Name: Roll No:

National Institute of Technology Calicut
Department of Computer Science and Engineering
CS2005 Data Structures and Algorithms

$\overline{}$
20

Time: 1 Hour **Test-#1 February 2016** Maximum Marks : 20

(Note: For all the questions given below write your answers only in the space provided in the question paper. Answers written elsewhere will not be evaluated)

Part -A: (Questions 1-3 and 5-8 carries 1 Mark each and Question 4 carries 2 marks) (9 Marks)

1. Let T(n) be the running time of recursive insertion-sort algorithm. Write down a recurrence relation for T(n).

Ans: $T(n) = \begin{cases} T(n-1) + n, & n > 1 \\ 1, & n = 1 \end{cases}$

2. For a problem P, algorithm A_1 and algorithm A_2 run in $O(n^2)$ and $O(n^3)$ time, respectively. Which algorithm is asymptotically quicker? Give the reason. Answer without reason will not carry any marks.

A) both perform equally well

B) A_1 performs quicker

C) A_2 performs quicker

D) Cannot be ascertained from the given information

Ans: D Algorithms A_1 & A_2 running times are given as $O(n^2)$ & $O(n^3)$ respectively. This implies the following:
1) Both the algorithms' running times may be $\Theta(n^2)$ & $\Theta(n^2)$
2) Algorithm A_1's running time may be $\Theta(n)$ and A_2's may be $\Theta(n^2)$
3) Algorithm A_2's running time may be $\Theta(n^2)$ and A_1's running time may be $\Theta(n)$

3. State the trichotomy property of real numbers. Does it hold for asymptotic relations? Justify your answer.

Ans: For any two real numbers, a & b exactly one of the following must hold: $a < b$, $a = b$ or $a > b$.
No. For example, the functions n and $n^{1+\sin n}$ can't be compared using asymptotic relationships. Since the value of the exponent in $n^{1+\sin n}$ oscillates between 0 & 2, taking on all values in between.

4. Solve the following recurrences using Master theorem. Assume T(n) is constant when $n \leq 2$.

a) $T(n) = 9\,T(n/3) + n$

b) $T(n) = 2T(n/2) + n \log n$

(You may write your answers overleaf)

a) $T(n) = 9T(n/3) + n$

In the given recurrence,

$a = 9,\ b = 3,\ f(n) = n$

Compare $n^{\log_b a}$ with $f(n)$ i.e $n^{\log_3 9}$ with n

$\Rightarrow n^2$ with n

Here, $f(n)$ can be expressed as, $f(n) = O(n^{\log_3 9 - \epsilon})$, where $\epsilon = 1$

and it falls under case I of Master's Theorem.

$\therefore f(n) = \Theta(n^2)$

b) $T(n) = 2T(n/2) + n \log n$.

In the given recurrence, $a = 2,\ b = 2,\ f(n) = n \log n$.

Compare $n^{\log_b a}$ with $f(n)$ i.e $n^{\log_2 2}$ with $n \log n$

$\Rightarrow n$ with $n \log n$

Here, Case I & II of Master's Theorem are not applicable as $n \log n$ is asymptotically larger than n.

Case III does not hold because $n \log n$ is not polynomially larger than n. The ratio $\frac{f(n)}{n^{\log_b a}} = \frac{n \log n}{n} = \log n$ is asymptotically less than n^{ϵ} for any positive constant ϵ.

As no cases conditions of Master's theorem are satisfied in this recurrence, none of them are applicable.

Name: Roll No:

5. What is the space complexity of the Merge-sort algorithm discussed in the class? Express your answer using asymptotic notations.

Ans:

$O(n)$, $O(n^2)$, $O(n^3)$, ...

$\Theta(n)$

$\Omega(n)$, $\Omega(1)$

6. Write the upper, lower and tight bounds for worst-case running time of Heap-sort algorithm.

Ans:

upper bound: $O(n \log n)$, $O(n^2)$, $O(n^3)$, ...

Lower bound: $\Omega(n \log n)$, $\Omega(n)$, $\Omega(1)$

Tight bound: $\Theta(n \log n)$

7. Is the function $\lceil \log n \rceil!$ polynomially bounded? Justify your answer.

Ans: No. Suppose $(\log n)! \leq c \cdot n^k \ \forall n \geq n_0$

$\Rightarrow (\log 2^n)! \leq c(2^{nK}) \ \forall n \geq n_0$

Let $2^K = \lambda$ (a constant) then $(\log_2 2^n)! \leq c\lambda^n$

$\Rightarrow n! \leq c \cdot \lambda^n$

$\frac{n!}{\lambda^n} \leq c$. $\Rightarrow\Leftarrow$ contradiction with Stirling's formula.

8. Write the minimum and maximum number of elements in a heap of height *h*.

Ans: Minimum : 2^h

Maximum : $2^{h+1} - 1$

Part-B: **(Max. marks: 5)**

9. Consider the following function named as *xy_function.* Derive tight bound on the worst case running time of *xy_function* using one of the methods discussed in the class. You may write your answers overleaf. **(2 Marks)**

xy_function (*disk*, *source*, *dest*, *spare*):
if *disk* = 0
then move *disk* from *source* to *dest*
else
xy_function(*disk* - 1, *source*, *spare*, *dest*)
move *disk* from *source* to *dest*
xy_function(*disk* - 1, *spare*, *dest*, *source*)

Let $f(n)$ be a recurrence for the given recursive function xy-function.

$$f(n) = 2f(n-1) + O(1)$$

Guess the Solution as $\theta(2^n-1)$

i.e $c_1(2^n-1) \leq f(n) \leq c_2(2^n-1)$

Guess the values of c_1 & c_2 as 1

Prove the following by Substitution Method.

$$2^n-1 \leq f(n) \leq 2^n-1$$

Base case: $n=1$

$$f(1) = 2^1 - 1 = 1$$

Assume the guessed Solution is true for $n-1$.

i.e $f(n-1) = 2^{n-1} - 1$

Induction Step:

$$f(n) = 2f(n-1)+1$$
$$= 2(2^{n-1}-1)+1$$
$$= 2^n - 1$$
$$f(n) = 2^n - 1$$

$$\therefore f(n) = \theta(2^n-1).$$

Name: Roll No:

10. In the following list of sorting algorithms, write *Yes* for those algorithms which are in-place/stable and *No* otherwise. Answers without reasons will not carry any marks. **(3 Marks)**

Sorting algorithms	Stable	In-place	Reason
Bubble sort	Yes	Yes	Stable: BS will never swap 2 equal valued elements In-place: BS uses only constant number of extra space
Quick sort	NO	Yes	Not stable: QS swaps equal valued elements in the partition function In-place: QS uses one temp variable for extra space
Heap sort	No	Yes	Not stable: HS output is obtained by removing elements from the created Heap w.r.t. size of Heap. Info. about the ordering of items in the sequence was lost during the heap creation itself. In-place
Insertion sort	* Yes	Yes	* May depend on choice of symbol. In-place: const. amount of extra space.
Merge sort	* Yes	No	Stable: MS never swaps 2 equal valued elements Not In-place: Requires O(n) additional space to perform Merge
Counting sort	Yes	No	Not In-place: 2 additional arrays apart from the I/P array A[1...n] i.e B[1...n] ⇒ O/P array, C[0...k] ⇒ Temp array where k = O(n)

Part-C: **(Max. marks: 6)**

11. There are n positive integers in an array A which have to be sorted. The sorting technique illustrated below uses two stacks, S1 and S2, which are infinite for all practical purposes, to sort the array. The stacks provide the functions

a. top(S) which return the value stored at S.top. NIL indicates empty stack
b. pop(S)
c. push (S, a)

The sorting algorithm is described next:

```
Stacksort(A)
n=A.length
for i = n downto 1
    a=A[i]
    while( (top(S1) != NIL) AND (top( S1) < a) )
        push(S2, (pop(S1))
    push(S1,a)
    while(top(S2)!= NIL)
        push(S1, pop(S2))
while(top(S1)!= NIL)
    i=i+1
    A[i]= pop(S1)
```

a) Prove the correctness of Stacksort. State loop invariants for all the loops. You may assume the truth of the while loops inside the for loop, for proving the correctness.

(3)

Loop Invariants

For Loop (st 2) : At the start of iteration with $i = k$, the stack S1 contains all the elements in $A[k+1..n]$ in sorted order and the elements $A[1..k]$ are in A in original sequence

While loop (9) : At the beginning of while iteration, with $i = k$, the array $A[0..i]$ contains the first i elements of sorted sequence of A, and the top of S1 points to the $i+1$ th element in the sorted sequence

While loop (4) : At the beginning of j th iteration S2 contains lowest $j-1$ elements of $A[i+1..n]$, all $< A[i]$, in decrease order and elements in S1 are the highest $n-j-i+1$ elements of $A[i+1..n]$ in increasing order from top to bottom

While loop (7) : Before k th iteration of loop the elements in S1 are the first $n-i-j+k+1$ elements of $A[i..n]$ in sorted order

Proof for 2 & 9

(2) Initially $i = n$, so $A[n+1..n] = \emptyset$ – trivially true

To prove maintenance :

Let it be true for $i = x$. Then S1 contains $A[x+1..n]$ in sorted order. By the termination condition of (4) S2 will contain lowest $j-1$ elements of $A[x+1..n]$ all less than $A[x]$ and elements in S1 are the largest $[n-j-x+1]$ elements, all $\geq A[x]$

When $A[x]$ is pushed. S1 will contain $n-j+x+2$ elements, all $\geq A[x]$, in sorted order, and S2 contains $j-1$ elements in reverse sorted order.

By termination of (7), all the ~~j-1 element~~ elements in S1 at the end of (7) will be $n-x-j+j-1+1$ elements i.e $n-x+1$ elements of A in sorted order

Hence at the beginning of next for iteration for $i = x+1$ the condition is true

A termination $i = 0$ so elements $A[1..n]$ are in sorted sequence in S1

Name: Roll No:

b) Give the running time function of Stacksort in the best and worst cases. (1)

Best case $T(n) = cn.$

Worse case $T(n) = c_1 n^2 + c_2 n + c_3$

c) Is the algorithm in place? Is it stable? Give reasons for your answer. (0.5)

Not inplace. Requires space from the stack of $O(n)$.

stable: Yes. If elements have equal keys, the element with larger index lies at a stack position below the one with smaller index, in S1.

When S1 is transferred to array, lower indexed elements get written earlier

d) Suggest modifications for improvement of the running time of the algorithm. Does your enhancement improve the time complexity (the asymptotic bounds) in any manner? Why? (1.5)

(Note: In addition to the space provided in this page, you may write your answers overleaf)

(Several answers possible)

a) Initially: $i = 0$ for S1's top pts to 1st element

~~Maintenance~~: From termination of ~~S2~~ 7, S1 contains first 1..n elements in increasing order.

Maintenance: if true for i then true for $i+1$ as element is lifted from S1 to array and top is decremented.

At then end $A[1..n]$ contains sorted elements