Dynamic Programming on Tree

MCPC-2020 Winter Training

Shizhe Zhao (eggeek)

Introduction

- DP is everywhere: contest, industry, research, daily life ...
- Tree is also everywhere ...
- DP on Tree is a good start point

Tree: definition

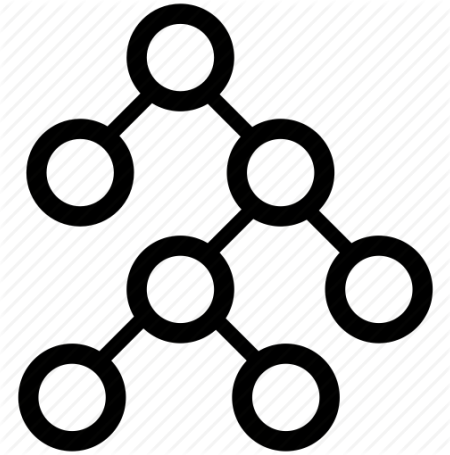


Figura: Unroot tree

- Tree: has 1 (or 0) root
- Node: has parent (1 or 0) and children (0 or many)
- Root: has 0 parent
- Leaf: has 0 child

Tree: definition

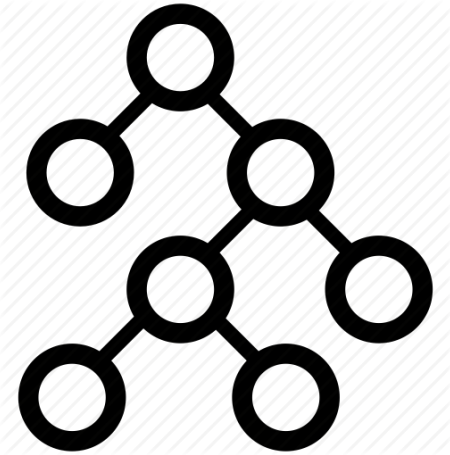


Figura: Unroot tree

- Tree: has 1 (or 0) root
- Node: has parent (1 or 0) and children (0 or many)
- Root: has 0 parent
- Leaf: has 0 child

Warm up Question: how many edges in a tree with n nodes?

Tree: definition

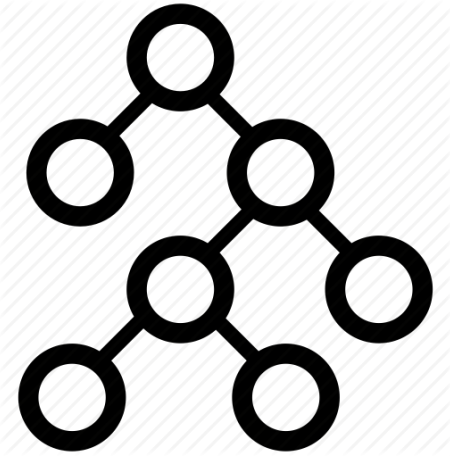


Figura: Unroot tree

- Tree: has 1 (or 0) root
- Node: has parent (1 or 0) and children (0 or many)
- Root: has 0 parent
- Leaf: has 0 child

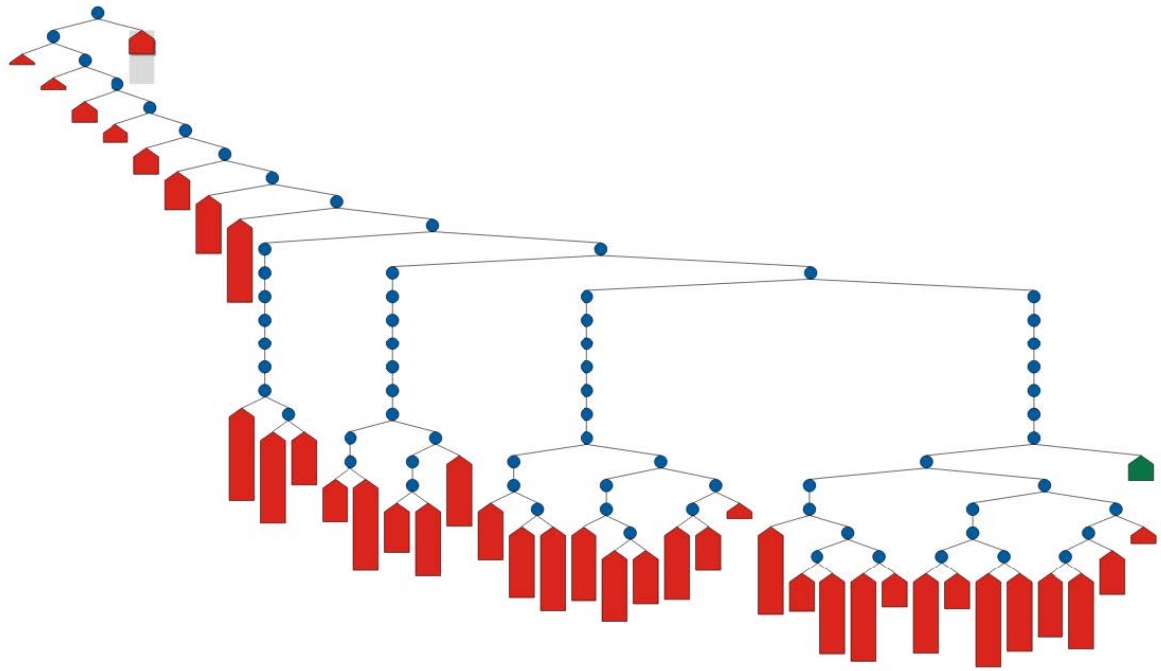
Warm up Question: how many edges in a tree with n nodes?
A: $\mathbf{n-1}$.

Tree: examples

- Tree (biological)
- File system
- Package manger system
- and more advance examples . . .

Tree: examples

Search space

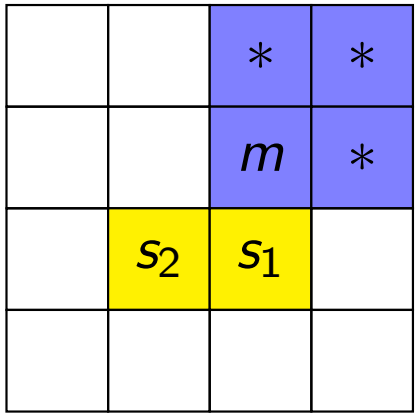


Search space of a MiniZinc solver

Tree: example

Dijkstra propagation is also a tree

- *Dijkstra*s (on different sources) are not independent
- $sp(s_2, m) + sp(m, *) = sp(s_2, *)$



Tree: diameter

Definition

Diameter of a tree is $\mathbf{max}(\mathbf{dist}(\mathbf{u}, \mathbf{v}) | \mathbf{u}, \mathbf{v} \in \mathbf{Tree})$

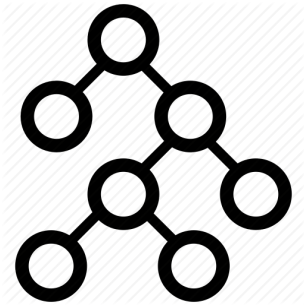
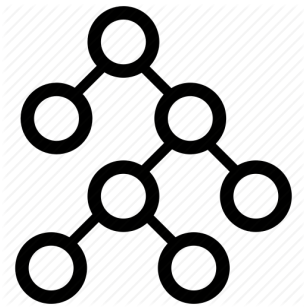


Figura: Diameter is 4

Tree: diameter

Definition

Diameter of a tree is $\mathbf{max}(\mathbf{dist}(\mathbf{u}, \mathbf{v}) | \mathbf{u}, \mathbf{v} \in \mathbf{Tree})$



Solution:

- randomly choice a root u

Figura: Diameter is 4

Tree: diameter

Definition

Diameter of a tree is $\mathbf{max}(\mathbf{dist}(\mathbf{u}, \mathbf{v})|\mathbf{u}, \mathbf{v} \in \mathbf{Tree})$

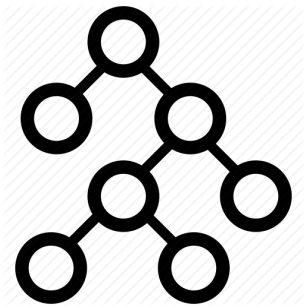


Figura: Diameter is 4

Solution:

- randomly choice a root u
- find the furthest node v to u

Tree: diameter

Definition

Diameter of a tree is $\mathbf{max}(\mathbf{dist}(\mathbf{u}, \mathbf{v}) | \mathbf{u}, \mathbf{v} \in \mathbf{Tree})$

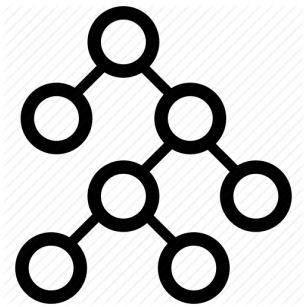


Figura: Diameter is 4

Solution:

- randomly choice a root u
- find the furthest node v to u
- find the furthest node w to v, $dist(w, v)$ is diameter

Tree: diameter

Definition

Diameter of a tree is $\mathbf{max}(\mathbf{dist}(\mathbf{u}, \mathbf{v}) | \mathbf{u}, \mathbf{v} \in \mathbf{Tree})$

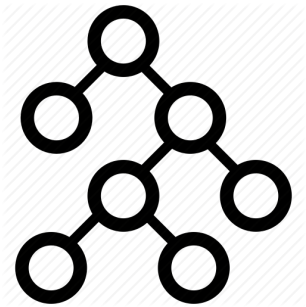


Figura: Diameter is 4

Solution:

- randomly choice a root u
- find the furthest node v to u
- find the furthest node w to v, $dist(w, v)$ is diameter

Reasoning:

v must be a leaf; and it must be one of the end point on the "diameter path"

(hint: proof by contradiction, or `http://courses.csail.mit.edu/6.046/fall01/handouts/ps9sol.pdf`)

Dynamic Programming

- **Dynamic:** consider the current 'status'
- **Programming:** making decision

Dynamic Programming

- **Dynamic:** consider the current 'status'
- **Programming:** making decision

In narrow sense: making decision based on current status to achieve global optimal.

Dynamic Programming

- **Dynamic:** consider the current 'status'
- **Programming:** making decision

In narrow sense: making decision based on current status to achieve global optimal.
In broad sense: divide and conquer, memorization, combinatorial counting, ...

Dynamic Programming on Tree

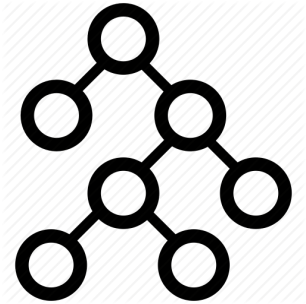
Classic DP examples: knapsack, coin change . . . ,
they are not friendly to beginners:

- context sensitive: any small changes on problem statement (i.e. scenario description) cause a very different model.
- abstract: models usually imply a DAG (Directed Acyclic Graph) in multi-dimension

while Tree DP is relative easier as the structure is given, and you only need to focus on fit your problem in tree model.

Dynamic Programming on Tree

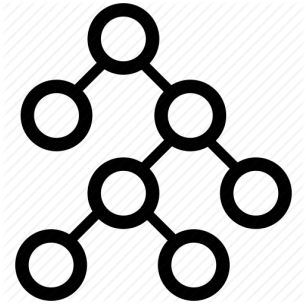
Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.



Dynamic Programming on Tree

Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.

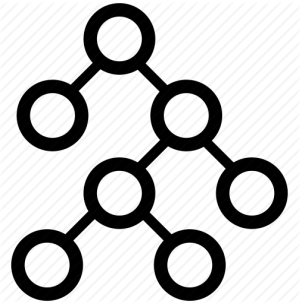
- Randomly choose a root r;



Dynamic Programming on Tree

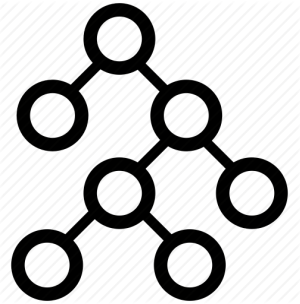
Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.

- Randomly choose a root r;
- Precompute $L_r[i]$ for each node i: the max distance from i to any leaf in it's subtree



Dynamic Programming on Tree

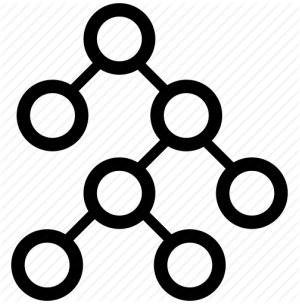
Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.



- Randomly choose a root r;
- Precompute $L_r[i]$ for each node i: the max distance from i to any leaf in it's subtree
- Compute the max distance of path that pass through r $(max(L_r[a] + 1 + L_r[b] + 1)_{a,b \in children(r)})$.

Dynamic Programming on Tree

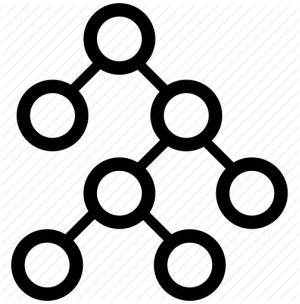
Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.



- Randomly choose a root r;
- Precompute $L_r[i]$ for each node i: the max distance from i to any leaf in it's subtree
- Compute the max distance of path that pass through r $(max(L_r[a] + 1 + L_r[b] + 1)_{a,b \in children(r)})$.
- Move the root to an adjacent node r', and regard r as a child of r'

Dynamic Programming on Tree

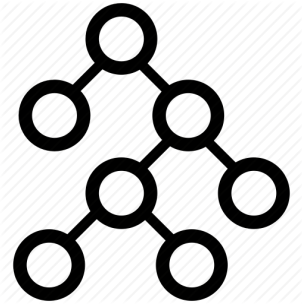
Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.



- Randomly choose a root r;
- Precompute $L_r[i]$ for each node i: the max distance from i to any leaf in it's subtree
- Compute the max distance of path that pass through r $(max(L_r[a] + 1 + L_r[b] + 1)_{a,b \in children(r)})$.
- Move the root to an adjacent node r', and regard r as a child of r'
- Notice for all nodes under the subtree rooted at r', $L_{r'}[i]$ is same as previous iteration(i.e. when root is r)

Dynamic Programming on Tree

Let's overkill the tree-diameter problem.
Observation: diameter passes through the root or present in any subtree.



- Randomly choose a root r;
- Precompute $L_r[i]$ for each node i: the max distance from i to any leaf in it's subtree
- Compute the max distance of path that pass through r $(max(L_r[a] + 1 + L_r[b] + 1)_{a,b \in children(r)})$.
- Move the root to an adjacent node r', and regard r as a child of r'
- Notice for all nodes under the subtree rooted at r', $L_{r'}[i]$ is same as previous iteration(i.e. when root is r)
- Repeat the 3rd step

Static memory location is limited

Only using static allocated memory can be a bit complicated in some cases:

- Store a graph: native way need $O(n^2)$ space

```
int nodes[N][N];
```

- Describe discrete multi-dimension status

```
// count the number of ways in grid
if (condition1) cnt[x][y] += cnt[x-1][y];
if (condition2) cnt[x][y] += cnt[x][y-1];
```

Using STL

```
1   // store a graph with at most N nodes
2   const int N = 1000;
3   vector<int> g[N];
4   // dynamically
5   vector<vector<int>> g;
6   g.resize(n);
7
8   // count the number of ways in grid
9   typedef pair<int, int> pii;
10  pii status = {x, y};
11  map<pii, int> cnt; // like Python dict
12  if (condition1) cnt[{x, y}] += cnt[{x-1, y}];
13  if (condition2) cnt[{x, y}] += cnt[{x, y-1}];
```

Using STL - example: max *dist* from root to leave

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1000;
vector<vector<int>> tree;
vector<int> dist;

// call dfs(0, -1) for root=0
void dfs(int cur, int pa) {
  dist[cur] = 0;
  for (auto& it: tree[cur]) if (it != pa) { // recurse to all neighbors except for the parent
    dfs(it, cur);
    dist[cur] = max(dist[cur], dist[it] + 1);
  }
}

void read_tree() {
  int n, u, v;
  cin >> n;
  tree.resize(n); dist.resize(n);
  // tree has exactly n-1 edges
  for (int i=0; i<n-1; i++) {
    cin >> u >> v;
    // add bidirectional edge
    tree[u].push_back(v); tree[v].push_back(u);
  }
}
```