

**CSE 101      Midterm Solutions      2/11/00**

Name: \_\_\_\_\_ Student ID: \_\_\_\_\_

*Conditions:* No notes, books, consultation, or any kind of assistance allowed. The time limit will be announced. Be sure to show your work where indicated.

1. This innocent-looking piece of dynamic programming code does a surprising task:

```

Input: Array  $A$  of size  $n$ 
Set  $x = 0$  and set  $y = 0$ 
For  $i = 1$  to  $n$  do
     $x = \max(x + A[i], 0)$ 
     $y = \max(x, y)$ 
    Print  $A[i]$ ,  $x$ , and  $y$ 
Return  $y$ 
    
```

**(2 pts) (a)** Trace the above algorithm on the input  $A[1], \dots, A[6] = (-2, 2, -1, 3, -6, 1)$ , printing out the values of  $A[i]$ ,  $x$  and  $y$  whenever the line “Print  $A[i]$ ,  $x$ , and  $y$ ” occurs. Do this in a table with columns labeled by values of  $i$ , and rows labeled by  $A[i]$ ,  $x$ , and  $y$ .

$i$		1	2	3	4	5	6
$A[i]$		-2	2	-1	3	-6	1
$x$		0	2	1	4	0	1
$y$		0	2	2	4	4	4

**(2 pts) (b)** Briefly describe what this algorithm does on a general array of integers. (hint: test on other arrays if necessary.)

This algorithm finds the largest sum of any subsequence of  $A[1], A[2], \dots, A[n]$ .

**(2 pts) (c)** Interpret the value printed for  $y$  when  $i = 2$ .

When  $i = 2$ ,  $y$  is the largest sum of any subsequence of  $A[1], A[2]$  which in this case is just the value of  $A[2]$ .

**2.** The algorithm **Bubble Sort** sorts a list of integers by successively comparing the  $i$ th number to the  $(i + 1)$ st number in the list and interchanging them if the  $i$ th number is larger. Here is the full algorithm:

**Bubble Sort**

Input: An array  $A[1], A[2], \dots, A[n]$  of integers.

Output: A listing of  $\{A[1], A[2], \dots, A[n]\}$  in increasing order.

Step 1. Set  $m = n - 1$ .

Step 2. For  $i = 1, 2, \dots, m$ , if  $A[i] > A[i + 1]$ , interchange  $A[i]$  and  $A[i + 1]$ .

Print  $A[1], A[2], \dots, A[n]$ .

Step 3. Decrease  $m$  by 1. If  $m \neq 0$ , return to Step 2. If  $m = 0$  stop and return  $A[1], \dots, A[n]$ .

**(3 pts) (a)** Trace through the above algorithm by listing  $A[1], A[2], \dots, A[n]$  each time the line “Print  $A[1], A[2], \dots, A[n]$ ” appears. Use as input the array  $A = (A[1], A[2], A[3], A[4], A[5]) = (8, 7, 10, 4, 1)$ .

$m$	$A[1]$	$A[2]$	$A[3]$	$A[4]$	$A[5]$
4	7	8	4	1	10
3	7	4	1	8	10
2	4	1	7	8	10
1	1	4	7	8	10

**(3 pts) (b)** Determine the worst-case time complexity of **Bubble Sort**. Be sure to fully justify your answer.

The algorithm requires  $n - 1$  comparisons the first time Step 2 is executed,  $n - 2$  comparisons the second time, etc. So the total number of comparisons is  $(n - 1) + (n - 2) + \dots + 1$ . It is well known that this sum is equal to  $\frac{n(n-1)}{2}$ . So **Bubble Sort** is  $O(n^2)$ .

**(8 pts) 3.** Suppose that you are given a sorted sequence of *distinct* integers  $A = (a_1, a_2, \dots, a_n)$ . Give a divide-and-conquer  $O(\lg n)$  algorithm in pseudocode to determine if there exists an index  $i$  such that  $a_i = i$ . For example, in  $(-10, -3, 3, 5, 7)$ ,  $a_3 = 3$ . In  $(2, 3, 4, 5, 6, 7)$  there is no such  $i$ . (hint: test the middle element, and break into cases.)

Note that since the  $a_i$ 's are all distinct, if  $a_i > i$  for some  $i$  then  $a_j > j$  for all  $j > i$ . Similarly if  $a_i < i$  for some  $i$  then  $a_j < j$  for all  $j < i$ . So use binary search as follows: first check if the middle element satisfies  $a_i = i$ . If so, return  $a_i = i$ . If not and  $a_i > i$ , repeat the search on the left half of the list and if  $a_i < i$  repeat the search on the right half of the list. The algorithm looks like:

$l = 1$

$r = n$

**Find**( $A, l, r$ )

$m = \lfloor \frac{l+r}{2} \rfloor$

if  $a_m = m$  then return( $a_m = m$ )

if  $a_m < m$  then  $l = m + 1$

if  $a_m > m$  then  $r = m - 1$

if  $r < l$  return("no index  $i$  such that  $a_i = i$ ") else **Find**( $A, l, r$ )