Solution This can be solved by dynamic programming. Let S(p, y) be the maximum possible saving after year y with plan p. We assume that c_i and p_i are positive. The final answer would be the $max_{1<i\leq n}S(i,m)$. We have to compute the maximal saving for a year given any plan. The recurrence can be defined as

$$S(p,y) = \begin{cases} p_1, & \text{if } y = 1 \text{ and } p = 1\ . \\ -\infty, & \text{if } y = 1 \text{ and } p \geq 2. \\ max(max_{1<i\leq n}S(i,y-1) - c_p, S(p,y-1) + p_i), & \text{otherwise.} \end{cases} \tag{2}$$

We create a table of $m * n$, and since for year 1, company has no savings, Susan can't switch to any other plan. The company can have only p_1 for first year, so we take this as base case and make $S(1,1) = p_i$ and other columns for year 1 as $-\infty$. Now, we initialize another array to store maximum savings of a year, for base case it would be p_i. Now, we use the recurrence formula to compute other entries in the table year wise. We are using the extra space to store the maximum of every year. When we compute for next year we store the maximum saving of that year which will be used later on, and will make every sub-problem solvable in constant time. Now, we can return the $max_{1<i\leq n}S(i,m)$ which will yield the maximum profit.

Proof of Correctness

Base Case : Susan has plan 1 for the first year and 0 savings. Since, she has zero savings she can't switch to any plan for the first year hence everything else will be -∞

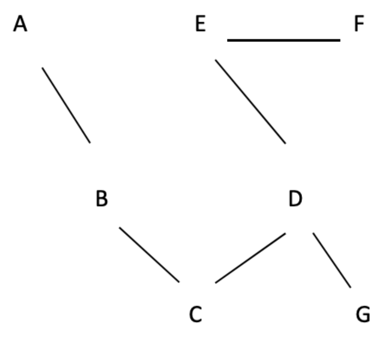
Inductive Hypothesis : Let us assume we have correctly calculated the sub problems till S(p, y-1)

Inductive steps : From the recurrence relation we can see that we are taking the max of $max_{1<i\leq n}S(i,y-1) - c_p$ and $S(p,y-1)+p_i$. The first part will cover the cases when Susan changes to plan p in the year y in which case we take maximum savings of last year and pick plan p for the current year by deducting the cost of plan p. Now, the second part is where she already had plan p in last year and she just accumulated the profit of this year which is p_i. Both these cases are possible, but we are trying to maximize the savings so we take the maximum of these two parts.

Runtime Complexity There are $m * n$ subproblems and it will take $O(n)$ for each sub-problem if we don't store the maximum saving for each year, but since we are saving it, each sub-problem can be solved in constant time which gives us total complexity of $O(mn)$

Note: For Q1 and Q2, please see the solutions here. Do not refer to the question numbering and marks in these scans. See the marks in the QP.

5. {6 marks} Perform a *breadth-first search* of the following graph, where E is the starting node. In other words, show the output if we issue the call `BFS(E)`. Provide two cases: (a) Use a counterclockwise ordering from the top (12 o'clock position); and (b) Use a clockwise ordering from the top.



ANSWER:

(a) When we visit adjacent nodes in a counterclockwise order from the top, the order in which we visit the nodes is:

E, D, F, C, G, B, A

(b) When we visit adjacent nodes in a clockwise order from the top, the order in which we visit the nodes is:

E, F, D, G, C, B, A

6. {6 marks} Perform a *depth-first search* of the same graph as in Question 5, but use D as the starting node. In other words, show the output if we issue the call `DFS(D)`. Provide two cases: (a) Use a counterclockwise ordering from the top (12 o'clock position); and (b) Use a clockwise ordering from the top.

ANSWER:

(a) When we visit adjacent nodes in a counterclockwise order from the top, the order in which we visit the nodes is:

D, E, F, C, B, A, G

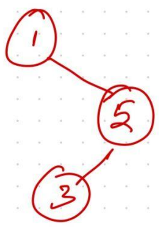
(b) When we visit adjacent nodes in a clockwise order from the top, the order in which we visit the nodes is:

D, G, C, B, A, E, F

Q3.

Solⁿ 3 (b) No Counter-example

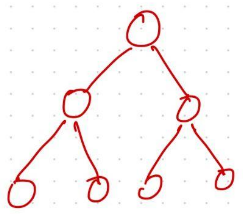
$\{1, 5, 3\}$



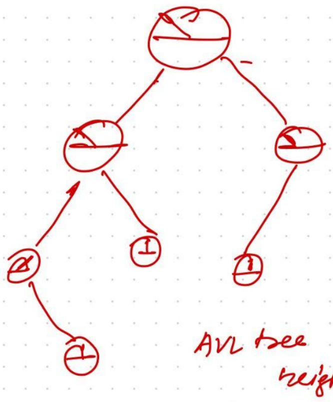
BST is of the worst possible height but elements are not sorted.

Hence, elements need not be in the sorted order.

Solⁿ 3 (a) Counter-example



AVL tree of height 2 (7 nodes)

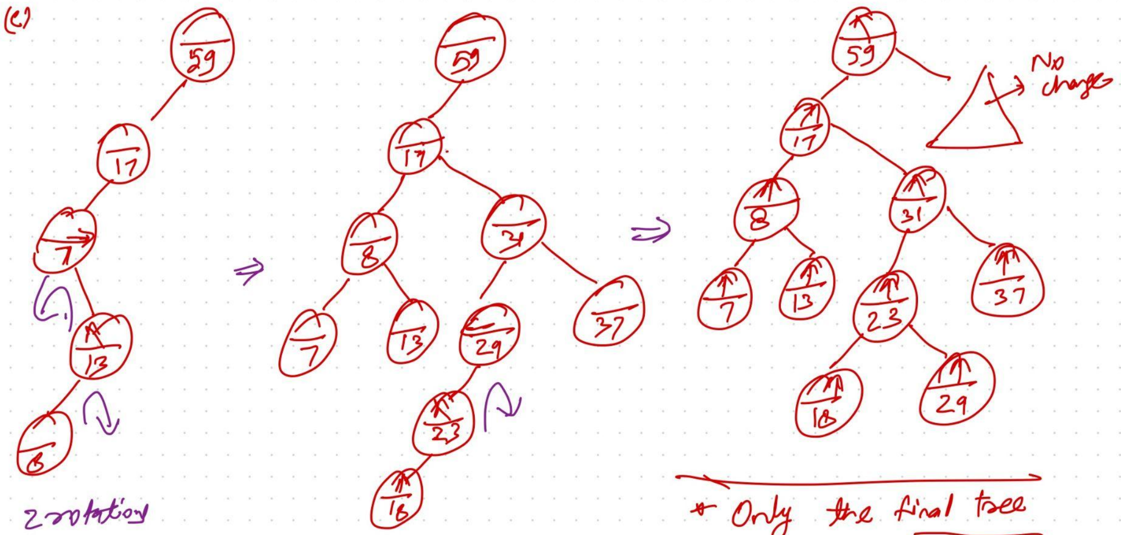


AVL tree of height 3 (7 nodes)

Hence, this is a counter example

(height $\underset{=3}{l}$ AVL tree contains same number of nodes as height $\underset{=2}{l-1}$, not strictly more.

Sol^n 3(c)



Insert 8 $\longrightarrow$ Insert 18

2 rotations   1 rotation

* Only the final tree to be shown — $\boxed{3 \text{ marks}}$

# of rotations $= 3$ $\boxed{1 \text{ mark}}$

Deduct 1 mark if balance factors are not shown / incorrect

Sol^n 4: → Choice 1 is correct [1 mark for mentioning]

Choice 2 is not optimal [4 marks]

Counter-example

Height	1 ,	2
Gown size	1.75 ,	2.75

as per Choice 2. $[2, 1.75]$, $[1, 2.75]$ avg diff. $= 1$

$$\Rightarrow \frac{.25 + 1.75}{2}$$

Optimal $[1, 1.75]$, $[2, 2.75] \Rightarrow$ avg diff. $= \underline{0.75}$