Hw 1

Page12：2.1-2, 2.1-4

Page16：2.2-2, 2.2-3

2.1

2.1-2 重写过程 INSERTION-SORT，使之按非升序(而不是非降序)排序。

书上非降序排序：

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

非升序排序：只要对上面第5行修改：A[i] < key

2.1-4 考虑把两个 n 位二进制整数加起来的问题，这两个整数分别存储在两个 n 元数组 A 和 B 中。这两个整数的和应按二进制形式存储在一个$(n+1)$元数组 C 中。请给出该问题的形式化描述，并写出伪代码。

输入：存储两个n位二进制数的n元数组A[1..n], B[1..n]

输出：存储A, B对应二进制数的和的n+1元数组C[1..n+1]

伪码：

```
BinaryAdd(A, B):
# 输入：A[1..n], B[1..n]。从数组最高位A[n], B[n]开始存二进制数。
# 输出：C[1..n+1]。也是从最高位C[n+1]开始存输入两个二进制数的和。
    # carry表进位
    carry = 0
    # 二进制从数组的最高位开始存，所以i=n to 1，循环体内用C[i+1]
    for(i=n to 1):
        sum = A[i]+B[i]+carry
        #  应该是C[i+1]，不是C[i]
        C[i+1] = sum % 2
        carry = floor(sum/2)
    # 最后要将C[1]置为carry
    C[1] = carry
```

- 如果用C[i]存carry，最后就不用C[1]=carry
- 也可以将二进制数倒着存，即从数组最低位开始存，其实这样更简单

法2：如果if-else分类讨论，则要考虑sum=0, 1, 2, 3的情况。

2.2

2. 2-2 考虑排序存储在数组 A 中的 n 个数：首先找出 A 中的最小元素并将其与 $A[1]$中的元素进行交换。接着，找出 A 中的次最小元素并将其与 $A[2]$中的元素进行交换。对 A 中前 $n-1$ 个元素按该方式继续。该算法称为**选择算法**，写出其伪代码。该算法维持的循环不变式是什么？为什么它只需要对前 $n-1$ 个元素，而不是对所有 n 个元素运行？用 Θ 记号给出选择排序的最好情况与最坏情况运行时间。

升序，选择排序

1. 伪码：

   ```
   SelectionSort(A):
   # 输入：A[1..n]
   # 输出：升序排好的A[1..n]
       n = len(A)
       # 最后一个元素不用排，所以是i to n-1
       for i = 1 to n-1:
           minId = i
           # 找i之后的元素更新minId，所以是从i+1开始
           for j = i+1 to n:
               if A[j] < A[minId]:
                   minId = j
           swap(A, i, minId)
   ```

2. 循环不变式（证明算法正确性）：
   1. 初始化
   2. 保持：排序好的部分保持有序
   3. 终止：能正确中止
3. 因为最后一个元素一定是最大的
4. 最好最坏都要依次在n-1, n-2,..,1个元素中找最小元素，然后交换，时间$\Sigma_{i=1}^{n-1} i = n(n-1)/2$

   $\theta(n^2)$

2.2-3 再次考虑线性查找问题(参见练习 2.1-3)。假定要查找的元素等可能地为数组中的任意元素，平均需要检查输入序列的多少元素？最坏情况又如何呢？用 Θ 记号给出线性查找的平均情况和最坏情况运行时间。证明你的答案。

2.1-3 考虑以下查找问题：
输入：n 个数的一个序列 $A=\langle a_1, a_2, \cdots, a_n\rangle$ 和一个值 v。
输出：下标 i 使得 $v=A[i]$ 或者当 v 不在 A 中出现时，v 为特殊值 NIL。
写出**线性查找**的伪代码，它扫描整个序列来查找 v。使用一个循环不变式来证明你的算法是正确的。确保你的循环不变式满足三条必要的性质。

1. 平均需要检查：

   v出现的位置求个期望：n个元素每个出现的概率都是$1/n$，故期望位置$\Sigma_{i=1}^{n} i/n=(n+1)/2$

   最坏需要检查n个元素

2. 平均和最坏都是 $\theta(n)$。证明：由 *1* ，平均和最坏都要检查$\theta(n)$量级的元素，一次检查耗时$\theta(1)$.

9月24日随堂测试

已知定理：

$f(\mathrm{x})$ 是任何连续单调上升函数，且 $f(\mathrm{x})$ 在整数点才可能取整数值，则：

1）$$\lfloor f(x) \rfloor = \lfloor f(\lfloor x \rfloor) \rfloor$$

2）$$\lceil f(x) \rceil = \lceil f(\lceil x \rceil) \rceil$$

证明：

$$\left\lceil \frac{\lceil n/a \rceil}{b} \right\rceil = \left\lceil \frac{n}{ab} \right\rceil \qquad \left\lfloor \frac{\lfloor n/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{n}{ab} \right\rfloor$$

（二者证明一个即可）

取$f(x)=x/b$，$x=n/a$即可。

HW2

2.3

Page22：2.3-3，2.3-5

2.3-3 使用数学归纳法证明：当 n 刚好是 2 的幂时，以下递归式的解是 $T(n)=n\lg n$。

$$T(n)=\begin{cases}2 & 若\ n=2\\ 2T(n/2)+n & 若\ n=2^k, k>1\end{cases}$$

（1）$n=2$时，$T(2)=2=2\lg 2=2$

$k=2$时，$n=2^2=4$，$T(4)=2T(2)+4=8=4\lg 4$

（2）设$n=2^k$时，$T(2^k)=2T(2^{k-1})+2^k=2^k\lg 2^k=k2^k$ 　　$(*)$

证 $n=2^{k+1}$时，$T(2^{k+1})=2T(2^k)+2^{k+1}=2^{k+1}\lg 2^{k+1}=(k+1)2^{k+1}$，

由(*)式，$T(2^{k+1})=2k2^k+2^{k+1}=(k+1)2^{k+1}$.

由（1）（2）得证。

2.3-5 回顾查找问题(参见练习 2.1-3)，注意到，如果序列 A 已排好序，就可以将该序列的中点与 v 进行比较。根据比较的结果，原序列中有一半就可以不用再做进一步的考虑了。**二分查找**算法重复这个过程，每次都将序列剩余部分的规模减半。为二分查找写出迭代或递归的伪代码。证明：二分查找的最坏情况运行时间为 $\Theta(\lg n)$。

2.1-3 考虑以下**查找问题**：
输入：n 个数的一个序列 $A=\langle a_1, a_2, \cdots, a_n\rangle$和一个值 v。
输出：下标 i 使得 $v=A[i]$或者当 v 不在 A 中出现时，v 为特殊值 NIL。
写出**线性查找**的伪代码，它扫描整个序列来查找 v。使用一个循环不变式来证明你的算法是正确的。确保你的循环不变式满足三条必要的性质。

二分查找递归伪码:

```
BinarySearch(A, v, low, high):
    # 二分查找递归算法
    # 输入：数组A[1..n]，待查找的值v，查找区间[low,high]
    # 输出：若找到，则返回一个匹配元素的下标i，若v不在A中，则返回NIL

    mid = floor((low+high)/2)
    # 注意是low≤high，v=A[high]时，low==high
    if low <= high:
        if v == A[mid]:
            return mid
        if v > A[mid]:
            BinarySearch(A, v, mid+1, high)
        else:
            BinarySearch(A, v, low, mid-1)
    return NIL
```

题目要求证明，至少要写出递归式。

最坏情况：v不在A中，或最后一次比较才命中

时间$T(n)=T(n/2)+\theta(1)$，解为$T(n)=\theta(\lg n)$，得证。

3.1

Page31：3.1-2，3.1-4，3.1-6

3.1-2 **证明**：对任意实常量 a 和 b，其中 $b>0$，有

$$(n+a)^b=\Theta(n^b) \tag{3.2}$$

法1.

记$f(n)=(n+a)^b$，$g(n)=n^b$，即证 $n\geq n_0$时，$0\geq c_1g\leq f\leq c_2g$。

> b>0，保证x^b单增。注意a为负的情况。

Note that

$$\begin{aligned} n+a &\le n+|a| \\ &\le 2n \qquad \text{when } |a| \le n\ , \end{aligned}$$

and

$$\begin{aligned} n+a &\ge n-|a| \\ &\ge \frac{1}{2}n \qquad \text{when } |a| \le \tfrac{1}{2}n\ . \end{aligned}$$

Thus, when $n \ge 2|a|$,

$$0 \le \frac{1}{2}n \le n+a \le 2n\ .$$

Since $b > 0$, the inequality still holds when all parts are raised to the power b:

$$0 \le \left(\frac{1}{2}n\right)^b \le (n+a)^b \le (2n)^b\ ,$$

$$0 \le \left(\frac{1}{2}\right)^b n^b \le (n+a)^b \le 2^b n^b\ .$$

Thus, $c_1 = (1/2)^b$, $c_2 = 2^b$, and $n_0 = 2|a|$ satisfy the definition.

法2. 一个具有启发意义的做法，by No.37

极限方法

由于

$$\lim_{n \to +\infty} \frac{(n+a)^b}{n^b} = \lim_{n \to \infty} \left(1 + \frac{a}{n}\right)^b = 1$$

即

$$\forall \varepsilon, \exists N, \forall n > N,\ 1 - \varepsilon \leqslant \frac{(n+a)^b}{n^b} \leqslant 1 + \varepsilon$$

取 $\varepsilon = 0.05$，故取 $c_1 = 0.95, c_2 = 1.05$ 时，有

$$\exists N, \forall n > N,\ 0.95 \leqslant \frac{(n+a)^b}{n^b} \leqslant 1.05$$

即为

$$\exists n_0, \forall n > n_0,\ 0.95 \cdot n^b \leqslant (n+a)^b \leqslant 1.05 \cdot n^b$$

故 $(n+a)^b = \Theta\left(n^b\right)$

3.1-4 $2^{n+1} = O(2^n)$成立吗？$2^{2n} = O(2^n)$成立吗？

（1）证$2^{n+1} = O(2^n)$

即证$n \geq n_1$时，$2^{n+1} \leq c_1 2^n$，不妨取$n_1 = 1, c_1 = 2$。

（2）证$2^{2n} \neq O(2^n)$

即证不存在n_2, c_2，使得$n \geq n_2$时，$2^{2n} \leq c_2 2^n$。因为要满足$c_2 \geq 2^n \to \infty$，故c_2不存在，得证。

3.1-6 **证明：一个算法的运行时间为 $\Theta(g(n))$当且仅当其最坏情况运行时间为 $O(g(n))$，且其最好情况运行时间为 $\Omega(g(n))$。**

$T = \theta(g) \Leftrightarrow n \geq n_0$时，$c_2 g \leq T \leq c_1 g$.　(命题1)

最坏时间$O(g) \Leftrightarrow n \geq n_1$时，$T \leq c_1' g$　(命题2.1)

最好时间$\Omega(g) \Leftrightarrow n \geq n_2$时，$T \geq c_2' g$　(命题2.2)

让$n_0 = \max\{n_1, n_2\}, c_1 = c_1', c_2 = c_2'$，则命题1 $\Leftrightarrow$ 命题2.1+2.2。

3.2

Page34：3.2-1，3.2-4，3.2-5

3.2-1 **证明：若 $f(n)$ 和 $g(n)$ 是单调递增的函数，则函数 $f(n)+g(n)$ 和 $f(g(n))$ 也是单调递增的，此外，若 $f(n)$ 和 $g(n)$ 是非负的，则 $f(n)\cdot g(n)$ 是单调递增的。**

设$x \le y$，则$f(x) \le f(y), g(x) \le g(y)$，$f(x)+g(x) \le f(y)+g(y)$，$f(g(x)) \le f(g(y))$，即$f+g$和$f(g)$单增。

$f, g \ge 0$时，$0 \le f(x)g(x) \le f(y)g(y)$，故$f, g$非负时，$f \cdot g$单增。

★3.2-4 **函数$\lceil \lg n \rceil!$ 多项式有界吗？函数$\lceil \lg \lg n \rceil!$ 多项式有界吗？**

> 这题比较复杂。
>
> 首先，多项式有界的定义：f多项式有界（上界）$\Leftrightarrow \exists\, n_0, c, k$，使$n \ge n_0$时，$f(n) \le cn^k$。

法1

考虑到直接证明阶乘有界很困难，

引理 f 多项式有界 $\Leftrightarrow$ $\lg f(n) = O(\lg n)$

记$r = \lceil \lg n \rceil!$，$s = \lceil \lg \lg n \rceil!$，根据 *引理* ，即判断 $\lg r$和$\lg s$是否等于$O(\lg n)$。

因为：

- $\lg n! = \theta(n \lg n)$：这是因为$n! = \theta(n^n)$，同时取lg。
- $\lceil \lg n \rceil = \theta(\lg n)$

（1）

$$\begin{aligned}\lg(\lceil \lg n \rceil !) &= \Theta(\lceil \lg n \rceil \lg \lceil \lg n \rceil) \\ &= \Theta(\lg n \lg \lg n) \\ &= \omega(\lg n)\,.\end{aligned}$$

所以，$\lg r \ne O(\lg n)$，r不是多项式有界。

（2）

$$\begin{aligned}\lg(\lceil \lg \lg n \rceil !) &= \Theta(\lceil \lg \lg n \rceil \lg \lceil \lg \lg n \rceil) \\ &= \Theta(\lg \lg n \lg \lg \lg n) \\ &= o((\lg \lg n)^2) \\ &= o(\lg^2(\lg n)) \\ &= o(\lg n)\,. \qquad (*)\end{aligned}$$

(*) 式是因为对$a, b > 0$，$\lg^b = o(n^a)$。

所以，$\lg s = O(\lg n)$，s多项式有界。

引理 的证明：

（1）证 f 多项式有界 $\Rightarrow$ $\lg f(n) = O(\lg n)$

f多项式有界，则$\exists\, n_0, c, k$，$n \ge n_0$时，$f(n) \le cn^k$。所以，$\lg f(n) \le kc \lg n$，即$\lg f(n) = O(\lg n)$。

（2）证 f 多项式有界 $\Leftarrow$ $\lg f(n) = O(\lg n)$

（1）的证明倒过来即证（2）。

法2

有的同学想用Stirling公式$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1+\theta(\frac{1}{n}))$直接证有界无界，请参考以下三位同学的证明：

No. 58：r用的Stirling，s没用，但非常漂亮地给出了s的多项式界。

3.2-4 $\lceil \lg n \rceil ! = \sqrt{2\pi m}\left(\frac{m}{e}\right)^m e^{\alpha_m}$ (反证) $\le C\cdot n^k \le C\cdot 2^{mk}$. $\forall m \ge m_0$ (常数k).

$\left(\frac{m}{e}\right)^m \le C\cdot 2^{mk}$. 矛盾! $\therefore \lceil \lg n \rceil !$ 多项式无界.

$\lceil \lg\lg n \rceil = m$. $m! < m^m < (2^m)^m < 2^{2^{m-1}}$

又 $\lg\lg n \ge m-1 \Rightarrow n \ge 2^{2^{m-1}} > m!$ $\therefore \lceil \lg\lg n \rceil !$ 多项式有界.

其实，r 非多项式有界直接给个反例就行（by No.43）：设$n = 2^k$，$r(n) = k!$，非多项式有界。

No. 47：r, s均用Stirling证的。

3.2-4 i. $n! > \sqrt{2\pi n}(n/e)^n$, $\sqrt{2\pi n}(n/e)^n \uparrow$, $n! < 2\sqrt{2\pi n}(n/e)^n$

$\lceil \lg n \rceil \ge \lg n$,

$\lceil \lg n \rceil ! > \sqrt{2\pi \lceil \lg n \rceil}\left(\frac{\lceil \lg n \rceil}{e}\right)^{\lceil \lg n \rceil} \ge \sqrt{2\pi \lg n}\left(\frac{\lg n}{e}\right)^{\lg n}$

$\lg(\lceil \lg n \rceil !) >$ $\lg(\sqrt{2\pi \lg n}(\lg n/e)^{\lg n}) = \text{Const} + \frac{1}{2}\lg^{(2)} n + \lg n \lg \frac{\lg n}{e}$,

记$f(n) :=$ 若 $f(n) = O(n^a), a>0, \exists c, n_0, \forall n \ge n_0$, $0 \le f(n) \le c g(n)$, $g(n) = n^a$,

多项式有界

$\lceil \lg n \rceil !$ 有 $\lg f(n) \le a \lg n + c$,

而 $\lg \frac{\lg n}{e}$ 无上界，对确定的 a，$(\lg \frac{\lg n}{e} - a)\lg n$ 无上界，

即对 $\forall c$，足够大的n使 $\lg f(n) > a\lg n + c$，

$\lceil \lg n \rceil !$ 非多项式有界 .

ii. $\lceil \lg\lg n \rceil \le \lg(2\lg n)$,

$\lceil \lg\lg n \rceil !$ $\cancel{\lceil \lg n \rceil}! < 2\sqrt{2\pi \lceil \lg\lg n \rceil}\left(\frac{\lceil \lg\lg n \rceil}{e}\right)^{\lceil \lg\lg n \rceil} \le 2\sqrt{2\pi \lg(2\lg n)}\left(\frac{\lg(2\lg n)}{e}\right)^{\lg(2\lg n)}$ (*)

$\lg\left(2\sqrt{2\pi \lg(2\lg n)}\left(\frac{\lg(2\lg n)}{e}\right)^{\lg(2\lg n)}\right) = \text{Const} + \frac{1}{2}\lg^{(2)}(2\lg n) + \lg(2\lg n)\lg\frac{\lg(2\lg n)}{e}$,

$\overset{\lg x < x}{<} \text{Const} + \frac{1}{2}(1+\lg\lg n) + (1+\lg\lg n)^2$ （在 $1+\lg\lg n > 0$ 时）

要证明 $(\lg\lg n)^2 = o(\lg n)$，故 $\forall a>0$, $\exists n_0$，使 $\forall n > n_0$,

$\lg(\lceil \lg\lg n \rceil !) <$ $\text{Const} + \frac{1}{2}(1+\lg\lg n) + (1+\lg\lg n)^2 <$ ~~$a\lg n + \text{Const}$~~ $a\lg n + c$，c也是任意的，

$\lceil \lg\lg n \rceil ! = o(n^a)$, $\forall a > 0$，则$\lceil \lg\lg n \rceil !$具有多项式上界，

***3. 2-5　如下两个函数中，哪一个渐近更大些：$\lg(\lg^* n)$还是 $\lg^*(\lg n)$？**

证明：设$2^{2^{\cdot^{\cdot^{2}}}}$共$k$个2，记作$\overset{k}{2}$。记$r = \lg(\lg^* n)$，$s = \lg^*(\lg n)$。

因为$\overset{k-1}{2} < n \le \overset{k}{2}$时，$\lg^* n = k$，故只要判断$n = \overset{k}{2}$，$r, s$哪个渐进更大即可。

因为$r(n) = \lg k$，$s(n) = \lg^* \overset{k-1}{2} = k-1$，明显$s$渐进更大。严格证明：

$\lim\limits_{k\to\infty}\frac{r(n)}{s(n)} = \lim\limits_{k\to\infty}\frac{\lg k}{k-1} = 0$，故$s$渐进更大。