Problem Set 1

Problem 1

(a)

证明 *A'* 中的元素都是 *A* 中的元素。

(b)

2～4行维持的循环不变式是：在第2～4行的for循环的每次迭代开始时，子数组 *A*[j, ..., n] 由原来在 *A*[j, ..., n] 中的元素组成，且 *A*[j] 是*A*[j, ..., n] 中最小的元素。

Initialization:

在第一次循环迭代之前 (当 j = *A.length*时) , 子数组 *A*[j, ..., n] 中仅有一个元素, 显然 成立。

Maintenance:

若前一次循环时循环不变式成立，则在本次循环中可以确定*A*[j-1] 是较小的那一 个元素。

非形式化地，for循环体的第3～4行规定，执行本次循环时，若 *A*[j-1] > *A*[j]，就 会交换 *A*[j-1] 和 *A*[j] 的位置，否则不进行交换。

case 1： 若 *A*[j-1] > *A*[j]，交换位置，*A*[j-1] 的值变成了原先*A*[j] 的值

∵ 原先*A*[j] 的值已经满足是 *A*[j, ..., n] 中的最小值,

∴ 交换后*A*[j-1] 是 *A*[j-1, ..., n] 中的最小值。

case 2： 若 *A*[j-1] < *A*[j] 或 *A*[j-1] = *A*[j],

满足 *A*[j-1] 是*A*[j-1, ..., n] 中的最小值。

∴ 对for循环的下一次迭代将保持循环不变式。

Termination:

当 j = i 时，循环终止，最后一次循环时 j = i + 1。

若 *A*[i] < *A*[i+1]，*A*[i] 与 *A*[i+1] 交换位置，此时*A*[i] 为*A*[i, ..., n] 中的最小值。

综上所述：循环不变式成立。

(c)

循环不变式是：在第1～4行的for循环的每次迭代开始时，子数组*A*[1, ..., i-1] 中的元素由为*A* 中最小的元素组成，切*A*[1, ..., i-1] 已经按从小到大的顺序排好。

Initialization:

在第一次循环迭代之前 (当 i =1时) , 子数组 *A*[1, ..., i-1] 中没有元素，循环不变式 成立。

Maintenance:

若前一次循环不变式成立，则本次循环时，*A*[1, ..., i-1] 将满足不变式的条件。

由(b)中证明的结论可知，第2～4行确定了$A[\,i\,]$ 是 $A[i, ..., n]$ 中最小的元素。

又$\because$ $A[\,1, ..., i-1]$ 中的元素由 A 中最小的(i - 1) 个元素组成，

$\therefore \forall$ x $\in A[1,..., i], A[\,i\,] > x$

又$\because$ $A[1, ..., i-1]$ 已经从小到大排好

$\therefore A[1, ..., i]$ 也已经从小到大拍好

又$\because$ $A[\,i\,]$ 是$A[i, ..., n]$ 中最小的元素

$\therefore A[\,i\,]$ 是 A 中第 i 个最小的元素

$\therefore$ $A[1, ..., i]$ 中的元素由 A 中最小的 i 个元素组成且按照从小到大的顺序排好。

Termination:

当 i = n 时，循环终止，最后一次循环时 i = n。

此时$A[1, ..., n-1]$ 满足循环不变式，

$\because$ $A[1, ..., n-1]$ 由 A 中最小的 (n-1) 个元素组成

$\therefore \forall$ x $\in A[1, ..., n-1], A[n] > x$

又$\because$ $A[1, ..., n-1]$ 已经从小到大排好

$\therefore A[1, ..., n]$ 也已经从小到大排好

综上所述：循环不变式成立。

在以上证明的最后一步中，得到$A[1, ..., n]$ 按照从小到大的顺序排好，

即A'是按照从小到大的顺序排列的，

即$A'[1] \leq A'[2] \leq \cdots \leq A'[n]$

Problem 2

(a)

$T(n) = c_1 + (n+2)c_2 + (n+1)c_3$

$=(c_2 + c_3)n + c_1 + 2c_2 + c_3$

$=\Theta(n)$

(b)

循环不变式是：经过第i个循环后，$y = c_{i-1} + c_{i-2} \cdot x + c_{i-3} \cdot x^2 + \cdots + c_n \cdot x^{n-i}$

$=\sum_{i=j}^{n} c_j \cdot x^{j-i}$

证明：

Initialization:

当 i = n 时，$y = c_n + x\cdot 0 = c_n$

循环不变式成立。

Maintenance:

假设在前一次循环中循环不变式已经成立，即：

$y = \sum_{i+1=j}^{n} c_j \cdot x^{j-i-1}$

$\therefore y' = c_i + xy = c_i + x\cdot (\sum_{i+1=j}^{n} c_j \cdot x^{j-i-1})$

$= \sum_{i=j}^{n} c_j \cdot x^{j-i}$

循环不变式成立。

Termination:

最后一次循环时，i = 0，

$y = c_0 + c_1\cdot x + \cdot\cdot\cdot + c_n \cdot x^n$

$= \sum_{j=0}^{n} c_j \cdot x^j$

循环不变式成立。

综上所述：循环不变式成立。

又$\because \sum_{j=0}^{n} c_j \cdot x^j = \sum_{k=0}^{n} c_k \cdot x^k = P(x)$

$\therefore$ 正确性得证。

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

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

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

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

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

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

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

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

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

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

Problem 4

1 = $n^{1/lgn}$ << $lg(lg^*n)$ << lg^*n = $lg^*(lgn)$ << 2^{lg^*n} << $lnlnn$ << $\sqrt{lgn}$ << lnn<< lg^2n << $2^{\sqrt{2lgn}}$ << $(\sqrt{2})^{lgn}$ << n = 2^{lgn} << $nlgn$ = $lg(n!)$ << n^2 = 4^{lgn} << n^3

<<$(lgn)!$ << $(lgn)^{lgn}$ = n^{lglgn} << $(3/2)^n$ << 2^n << $n \cdot 2^n$ << e^n << $n!$ << $(n+1)!$ << 2^{2^n}<<$2^{2^{n+1}}$

Problem 5

Brief overview:

假设两个栈分别为B、C，B的栈顶从数组下标为0处开始，另一个栈C的栈顶从数组下标为n+1处开始，栈B使用数组的前半部分，栈C使用数组的后半部分，栈B所使用的部分的最右边的位置就是B的栈顶，栈C所使用的部分的最左边的位置就是C的栈顶。

进行$push$ 操作时，栈B向右扩充，栈C向左扩充，当两栈的栈顶相遇时，就溢出。

进行pop 操作时，两栈分别向两边缩减直到栈顶回到初始位置。

Pseudocode:

$PUSH(S,x)$

if $S == B$

    if $B.top == C.top - 1$

        error "$overflow$"

    else $B.top = B.top + 1$

            $B[B.top] = x$

    end if

end if

if $S == C$

    if $C.top == B.top + 1$

        error "$overflow$"

    else $C.top = C.top - 1$

        $C[C.top] = x$

    end if

end if

$POP(S)$

if $S == B$

  if $B.top == 0$

    error "$underlow$"

  else $B.top = B.top - 1$

      return $B[B.top]$

  end if

end if

if $S == C$

  if $C.top == n + 1$

      error "$underlow$"

    else $C.top = C.top + 1$

        return $C[C.top]$

    end if

  end if

Problem 6

Brief overview:

定义两个栈分别为A，B。当要求要实现的队列增加元素时，将增加的元素压进栈A中；当要求要实现的队列删除元素时，将A中的元素pop进入栈B，因为栈A只能实现后进先出，所以在B中的元素的顺序与原先A中的顺序相反，此时再pop出B中的元素，就能够实现整个队列的先进先出。

Pseudocode:

$ENQUEUE(S, x)$

if $B\ is\ empty$

  $A.push(x)$

else

  do $A.push(B.pop())$ **until** $B\ is\ empty$

  $A.push(x)$

end if

$DEQUEUE(S)$

if $A\ is\ not\ empty$

  do $B.push(A.pop())$ **until** $A\ is\ empty$

  return $B.pop()$

else

  if $B\ is\ empty$

    return $EmptyError$

  else

    return $B.pop()$

  end if

  end if

Running time:

Enqueue: $T(n) = \Theta(1)\ in\ the\ best\ case$ (元素都在A中)

$T(n) = \Theta(n)\ in\ the\ worst\ case$（B非空）

Dequeue: $T(n) = \Theta(1)\ in\ the\ best\ case$ (元素已经都在B中)

$T(n) = \Theta(n)\ in\ the\ worst\ case$ (A非空)

Bonus Problem

Brief Overview:

定义一个数组A，初始时A为空数组。当执行add操作时，将元素从数组一端添加进入数组A。当执行remove操作时 使用random()实现随机移除。

Pseudocode:

$ADD(A, x)$

$A[A.length] = x$

$A.length = A.length + 1$

$REMOVE(A)$

$i = random(A.length) - 1$

$A.length = A.length - 1$

$swap(A[i], A[A.length])$

return $A[A.length]$

Correctness Argument:

循环不变式是：$A[0], A[1], ..., A[A.length-1]$ 都是A中的元素。

Initialization:

当A中没有元素时，循环不变式显然成立。

Maintenance:

若已经执行了一次add操作使得$A.length$加一

且确定$A[0], A[1], ..., A[A.length-2]$都是$A$中的元素

再次执行add，使得新加入的数为$A[A.length-1]$,

故$A[0], A[1], ..., A[A.length-1]$都是$A$中的元素。

不变式成立。

若已经执行了一次remove，留下的元素都是A中的元素

再次执行remove，除了被删除的元素其他元素都没有改变,

故不变式成立。

Time Complexity:

$T(n) = \Theta(1) in\ all\ cases$