# Trees

## Introduction :

Trees are a special type of graphs with some specific properties.

First of all, what are graphs ?

Graph is a way to represent the relations between data  or objects. Graphs are made of points called vertices (nodes) which are connected with edges (Relations). Nodes in graphs may represent : people , places and any things, where edges represents the connection and the relations between these nodes.

There is two types of edges : directed , which take you from a to b and not vice versa, and undirected, which take you from a to b and from b to a.

How to build graphs :

Graphs are usually build by giving the user the number of nodes (n) and the number of edges (m) in the graph then (m) pairs v , w, which represent a connection between node u and node v

Lets take an example :

If we want to assume that nodes are Cities and edges are roads that connects the cities

As we see there are many ways to go from node 1 to node 4 which may differ in length,

Also there is no way to go from node 2 to 5 so we call them disconnected

Graphs have n nodes and m edges and not every node is connected.

Cycles : means that you start from one node follow so edges (While using each edge once) and go back to itself.

## Properties :

1 - Trees are a connected graph with undirected edges.

2 - In trees every node is connected by exactly one unique path.

3 - It has no cycles.

4 - Tree of n nodes have n-1 edges.

5 - If you remove any edge from tree it becomes disconnected

## Ways to represent trees :

1 - Give edges pairs (u , v).

2 - For each node give parent.

3 - In binary trees give left and right childs.

4 - Give all childs of the current node.

## Tree Traversing :

If we trying to traverse a tree we only need to make sure to go back up from the current node (Moves from node u to it’s parent is not allowed because to will cause infinite loops) so while traversing a tree we just need to keep track of the number of the current node and its parent.

In case of dealing with pointers we don’t need to know what is the parent

Task 1 : find the size of each subtree (n <= 1000).

Task 2 : find the size of each subtree (n <=10^5).

## Tree numbering:

Tree numbering technique is used by giving each node a new value based in the time we enter it and the time we leave it.

\* If a node’s v enter and leave values are inside node u enter and leave values thats mean that v is inside u subtree

Task 1 : Given node u , v check if v is in the subtree of u

Task 2 : Given node u find the sum of values in subtree u;

Task 3 : <https://www.hackerrank.com/challenges/tree-preorder-traversal/problem> // Preorder

Task 4 : <https://www.hackerrank.com/challenges/tree-postorder-traversal/problem> // Postorder

Task 5 : <https://www.spoj.com/problems/TREEDEGREE/> // Restore tree using euler array

## Requirements :

Vectors :

<https://www.geeksforgeeks.org/vector-in-cpp-stl/>

Pointers :

<https://www.tutorialspoint.com/cplusplus/cpp_pointers.htm>

<https://www.w3schools.com/cpp/cpp_pointers.asp>

<http://www.cplusplus.com/doc/tutorial/pointers/>

<https://www.youtube.com/watch?v=DTxHyVn0ODg>

## Problems :

<https://www.hackerrank.com/domains/data-structures?filters%5Bsubdomains%5D%5B%5D=trees> // Nice Tree problems

<https://codeforces.com/contest/796/problem/C>

<https://www.spoj.com/problems/UCV2013J/>

<https://www.spoj.com/problems/GCPC11J/>

<https://www.spoj.com/problems/TREEISO/> //tree Isomorphism not rooted

<https://www.hackerrank.com/challenges/jenny-subtrees/problem> //Isomorphism cuts

<https://www.hackerrank.com/challenges/balanced-forest/problem>

Done :

<https://cses.fi/problemset/task/1700> //tree Isomorphism rooted

<https://codeforces.com/problemset/problem/1336/A?fbclid=IwAR1T6yOmAFdBqfHmeeQrjQwh61VD1SfHfzYY63ij6lApkyKT6T7JwpgyL5M> // greedy

<https://www.hackerrank.com/challenges/tree-level-order-traversal/problem> // Print by level

<https://www.hackerrank.com/challenges/tree-top-view/problem> //Tree traversing

<https://codeforces.com/contest/34/problem/D> // Find the new Parent array

<https://codeforces.com/contest/828/problem/D> // Hard + Constructive

<https://codeforces.com/contest/1189/problem/D1> // Greedy

<https://www.hackerrank.com/challenges/tree-preorder-traversal/problem> // Preorder

<https://www.hackerrank.com/challenges/tree-postorder-traversal/problem> // Postorder

<https://www.spoj.com/problems/TREEDEGREE/> // Restore tree using euler array

<https://www.hackerearth.com/practice/algorithms/graphs/graph-representation/practice-problems/algorithm/monk-at-the-graph-factory/> //Check if tree

[Find size of graph](https://www.hackerearth.com/practice/algorithms/graphs/graph-representation/practice-problems/algorithm/monk-in-the-real-estate/description/)

<https://codeforces.com/contest/913/problem/B> // Easy - Tree - Implemeination

## More Resources :

<https://www.youtube.com/watch?v=zEQZpTizgLo>

<https://www.youtube.com/watch?v=p8MFuDxvnuo>

<https://en.wikipedia.org/wiki/Tree_(graph_theory)>

<https://www.geeksforgeeks.org/difference-between-graph-and-tree/>