Binary Trees and Heaps

CS 491 – Competitive Programming

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Spring 2023

Objectives

- Write code to implement binary search trees, regular trees, and heaps.
- Explain the differences between these data structures.

Binary Search Trees

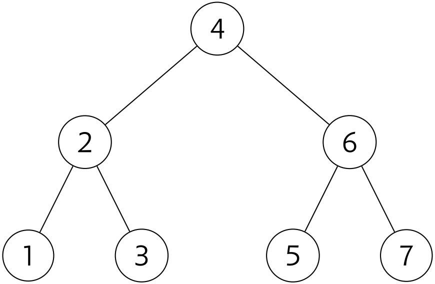
A *Binary Search Tree* is a set of vertices v_i such that:

- There is precisely one vertex with no parents, called the *root*.
- Each vertex has 0,1, or 2 children.
- Each vertex has a value of a type that supports ordering
- If a child value is less than the parent, it must be the left child.
- If a child value is greater than the parent, it must be the right child.
- If the values are equal: go left, go right, or delete. Just be consistent.

Important notes:

- Expected height of the tree is $\mathcal{O}(\log_2 n)$.
- Worst case height is $\mathcal{O}(n)$. When does this occur?

Picture



Implementing

- Use a struct for simplicity

```
struct bst<T> {
  T value;
  bst<T> *left, *right;

  bst<T>(T value) {
      this->value = value;
      left = right = NULL;
  }
}
```

- You could also be clever and use a sized-2 vector for the children, or a pair.

Add

```
bst<T> add(bst<T> *root, T value) {
  if (root == NULL) {
    return bst<T>(value);
  }
  bst<T> *parent = root;
  while (true) {
    if (value < root->value) {
      if (parent->left == NULL) {
        return parent->left = bst<T>(value);
      } else {
        parent = parent->left;
      }
    } else {
      // Same thing, but go right.
}}}
```

Find

```
bool find(bst<T> *root, T value) {
  while (root) {
    if (root->value == value) {
      return true;
    }
    if (value < root->value)
      root = root->left;
    else
      root = root->right;
    }
  return false;
}
```

Deletion

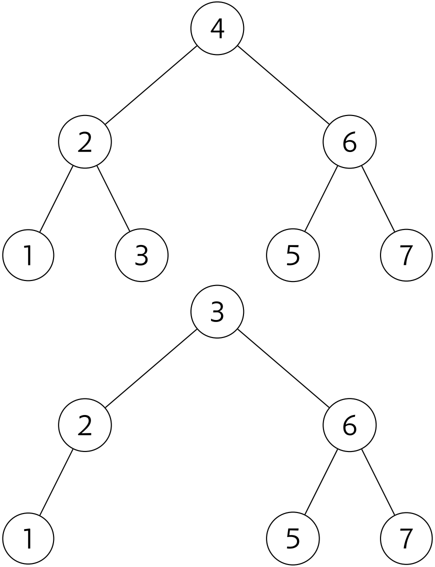
Deletion is a bit of a pain. The steps:

- Find the victim node
- Get the In Order Predecessor (IOP) of the victim node.
- Replace the victim value with the IOP value.
- Delete the IOP from the child branch.

There are edge cases!

- Deleting the last vertex
- Nodes without an IOP

Picture: Deleting 4



Regular Trees

If we don't constrain order or number of children, we just have a tree.
Trees have special properties!

- $|E| = |V| - 1$. Adding even one more edge makes it not a tree anymore.
- There are no cycles.
- Equivalently: there is exactly one path between any two nodes.

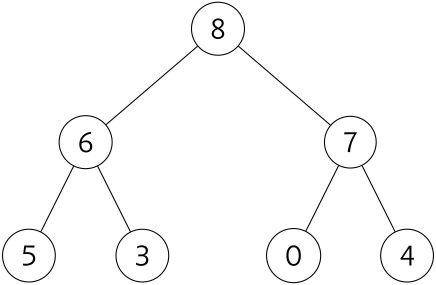
Representation:

- Usually we would use the adjacency list representation from last time to construct the tree.

Remember Heaps?

- A heap is also a binary tree.
  - Each child is smaller (max-heap) or larger (min-heap) than the parent.
- We use vectors to represent them.
  - Leave the 0 element empty as a sentinel. The math is cleaner this way.
- You rarely need to use heaps as heaps, but this method of storing a binary tree is often very efficient!

Heap Visualization



0	1	2	3	4	5	6	7
S	8	6	7	5	3	0	4

Implementing

```
int goLeft(int i) {
    return 2*i;
}

int goRight(int i) {
    return 2*i + 1;
}

int goUp(int i) {
  if (i>1) {
    return i / 2;
  }
}
```