CS302_DAAA_180428_HARIS_MANZOOR

Q no. 1

A) increasing order is

$\lg n < n^{1/3} < \sqrt{n} < n^{20} < 2^{2n} < 10^n < 2^{2^n} < n!$

B)

$f(n)$	1 sec	1 min	no. of ~~m~~minutes represented	no. of ~~minutes~~ hours represented by last 4 digits	no. of hours represented by last 4 digits	process
10^n	3	5	7.41 mili = 25680000	12.18 15408000000000		obtained no of miliseconds $m = 10^n$ $n = \frac{\log m}{\log 10}$
2^{2n}	5	8	24.61	12.3		$m = 2^{2n}$ $n = \frac{\log m}{2\log 2}$
n^{20}	2	2	2.346	4.064		$m = n^{20}$ $\log m = 20 \log n$ $n = \text{Antilog}(\frac{\log m}{20})$
$\lg n$	2^{1000}	$2^{6 \cdot 10^4}$	$2^{25680000}$	$2^{15408000000000}$		$m = \log n$ $n = \text{Antilog}(n)$
$n!$	7	9				
2^{2^n}	4	4	4	4		$m = 2^{2^n}$ $2^n = \frac{\log m}{\log 2}$

$$n = \frac{\log\left(\frac{\log n}{\log 2}\right)}{\log 2} = n$$

CS-302 - i180428 - HARIS MANZOOR

Q no. 2

A) $T(n) = 3T(n^{1/3}) + \log n$, $T(1) = 1$

level	no. of times func is called	Tree	cost
0	1	$\log n$	$\log n$
1	3	$\log n^{1/3}$ $\log n^{1/3}$ $\log n^{1/3}$	$3 \log n^{1/3}$
2	9^2	$\log n^{1/9}$ $\log n^{1/9}$ $\log n^{1/9}$	$9 \log n^{1/9}$
⋮		⋮	
k	3^k	$\log n^{1/3^k}$	$3^k \log n^{1/3^k}$
⋮		⋮	
h	3^h	$T(1)$	$3^h \log n^{1/3^h}$

let us suppose that $k = h - 1$

$\Rightarrow$ base case :- $3^k = n^{1/3} \Rightarrow h = \frac{\log(\log n)}{\log 3}$

$\Rightarrow T(n) = 3^h T(1) + 3^h \log n^{1/3} + 3^k \log n^{1/3^k} + \dots \log n$

$\Rightarrow 3^h T(1) + \sum_{i=0}^{k-1} 3^i \log n^{1/3^i}$

$\Rightarrow n + \sum_{i=0}^{\log n} \frac{1}{3^i} \times 3^i \log n$

CS302 - DAAA - i180428_ HARIS

Qno. 2 (B)

$$T(n) = T(n/2) + T(n/4) + n^2$$

level	no. of time func is called	Tree	cost
0	1	n^2	n^2
1	2	$T(n/2)$ $(n/4)^2$ $(n/2)^2$	$\frac{5}{16} n^2$
2	$(2)^2$	$\left(\frac{n}{4}\right)^2$ $\left(\frac{n}{8}\right)^2$	$\left(\frac{5}{16}\right)^2 n^2$
3	$(2)^3$	$\left(\frac{n}{8}\right)^2$ $\left(\frac{n}{16}\right)^2$	$\left(\frac{5}{16}\right)^3 n^2$
k	$(2)^k$	$\left(\frac{n}{2^k}\right)^2$ $\left(\frac{n}{2^{k+1}}\right)^2$	$\left(\frac{5}{16}\right)^k n^2$
h	2^h	$T(1)$	$\left(\frac{5}{16}\right)^h n^2$

base case :- $n = 2^h$

$h = \log n$

$\Rightarrow 2^h T(1) + n^2 + \frac{5}{16} n^2 + \cdots \left(\frac{5}{2^k}\right)^2 n^2$

$\Rightarrow 2T(1) + \sum_{i=0}^{h} \left(\frac{5}{16}\right)^i n^2$

CS-302- DAAA- 180428- HARIS MANZOOR

Qno.3 You are provided with ....

Algo X:-

$$T(n) = 8T(n/2) + c$$

using master method.

$a=8, \quad b=2, \quad k=0, \quad p=0$

$\log_b a = 3$

$\Rightarrow \log_b a > k$

$\Rightarrow \Theta(n^3)$

Algo Y:-

$$T(n) = 7T(n/3) + n^2$$

$\Rightarrow a=7, b=3, k=2, p=0.$

$\Rightarrow \log_b a = \log_3 7 = 1.77$

$\Rightarrow \log_b a < k.$

$\Rightarrow$ case 3

$\Rightarrow \Theta(n^2)$

CS-302 - i180428 - Haris Manzoor

7 Hair.

It is clear than complexity of Y is better than X. I would suggest Y

CS-302-DAAA- i180422 HARIS_ MANZOOR

Qno. 4

part (A)

I would use inorder topological in which i would calculate the indegree of all vertices first by visiting each node

vertices	indegree
A	0
B	0
C	1 : 0
D	2 : 1 : 0
E	1 → 0
F	2 : 1 : 0
G	1 : 0
H	2 : 1 : 0

load the vertices having indegree of 0 is stack.

A
~~B~~

B E G A D C F H
topoligical order.

now pop B and check its neighbours i-e D and E. reduce the indegree and if indegree become 0, push it back in stack, now stack has A and E

CS-302 - i180428 - HARIS MANZOOR

7 HW

then get to the vertices in neighbourhood of E i-> G, decrease indegree and push G into stack. now stacks has

A, G.

pop G, check its neighbours i-e H donot push this time because indegree is not 0.

now pop again, we get A, neighbours are C and D, push them as Both have 0 indegree

stack = C D

- pop D   stack = C
- reduce F
- pop C   stack = empty
- reduce F

now push F   Stack = F

- pop F
- reduce H   stack = Empty
- push H   stack = H
- pop H

to generate more topological orders, we change the order of pushing for e.g if two vertices have same indegree, we would push them alternatively.

=> BEGACDFH, BEGADCFH, BEAGDCFH, BEAGCDFH - - - .

CS-302: DAAA - i180428_ HARIS

Haris.

Part (B)

Provide all

starting from 8

8 -> 3 -> 7 -> 0 -> 4 -> 5 -> 9 -> 1 -> 6 -> 2

again

8 -> 7 - 3 - 0 -> 9 -> 5 -> 4 -> 1 -> 6 -> 2

again

8 -> 0 -> 7 -> 3 -> 1 -> 9 -> 4 -> 5 -> 6 -> 2

CS-302 DAAA- i180428- HARIS_MAN2002

Q no.7

M.Hari

part a) You are running... ?

since the size of integers is fixed, we can use radix sort. as radix sort works best as we have given fixed length integers. time complexity for radix sort is $\Theta(nk)$ where n will be fixed everytime.

part b) You are organizing....?

since we have the ~~bes~~ case very close to best case, we can use either insertion sort or bubble sort because both give the best case as $\Omega(n)$

CS-302 DAAA - i180428.

HARIS MANZOOR

Haris.

Q no. 8

a) **optimal substructure property:**

If a city c lies between the shortest path from u to v then the shortest path from u to v is a combination of shortest path from u to c and c to v.

b) **subproblems:-**

There can be n subproblems i-e we need to search each city and by having path from having each city in-between and finding the minimum i-e min[~~...~~(u,v) or (u,c),(c,v)]

and each subproblem will need n^2 time to complete.

c) **Recurrence :-**

$$A^{k}[i][j] = \min \begin{cases} A^{k-1}[i][j], \\ A^{k-1}[i][k] + A[k][j] \end{cases}$$

~~the~~ where A^0 = the matrix of all the direct paths between cities.

d) **Pseudo code :-**

```
D[n][n],        D^0 = W.        W = weight of direct edges.
for k=1 to n
    for i=1 to n
        for j=1 to n.
            D[i][j]^k
```

Scanned with CamScanner