CS20BTECH11063

TOPIC Tanmay Garg DATE Algo Final Exam

1. (a) $T(n) = 2T(n/4) + 1$

$n = 4^k$

$T(4^k) = 2T(4^{k-1}) + 1$

$S(k) = T(4^k)$ $\quad k = \log_4 n$

$S(k) = 2S(k-1) + 1$

$\frac{R(k)}{2^k} = \frac{R(k-1)}{2^{k-1}} + 1$ $\quad \Rightarrow P(k) = P(k-1) + 1$

$P(k) = 2^k$

$P(k) = P(k-1) + 1$

$P(k-1) = P(k-2) + 1$

$\vdots$

$P(1) = P(0) + 1$

$P(k) = k + 1$

$\frac{R(k)}{2^k} = (k+1)$ $\quad S(k) = k+1 = T(4^k)$

$\Rightarrow T(n) = \log_4 n + 1$

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

$R(k) = \frac{S(k)}{2^k}$

$2^k R(k) = 2^k R(k-1) + 2^{-k}$

$R(k-1) = R(k-2) + 2^{-(k-1)}$

$\vdots$

$R(k) = \frac{2^{k+1} - 1}{2^k}$

$S(k) = 2^{k+1} - 1$

$T(4^k) = 2^{k+1} - 1$

$T(n) = 2^{1/2 \log_2 n + 1} - 1$

$\therefore T(n) = O(\sqrt{n})$

TOPIC .....

6) $T(n) = 3T(n/3) + n$

$n = 3^k$

$T(3^k) = 3T(3^{k-1}) + 3^k$

$T(3^k) = S(k)$

$\Rightarrow S(k) = 3S(k-1) + 3^k$

$R(k) = \frac{S(k)}{3^k}$

$3^k \cdot R(k) = 3 \cdot 3^{k-1} R(k-1) + 3^k$

$R(k) = R(k-1) + 1$

$R(k-1) = R(k-2) + 1$

$R(k) = k+1$

$\frac{S(k)}{3^k} = k+1$

$S(k) = 3^k(k+1)$

$T(3^k) = 3^k(k+1)$

$T(n) = n \log_3 n + 1$

$\therefore T(n) = \Theta(n \log n)$.

2. The total no. of possibilities on n numbers assuming all distinct is $n \cdot {}^{n-1}C_{\frac{n-1}{2}}$

$\therefore$ minimum no. of comparisons will be $\log_2 n \cdot {}^{n-1}C_{\frac{n-1}{2}}$

because each possibility will either be correct or it won't be, so height of such decisions = $\log_2 k$

k is total no. of possibilities

TOPIC ..... DATE.....

$\log_2 \frac{n!}{(\frac{n-1}{2})!\,(\frac{n-1}{2})!}$ * applying stirling approx

$\frac{1}{\ln 2}\left(\cancel{\ln(n+1)}\; n\ln n - n - (n-1)\ln\left(\frac{n-1}{2}\right) + n - 1 \right)$

$\approx \frac{1}{\ln 2}\left(n\ln\left(\frac{2n}{n-1}\right) + \ln\left(\frac{n-1}{2}\right) - 1 \right)$

assuming large n

$\frac{1}{\ln 2}\left(n\ln 2 + \ln n - \ln 2 - 1 \right)$

$\approx (n-1) + \frac{\ln n - 1}{\ln 2} \geq n-1.$

Hence Proved

3.

$$funk(l,m,n) = \begin{cases} 1 + funk(l-1, m-1, n-1) & \text{if } (a[l]=b[m]=c[n] \\ \max \begin{cases} funk(l-1, m, n) \\ funk(l, m-1, n) \\ funk(l, m, n-1) \end{cases} & \text{else} \end{cases}$$

TOPIC ..... DATE .....

4. Idea of the algorithm:

We will use kruskal's algorithm to check if the edge 'e' is chosen for MST or not

If after choosing 'e' kruskal's algorithm doesn't get a cycle then it will choose it & move to the next edge

Algorithm:

```
sort (edges)
DFS( G' = vertices & edges of weight < e
DFS(G,1)
set_G = DFS(G')
if (end)
{
search end points of 'e' in set_G
if ( set_G(e1) != set_G(e2) )
    return TRUE
else
    return False
```

Runtime = O(m+n)

TOPIC ..... DATE.....

5. To prove: G is undirected connected graph, edges cost are distinct
G has unique MST

Proof:
let T_1 & T_2 be 2 MST*
~~T_1 & T_2 have edges that~~
T_1 would have edge that T_2 didn't choose
vice versa
let 'e_1' be the min weight edge in T_1 (subtracted all T_2 edges)
let 'e_2' be for T_2
let us assume $e_1 \leq e_2$
∴ graph $T_2 \cup \{e_1\}$ will have cycle going through e_1
let us take an edge e_3 which is edge in the cycle but not in T_1
∵ $e_1 \in T_1$ & $e_3 \neq e_1$
∴ e_3 is in T_2 (subtracted all T_1 edges)

TOPIC ..... DATE .....

$\therefore e_3 \geq e_2 \geq e_1$

let us take tree $T_3 = T_2 \cup \{e_1\} - \{e_3\}$

$\therefore$ weight of T_3 = weight of $T_2 + e_1 - e_3$

$\therefore$ weight of $T_3 \leq$ weight of T_2

but T_2 was MST $\therefore T_3 = T_2$ $\therefore T_3$ also MST

$\therefore e = e_3$

Hence proved

6. Idea of Algorithm:

We will apply cycle property which states that if G_1 has cycle C & an edge e has weight greater than sum of all other edges in C then 'e' will not be in MST

MST

~~Algorithm~~ (G_1):

Do $\{ G_1' = BFS(G_1)$ // G_1' will return the

while: (cycle not detected) // cycle from G_1

~~sum_edge_G_1' = sum of weights of G_1'~~

~~sum_edge_G_1 = sum of weights of G_1~~

if ($w(e) \in G_1'$ && $(w(e) > (sum\_edge\_G_1' - w(e))$)

$G_1 = G_1 - \{e\}$

~~MST(G_1)~~

do above multiple times till we get G_1 & return G_1

G_1 will have $n-1$ edges & will be MST of G_1

TOPIC ..... DATE.....

6. Runtime : BFS = $O(m+n)$

$\therefore$ $n+8 \geq m$

checking greatest weight is ~~$O(n)$~~ $O(m+n)$

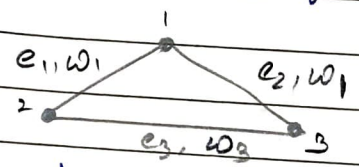
$\therefore$ $O(n)$

Correctness :

The correctness of the algorithm can be taken from cycle property of graphs & MST

7. Counter example :

let us take the following graph



let e_i denote edges & w_i denote weights

The spanning trees are :

e_1e_2 , e_2e_3, e_1e_3

$\downarrow$ $\downarrow$ $\downarrow$

$2w_1$, w_1+w_3 w_1+w_3

$w_1 \neq w_3$

~~Case 1~~ $w_3 < w_1$

e_2e_3 , e_1e_3 will be MST but e_1e_2 will not be MST

~~Case 2~~ ~~$w_3 > w_1$~~

~~e_1e_2~~ both e_1 & e_2 are not in MST together

~~Hence~~ ~~proved~~

TOPIC ..... DATE.....

8. To prove: The statement in the question is true

Proof (By induction):

let us take 2 trees of H

$T_1 = (V_1, E_1)$

$T_2 = (V_2, E_2)$

let us use the difference in number of edges in both trees & do mathematical induction on them

Base case $E_2 - E_1 = 1$

when we construct H we connect, the 2 edges & it ensures connected graph & both T_1 & T_2 are neighbouring 'nodes' & are connected by edge of length 1 unit

Assumed $E_2 - E_1 = n$ is true

to $E_2 - E_1 = n+1$

let us choose some edge 'e' in T_2 but not in T_1

$\therefore T_1 \cup \{e\}$ will have cycle with edge 'e'

This cycle will also have edge 'e_2' not in T_2 as well

let us take $T_3 = T_1 \cup \{e\} \setminus \{e_2\}$

T_3 will have E_3 no. of edges

T_3 & T_2 will have n edges ~~no unique~~ which are not common

$\therefore$ We can construct H from T_2 & T_3 using n edges

$\therefore$ T_1 & T_3 are neighbouring 'nodes'

$\therefore \exists$ a path b/w T_1 & T_2 with $n+1$ length

TOPIC ..... DATE.....

9. By Max Flow Min cut theorem, there is some s-t cut (A, B) in original G of capacity c_e

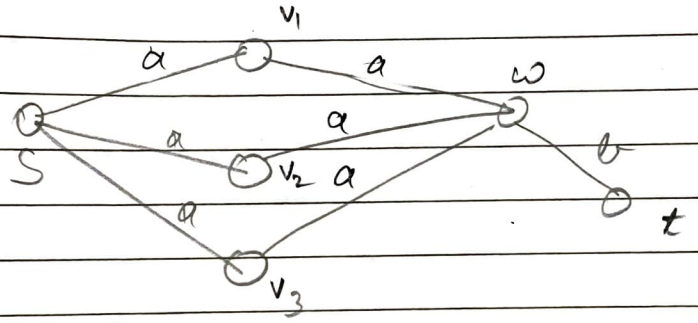
~~All edges crossing (A,B) have capacity in G'~~

All edges have same capacity in G2 & in G
∴ c_e reduced by 1 so capacity of (A,B) in new network can be at most $c_e - 1$

~~With this~~

Finding augmenting path in residual graph takes $O(m+n)$ time

10. Counter example:



$A = \{s\}$

$B = V - \{s\}$ 3a

this gives minimum cut with flow 3a but if we add 1 to all edges then this becomes 6a

but when the cut is on $A = \{t\}$

$B = V - \{t\}$

it is ~~c(t)~~ $c_t + 1$

hence it is wrong

$c_t + 1$ may or may not be equal to 6a

TOPIC ..... DATE.....

11. Let us build bipartite graph with edges such that if P_i works on night d_j
let us use maximum matching technique &
~~apply maximum flow~~
source 's' has edges to all people
sink 't' has edges from all nights

If we have a feasible dinner schedule then P_i is mapped to only one d_j & each person is joined to a unique d_j
combining all edges will give us perfect maximum matching
If we have perfect matching, then each P_i is mapped to a unique d_j
∴ Each P_i works one only & someone is there to cook on each night
∴ This gives us a feasible dinner schedule

Hence proved

TOPIC ..... DATE.....

12. (a) Let us take matrix where only 1 row has all entries 1 & only 1 column as all entries 1 rest all entries are 0

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad \text{etc.}$$

(b)