Data Structures

Binary Tree Serialization

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

A unique tree representation

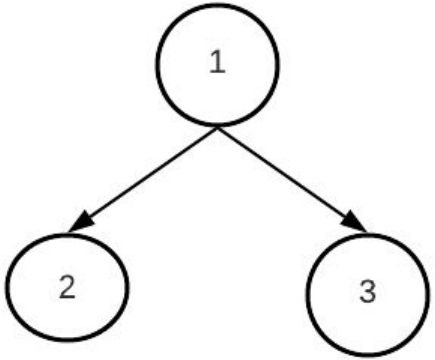
- So far we learned 2 representations for a tree to be able to reconstruct it
  - E.g. Inorder + preorder or Inorder + postorder or Inorder + level-order
- But this means we need 2 arrays to represent a tree!
- Why one representation was not enough?!
  - Because we don’t know if these values are for left or right subtrees!
  - In other words, nothing indicate null subtrees!
- To have one unique representation, simply change it to indicate null trees!
- Try implementing: void print_preorder_complete()
  - Its preorder representation is uniquely a tree
  - Assume all tree values are >= 0

Full information preorder

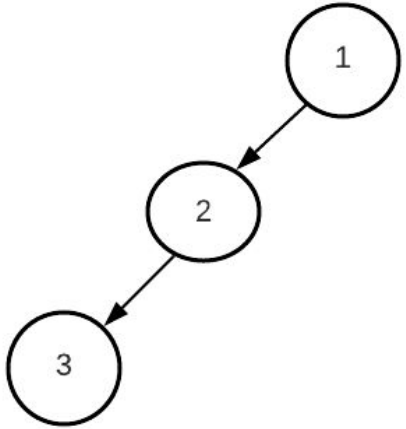
- Simply when we have a null node, print something indicate that
- E.g. -1, assuming no -1 in tree values

```
void print_preorder_complete() {
    cout<<data<<" ";

    if (left)
        left->print_preorder_complete();
    else
        cout<<"-1 ";    // 2 null pointers

    if (right)
        right->print_preorder_complete();
    else
        cout<<"-1 ";    // 2 null pointers
}
```

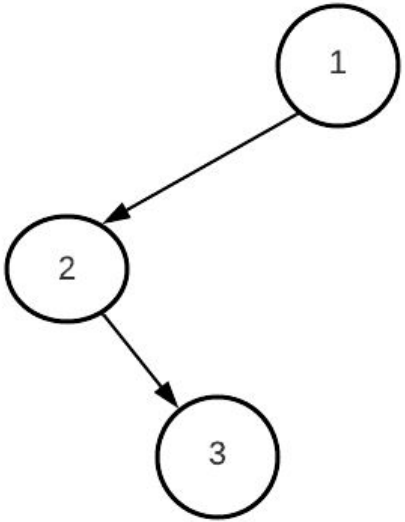
Full information preorder



1 2 **-1 -1** 3 **-1 -1**



1 2 3 **-1 -1** -1 -1



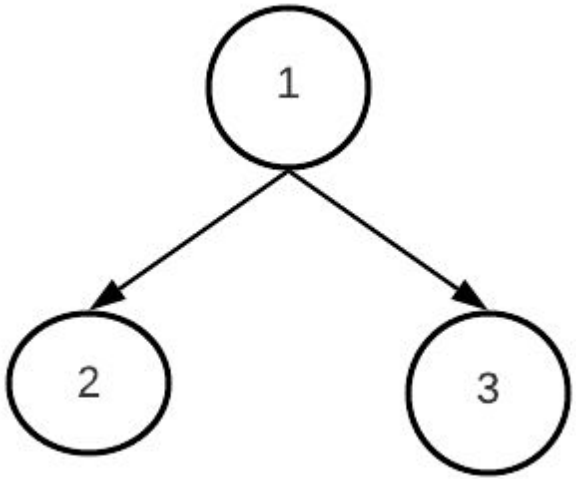
1 2 -1 3 -1 -1 -1

Seralization

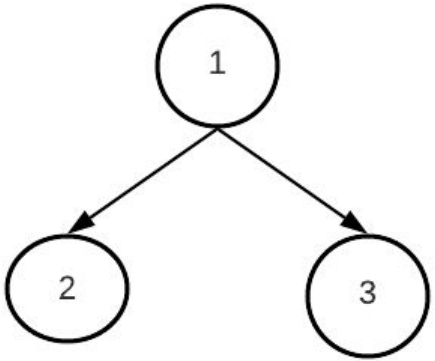
- Serialization is the process of converting a data structure to a representation that can be stored e.g. in a file
- We learned how to get a uniquely representative preorder representation
- Another interesting representation is to parenthesize the tree!
- Each tree representation is
  - (
  - Left sub-tree representation
  - Right sub-tree representation
  -)
- Then a null child is represented as ()

Parenthesizing a tree

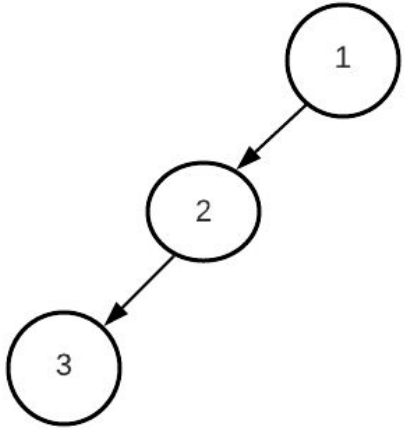
- Node 2 representation:
  - (2()())
- Node 3 representation:
  - (3()())
- Node 1 representation:
  - (1LR)
  - (1 (2()()) (3()()))



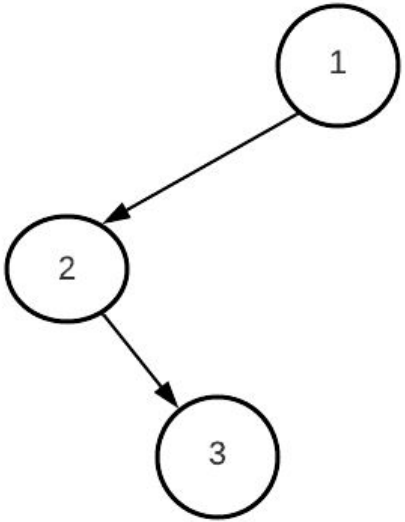
Parenthesizing a tree



(1(**2()()**)(**3()()**))



(1(2(3()())())())



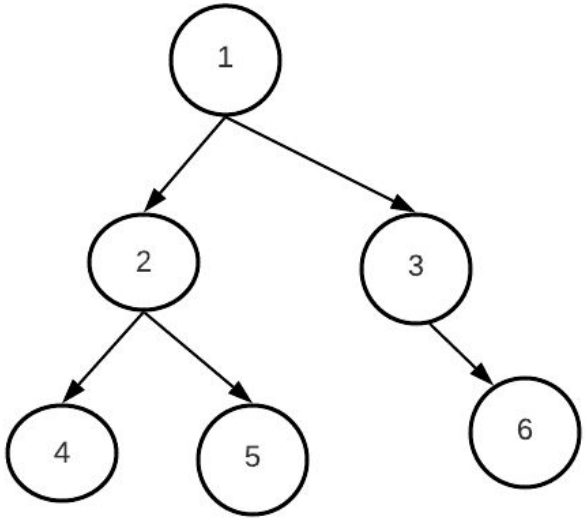
(1(2()(3**()()**))())

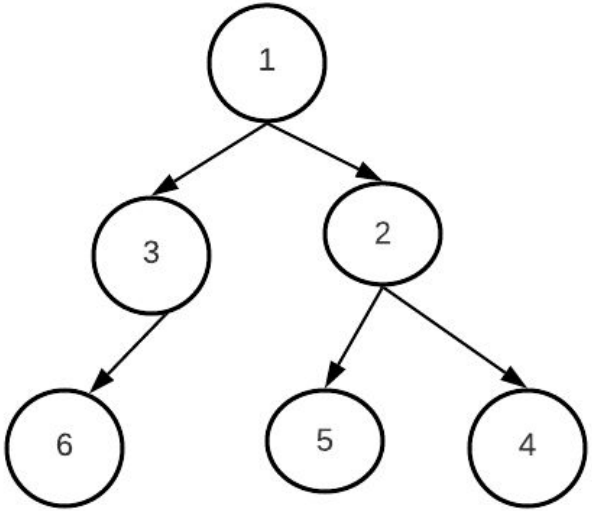
Parenthesizing a tree

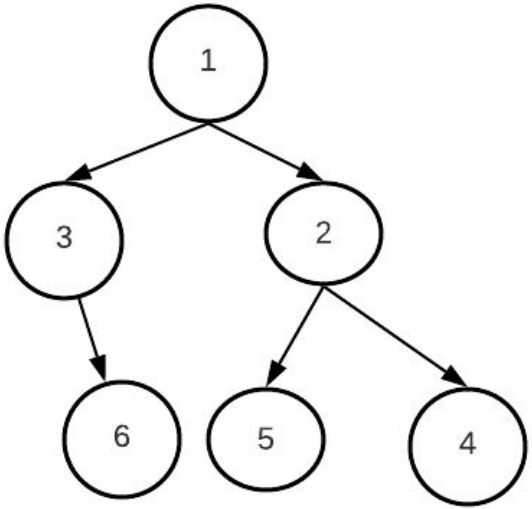
```
string parenthesize() {
    string repr = "(" + toStr(data);

    if (left)
        repr += left->parenthesize();
    else
        repr += "()";   // null: no child

    if (right)
        repr += right->parenthesize();
    else
        repr += "()";   // null: no child
    repr += ")";

    return repr;
}
```

Canonicalizing a tree

- If you have several arrays: how can u check if they have the same values?
  - Sort each array and compare them.
- Below are close trees: how can we **sort a tree**? E.g. for comparisons
  - *Still each subtree must have its old children!*







Canonicalizing a tree

- The core of the idea is simple
- When you have the **representation** of every child, put them in vector and **sort** them
- The easiest to code is with parathesizing
- Tip: if nodes values are unique we don’t need sort()

```
string parenthesize_canonical() {
    string repr = "(" + toStr(data);

    vector<string> v;

    if (left)
        v.push_back(left->parenthesize_canonical());
    else
        v.push_back("()");

    if (right)
        v.push_back(right->parenthesize_canonical());
    else
        v.push_back("()");

    sort(v.begin(), v.end());
    for (int i = 0; i < (int)v.size(); ++i)
        repr += v[i];

    repr += ")";

    return repr;
}
```

Tackling Problems

- There are many problems that depends on tree unique representation
  - Check if 2 trees are identical
  - Check if 2 trees are mirrors
  - Check if a tree has duplicate subtrees
  - Check if a tree is a subtree of another
  - Find the largest identical 2 subtrees of a tree
- We maybe able to:
  - Develop a recursive technique that tries to answer
  - Serialize each (sub)tree and easily compare
    - This is usually less buggy and less thinking!

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."