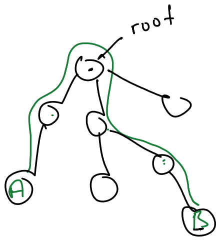
Some Problems on Trees

- Distance between two nodes of a tree (simple DFS O(N)
  LCA with parent O(H)

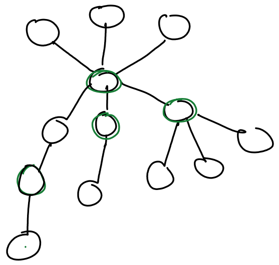
- Print a tree level by level (BFS O(N))

- Diameter of a tree (longest distance) (BFS O(N))

BFS(root) → B
BFS(B) → A



- Minimum vertex cover on a tree
  (Greedy solution)



- Binary tree. Given in-order and post-order traversals. ← Divide and conquer.

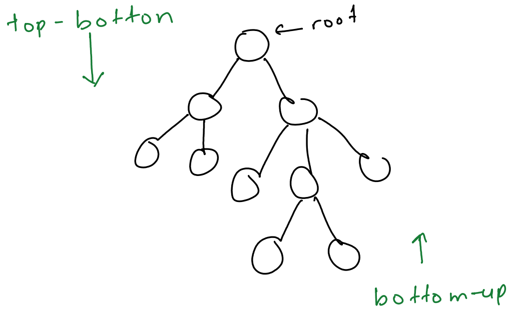
  Find pre-order traversal.

DP on Trees

Original Problem : the input tree

Sub Problems : sub-trees

Bas Cases : leaf, empty subtree

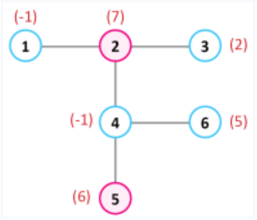


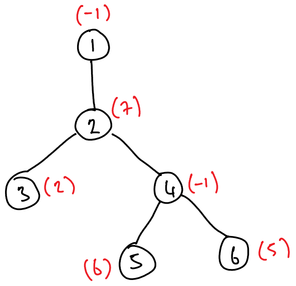
Problem 1: Maximum Sum Path

Given a tree T of N (1 <= N <= 100,000) nodes, where each node i has a value Vi (-10,000 <= Vi <= 10,000). You have to find a path between root and another node such that sum of values of the chosen nodes on the path is the largest when no two adjacent nodes (i.e. nodes connected directly by an edge) are chosen. The length of the path can be 0 and the node 1 is the root node.

```
Sample Input          Sample Output

6                13
-1 7 2 -1 6 5
1 2
3 2
4 6
5 4
2 4
```





$\overset{-1}{1} \to \overset{7}{\underline{2}} \to \overset{2}{3} \Leftarrow 7$

$\overset{-1}{1} \to \overset{7}{\underline{2}} \to \overset{-1}{4} \to \overset{5}{\underline{5}} \Leftarrow 12$

$\overset{-1}{1} \to \overset{7}{\underline{2}} \to \overset{-1}{4} \to \overset{6}{\underline{6}} \Leftarrow 13$

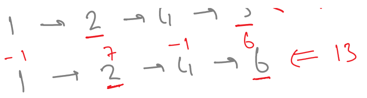
How to solve the same problem on an array ?

i

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

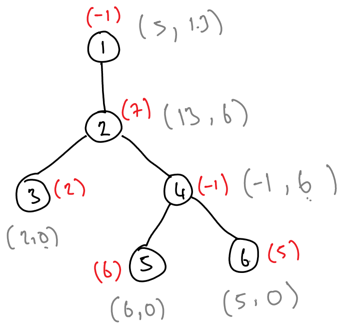
	0	1	2	3	4	5	6	7	8
A	2	-4	6	7	2	8	-6	9	4
dp in	2	-4	8	9	10	17	4	26	21
dpout	0	2	2	8	9	10	17	17	26

} max is the answer.



dpIn[i] = A[i] + dpOut[i-1]

dpOut[i] = max(dpIn[i-1], dpOut(i-1]);



```
int N;
vector<int> nodes;
vector<vector<int>> adjList;
vector<int> inVec, outVec;

void readInput()
{
        N, nodes, adjList
}

int main()
{
        inVec.resize(N+1, 0);
        outVec.resize(N+1, 0);

        dfs(1,1); // root is 1.
    cout << max(inVec[1], outVec[1]); }
}
```

```
void dfs(int u, int p) // current node, parent of it
{
        int maxChildIn = 0, maxChildOut = 0;
        for(int v: adjList[u])
        {
                if (v==p)
                        continue;
                dfs(v, u);
                maxChildIn = max(maxChildIn, inVec[v]);
                maxChildOut = max(maxChildOut, outVec[v]);
        }
        inVec[u] = nodes[u] + maxChildOut;
        outVec[u] = max(maxChildI, maxChildOut)
}
```

Problem 2: Maximum Sum Path 2

Given a tree T of N (1 <= N <= 100,000) nodes, where each node *i* has a value Vi (-10,000 <= Vi <= 10,000). You have to find a path between any two nodes such that sum of values of the chosen nodes on the path is the largest when no two adjacent nodes (i.e. nodes connected directly by an edge) are chosen. The length of the path can be 0.

Sample Input **Sample Output**

9			19

6 -1 -2 9 3 -1 7 8 2

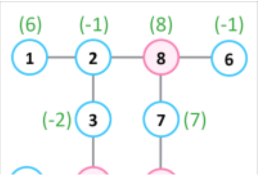
1 2

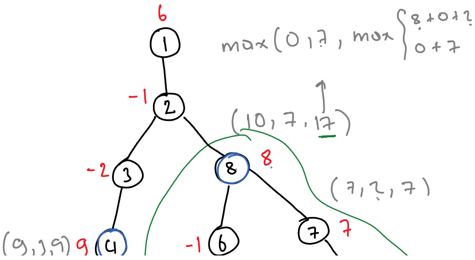
3 2

2 8

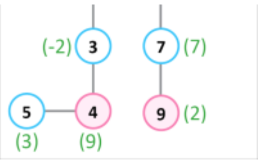
3 4

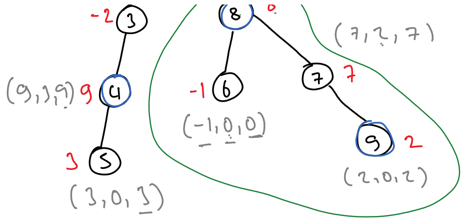
4 5

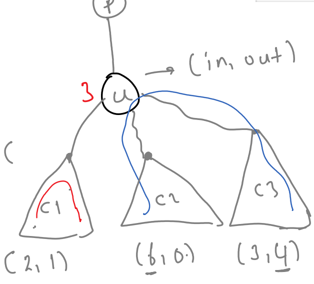




```
2 8
3 4
4 5
6 8
7 8
7 9
```





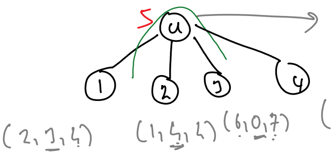


(in, out, sol)

solIn[u] =

solOut[u] = 6 + 4 = 10

dp[u] ← best result in subtree u.



$$\max \begin{cases} 5+4+5 = 14 \\ 6+5 = 11 \end{cases}$$

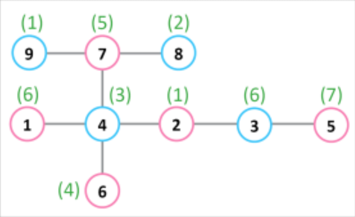
```
void dfs ( int u, int p )
{
    int max1ChildIn, max2ChildIn, max1ChildOut, max2ChildOut ;
```

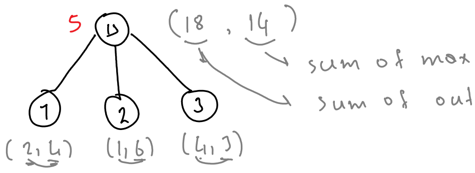
Problem 3: Maximum Sum Subset

Given a tree T of N (1 <= N <= 100,000) nodes, where each node i has a value Vi (1 <= Vi <= 10,000). You have to choose a subset of nodes such that no two adjacent nodes (i.e. nodes connected directly by an edge) are chosen and sum of values of the nodes in chosen subset is maximum.

Sample Input			Sample Output
9			23

```
6 1 6 3 7 4 5 2 1
2 3
2 4
5 3
6 4
7 4
4 1
8 7
7 9
```





(in, out)

Problem 4: K-Leaf Tree

You are given an unweighted, undirected tree with N (1 <= N <= 100,000) nodes. There is a weight wi (1 <= wi <= 1000) assigned to each node. The aim is to delete enough nodes from the tree so that the tree is left with precisely K (0 <= K <= 1000) leaves.

The cost of such a deletion is the sum of the weights of the nodes deleted. After deleting some nodes, tree still should be connected.

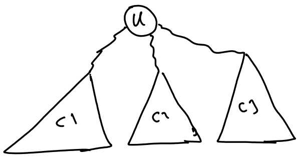
What is the minimum cost to reduce the given tree into a tree with K leaves?

PS: Root is not considered as a leaf node.

precalculation: total weight of every subtree

treeWeight[u] → total weight of subtree u.

treeCost[u][k] ← best cost for the subtree u if it has k leaves.



treeCost[leaf][0] = nodes[leaf]

treeCost[leaf][1] = 0

treeCost[leaf][k] = ∞ k > 1

} base case

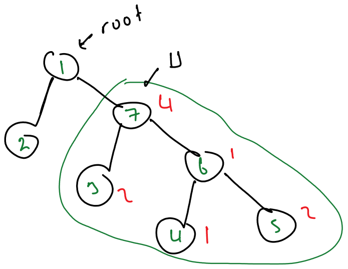
treeCost[intNode][0] = treeWeight[u]

treeCost[7][3] = 0

treeCost[7][2] = 1

treeCost[7][1] = 3

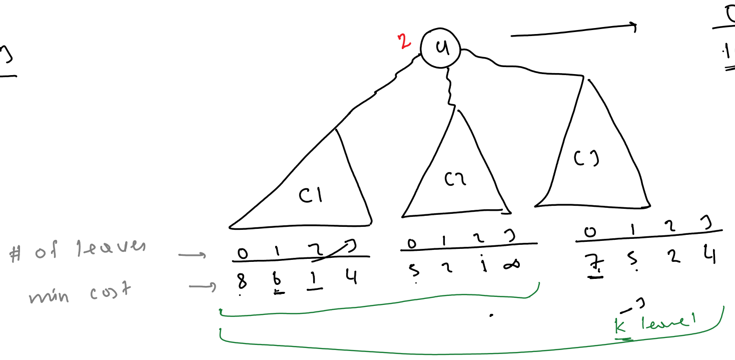
treeCost[7][0] = 10



treeCost[intNode][0] = treeWeight[intNode]

treeCost[intNode][k] = ? k → 1 to K

K = 3



0	1	2	3
16	6		

C[i][j] ← for the first i subtrees if we have j leaves, best cost.

C[i][j] ← min cost for j leaves if we consider the first i children.

C[i][0] = treeWeight[1] + treeWeight[2] + . . + treeWeight[i] or
treeCost[1][0] + treeCost[2][0] + . .

C[1][j] = treeCost[child1][j]

C[i][j] = min (C[i][j], C[i-1][k] + treeCost[i][j-k]) k → 0 to j