抄写:

```
INSERTION-SORT(A)
for j=2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1...j-1].
    i = j-1
    while i>0 and A[i] > key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key
```

```
MERGE-SORT(A, p, r)
if p<r
    q = ⌊(p+r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q+1, r)
    MERGE(A, p, q, r)
```

```
MERGE(A, p, q, r)
n1 = q-p+1
n2 = r-q
let L[1...n1+1] and R[1...n2+1] be new arrays
for i=1 to n1
    L[i] = A[p+i-1]
for j=1 to n2
    R[j] = A[q+j]
L[n1+1] = ∞
R[n2+1] = ∞
i=1
j=1
for k=p to r
```

```
        if L[i] ≤ R[j]
            A[k] = L[i]
            i = i+1
        else A[k] = R[j]
            j = j+1
```

1.2-2 要使插入排序优于归并排序.

则，$8n^2 < 64n\lg n$

即 $2^n < n^8$

∴求得 $n \le 43$

故对于不超过43的n值，插入排序优于归并排序.

1.2-3 $100n^2 < 2^n$

求得 $n \geq 15$

∴n的最小值为15

2.2-2 SELECTION-SORT(A)

```
for j = 1 to A.length-1
    k = j
    for i = j+1 to A.length
        if A[i] < A[k]
            k = i
    temp = A[j]
    A[j] = A[k]
    A[k] = temp
```

该算法维持的循环不变式：
在第1~8行的for循环的每次迭代开始时，子数组 $A[1\cdots j-1]$ 由原来 $A[1\ldots n]$ 中从最小数开始由小到大的 $j-1$ 个元素排列组成。
当对前 $n-1$ 个元素进行选择排序后，其已经是 $A[1\cdots n]$ 中 $n-1$ 个较小数的排列序列，故所剩的最后一个元素必为最大值，无需进行多余的排序。
对最好或最坏情况，均为 $\Theta(n^2)$

3.1-5 证明：

① $f(n)=\Theta(g(n)) \Rightarrow f(n)=O(g(n))$ 且 $f(n)=\Omega(g(n))$

由 $f(n)=\Theta(g(n))$，可知，$\exists n_0, c_1, c_2>0$，满足 $\forall n \geq n_0$，有

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$$

故 $\exists c_1, n_0>0$，满足 $\forall n \geq n_0$，有 $0 \leq c_1 g(n) \leq f(n)$，即 $f(n)=\Omega(g(n))$

故 $\exists c_2, n_0>0$，满足 $\forall n \geq n_0$，有 $0 \leq f(n) \leq c_2 g(n)$，即 $f(n)=O(g(n))$

② $f(n)=O(g(n))$ 且 $f(n)=\Omega(g(n)) \Rightarrow f(n)=\Theta(g(n))$

由 $f(n)=O(g(n))$，可知 $\exists n_1, c_3>0$，满足 $\forall n \geq n_1$，有 $0 \leq f(n) \leq c_3 g(n)$

由 $f(n)=\Omega(g(n))$，可知 $\exists n_2, c_4>0$，满足 $\forall n \geq n_2$，有 $0 \leq c_4 g(n) \leq f(n)$

$\therefore$ 取 $n_3=\max\{n_1, n_2\}$。

故 $\exists n_3, c_3, c_4>0$，满足 $\forall n \geq n_3$，有 $0 \leq c_4 g(n) \leq f(n) \leq c_3 g(n)$

即 $f(n)=\Theta(g(n))$

4.1-5 该算法是最大子数组问题的一个非递归、线性时间的算法.

算法思想则是当求得 $A[1\cdots j]$ 的最大子数组后,求 $A[1\cdots j+1]$ 的最大子数组只可能出两种情况; ① 仍为 $A[1\cdots j]$ 的最大子数组; ② 最大子数组为 $A[i\cdots j+1]$. $(1\le i\le j+1)$. 故以此思想设计出该算法.

1-3 行作初始化, n 为 A 数组元素个数. max-sum 为最大和. ending-here-sum 为 $A[i\cdots j]$ 的和. 置其为无穷小

4-13 行进行迭代, 从数组 $A[1]$ 开始, 开始时, $A[i\cdots j]$ 为 $A[1]$. 将其置入 max-sum 与 ending-here-sum 中.

5-9 行进行 $A[i\cdots j]$ 的获取. 若此前 $A[i\cdots j-1]$ 的和为正, 则表明存在 $A[i\cdots j]$ 为最大和的可能, $A[1\cdots j]$ 中包含 $A[j]$ 的最大子数组为 $A[i\cdots j]$.

而若 $A[i\cdots j-1]$ 的和不为正, 那么可视 $A[i\cdots j-1]$ 为累赘, 将其丢弃. 故 $A[1\cdots j]$ 中包含 $A[j]$ 的最大子数组为 $A[j]$.

10-13 行则对两种情况进行比较. 取两种情况的最大值为当前最大子数组, 存入 low, high 与 max-sum 中.

low, high 与 max-sum 为对应最大子数组的有关属性.

进行 n 次迭代, 最终的 low, high 与 max-sum 为对应最大子数组的有关属性.

每次迭代的程序行都固定, 为一常数为 c. 故 n 次迭代用时为 cn.

故该算法时间复杂度为 $\Theta(n)$

4.3-2. 证明: 假设 $T(n)\le c\lg(n-2)$, c 为常数

$$\therefore T(n)\le c\lg(\lceil \tfrac{n}{2}\rceil-2)+1 \le c\lg(\tfrac{n}{2}+1-2)+1$$

$$= c\lg\frac{n-2}{2}+1 = c\lg(n-2)+1-c\lg 2$$

$\therefore$ 当 $c\ge 1$ 时, $T(n)\le c\lg(n-2)$ 成立.

即当 $c\ge 1$ 时, $T(n)\le c\lg n$. 故 $T(n)=O(\lg n)$ 得证.

4.3-9. 令 $m=\lg n$. ∴ $n=2^m$

由 $T(n)=3T(\sqrt{n})+\lg n$. $T(2^m)=3T(2^{\frac{m}{2}})+m$

再设 $S(m)=T(2^m)$. ∴ $S(m)=3S(\frac{m}{2})+m$

假设 $S(m)\le cm^{\lg 3}+dm$.

∴ $S(m)\le 3[c(\frac{m}{2})^{\lg 3}+d(\frac{m}{2})]+m=\frac{3cm^{\lg 3}}{3}+(\frac{3}{2}d+1)m$

$\le cm^{\lg 3}+dm \quad (d\le -2)$

假设 $S(m)\ge cm^{\lg 3}+dm$

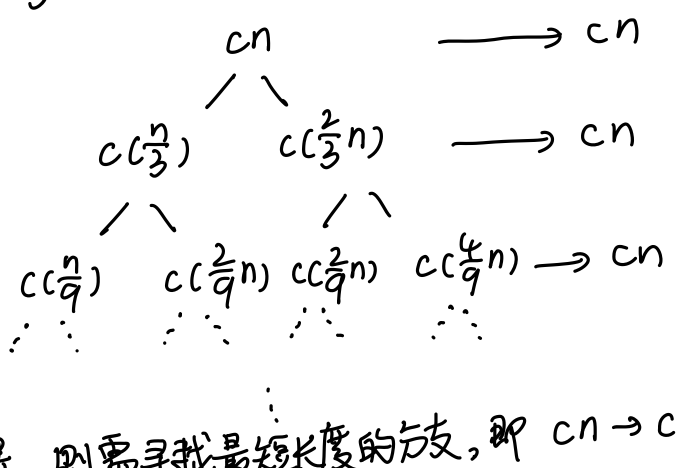
∴ $S(m)\ge 3[c(\frac{m}{2})^{\lg 3}+d(\frac{m}{2})]+m=cm^{\lg 3}+(\frac{3}{2}d+1)m$

$\ge cm^{\lg 3}+dm \quad (d\ge -2)$

∴ $S(m)=O(m^{\lg 3})$

得到 $T(n)=T(2^m)=S(m)=O(m^{\lg 3})=O(\lg^{\lg 3}n)$

4.4-6. $T(n)=T(\frac{n}{3})+T(\frac{2}{3}n)+cn$



需求一个下界，则需寻找最短长度的分支，即 $cn\to c(\frac{n}{3})\to c(\frac{n}{9})\to\cdots\to 1$

∴ $\frac{n}{3^k}=1$ 得 $k=\log_3 n$ ∴ $T(n)\ge cn(\log_3 n+1)\ge cn\log_3 n=\frac{c}{\lg 3}n\lg n$

∴ $T(n)=\Omega(cn\lg n)$

4.5-1. b) $f(n)=n^{\frac{1}{2}}$, $a=2$, $b=4$

$\therefore f(n)=n^{\log_4 2} = \Theta(n^{\log_b a})$

由主定理: $T(n)=\Theta(n^{\frac{1}{2}}\lg n) = \Theta(\sqrt{n}\lg n)$

d) $f(n)=n^2$. $a=2$, $b=4$

$\therefore f(n)=\Omega(n^{\log_b a+\frac{3}{2}})$

且 $\exists\ \frac{1}{8}\le c<1$, 对 $n\geqslant 1$, 有 $af(\frac{n}{b})=2f(\frac{n}{4})\le cf(n)$

由主定理: $T(n)=\Theta(f(n)) = \Theta(n^2)$

4.5-4. $a=4$. $b=2$. $f(n)=n^2\lg n \ne O(n^{2-\varepsilon}) \ne \Theta(n^2) \ne \Omega(n^{2+\varepsilon})$

$\therefore$ 无法应用主定理. 故使用代入法.

假设 $T(n)\le cn^2\lg^2 n$.

$\therefore T(n)\le 4c(\frac{n}{2})^2\lg^2\frac{n}{2} + n^2\lg n = cn^2(\lg n-1)^2+n^2\lg n$

$= cn^2\lg^2 n - 2cn^2\lg n + cn^2 + n^2\lg n$

$= cn^2\lg^2 n + (1-c)n^2\lg n - cn^2\lg n + cn^2$

$\therefore$ 当 $c\geqslant 1$ 时, $T(n)\le cn^2\lg^2 n - cn^2\lg\frac{n}{2} \le cn^2\lg^2 n$

$\therefore T(n)=O(cn^2\lg^2 n)$