作业1 刘佳璇 231220105

第一章

1.2 三个数的中位数

1)

```
int median(int x, int y, int z)
{
    int min = (x < y) ? x : y;

    if(min > z)
        return min;
    else
    {
        int max = x + y - min;
        return (max < z) ? max : z;
    }
}
```

2)

最坏情况：3 次

平均情况：$2 \times \frac{1}{3} + 3 \times \frac{2}{3} = \frac{8}{3}$ 次

3)

最坏情况至少进行 3 次比较

```
             x<y?
           /      \
      true/        \false
         /          \
       x<z?          y<z?
   true/  \false true/  \false
     y<z?  x       x<z?  y
true/ \false  true/ \false
   y   z         x   z
```

1.3 集合最小覆盖问题

1)

$U = \{1, 2, 3, 4, 5\}$

$S_1 = \{1, 2, 3, 4\}, S_2 = \{2, 3, 4, 5\}, S_3 = \{5\}$

2)

选择 S 中与 U 的交集最大的集合 S_i，从 U 中删除 S_i 与 U 的交集；重复上述步骤直至 U 中的所有元素都被覆盖。

证明：在算法停止前，全集 U 中的所有元素都能被算出的集合覆盖。

3)

不能保证。

反例：$U=\{1,2,3,4\}$

$S_1=\{1,2\},S_2=\{2,3\},S_3=\{3,4\}$

在不同实现中，算法可能先选出 S_1，再选出 S，最终得到的集合覆盖为 $\{S_1,S_3\}$，是最小覆盖；也可能先选出 S_2，再选出S_1，最后选出 S_3，最终得到的集合覆盖为 $\{S_1,S_2,S_3\}$，不是最小覆盖。

1.7 多项式计算

证明：

①当 $n=0$ 时，$p=a_0$，成立.

②假设对 $n=k$，算法成立，则

$$P_k(x)=a_kx^k+a_{k-1}x^{k-1}+\cdots+a_1x+a_0=(((a_nx+a_{n-1})x+a_{n-2})x+\ldots)x+a_0.$$

考虑 $n=k+1$ 时，

$$P_{k+1}(x)=a_{k+1}x^{k+1}+a_kx^k+\cdots+a_1x+a_0.$$

根据 HORNER 算法，将其重写为

$$P_{k+1}(x)=(a_{k+1}x+a_k)x+a_{k-1})x+\ldots)x+a_0$$

展开后与原多项式一致，结论成立.

③故 HORNER 算法对任意 n 均成立.

1.8 整数相乘

1)

当 $z\neq 0$ 时，令 $z=2k+r(r=0,1)$，则$\lfloor\frac{z}{2}\rfloor=k,z\bmod 2=r.z=2\cdot\lfloor\frac{z}{2}\rfloor+(z\bmod 2)$.
故 $y\cdot z=y\cdot(2\cdot\lfloor\frac{z}{2}\rfloor+(z\bmod 2))=2\cdot y\cdot\lfloor\frac{z}{2}\rfloor+y\cdot(z\bmod 2)$.

2)

当 $z\neq 0$ 时，令 $z=ck+r(r=0,1,\cdots,c-1)$，则$\lfloor\frac{z}{c}\rfloor=k,z\bmod c=r.z=c\cdot\lfloor\frac{z}{c}\rfloor+(z\bmod c)$.
故 $y\cdot z=y\cdot(c\cdot\lfloor\frac{z}{c}\rfloor+(z\bmod c))=c\cdot y\cdot\lfloor\frac{z}{c}\rfloor+y\cdot(z\bmod c)$.

1.9 计算平均情况时间复杂度

$$A(n)=\sum_{I\in D_n}Pr(I)\cdot f(I)=10\cdot\frac{1}{n}+20\cdot\frac{2}{n}+30\cdot\frac{1}{4n}+n\cdot\frac{1}{4n}=\frac{1}{4}+\frac{115}{2n}.$$

1.10 UNIQUE算法

1)

最坏情况为数组元素两两不同，此时算法时间复杂度为 $W(n)=(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}$.

2)

两个相等元素出现在不同位置的情况共有 C_n^2 种，时间复杂度平均分布在 1 到 C_n^2 之间。因此

$$A(n)=\sum_{i=1}^{C_n^2} i\cdot\frac{1}{C_n^2}=\frac{1+\frac{n(n-1)}{2}}{2}=\frac{n^2-n+2}{4}.$$

3)

假设 UNIQUE 算法在遍历到 $A[i]$ 与 $A[j]$ 时跳出循环，即 $A[i]$ 与 $A[j]$ 是数列中最小的相等元素对，则

$$Pr(i,j)=\frac{C_k^{j-2}\cdot(k-j+2)}{n^k}\quad(j\le k+1),$$

$$f(i,j)=\sum_{s=n-1}^{n-i} s+(j-i)=\frac{(2n-i-1)\cdot i}{2}+j-i=\frac{2ni-i^2-3i+2j}{2}$$

故

$$A(n)=\sum_{0\le i<j\le k+1} Pr(i,j)\cdot f(i,j)=\sum_{0\le i<j\le k+1}\frac{C_k^{j-2}\cdot(k-j+2)\cdot(2ni-i^2-3i+2j)}{2n^k}$$

第二章

2.2 取对数性质

证明：

当 $2^k\le n\le 2^{k+1}-1$ 时，$左边=\lceil\log(n+1)\rceil=k+1$，$右边=\lfloor\log n\rfloor+1=k+1$. $左边=右边$.

2.5 二叉树节点个数

1)

证明：

因为 T 为 2-tree，所以$n_1=0$，$n=n_0+n_1+n_2=n_0+n_2$. 因为$\sum deg^+(v)=\sum deg^-(v)$，$\sum deg^+(v)=2n_2$，$\sum deg^-(v)=n-1$，所以 $n-1=n_0+n_2-1=2n_2$，整理得 $n_0=n_2+1$.

2)

满足。对任意二叉树，$n_1\neq 0$，$\sum deg^+(v)=n_1+2n_2$，$\sum deg^-(v)=n-1$，所以 $n-1=n_0+n_1+n_2-1=n_1+2n_2$，整理得 $n_0=n_2+1$.

2.7 函数渐近增长率的基本性质

1）传递性

①已知 $f(n)=\mathrm{O}(g(n))$，$g(n)=\mathrm{O}(h(n))$，则 $\exists c_1>0$ 和 $n_1>0$，满足 $0\le f(n)\le c_1g(n)$ 对所有 $n\ge n_1$ 均成立；$\exists c_2>0$ 和 $n_2>0$，满足 $0\le g(n)\le c_2h(n)$ 对所有 $n\ge n_2$ 均成立. 因此有 $0\le f(n)\le c_1c_2h(n)$ 对所有 $n\ge max(n_1,n_2)$ 均成立，即 $f(n)=\mathrm{O}(h(n))$.

②已知 $f(n)=o(g(n))$，$g(n)=o(h(n))$，则 $\forall c>0$，$\exists n_1>0$，满足 $0\le f(n)<cg(n)$ 对所有 $n\ge n_1$ 均成立；$\forall c>0$，$\exists n_2>0$，满足 $0\le g(n)<ch(n)$ 对所有 $n\ge n_2$ 均成立. 因此有 $0\le f(n)<c^2h(n)$ 对所有 $n\ge max(n_1,n_2)$ 均成立，即 $f(n)=o(h(n))$.

③已知 $f(n)=\Omega(g(n))$，$g(n)=\Omega(h(n))$，则 $\exists c_1>0$ 和 $n_1>0$，满足 $0\le c_1g(n)\le f(n)$ 对所有 $n\ge n_1$ 均成立；$\exists c_2>0$ 和 $n_2>0$，满足 $0\le c_2h(n)\le g(n)$ 对所有 $n\ge n_2$ 均成立. 因此有 $0\le c_1c_2h(n)\le f(n)$ 对所有 $n\ge max(n_1,n_2)$ 均成立，即 $f(n)=\Omega(h(n))$.

④已知 $f(n)=\omega(g(n))$, $g(n)=\omega(h(n))$，则 $\forall c>0$， $\exists n_1>0$，满足 $0\le cg(n)<f(n)$ 对所有 $n\ge n_1$ 均成立；$\forall c>0$，$\exists n_2>0$，满足 $0\le ch(n)<g(n)$ 对所有 $n\ge n_2$ 均成立. 因此有 $0\le c^2h(n)<f(n)$ 对所有 $n\ge max(n_1,n_2)$ 均成立，即 $f(n)=\omega(h(n))$.

⑤已知 $f(n)=\Theta(g(n))$, $g(n)=\Theta(h(n))$，则 $\exists c_1>0$, $c_2>0$, $n_1>0$，满足 $0\le c_1g(n)\le f(n)\le c_2g(n)$ 对所有 $n\ge n_1$ 均成立；$\exists c_3>0$, $c_4>0$, $n_2>0$，满足 $0\le c_3h(n)\le g(n)\le c_4h(n)$ 对所有 $n\ge n_2$ 均成立. 因此有 $0\le c_1c_3h(n)\le f(n)\le c_2c_4h(n)$ 对所有 $n\ge max(n_1,n_2)$ 均成立，即 $f(n)=\Theta(h(n))$.

2）自反性

①$\exists\, c\ge 1$, $n_0>0$，满足 $0\le f(n)\le cf(n)$ 对所有 $n\ge n_0$ 均成立，故 $f(n)=\mathrm{O}(f(n))$.

②$\exists\, 0<c<1$, $n_0>0$，满足 $0\le cf(n)<f(n)$ 对所有 $n\ge n_0$ 均成立，故 $f(n)=\Omega(f(n))$.

③$\exists\, 0<c_1<1$, $c_2\ge 1$, $n_0>0$，满足 $0\le c_1f(n)\le f(n)\le c_2f(n)$ 对所有 $n\ge n_0$ 均成立，故 $f(n)=\Theta(f(n))$.

3）等价关系

Θ 关系满足传递性和自反性以证明。以下证明其满足对称性。

已知 $f(n)=\Theta(g(n))$，则 $\exists\, c_1>0$, $c_2>0$, $n_0>0$，满足 $0\le c_1g(n)\le f(n)\le c_2g(n)$ 对所有 $n>n_0$ 均成立。则有 $0\le \frac{1}{c_2}f(n)\le g(n)\le \frac{1}{c_1}f(n)$ 对所有 $n\ge n_0$ 均成立，即 $g(n)=\Theta(f(n))$.

4）满足 Θ 关系的充分必要条件

充分条件：已知 $f(n)=\Theta(g(n))$，则 $\exists\, c_1>0$, $c_2>0$, $n_0>0$，满足 $0\le c_1g(n)\le f(n)\le c_2g(n)$ 对所有 $n>n_0$ 均成立. 则有 $0\le c_1g(n)\le f(n)$ 和 $0\le f(n)\le c_2g(n)$ 对所有 $n>n_0$ 均成立，即 $f(n)=\mathrm{O}(g(n))$ 且 $f(n)=\Omega(g(n))$.

必要关系：已知 $f(n)=\mathrm{O}(g(n))$ 且 $f(n)=\Omega(g(n))$，则 $\exists c_1>0$ 和 $n_1>0$，满足 $0\le f(n)\le c_1g(n)$ 对所有 $n\ge n_1$ 均成立； $\exists c_2>0$ 和 $n_2>0$，满足 $0\le c_2g(n)\le f(n)$ 对所有 $n\ge n_2$ 均成立. 则 $\exists\, n_0>max(n_1,n_2)$，满足 $0\le c_2g(n)\le f(n)\le c_1g(n)$ 对所有 $n>n_0$ 均成立，即 $f(n)=\Theta(g(n))$.

故 $f(n)=\Theta(g(n))\ iff\ f(n)=\mathrm{O}(g(n))$ 且 $f(n)=\Omega(g(n))$.

5）对偶关系

①充分条件：已知 $f=\mathrm{O}(g)$， 则 $\exists c>0$ 和 $n_0>0$，满足 $0\le f\le cg$ 对所有 $n\ge n_0$ 均成立，则 $0\le \frac{1}{c}f\le g$ 对所有 $n\ge n_0$ 均成立，即 $g=\Omega(f)$;

必要条件：已知 $g=\Omega(f)$， 则 $\exists c>0$ 和 $n_0>0$，满足 $0\le cf\le g$ 对所有 $n\ge n_0$ 均成立，则 $0\le f\le \frac{1}{c}g$ 对所有 $n\ge n_0$ 均成立，即 $f=\mathrm{O}(g)$.

②充分条件：已知 $f=o(g)$，则 $\forall c>0$， $\exists n_0>0$，满足 $0\le f(n)<cg(n)$ 对所有 $n\ge n_0$ 均成立，则 $0\le \frac{1}{c}f(n)<g(n)$ 对所有 $n\ge n_0$ 均成立，即 $g=\omega(f)$.

必要条件：已知 $g=\omega(f)$，则 $\forall c>0$， $\exists n_0>0$，满足 $0\le cf(n)<g(n)$ 对所有 $n\ge n_0$ 均成立，则 $0\le f(n)<\frac{1}{c}g(n)$ 对所有 $n\ge n_0$ 均成立，即 $f=o(g)$.

6）交集 $=\emptyset$

①假设 $f(n)=o(g(n))$ 且 $f(n)=\omega(g(n))$，则 $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$, $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=\infty$ 不可能同时满足，故 $o(g(n))\cap\omega(g(n))=\emptyset$.

②假设 $f(n)=\Theta(g(n))$ 且 $f(n)=o(g(n))$，则 $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=c(0<c<\infty)$, $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=0$ 不可能同时满足，故 $\Theta(g(n))\cap o(g(n))=\emptyset$.

③假设 $f(n)=\Theta(g(n))$ 且 $f(n)=\omega(g(n))$，则 $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=c(0<c<\infty)$, $\lim\limits_{n\to\infty}\frac{f(n)}{g(n)}=\infty$ 不可能同时满足，故 $\Theta(g(n))\cap\omega(g(n))=\emptyset$.

2.8 函数按照渐近增长率排序

1)

$$\log n < n < n\log n < n^2 = n^2 + \log n < n^3 < n - n^3 + 7n^5 < 2^n.$$

2)

$$\log\log n < \log n = \ln n < (\log n)^2 < \sqrt{n} < n < n\log n < n^{1+\varepsilon} < n^2 = n^2 + \log n < n^3 < n - n^3 + 7n^5 < 2^{n-1} = 2^n < e^n.$$

2.16 计算 $T(n)$ 的渐近增长率

1)

$\because T(n) = 2T(n/3) + 1,\ \log_b a = log_3 2,$

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

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

2)

$\because T(n) = T(n/2) + c\log n,\ \log_b a = 0,$

$\therefore f(n) = c\log n = \Omega(n^0),$ 但$\forall\varepsilon,\ f(n) = \Omega(n^\varepsilon)$均不成立.

$\therefore$ 不能用 $Master$ 定理求解，将 $T(n)$ 展开求和，得

$$\begin{aligned} T(n) &= T(n/2) + c\log n \\ &= T(n/4) + c(\log n + \log\frac{n}{2}) \\ &= \cdots \\ &= T(1) + c\sum_{i=0}^{k}\log\frac{n}{2^k}\ (k = \log n) \\ &= 1 + c(\log^2 n - \frac{k(k+1)}{2}\log 2) = \Theta(\log^2 n) \end{aligned}$$

3)

$\because T(n) = T(n/2) + cn,\ \log_b a = 0,$

$\therefore f(n) = cn = \Omega(n^{0+1}),$ 且 $\exists k = \frac{3}{4}$ s.t. $f(n/2) \le kf(n)$

$\therefore T(n) = \Theta(n).$

4)

$\because T(n) = 2T(n/2) + cn,\ \log_b a = 1,$

$\therefore f(n) = cn = \Theta(n)$,s

$\therefore T(n) = \Theta(n\log n).$

5)

$\because T(n) = 2T(n/2) + cn\log n,\ log_b a = 1,$

$\therefore f(n) = cn\log n = \Omega(n),$ 但$\forall\varepsilon,\ f(n) = \Omega(n^{1+\varepsilon})$均不成立.

$\therefore$ 不能用 $Master$ 定理求解，将 $T(n)$ 展开求和，得

$$
\begin{aligned}
T(n) &= 2T(n/2) + cn\log n \\
&= 2(2T(n/4) + c\frac{n}{2}\log\frac{n}{2}) + cn\log n = 4T(n/4) + cn(\log n + \log\frac{n}{2}) \\
&= \cdots \\
&= 2^{\log_2 n}T(1) + cn(\log n + \log\frac{n}{2} + \cdots + \log\frac{n}{2^{\log_2 n}})
\end{aligned}
$$

由 2) 知，$T(n) = \Theta(n\log^2 n)$.

6)

$\because T(n) = 3T(n/3) + n\log^3 n,\ \log_b a = 1,$

$\therefore f(n) = n\log^3 n = \Omega(n),$ 但$\forall\varepsilon,\ f(n) = \Omega(n^{1+\varepsilon})$均不成立.

$\therefore$ 不能用 $Master$ 定理求解，将 $T(n)$ 展开求和，得

$$
\begin{aligned}
T(n) &= 3T(n/3) + n\log^3 n \\
&= 3(3T(n/9) + \frac{n}{3}\log^3\frac{n}{3}) + n\log^3 n = 9T(n/9) + n(\log^3 n + \log^3\frac{n}{3}) \\
&= \cdots \\
&= 3^{\log_3 n}T(1) + n\sum_{i=0}^{k}(\log n - \log 3^i)^3\ (k = \log n) \\
&= n + n\log^4 4 + nM\ (M\text{为低次项}) \\
&= \Theta(n\log^4 n)
\end{aligned}
$$

7)

$\because T(n) = 2T(n/2) + cn^2,\ \log_b a = 1$

$\therefore f(n) = cn^2 = \Omega(n^{1+\varepsilon}),$ 且对 $\frac{1}{2} \le c < 1,\ af(n/b) \le kf(n)$

$\therefore T(n) = \Theta(n^2)$

8)

$\because T(n) = 49T(n/25) + n^{3/2}\log n,\ \log_b a = \log_5 7$

$\therefore f(n) = n^{3/2}\log n = \Omega(n^{\log_5 7+\varepsilon}),$ 且对 $\frac{49}{125} \le k < 1,\ af(n/b) \le kf(n)$

$\therefore T(n) = \Theta(n^{3/2}\log n)$

9)

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

10)

$$
\begin{aligned}
T(n) &= T(n-1) + n^c = \cdots = 1^c + 2^c + \cdots + n^c \\
&= \int_1^n x^c dx = \frac{n^{c+1}}{c+1} \\
&= \Theta(n^{c+1})
\end{aligned}
$$

11)

$$
\begin{aligned}
T(n) &= T(n-1) + c^n = \cdots = 1 + c^2 + \cdots + c^n \\
&= \frac{c^{n+1} - 1}{c-1} - c = \Theta(c^n)
\end{aligned}
$$

12)

$T(n) = T(n-2) + 2n^3 - 3n^2 + 3n - 1 = T(n-2) + n^3 + (n-1)^3,$

$T(1) = 1,\ T(2) = 9$

当 n 为奇数时，

$$\begin{aligned}T(n) &= T(n-2) + n^3 + (n-1)^3 = \cdots = T(1) + \sum_{i=2}^{n} i^3 \\ &= \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{2} - n^3 \\ &= \Theta(n^4)\end{aligned}$$

当 n 为偶数时，

$$\begin{aligned}T(n) &= T(n-2) + n^3 + (n-1)^3 = \cdots = T(2) + \sum_{i=3}^{n} i^3 \\ &= \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{2} - n^3 \\ &= \Theta(n^4)\end{aligned}$$

综上，$T(n) = \Theta(n^4)$.

13)

$T(n) = T(n/2) + T(n/4) + T(n/8) + n.$

假设 $T(n) = \mathrm{O}(n)$，则 $\exists\, c > 0 \quad s.t.\ T(n) \le cn$。代入递归式，得到

$$\begin{aligned}T(n) &\le c(\frac{n}{2} + \frac{n}{4} + \frac{n}{8}) + n \\ &= c\frac{7n}{8} + n \le cn,\end{aligned}$$

只需 $c \ge 8$.

假设 $T(n) = \Omega(n)$，则 $\exists\, c > 0\ s.t.\ T(n) \ge cn$。代入递归式，得到

$$\begin{aligned}T(n) &\ge c(\frac{n}{2} + \frac{n}{4} + \frac{n}{8}) + n \\ &= c\frac{7n}{8} + n \ge cn,\end{aligned}$$

只需 $c \le 8$.

综上，$\exists\, c = 8 \quad s.t. \quad T(n) = \Theta(n)$.

2.18 计算 $T(n)$ 的时间复杂度

$$\begin{aligned}T(n) &= \sqrt{n} \cdot T(\sqrt{n}) + \mathrm{O}(n) = n^{\frac{3}{4}} T(n^{\frac{1}{4}}) + 2\mathrm{O}(n) \\ &= \cdots \\ &= n^{1-\log_n 2} \cdot T(2) + \log_2 \log_2 n \cdot \mathrm{O}(n) \\ &= \frac{n}{2} + \log_2 \log_2 n \cdot \mathrm{O}(n) = \mathrm{O}(n \cdot \log\log n)\end{aligned}$$

假设 $T(n) = \Omega(n \cdot \log\log n)$，则 $\exists c > 0\ s.t.\ T(n) \ge cn \cdot \log\log n$，代入递归式，得到

$$\begin{aligned}T(n) &\ge \sqrt{n} \cdot c\sqrt{n} \cdot \log\log\sqrt{n} + \mathrm{O}(n) \\ &= cn \cdot (\log\log n - \log 2) + \mathrm{O}(n) \\ &= cn \log\log n + O(n) \ge cn\end{aligned}$$

对任意 c 均成立，故 $T(n) = \Theta(n \cdot \log\log n)$.

2.19 Master定理不能覆盖的情况

$$T(n) = 2T(n/2) + n\log n$$

2.22 算法 ALG1&2

1)

两个算法的输出均为 $A[1\ldots n]$ 的最小值。

2)

ALG1：$T(n) = T(n-1) + 1,\ T(1) = 1. \quad \therefore T(n) = \Theta(n).$

ALG2：$T(n) = 2T(n/2) + 1,\ T(1) = 1. \quad \therefore T(n) = \Theta(n).$

2.24 计算算法结果和最坏情况下的运行时间

MYSTERY：

$$\begin{aligned} T(n) &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k=1}^{j} 1 = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} j = \sum_{i=1}^{n-1}\frac{(i+1+n)(n-i)}{2} \\ &= \frac{n(n-1)(n+1)}{3} = \mathrm{O}(n^3) \end{aligned}$$

PERSKY：

$$\begin{aligned} T(n) &= \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i}(i+1) = \sum_{i=1}^{n} i(i+1) \\ &= \frac{n(n+1)(n+2)}{3} = \mathrm{O}(n^3) \end{aligned}$$

PRESTIFEROUS：

$$\begin{aligned} T(n) &= \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j}\sum_{l=1}^{i+j-k} 1 = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i+j}(i+j-k) = \sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=1}^{i} 1 \\ &= \sum_{i=1}^{n} i\cdot\frac{i(i+1)}{2} = \sum_{i=1}^{n}\frac{i^3+i}{2} \\ &= \frac{n(n+1)(n+2)(3n+1)}{2} = \mathrm{O}(n^4) \end{aligned}$$

CONUNDRUM：

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=i+j-1}^{n} 1 = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(n+2-i-j),\ (i+j<n+2) \\
&= \sum_{i=1}^{n}\sum_{j=i+1}^{n+1-i}(n+2-i-j) = \sum_{i=1}^{n}\sum_{j=1}^{n+1-2i} 1 \\
&= \sum_{i=1}^{\frac{n}{2}}\frac{(n+2-2i)(n+1-2i)}{2} = \sum_{i=1}^{n}\frac{(n+2-i)(n+1-i)}{2} \\
&= \sum_{i=1}^{n-1}\frac{(i+2)(i+1)}{2} \\
&= \frac{n(n+2)(2n-1)}{24} = \mathrm{O}(n^3)
\end{aligned}
$$

第三章

3.2 冒泡排序

1)

证明：

对循环 $i=k$，取子数组 $A[1\ldots k+1]$ 为循环不变式

当 $i=n$ 时，循环不变式为 $A[1\ldots n]$，成立。

当 $i=k$ 时，for 循环体将子数组 $A[1\ldots k]$ 中最大的元素移动到最右边，其他元素相对位置不变，循环不变式仍成立。

当算法结束时，整个数组的元素不变且已按升序排列。故算法正确。

2)

最坏情况和平均情况的时间复杂度均为 $T(n)=\sum_{i=n}^{2}\sum_{j=1}^{i-1}1=\sum_{i=2}^{n}(i-1)=\frac{n(n-1)}{2}=\Theta(n^2)$.

3)

不会影响最坏情况的时间复杂度

平均情况的时间复杂度常系数会减小，但仍为 $\Theta(n^2)$。

3.5 PREVIOUS_LARGER算法

改进算法：

```
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 [1...n];
```

正确性证明：

对某一给定的 i，$\forall\, k\,(j<k<i)$，$A[k]\le A[i]$ 为循环不变式。

当 $i=1$ 时，$j=0$，k 不存在，循环不变式成立。

在一次循环中，若当前的 $j>0$ 且 $A[j]\le A[i]$，则 $j=P[j]$。$A[P[j]]$ 到 $A[j]$ 之间的元素均小于 $A[i]$。直至 $j=0$ 或 $A[j]>A[i]$，循环结束，循环不变式仍成立。

当算法结束时， 已通过 $A[1\ldots n]$ 计算出 $P[1\ldots n]$。故算法正确。

时间复杂度：

3.6 设计数组算法

1)

```
for i=1 to k do
 |  for j=i to n-1 do
 |   |_ A[j]=A[j+1];
 |_ A[n]=A[i];
```

2)

```
create P[1..n]
for i=1 to k do
 |_ P[i]=A[i];
for i=k+1 to n do
 |_ A[i-k]=A[i];
for i=1 to k do
 |_ A[i+n-k]=P[i];
```

3)

```
reverse(A,n,m):
 |  for i=n to (n+m-1)/2 do
 |   |_swap(A[i],A[n+m-i]);
 |_____________________________

reverse(A,1,k);
reverse(A,k+1,n);
reverse(A,1,n);
```

3.8 微博名人问题

1)

可能至多有一个。

2)

```
isSub(a,b): return (a subscribes b)
isCel[1..n]={true};
for i=1 to n do
 |  if isCel[i]=true then
 |   |   for j=1 to n and j!=i do
 |   |   |  if isSub(i,j) then
 |   |   |   |_ isSub[i]=false;
 |   |   |  else
 |_  |_  |_  |_ isSub[j]=false;
for i=1 to n do
 |  if isCel[i]=true then
 |_  |_ return i;
return 0;
```

3.9 最大和连续子序列

1)

```
max=0;
for i=1 to n do
 |  for j=i to n do
 |   |  sum=0;
 |   |  for k=i to j do
 |   |   |  sum+=S[k];
 |   |   |  if(sum>max) then
 |_  |_  |_  |_ max=sum;
return max;
```

2)

```
max=0;
for i=1 to n do
 |  sum=0;
 |  for j=i to n do
 |   |   sum+=S[j];
 |   |   if(sum>max) then
 |_  |_  |_ max=sum;
return max;
```

3)

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
{
    left-sum = -∞
    sum = 0
    for i = mid downto low
        sum = sum + A[i]
        if sum > left-sum
            left-sum = sum
            max-left = i

    right-sum = -∞
    sum = 0
    for j = mid + 1 to high
        sum = sum + A[j]
```

```
        if sum > right-sum
            right-sum = sum
            max-right = j

    return (max-left, max-right, left-sum + right-sum)
}

FIND-MAXIMUM-SUBARRAY(A, low, high)
{
    if high == low
        return (low, high, A[low])
    else
        mid = floor((low + high) / 2)
        (left-low, left-high, left-sum) = FIND-MAXIMUM-SUBARRAY(A, low, mid)
        (right-low, right-high, right-sum) = FIND-MAXIMUM-SUBARRAY(A, mid+1, high)
        (cross-low, cross-high, cross-sum) = FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
        if left-sum >= right-sum and left-sum >= cross-sum
            return (left-low, left-high, left-sum)
        else if right-sum >= left-sum and right-sum >= cross-sum
            return (right-low, right-high, right-sum)
        else
            return (cross-low, cross-high, cross-sum)
}
```

4)

```
max = S[1];
sum = 0;
for i = 1 to n do
 |   current = current + S[i];
 |   if (current > max) then
 |   |_ max = current;
 |   if (current < 0) then
 |_  |_ current = 0;
return max;
```

5)

```
dp[1] = S[1];
max = S[1];
for i = 2 to n do
 |  dp[i] = max(S[i], dp[i-1] + S[i]);
 |  if (dp[i] > max) then
 |_ |_ max = dp[i];
return max;
```