

Practise Midterm Problems

1.) (i) Order the following functions according to their asymptotic growth rate. Indicate which functions belong to the same complexity class. Explain your answers.

$$4n \log n, 2^n \log n, n^2 + 8n \log n, 2^8, 2^n, (n + 4)(n - 6), n!, n^{2^6}, \sqrt{n}, 2^{n/2}, 2^{n^6}, (n - 2)!$$

(ii) Which are true? Explain your answers.

$$5n = O(n \log n)$$

$$12n^2 = O(n \log n)$$

$$\frac{n}{\log n} = \Theta(n)$$

$$4n = \Omega(\sqrt{n} \log n)$$

2.) Assume A is an array of size n containing integers in arbitrary order, A_s is an array of size n containing integers in sorted order, and A_h is an array of size n containing integers arranged in a min-heap. Give the running times (in big-O notation) for the specified operations on a given element x . Give a brief explanation of each entry below the table.

	determine whether x is in the array	determine whether x occurs at least $3n/4$ times	determine whether x is smaller than the smallest element in the array
A (not sorted)			
A_s (sorted)			
A_h (min-heap)			

3.) (i) Determine whether the Master Theorem can be applied. If yes, use it to find a tight asymptotic bound.

$$T(n) = 8T(n/4) + 6n \text{ for } n > 4 \text{ and } T(i) = 4 \text{ for } i \leq 4$$

$$T(n) = 8T(n/4) + 3n^2 + 4n \log n \text{ for } n > 4 \text{ and } T(i) = 14 \text{ for } i \leq 4$$

$$T(n) = 4T(n/3) + n^2 \log n \text{ and } T(i) = 1 \text{ for } i \leq 3$$

$$T(n) = 2T(n/4) + \sqrt{n} \text{ and } T(i) = 1 \text{ for } i \leq 4$$

(ii) Consider the recurrence relation $T(n) = 3T(n - 1) + 2$ with $T(1) = 1$.

Show by induction that $T(n) = O(3^n)$.

4.) (i) Let S be a set of size n containing integers. Design an $O(n)$ time algorithm to determine an integer *not* in S . Give an argument why $\Omega(n)$ is a lower bound for this problem.

(ii) Describe a linear time algorithm for, given an unsorted array of size n , finding the sum of the $2 \log n$ smallest elements.

(iii) Is it true that counting sort will always sort an array of n integers from $\{1, 2, \dots, m\}$ in $O(n)$ time?

5.) For an algorithm solving a problem of input size n , let $B(n)$ be the best-case, $A(n)$ be the average, and $W(n)$ be the worst case time performance. For each statement below give an algorithm seen in class which satisfies the claim. Give the name of the algorithm (or a brief description) and state its time bounds (if not already given).

(i) $B(n) = A(n) = W(n)$ (i.e., all three bounds are asymptotically the same)

(ii) $B(n) = O(1)$ and $W(n) = O(\log n)$

(iii) $A(n) = O(W(n))$, but $W(n) = O(A(n))$ does not hold

6.) (i) Give the running time in big-O notation (in terms of the number of times F is called) for the following code segment:

```
for  $i = 1$  to  $n$  by 1 do
     $k = i$ 
    while  $k > 1$  do
         $F(i, k)$ 
         $k = \lfloor k/4 \rfloor$ 
    endwhile
endfor
```

(ii) Given is a sorted array A of integers. The entries can be negative and there are no duplicates. Describe an $O(\log n)$ time algorithm to determine whether there exists an index i such that $A[i] = i$.

7.) Given a sequence A of n distinct integers, a_1, a_2, \dots, a_n , the goal is to find a longest increasing subsequence of A . For example, if A is 10, 4, 5, 20, 7, 30, 9, then a longest increasing subsequence of A is 4,5,7,9.

(i) Give another longest increasing subsequence in the above example.

(ii) Consider the greedy algorithm described below. Give the running time and explain your answer.

```
for  $i = 1$  to  $n$  do
    Start a (sub)sequence  $S_i$  that has  $a_i$  as the first element
    for  $j = i + 1$  to  $n$  do
        Add  $a_j$  to  $S_i$  if it is greater than the current last element in  $S_i$ 
    endfor
endfor
```

Output the subsequence S_i that has the maximum length.

(iii) Does the above greedy algorithm always output a longest increasing subsequence?