SOLO

Assignment - 6

CS-202

Anukool Dwivedi
B19074
Grp-13

Q1

The worst-case depth (~~min~~ maximum depth) of a B-tree is ~~$\log_{m/2} n$~~ $\log_{m/2} n$.

The best-case depth (minimum depth) of a B-tree is $\log_m n$

Q2

⇒ Drop eggs at floors $2\sqrt{n}$, $3\sqrt{n}$, $4\sqrt{n}$, ..., $\sqrt{n}\sqrt{n}$ [n is the number of floors]

⇒ Let us assume that the egg is broken at the level $k\sqrt{n}$.

⇒ Now, with the second egg, perform a linear search in the interval $(k-1)\sqrt{n}$ to $k\sqrt{n}$.

⇒ In this way we can find the floor in $2\sqrt{n}$ trials.

So $E(n) = \Theta(\sqrt{n})$

Q3

Algorithm is as follows :-
Let A be the given array.
Initialize :
max_so_far = 0
max_ending_here = 0

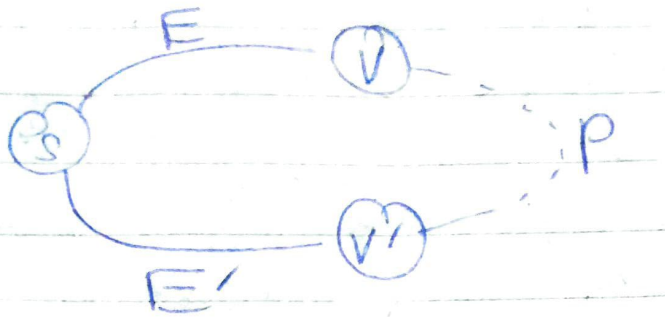
Loop for each element of the array :-
(a) max_ending_here = max_ending_here + A[i]
(b) if(max_so_far < max_ending_here)
max_so_far = max_ending_here
(c) if(max_ending_here < 0)
max_ending_here = 0
return max_so_far.

So, as it is clear that we are running only one loop on n elements.
So, time complexity is $\boxed{\Theta(n)}$.

Q4)

According to the question, we know that there is no negative weight cycle in the graph G and all negative weight edges are connected to the source vertex s.
So, we just need to prove that :-
if some vertex $v \neq s$ is connected with some negative weight edges, the shortest path from s to v must cover the negative weight edges.

Proof :- By contradiction :-



Both E and E' are negative weight edges.

Assume that the shortest path from S to V is $S \xrightarrow{E'} V' \xrightarrow{P} V$ instead of $S \xrightarrow{E} V$. Then we could get the eqn.

$\Rightarrow E' + P < E$

$\Rightarrow E + E' + P < 2E < 0$ // negative cycle.

So, it is a contradiction

(Q 5)

We have a Graph G (undirected), source vertex s.
Destination vertex d.
While finding the shortest distance of d from s. First task is to search the vertex d, starting our traversal from s.
So, in worst case, we might explore whole graph before reaching vertex d. Let us suppose we start search from

s using DFS and maintains parent array, distance array and while visiting each node, we could update parent and distance array accordingly.
So, in worst case we could reach d after visiting all other vertices.
So, in this case we may end up calculating the distance of all vertices from s.

Hence, in worst case, finding the shortest path from s to d is as hard as finding the shortest path from s to all other vertices in V.