2196. Create Binary Tree From Descriptions

```
/*
    Time complexity: O(nlogn)
    Extra space complexity: O(n): adjacency list,
                            O(n): recusrion,
                            O(n): has_parent

    [[40,50,1],[40,60,0],[60,70,0],[10,20,1],[20,40,1],[10,30,0]]

    [[39,70,1],[13,39,1],[85,74,1],[74,13,1],[38,82,1],[82,85,1]]
*/
class Solution {
   public:
      std::map<int,std::vector<ii>> adj;
   public:
      void build_adjacency_list(std::vector<std::vector<int>>& descriptions){
         for(auto& desc: descriptions){
            adj[desc[0]].push_back({desc[1],desc[2]});
         }
      }

      int find_root(std::vector<std::vector<int>>& descriptions){
         std::map<int,bool> has_parent;
         for(auto& desc: descriptions){
               has_parent[desc[1]]=true;
         }
         for(auto& desc: descriptions){
               if (!has_parent[desc[0]]) return desc[0];
         }
         return -1;
      }

      TreeNode* createBinaryTree(std::vector<std::vector<int>>& descriptions){

         build_adjacency_list(descriptions);
         int root_value=find_root(descriptions);

         auto solve=[&](int node_value,auto& self)->TreeNode*{
            TreeNode* node=new TreeNode(node_value);
            for(auto& p: adj[node_value]){
               TreeNode* res=self(p.first,self);
               if(p.second==1) node->left=res;
               else node->right=res;
            }
            return node;
         };
         return solve(root_value,solve);
      }
};
```