Tanmay Garg CS20BTECH11063

TOPIC ..... Algorithms Midterm ..... DATE ..... Exam 2 .....

1. **False**

Let us take the ~~match~~ preference list

$m_1 \rightarrow \{w_1 > w_2\}$

$m_2 \rightarrow \{w_2 > w_1\}$

$w_1 \rightarrow \{m_2 > m_1\}$

$w_2 \rightarrow \{m_1 > m_2\}$

In this situation, we can never get a stable matching. Let us take pair up $m_1$ with his most preferred w which is $w_1$.

$w_1$ then is not paired with her highest preferred m which is $m_2$

Similarly if we pair $(m_2, w_2)$, then $w_2$ is not paired with $m_1$

∴ There is no way to include an m in a pair such that it can be paired with its highest preferred & we can't have w who can be paired with her highest preferred.

2. **True**

Let us assume that there is a stable matching S for this instance (where m is ranked highest on preference list of w & w is ranked highest on preference list of m) ∎

the pairs $(m, w')$ & $(m', w)$ are taken.

Since w is ranked top on m's list, 'm' must choose w, & since m is ranked top on w's list w must choose m. But this is instable as m & w both prefer each other to their current pairs.

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

this is a contradiction, hence $(m, w)$ must be in $S$

3. (a)

(b) · There is no stable matching
· let A have 2 shows $\{a_1, a_2\}$ with rating $\{2, 4\}$.
· let D have 2 shows $\{d_1, d_2\}$ with rating $\{1, 3\}$.
· If $a_1$ is paired with $d_1$, then D will want to switch the order of the shows in its schedule
· If $a_1$ is paired $d_2$, then A will want to switch the order of the shows.

4. (a) Doesn't provide optimal schedule
eg:- A ⇒ start time = 0, end time = 10
B, C, D, E, ⇒ start time = 0, end time = 8 evenly spaced & of equal length (2)
the algo will choose A but optimal will choose B, C, D, E

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

4(b) Doesn't provide optimal solution
Using same scenario as used in (a), algo will choose A whereas optimal soln chooses B, C, D, E

(c) This provides us with optimal solution
Proof: let us take x as mentioned in the question. Let S be the schedule that doesn't crave x, let y be the last course in S.
∴ S[y] < S(x)
∴ F(i) < S(y) < S(x) for every i in S
∴ S' = S - y + x is a valid schedule
∴ S is an optimal soln so is S'
Hence proved

(d) This doesn't provide optimal soln
eg:- class (start, end)
A(5, 6)
B(1, 5.5) C(5.5, 10)
the optimal soln chooses B, C while algo chooses A

(e) This doesn't provide optimal soln
eg:- A(0,5) B(5,10) C(10,15) D(15,20)
E(3,6) F(4,7) G(4.5,8) H(8,11) I(12,16)
J(13,17) K(14,18)
Algo will choose H, G, I while optimal soln will choose A, B, C, D

(f) This doesn't provide optimal soln
using same scenario as (d), algo choose A while optimal is B, C

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

(g) This doesn't give us optimal sol$^n$
using same eg from (e) but with an extra course
L(9,12). Algo chooses A, D, L while optimal
chooses A, B, C, D

5. Algorithm:
- Sort the intervals in ~~increasing~~ accordance to L & then R. ~~but~~
- Select the first interval with the farthest endpoint
- ~~Then~~ keep a counter to keep track of the endpoint
- Then choose the interval with the farthest end point given that its starting point is less than or equal to the current ~~end~~ 'counter'
- Repeat the steps till we get the last endpoint to cover all the intervals length
- This is a greedy algorithm, the major time taken will be done by sorting which can take O(n log n)

- Let us say that our claim is wrong, ~~our~~ optimal sol$^n$ containing the first interval,
- Let 'a' be the interval our new optimal sol$^n$ chooses & 'b' be the original interval chosen
- if 'a' starts at any later time than 'b' then it invalid sol$^n$
- 'a' also can't end later than 'b' as in our algorithm
- Let us make a third optimal sol$^n$ that is the second optimal sol$^n$ but with 'a' replaced with 'b'
- This is a valid sol$^n$ of the same size as that of our actual optimal sol$^n$

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

6. A) We can take the frequencies in the following pattern

Let $a_1, a_2, \ldots, a_n$ be letters sorted in increasing order of frequencies

Let $f(a_i)$ denote the frequency of letter $a_i$

$$f(a_i) > \sum_{j=1}^{i-2} f(a_j)$$

In this case, the lower frequency letters when grouped together should have combined frequency less than or equal to next letter frequency

Each step of Huffman coding reduces the no. of letters by 1 & increases the depth by 1

$\therefore$ total depth = $n-1$

B) To make it minimum $f(a_i) = \sum_{j=1}^{i-2} f(a_j) + 1$

$\therefore f(a_i) = f(a_{i-1}) + f(a_{i-2})$

As we have proved before, we shall now begin with frequency 1 to include the letter in Huffman coding

$f(a_1) = 1$
$f(a_2) = 1$
$f(a_3) = 1$
$f(a_4) = 3$
$f(a_5) = 4$
$f(a_6) = 7$

As we can see this follows a pattern

(we have take the equality constraint to minimize the next frequency)

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

$$f(a_i) = \left(\frac{1+\sqrt{5}}{2}\right)^{i-2} + \left(\frac{1-\sqrt{5}}{2}\right)^{i-2} \quad \forall i \geq 5$$

7. NUM_GROUPS(A[1....n]):

// strip -ve entries from start & end  i = 0

while (A[i] ≤ 0 && i ≤ n):

i++

j = n

while (A[j] ≤ 0 && j ≥ 1):

j--

if i > j : return 0

sum = 0

calculate sum of subarray from i to j

if sum > 0:

return 1

pos_min = -1

min_num = ∞

for k = i to j:

if A[k] < min:

pos_min = k

min = A[k]

return NUM_GROUPS(A[i...k-1]) + NUM_GROUPS(A[k+1...j])

Idea: if sum of subarray ≤ 0 & first & last element are +ve then min no will be -ve will not be present in any of the subarrays which constitute the minimum

9. a, b, c, d, e, f  a, d, e, b, c, f

a, b, d, c, e, f

a, b, d, e, c, f  total 6

a, d, b, c, e, f

a, d, b, e, c, f

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

10.
- Pick any vertex as root
- Start a DFS, maintain DFS tree
- When a new vertex is seen, then add to the tree
- If encounter an edge from a node to an already visited node, then cycle exists
- Cycle can be returned by tracing the path from each node to the latest edge backwards in tree
- If DFS completes, then no cycle exists
- Entire procedure is dependent on DFS which takes $O(m+n)$ time

11.
- Let us say one of the connected components is $V_1, \ldots, V_m$ & is cycle
- Let DFS visit $V_1$ first, & after some iteration $V_2$ is visited.
- While visiting all the other they will be marked visited, $V_m$ is visited then $V_1$ is adjacent but is also visited, hence cycle will be returned

11.
- Assume graph is given & in-degree of each node is calculated.
- Select a node with indegree equal to 0 & not visited. If there is none & total visited no. of nodes is less than total no. of nodes then there there is cycle.
- If all nodes have visited give the ordering
- For selected node, mark it visited & put it in the ordering sequence
- Reduce the in degree for all the nodes that this node points to
- Then repeat the above steps

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

12. · Let us take {a, b} edge in $G_1$ that doesn't belong to T
· T is a DFS tree, let a be ancestor of b
· Since in BFS tree, every node visited at the same level are equidistant from its parent that is 1 unit distance
· As a is the ancestor b & BFS tree & DFS tree are same then a is parent of b
∴ This is a contradiction as then {a, b} edge is present in T

13. Proof (By contradiction)
Let us assume $G_1$ is not connected
∴ it has atleast 2 components $G_1$ & $G_2$
∵ every node must have degree of at least $n/2$
∴ a node in $G_1$ is connected to $n/2$ nodes
∴ there are $(n/2) + 1$ nodes in $G_1$
Similarly we can say in the case for $G_2$

∴ the total nodes will become
$(n/2 + 1)2 = n + 2$

∴ this is a contradiction as we have only n nodes
∴ $G_1$ is connected
We can similarly expand this proof for k components in that case the total no. of nodes will come as $\frac{nk}{2} + k$ but that will contradict the no. of nodes to be n

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

14. Proof:

Let us take a BFS tree T made from G with S as root

Each node k is at a length such that it is shortest path from S to k

∵ All paths from S to t have length greater than n/2 ∴

∴ let $l = n/2 + 1$

∴ if each n/2 distance node has 2 or more nodes then total no. of nodes won't be n

∴ ∃ a node v at it lies in distance less than n/2

∴ if we delete v, then all paths b/w S & t will be destroyed

Algorithm:

We will use a modified BFS Algorithm while doing BFS, each node at the same distance from root node S will be put in an array

A[i][a]. i is distance from root node
a is node address or node no.

After doing BFS

We find index i such that

A[i] has only 1 element throughout

& that is the node we require

& return that node

Running time will be O(m+n) as we are doing BFS on the entire G to find that particular node

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

15. Let us begin with BFS from a node v, we will keep a track of levels from v

We know that in BFS we always get a shortest path b/w nodes

Let us calculate the no. of shortest path from v to u. Let A be this no. for u

Let us take a node x at a level 'l'. the shortest v-x path are of the form such that it is the shortest path to some node 'a' in a level before 'l' & it takes only 1 unit to go from a to x

∴ We can calculate the sum of all paths from a to x & so on upto u & similarly from v to u

The time to compute 'A' is at most the degree of the node u & sum of all degrees of nodes is 2m

∴ The overall running time is decided by BFS algo which is O(m+n)

16. Let us make a directed graph such that 2 vertices bi & di are made for each person Pi

Add edges b/w (bi, dj) Pi was born before Pj died

Add edges b/w (di, bj) Pi died before Pj was born

Add edges (bi, dj), (bj, di) if lifespan of Pi & Pj overlap in some timespan

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

Let us say $G_t$ has a cycle.
Then each event associated with the nodes must precede the next but then no event will be the first. This is incorrect

If $G_t$ doesn't have a cycle. then we can create a topological sorting it is possible wrt to birth & death time of all people & hence this will be consistent with the facts

17. Let us construct a directed graph $G_t$
For each triple $(C_i, C_j, t_a)$ in trace data we make nodes $(C_i, t_a)$ & $(C_j, t_a)$ & create directed edges joining in both directions
Let us create arrays for each of the nodes & push the nodes into these arrays
We create an edge between $(C_i, t_a)$ & $(C_i, t_a)$ if that triple has been encountered before

Let us decide if a virus introduced to $C_i$ at time $a$ will infect $C_j$ at time $b$.
Let us go through array of $C_i$, till we reach at some $a'$ such that $a' \leq a$

Let us run a BFS from this node till we can reach a node $(C_j, b')$ such that $b' \leq b$, if such a node is reached then $C_j$ is infected by time $b$.

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

First we claim, if $\exists$ a path from $(c_i, a')$ to $(c_j, b')$ then $c_j$ is infected by time $b$

The virus moves b/w computers $c_p$ & $c_q$ at time $t_4$ when edge from $(c_p, t_4)$ to $(c_q, t_4)$ is traversed in BFS
This is a possible virus transmission

Suppose there is a sequence of virus transmission such that virus leaves $(c_i, t_a)$ & arrives $(c_j, t_b)$ then we can start at node $(c_i, t_a)$ & follow the edges in BFS
Each moment virus goes from $c_p$ to $c_q$ at $t_4$ we add edges from $(c_q, t_4)$ to $(c_q, t')$
when virus arrives at node $c_j$ we should have added a node $(c_j, t'')$
$\therefore t'' \leq t_b$
$\therefore$ There must be same sequence of edges completing the path

As we are doing a BFS & that defines the entire running time of the algorithm
$\therefore O(n+m)$

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

8.

```
Steps Counter (n)
    temp ← n
    count ← 0
    while ( temp != 1 )
        if ( temp % 2 == 0 )
            temp /= 2
            count ++
        else
            temp --
            count ++
    return count
```

Let us say our number 'n' is a power of 2 then to reach the number we will keep on multiplying by 2 to reach it

it will take $O(\log n)$ time for that

Let us say our no. 'n' is not a power of 2

$T(n) = \min \{ T(n/2), T(n-1) \} + 1$

if n is even then $T(n/2)$ will be smaller so each time a no. is even it must be divided by 2

if n is odd then $T(n-1)$ is defined after which it will divided by 2 as $n-1$ will be even giving us the least no. of steps

Worst Case : a no. is such that it always does the operation -1, /2 in this order till it reaches 1

Doing this in a loop it will again take $O(\log n)$ time

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

12. • Let us say $G_1$ is an undirected tree T
• DFS & BFS will then both produce tree & will contain all edges of T
• DFS & BFS both will be identical if same root & edges are present & will produce T
• Let us prove by contradiction
• let $G_1$ contain cycle C
• let the cycle C be the nodes $V_1, V_2, V_3 \cdots, V_i, V_1$
• In DFS tree, $V_1, \cdots, V_i$ will be on same path from root to leaf.
• Let us say $V_m$ is the node at which it arrives first
• The other nodes will be visited at some point while visiting neighbours of $V_m$
• In BFS this cause branching while visiting $V_m$
• ∴ BFS & DFS will produce same tree iff $G_1 = T$

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

7. Time Analysis : $j - i = l$

time taken till part where sum is calculated

$\Theta(l)$ Time

time for rest part is also $\Theta(l)$

$T(l) = \Theta(l) + \Theta(l) + T(p) + T(l-p)$

$T(n) = \Theta(n) + T(p) + T(n-p)$

Worst case $p = 1$

$T(n) = \Theta(n^2)$

Best case all are +ve nos

$T(n) = \Theta(n)$

Proof by contradiction

- ∃ a subarray A which contains min element
- & is in set of min subarrays
- Let A be divided into $A_1, A_2$ along with min element
- -ve nos b/w $(B, A_1)$ & $(C, A_2)$ are lesser in magnitude than min element
- We can conclude $(B, A_1) \rightarrow B_1$ & $(C, A_2) \rightarrow C_1$
- $(E, B_1, C_1, D)$ are only 4 subarray which satisfy our condition but in diagram shows 5
- Hence contradiction