

**CS2223**  
**HW#4**

**DUE:** Monday, April 10

1. (10 points) Suppose you have an array  $A$  of  $n$  elements, but each element is a member of the set  $\{0, \dots, k-1\}$  for some fixed  $k \geq 1$ . That is,  $A[i] \in \{0, \dots, k-1\}$  for  $1 \leq i \leq n$ .

Consider the problem of sorting  $A$ .

*a* Show that the complexity of sorting  $A$  is  $O(n)$ .

*b* Show that the complexity of sorting  $A$  is  $\Omega(n)$ .

2. (6 points) Suppose that  $A[1..n]$  is a list of integers, and  $B[1..n]$  is a list of integers, and  $m$  is an integer. We want to know if there exist  $i, j$ ,  $1 \leq i, j \leq n$ , such that  $A[i] + B[j] = m$ . Show a worst case  $O(n \lg n)$  upper bound on the complexity of solving this problem.

3. (4 points) Do **Problem 6-1 a** on page 142 of our text.

**CS2223**  
**HW#4 SOLUTIONS**

1. **a** To get a linear time upper bound, we use an algorithm which counts the number of occurrences of each  $i \in \{1, \dots, k\}$ .

```

for  $i \leftarrow 0$  to  $k-1$  do
     $Count[i] \leftarrow 0$ 
for  $i \leftarrow 1$  to  $n$  do
     $Count[A[i]] \leftarrow Count[A[i]] + 1$ 
 $j \leftarrow 1$ 
for  $i \leftarrow 1$  to  $k$  do
    for  $l \leftarrow 1$  to  $Count[k]$  do
         $A[j] \leftarrow i$ 
         $j \leftarrow j + 1$ 

```

**b** To get a linear time lower bound, we assume there is an algorithm which solves the problem with  $< n$  steps. The algorithm can't have examined at least one member of  $A$ , say

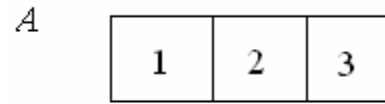
$A[i^*]$ . Form  $B$  from  $A$  by  $B[i] = \begin{cases} A[i], & \text{if } i \neq i^* \\ A[i] + 1 \pmod k, & \text{if } i = i^* \end{cases}$ . Because the algorithm never

examined  $A[i^*]$ , and  $B[i] = A[i]$  for  $i \neq i^*$ , the algorithm sees exactly the same values in  $B$  as it saw in  $A$ . Hence, it must return the same answer in  $B$ , which is now incorrect. By this contradiction, there can not be an algorithm which solves the problem with  $< n$  steps.

2.	HEAPSORT( $A$ )	$O(n \lg n)$
	HEAPSORT( $B$ )	$O(n \lg n)$
	$j \leftarrow n$	$O(1)$
	$i \leftarrow 1$	
	<b>repeat</b>	
	<b>if</b> $A[i] + B[j] = m$ <b>then return</b> $i, j$	
	<b>else if</b> $A[i] + B[j] > m$ <b>then</b> $j \leftarrow j - 1$	
	<b>else</b> $i \leftarrow i + 1$	
	<b>until</b> $i > n$ <b>or</b> $j < 1$	

The body of the **repeat**-loop takes time in  $O(1)$ , and it is executed  $O(n)$  times.

3. BUILD-MAX-HEAP and BUILD-MAX-HEAP' do not always construct the same heap when run on the same input array. On input array



BUILD-MAX-HEAP( $A$ ) constructs



but BUILD-MAX-HEAP'( $A$ ) constructs

