001 算法 Problem Set 1

1.

(a) 还需证明：

第 i 次迭代后满足 $A'[1] \le A'[2] \le \dots \le A'[i+1]$

(b) 循环变量为 i 时的 for 循环

循环不变式表示 $A'[1] \sim A'[i+1]$ 顺序已排好，不再改变

证明：初始 $i=1$，$A[1]$ 满足

保持：由第3、4行代码可知

循环变量为 $i+1$ 时，每次可能使排序调整的

$A[j] < A[j-1]$ 判断式必然包含新元素 $A[i+1]$

即循环不变式保持

(c) 由于对每个 i，循环不变式保持

所以对于 $i=1$ to $n-1$ 均成立

不等式证明：

$\forall i = 1$ to $n-1$，由于循环不变式，$A'[1] \le A'[2] \le \dots \le A'[i+1]$

$\therefore$ (1) 式成立

		代价	次数
2. (a)	$y \leftarrow 0$	c_1	1
	for ($i = n$ downto 0) do	c_2	$n+1$
	$\quad y \leftarrow c_i + xy$	c_3	n

$\Rightarrow \Theta(n)$

(b) 循环变量为 i 时 $y = \sum_{k=i}^{n} c_k x^{n-k}$ 为循环不变式

证明：初始成立，假设为 i 成立，为 $i-1$ 时

$y \leftarrow c_{i-1} + x\sum_{k=i}^{n} c_k x^{n-k} = \sum_{k=i-1}^{n} c_k x^{n-k} \Rightarrow$ 成立

算法正确性：

当 $i=0$ 时循环不变式为 $\sum_{k=0}^{n} c_k x^{k}$

No / Date /

3.

a) $f \in \Theta(g)$

b) $f \in O(g)$

c) $f \in \Theta(g)$

d) $f \in \Theta(g)$

e) $f \in \Theta(g)$

f) $f \in \Theta(g)$

g) $f \in O\cancel{(g)}\ \Omega(g)$

h) $\cancel{f \in \Theta(g)}$ $f \in \Omega(g)$

(i) $\cancel{f \in \Theta(g)}$ $f \in \Omega(g)$

(j) $f \in \Omega(g)$ $(\lg n+1)\lg(\lg n)$ $\lg n$

(k) $f \in \Omega(g)$

(l) $f \in \Theta(g)$ $0.4\log_5 n$ $\log_e n$

(m) $f \in O(g)$

(n) $f \in \Theta(g)$

(o) $f \in \Omega(g)$

(p) $f \in O(g)$ $\log_2 n$ $2^{\log_2 n}$

4.

$$1 \le n^{1/\lg n} \le \lg(\lg^* n) \le \lg^*(\lg n) \le \lg^* n \le 2^{\lg^* n}$$

$$\le \ln(\ln n) \le \lg^2 n \le \sqrt{\lg n} \le \ln n \le n \le \lg(n!) \le n \lg n$$

$$\le n^2 \le n^3 \le 2^{\sqrt{2\lg n}} \le (\sqrt{2})^{\lg n} \le 2^{\lg n} \le 4^{\lg n} \le (\lg n)!$$

$$\le n^{\lg\lg n} = (\lg n)^{\lg n} \le \left(\tfrac{3}{2}\right)^n \le 2^n \le n \cdot 2^n \le e^n$$

$$\le n! \le (n+1)! \le 2^{2^n} \le 2^{2^{n+1}}$$

5. 使数组头尾部各代表一个栈

从两边向中间为顺序进行压栈

从中间向两边为顺序进行弹栈

~~Stack S1~~

~~Stack S2~~

Push1(a):

A[i] = a

i += 1

Push2(b):

A[n-j] = b

j += 1

PoP1():

x = A[i-1]

A[i-1] = 0

i -= 1

return x

Pop2():

x = A[n-j+1]

A[n-j+1] = 0

j -= 1

return x.

主函数 Function():

i = j = 0 // i和j是全局变量

```
i = j = 0    // i和j是全局变量
```

6. 思路：入栈时存入A
出栈时将A中依次放入B，只留一个，pop
随后A、B互换，由B pop出

```
push (a):
    enqueue (A, a)          ----- Θ(1)

pop ( ):
    while (A.size() > 1)    ---- Θ(n)
        x = dequeue (A)     ---- Θ(n)      ⇒ pop 为 Θ(n)
        enqueue (B, x)      ---- Θ(n)
    y = dequeue (A)         ---- Θ(1)
    swap (A, B)
    return y
```