$$

(12)

$$n为奇数,\ T(n)=T(n-2)+2n^3-3n^2+3n-1=T(1)+\Theta(n^4)$$
$$n为偶数,\ T(n)=T(n-2)+2n^3-3n^2+3n-1=T(2)+\Theta(n^4)$$
$$T(n)=\Theta(n^4)$$

(13)

$$猜测T(n)=\Theta(n)$$
$$T(n)<=T(\frac{n}{2})+T(\frac{n}{4})+T(\frac{n}{8})+n=(\frac{7}{8}c+1)n$$
$$当c>=8,T(n)<=cn,当c<=8,T(n)>=cn$$
$$所以T(n)=\Theta(n)$$

2.18

$$T(n)=\sqrt{n}T(\sqrt{n})+O(n)$$
$$\frac{T(n)}{n}=\frac{T(\sqrt{n})}{\sqrt{n}}+O(1)$$
$$令S(n)=\frac{T(n)}{n},\ 则S(n)=S(\sqrt{n})+O(1)$$
$$令m=logn,\ R(m)=S(2^m)$$
$$R(m)=R(\frac{m}{2})+O(1)$$
$$R(m)=\Theta(logm),\ S(n)=\Theta(loglogn),\ T(n)=\Theta(nloglogn)$$

2.19

$$a=2,b=2,f(n)=nlogn$$

2.22

输出都是数组A中的最小元素

$$ALG1: T(n) = T(n-1) + O(1) = \Theta(n)$$

$$ALG2: T(n) = 2T(\frac{n}{2}) + O(1) = \Theta(n)$$

2.24

MYSTERY

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{j}1 = \sum_{i=1}^{n-1}\frac{(i+1+n)(n-i)}{2} = \frac{(n-1)n(n+1)}{3} --- O(n^3)$$

PRESKY

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j}1 = \sum_{i=1}^{n}i(i+1) = \frac{n(n+1)(n+2)}{3} --- O(n^3)$$

PRESTIFEROUS

$$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j}\sum_{l=1}^{i+j-k}1 = \frac{n(n+1)(n+2)(3n+1)}{24} --- O(n^4)$$

CONUNDRUM

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=i+j-1}^{n}1 = \frac{n(n+2)(2n-1)}{24} --- O(n^3)$$

3.2冒泡排序

(1)数学归纳法
1.n=2,ifA[0]>A[1] then SWAP(A[0],A[1])，则A[1]>=A[0],显然正确
2.假设n=k时算法正确，则当n=k+1时

$$i := k+1，j := 1\ to\ k, if\ A[j] > A[j+1]\ Then\ SWAP, \text{使得} A[k] = \max_{0\le i\le k} A[i]$$

$$i := k\ to\ 2，\text{由} n = k \text{时算法正确可得} A[0] \le A[1] \ldots \le A[k-1],$$

$$\text{综合} i = k+1 \text{时的循环可得} A[0] \le A[1] \ldots \le A[k-1] \le A[k], \text{所以} n = k+1 \text{时算法正确}$$

3.由数学归纳法可得，冒泡排序正确

(2)最坏情况和平均情况都是

$$\sum_{i=2}^{n}\sum_{j=1}^{i-1}1 = \frac{n(n-1)}{2} --- O(n^2)$$

(3)

$$\text{坏情况原数组逆序，改进不影响比较次数，所以不会影响最坏情况的时间复杂度}$$
$$\text{平均情况比较次数会减少，但是时间复杂度仍然是} O(n^2)$$

3.5

```
Algorithm: PREVIOUS-LARGER(A[1..n])
for i:=1 to n do
    j:=i-1;
    while j>0 and A[j]<=A[i] do
        j:=P[j];
    P[i]:=j;
return P[1..n];
```

正确性证明:

$$\text{算法正确的前提下，循环中}j=j-1\text{向前遍历，如果}j-1>P[j],\text{则}A[j-1]\le A[j]\le A[i],\\ \text{直到}j-1=P[j],\text{才有可能}A[j-1]>A[i],\text{所以该算法肯定正确。}$$

时间复杂度计算:

$$\text{计算}P[i]\text{的比较次数为}k_i\text{，此时}P[1..i-1]\text{已经计算完成}\\ \text{若}A[i-1]>A[i],\text{则}k_i=1\\ \text{若}A[i-1]\le A[i],\text{则}p[i]\text{一定在}p[0..i-1]\text{中，则}k_i>1,p[i]=p[i-k_i+1]\\ \text{同理有}k_{i+1}=1\text{或}k_{i+1}>1\text{且有关系}k_{i+1}\le i-k_i+2,\text{即}k_i+k_{i+1}<=i+2\\ \text{因为}p[i+1]\text{一定在}p[1..i]\text{中，}p[i]=p[i-k_i+1]\\ \text{同理}k_{i+m}=1\text{，或}k_{i+m}\le i-(k_i+...+k_{i+m})+m+1,\text{即}k_i+...+k_{i+m}\le i+m+1\\ \text{令}i=2\text{，}m=n-2,\text{则}k_2+...+k_n\le n+1\\ \text{又}k1=0,\text{所以复杂度为}\Theta(n)$$

3.6

(1)所有的元素像右移动1个（最后一个移到第一个）移动n-k次完成位置调换

```
Algorithm: POSITION-CHANGE(A[1,2,...,n])
for i:= 1 to n-k do
    temp := A[n];
    for j:=2 to n do
        A[j]=A[j-1];
    A[0]=A[n];
return A[1..n]
```

(2)所有元素A[i]放入B[(i-k)%n] 中，B[1..n]即为完成位置调换的结果

```
Algorithm: POSITION-CHANGE(A[1,2,...,n])
B[n] := {0}
for i:= 1 to n do
    B[(i-k)%n] := A[i]
return B[1..n]
```

(3)分别翻转1-k, k+1-n, 再整体翻转，结果即为完成位置调换的结果

```
Algorithm: POSITION-CHANGE(A[1,2,...,n])
reverse(A,1,k);
reverse(A,k+1,n);
reverse(A,1,n);
return A[1..n]
```

3.8微博名人问题

(1)可能有0或1个名人

(2)思路：一开始所有人都可能是名人，然后对他们进行遍历，如果关注了别人，那么自己不是名人，如果没关注那个人那么那个人不是名人，如果这个人没有关注任何人，则对其他人进行遍历，如果其他人都关注了他，那么他是名人，如果有人没有关注他，那么所有人中没有名人。

```
Algorithm: MICROBLOG-CELEBRITY(A[1,2,...,n])
flag:=true
possible:={true}
for i:= 1 to n do
   if possible[i] = true then
     flag:= true
     for j:= 1 to n do
       if j != i then
         if A[i] has no j then
            possible[j] := false;
         else
            flag := false;
   if flag = true then
      for k:=1 to n do
         if k!=i and A[k] has no i then
              return 0;
      return i;
 return 0;
```

3.9最大和连续子序列

(1)遍历所有子串并求和

```
Algorithm: MAXSUM-SUBARRAY(A[1,2,...,n])
max := A[1]; temp := 0;
for i:= 1 to n do
    for j:= i+1 to n do
        for k:= i to j do
            temp := temp + A[k];
        if temp>max then
            max := temp;
        temp := 0;
return max;
```

(2)求和的时候用累加

```
Algorithm: MAXSUM-SUBARRAY(A[1,2,...,n])
max := A[1]; temp := 0;
for i:= 1 to n do
    for j:= i+1 to n do
        temp := temp + A[j];
        if temp>max then
           max := temp;
    temp:= 0;
return max;
```

(3)将数组平分，分别求出求出连续子序列的最大和，同时从中间元素开始分别向左和向右算出最大的和并相加，三个最大和中最大的就是整个数组中连续子序列的最大和

```
Algorithm: MAXSUM-SUBARRAY(A[1,2,...,k])
center := (k+1)/2;
```

```
if center = 0 then
   return A[1];
leftmax := MAXSUM-SUBARRAY(A[1,2,...,center-1])
rightmax := MAXSUM-SUBARRAY(A[center+1,...,k])
midLmax := 0; midRmax := 0; temp := 0;
for i:= center downto 1 do
   temp := temp+A[i];
   if temp > midLmax then
       midLmax := temp;
temp := 0;
for j:= center+1 to k do
   temp := temp+A[j];
   if temp > midRmax then
        midRmax := temp;
midmax := midLmax+midRmax;
return MAX(leftmax,midmax,rightmax);
```

(4)遍历同时求和，如果和小于0就舍弃，记录过程中的最大和即为结果

```
Algorithm: MAXSUM-SUBARRAY(A[1,2,...,n])
max := A[1]; temp := 0;
for i:= 1 to n do
    temp := temp + A[i];
    if temp > max then
          max := temp;
    if temp < 0 then
          temp := 0;
return max;
```

(5)dp[n]：以n为结尾的连续子序列的最大和，以dp[i] := MAX(0,dp[i-1])+A[i]进行动态规划，答案为MAX(dp[i])

```
Algorithm: MAXSUM-SUBARRAY(A[1,2,...,n])
dp := {0}; dp[1] := A[1]; max := A[1];
for i:= 2 to n do
    dp[i] := MAX(0,dp[i-1])+A[i]
    max := MAX(max, dp[i]);
return max;
```