Tree and Recursion

Monash ICPC workshop

Shizhe Zhao

About the workshop

What's not?

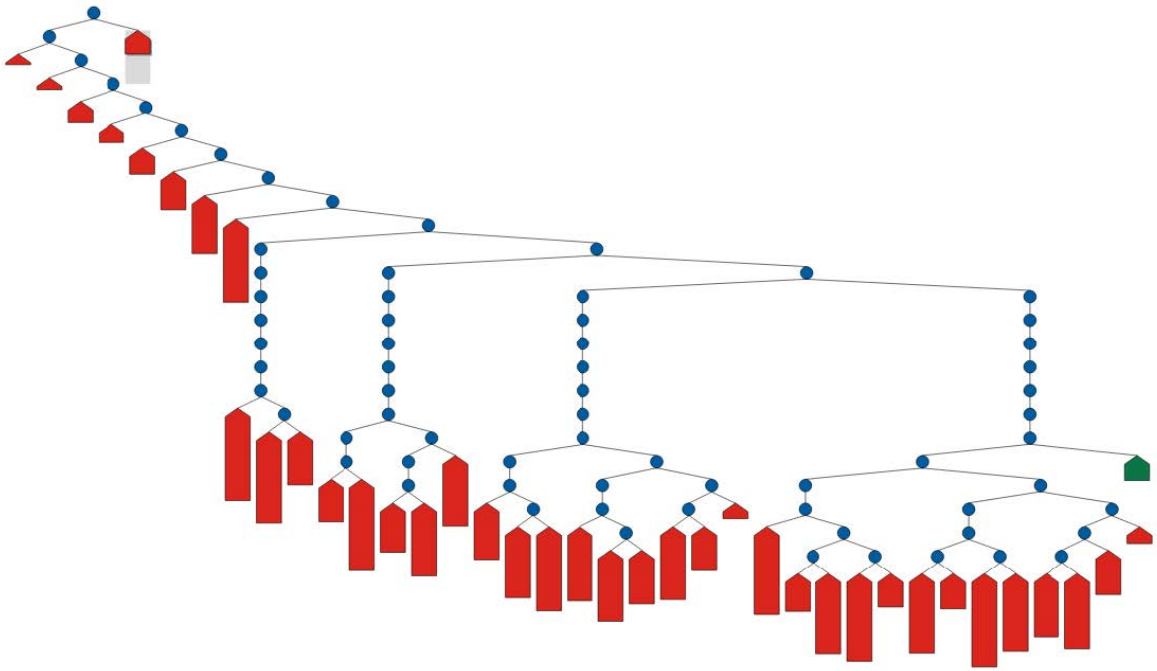
- Not a revising of lectures
- No spoon-feeding

What's it?

- Beyond your algo-units: extensions, application ...
- Filling up the gap between knowledge and practice.
- Develop your problem-solving skill

Intro: Tree structure in CS

Search space

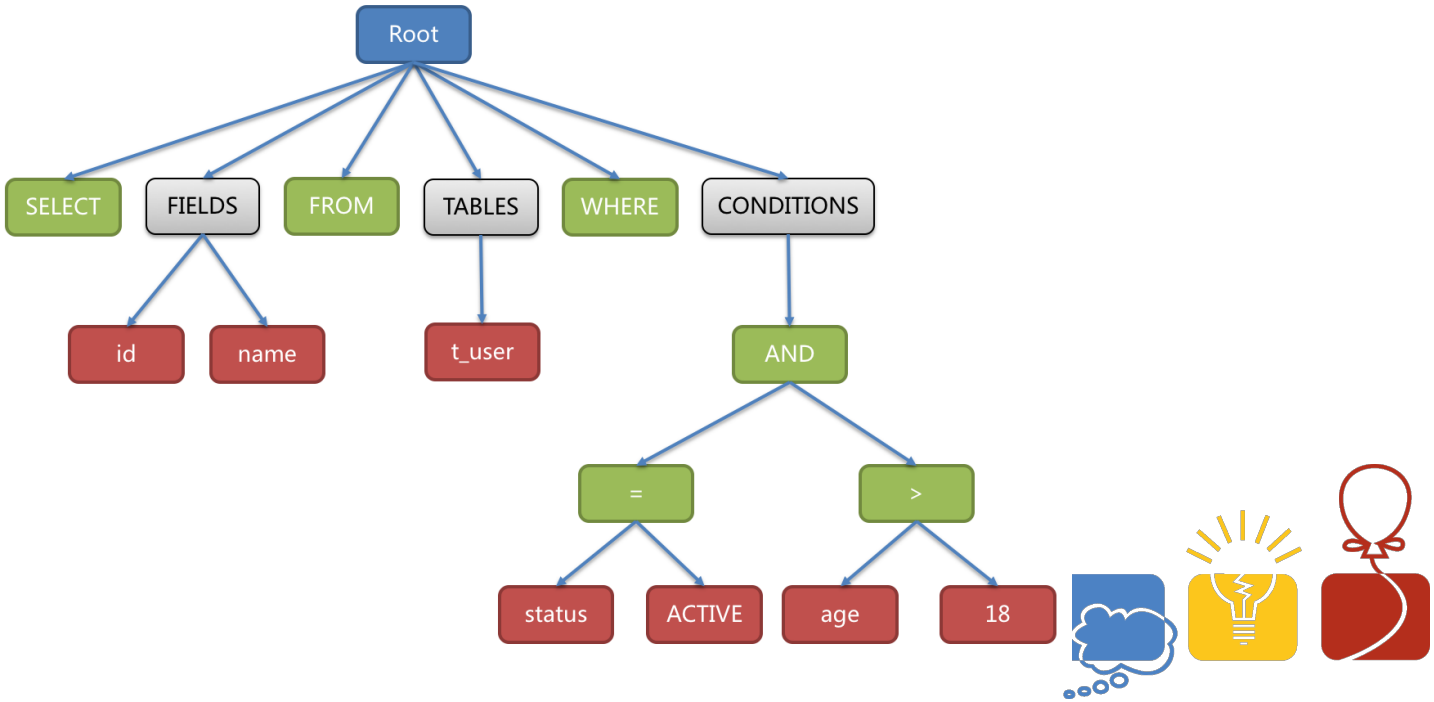


Search space of a MiniZinc solver

Intro: Tree structure in CS

Syntax

```
SELECT id, name FROM t_user WHERE status = 'ACTIVE' AND age > 18
```



Intro: Tree structure in CS

The most common recursion structure, classic examples:

- Tree traversal (Pre/In/Post, DFS)
- Dynamic Programming on tree

Tree traversals - quick review

```python
# Binary tree
def preorder(root):
  if root:
    visit(root)
    preorder(root.left)
    preorder(root.right)

def inorder(root):
  if root:
    inorder(root.left)
    visit(root)
    inorder(root.right)
```

```python
def postorder(root):
  if root:
    postorder(root.left)
    postorder(root.right)
    visit(root)

# Generic tree
def dfs(root):
  visit(root)
  for c in root.children:
    dfs(c)
  leave(root)
```

Applications of traversals

- Expression tree: pre/ dfs order

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.
  - Then you can feed the vector to a neural network, magic!

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.
  - ~~Then you can feed the vector to a neural network, magic!~~

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.
  - ~~Then you can feed the vector to a neural network, magic!~~
  - Then you can represent a subtree by "slicing", magic!

Applications of traversals

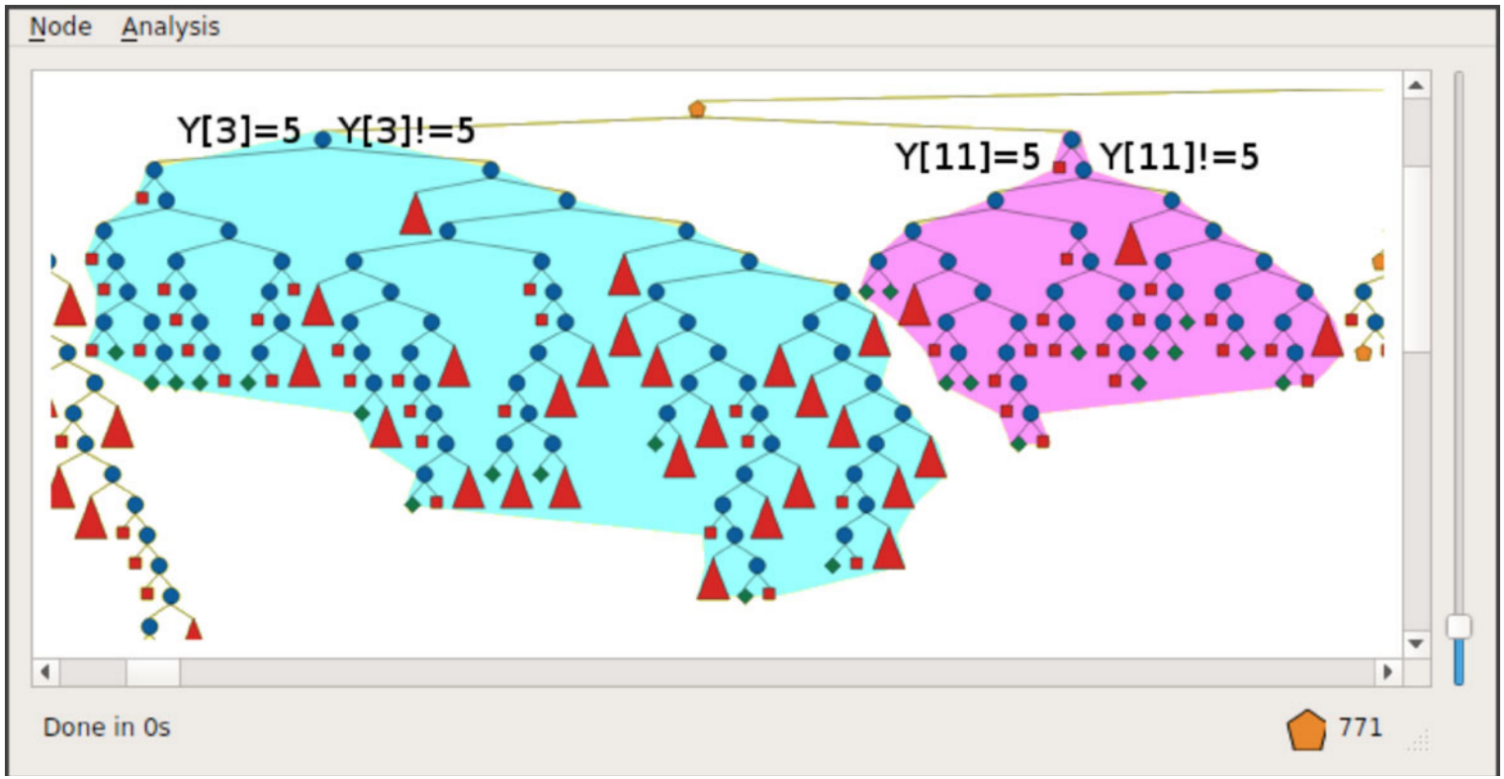
- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.
  - ~~Then you can feed the vector to a neural network, magic!~~
  - Then you can represent a subtree by "slicing", magic!
  - LCA (Lowest Common Ancestor).

Applications of traversals

- Expression tree: pre/ dfs order
- Binary search tree: in order
- Deleting the tree: post order
- Map a tree to a vector.
  - ~~Then you can feed the vector to a neural network, magic!~~
  - Then you can represent a subtree by "slicing", magic!
  - LCA (Lowest Common Ancestor).

Hope we can cover these topics in the future.

Example: Tree Visualisation

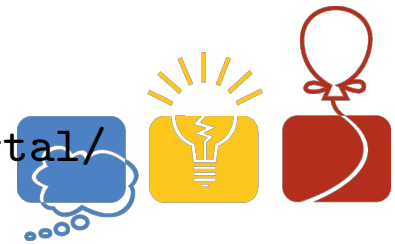


Visualize a tree with more than 10M nodes.
Scroll left/right/up/down.

Example: Tree Visualisation

- solution-1: `render(tree)` and only show visible part [1]
  - rendering: 15s
  - scrolling delay: 200ms
- solution-2: `render(leftMostId, rightMostId)` and show [2]
  - rendering: 200ms
  - scrolling delay: 20ms

Data structure is important!



[1] https://researchmgt.monash.edu/ws/portalfiles/portal/257776846/3566886_oa.pdf

[2] https://github.com/eggeek/cp-profiler

Example 2: Company Party

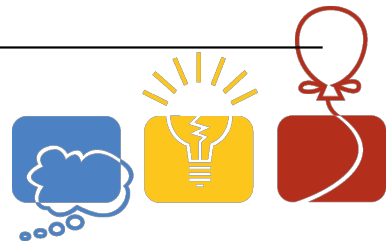
Brief:

- There are n employees, each one has 0 or 1 direct leader - forest structure
- Employees can not in a party group that contains their ancestor
- What's the minimum number of group?

Example 2: Company Party

```
1  def dfs(i):
2    if fa[i] == -1:
3      return 1
4    return dfs(fa[i]) + 1
5
6  n = int(input())
7  fa = [0] * (n+1)
8  for i in range(1, n+1):
9    fa[i] = int(input())
10 res = max([dfs(i) for i in range(1, n+1)])
11 print (res)
```

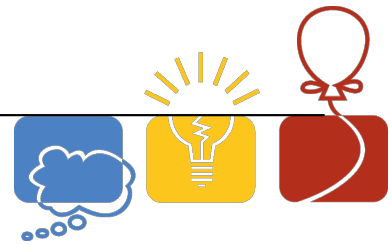
Bottom to top - $O(n^2)$.



Example 2: Company Party

```
1  def dfs(i, dep):
2          if fa[i] == -1:
3                  return 1
4          if dep[i] > 0:
5                  return dep[i]
6          dep[i] = dfs(fa[i], dep) + 1
7          return dep[i]
8
9  n = int(input())
10 fa = [0] * (n+1)
11 for i in range(1, n+1):
12         fa[i] = int(input())
13 dep = [0] * (n+1)
14 res = max([dfs(i, dep) for i in range(1, n+1)])
15 print (res))
```

With memorization - $O(n)$.



Common Patterns

```
1  import sys
2  # you may need to reset stack limit,
3  # usually default is 1000
4  sys.setrecursionlimit(...)
5
6  # Undirected
7  def dfs(pre, node, graph):
8    for i in graph.neighbors:
9      if i != pre:
10       dfs(node, i, graph)
```

Line 9 is important!

Coding time!

- https://vjudge.net/contest/364479