DP on Trees tutorial:

Basic DP Question

Q. You are given an array consisting of n integers

Task: You need to find the subsequence from the array such that the sum of all the elements in the subsequence is maximum and you are not allowed to take the two adjacent elements of the array in the subsequence.

Approach 1:

Let’s define the state dp[i] -> the ans considering i-th element as the last element in the array.

Recursive formula:

dp[i]=max(dp[i-1],ar[i]+dp[i-2])

Base condition:

dp[0]=0,dp[1]=ar[1]

Approach 2:

Let’s define the state as dp[i][0] -> means ans till i and not including i-th element in the ans.

dp[i][1] -> means ans till i and including the i-th element in the ans

Recursive formula:

dp[i][0] = max(dp[i-1][1],dp[i][0])

dp[i][1] = ar[i] + dp[i-1][0]

Base condition:

dp[1][1] = ar[1] , dp[1][0] = 0

DP on trees:

Q. You are given a Tree with N nodes and M edges , where each node i has Ci coins attached with it.

Task: You need to choose the subset of the nodes such that no two adjacent(nodes directly connected by an edge) nodes are chosen and the sum of the values of the coins linked with that is maximum.

Special things in dp on tree:

While solving the above problem we are solving for the first I elements but in case of the trees we solve for the subtrees of that tree.

So similar to the above question we also have two approaches.

For considering the subtrees of a tree we first need to root the tree so lets suppose our tree is rooted at 1.

Approach 1:

Let’s define the state as dp[v] -> ans for the subtree at node V, so from here our final answer is dp[1].

Recursive Relation:

dp[v] = max( , C(v) + )

Approach 2:

Let’s define the state as dp[v][0] -> ans for the subtree when we not include node v in our ans, dp[v][1] -> ans for the subtree at node V when we include node v in our ans.

So our final answer is max(dp[v][0],dp[v][1])

Recursive Relation:

dp[v][1] = C[v] +

dp[v][0] =

here x is the no. of the children of node v.

Q. You are given a tree with N nodes. Task: You need to find the diameter of the tree.

So let’s say that there is a node X :

1. There is a longest path f(x) in the subtree of node X starting at X.
2. The longest path passing through the node X is g(x).

Recursive Relation:

1. f(V) = 1 + max(f(v1),f(v2),….f(vn)), where v1…vn are the children of V.
2. g(V) = 1 + sum of two max element in (f(v1),f(v2)…f(vn)).

Q. You are given a tree with N nodes and an integer k.

Task: You need to find the number of different sub trees with size less than or equal to k.

Sub tree -> It’s a subset of nodes of the original tree such that this subset is connected.

First let’s see how to count the number of sub trees.

States for the problem:

1. dp[v][1] -> count when including the node V.
2. dp[v][0] -> count when not including the node v.

Recursive Relation:

dp[v][1] =

dp[v][0] =

so here our answer is dp[1][1] + dp[1][0].

Q. You are given a tree where node i has cost Ci, you need to start at some node considering that node as root and navigates to the node which is not visited yet at random, you will stop once there are no unvisited nodes, such a path takes total time equal to sum of costs of all the nodes visited.

Task: What node should be assigned as root such that expected total time is minimized?

Expected Value:

Let’s suppose we have to roll a die with 3 faces

E.V = (1/3) \* 1 + (1/3) \* 2 + (1/3) \* 3 = 2

OR

E.V = (1 / 3) \* (1 + 2 + 3) = 2

Approach:

Let’s suppose our tree is rooted at 1 define the state f[v] -> expected total time assuming that we need to calculate the answer for subtree at node v rooted at v.

Recursive relation:

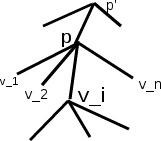
f[v] = C[v] + ()/x

so we can see that from this recursive relation we can find the value for f[1] which is the answer if we root at tree 1,

so if we calculate like this we will see that the time complexity of this approach will be O(n^2) as we need to root the tree at each vertex

As we have calculated f[v] which cannot be the answer if we have calculated f[v] when tree is rooted at u (u != v) because we have included all its children but haven’t included it’s parent

So let’s define a state g[v] -> it is the total expected time not including the subtree of v and including the parent.



Recursive Relation:

Now if we want to root the tree at node v and calculate the total expected time is:

C[v] + (g[v] + )/(n + 1)

g[v[i]] = C[p] + (g[p] + )/n

both g and f can be calculated in O(n) time complexity.