Announcements
Trees

Def: A tree \( T \) is a set of nodes storing elements in a parent-child relationship.

\[ T \text{ has a special node } r, \text{ called the root.} \]

Each node (except \( r \)) has a unique parent.
More defns

- child
- siblings
- leaves - no children
- internal nodes - have children if are not root
- rooted subtree
- descendant / ancestor
Examples

File hierarchies

Root

CS 2100

English 1900

Temp

Family trees...
General Tree Implementation

Pointer based:

Need a list of children in each node:

/home

CS 2100 Downloads
Applications

Anything where relationships are more complex than linear orderings.

Ex:
- Family tree
- File systems
- Numeric expressions

\[(5 \times 3) + 6\]
Binary Tree
- Every node has $\leq 2$ children.

Full tree: every node has $\geq 2$ children
**Depth + Height** - defined recursively

- **depth**: \( \text{depth}(r) = 0 \)
- \( \text{depth}(v) = \text{depth}(\text{parent}(v)) + 1 \)

- **height**: \( \text{height}(\text{leaf}) = 0 \)
- \( \text{height}(v) = \max(\text{height of children}) + 1 \)
Nice trick

Can be pointers or array based!

left(i) = 2i + 1
right(i) = 2i + 2
Potential downside (of array)

Array:

\[
\begin{array}{cccccccccc}
6 & 2 & 3 & 4 & 5 & \ldots & 3 & 4 & 5 & \ldots \\
1 & 2 & 3 & 4 & 5 & \ldots & 1 & 2 & 3 & 4 \\
7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & \\
\end{array}
\]

How big?

n pieces of data \rightarrow 2^n size array
Uses of trees in D.S:
- Binary Search Trees
  ▫ Balanced
Data Structure:

Priority Queue: supports the following operations

- `insert(e)`: adds element e to the data structure
- `removeMax()`: removes maximum element
- `getMax()`: returns maximum element

How to build?
Why?

Good if you need limited sorting.

Ex:

How to implement?

Many options: Array [Vector], Linked List, tree-based
Vector implementation:

if unsorted vector:

get or find Max: linear search $O(n)$

insert: $O(1)$ amortized (just push back)

Sorted vector:

got or find Max: $O(1)$ (look at end!)

insert: find (binary search) + then insert $O(k)$
Heaps

A binary tree where:

- For every node v (other than root), the key stored at v is ≤ key stored at v’s parent.

- The tree is complete if levels 0 to h-1 are full, and level h is filled in left to right order.
Max Heap

of integers

can return root.
Insert

insert (2)
insert (52) ← problem
insert (7)

insert at bottom
while (new one ≥ parent)
swap new one with parent
move up
Running times

How many comparisons/ swaps?

Each insert or remove Max travels + swaps along entire root to leaf path.

\[ d \text{ nodes} \geq \sum_{i=0}^{d} 2^i = n \]

\[ 2^0 + 2^1 + 2^2 + \ldots + 2^d = n \]

\[ = 2^{d+1} - 1 \]
\[ 2^{d+1} - 1 = n \]
\[ \log_2 (2^{d+1}) - \log_2 (n-1) \]
\[ \log_2 2^{d+1} = (d+1) \log_2 2 = d+1 \]

So:
\[ d = \log_2 (n-1) + 1 \]
\[ 2 = O(\log_2 n) \]
Code for this class

- Array-Based. Why?

These are nearly complete trees, so saves space.
To do: Code