Software 2 (Theory)

Lecture 4: Relational Data Structures

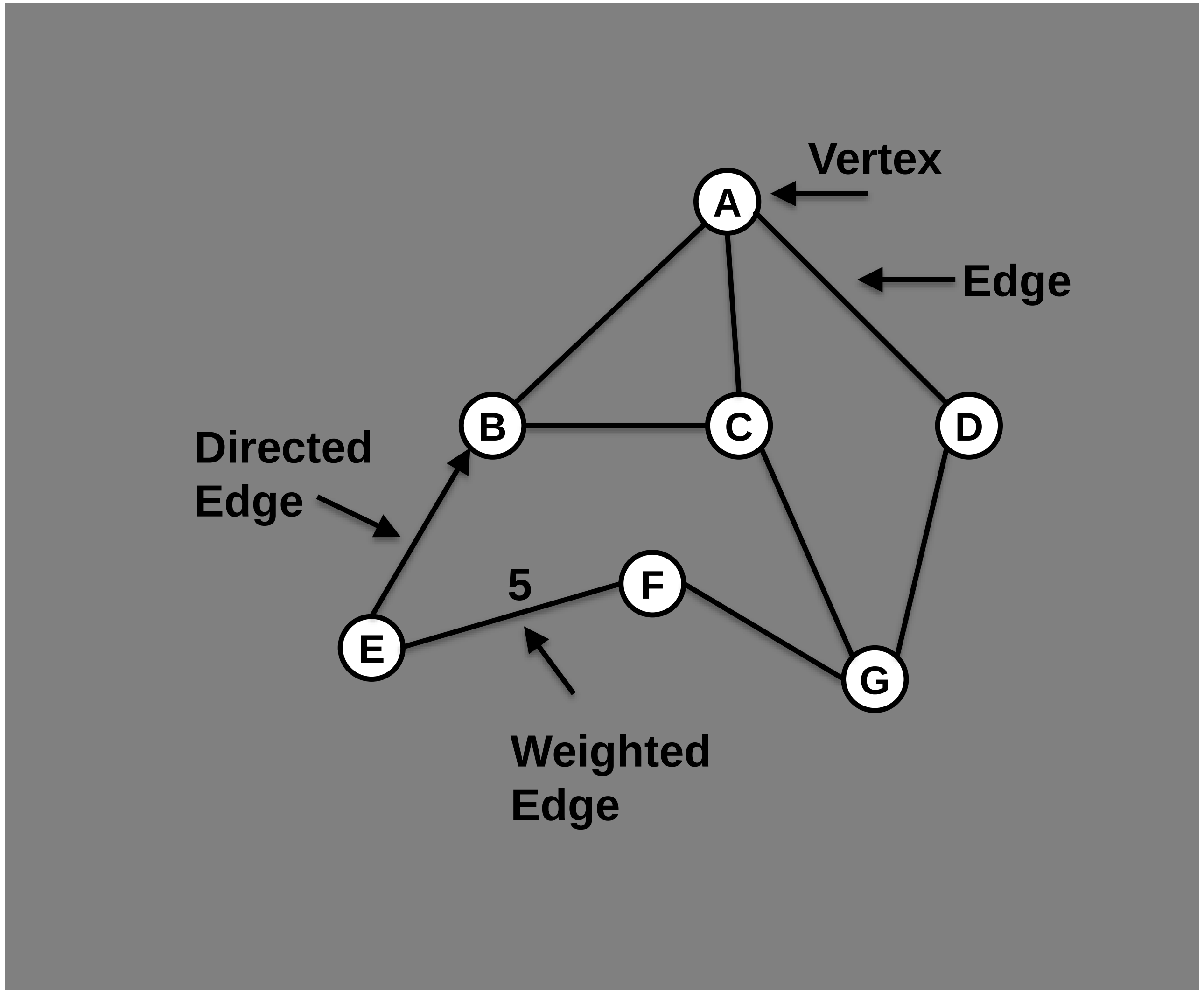
THE UNIVERSITY of York

Today's Topics

- Explore some more abstract data types which use links to organise data
- Introduce trees and graphs
- Study the use of trees and their application in other algorithms
- Explore some more algorithms for graphs

Graph Refresher

- Graphs and their representation were introduced in THE1
- They have vertices (nodes) and edges which connect vertices
- Edges may be undirected (two-way) or directed (one way)
- The edges may be unweighted (no information on them) or weighted (numeric weight attached to edge)
- B is adjacent to A if there is an edge connecting A to B

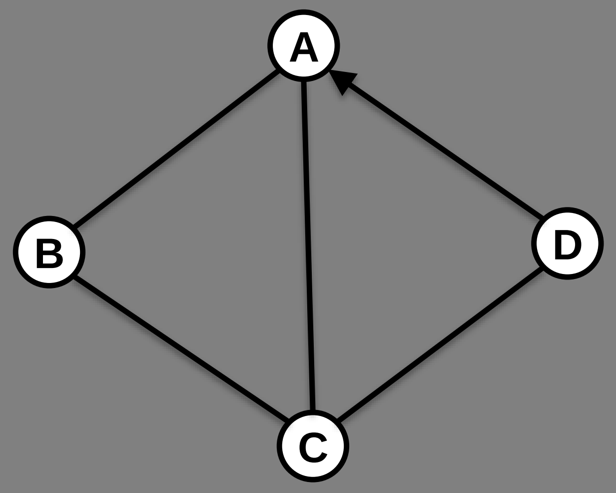


Adjacency matrix representation

- The adjacency matrix is a matrix with one row and one column for each vertex

$$\begin{array}{c c} & \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

- Insert 1 where there is an edge.
- We can use the weights instead of 1 for a weighted graph.
- The zeros may be replaced by ∞ dependent on the application.



Adjacency/Edge list representation

- The *adjacency list* is a list where each entry is a list of adjacent vertices:

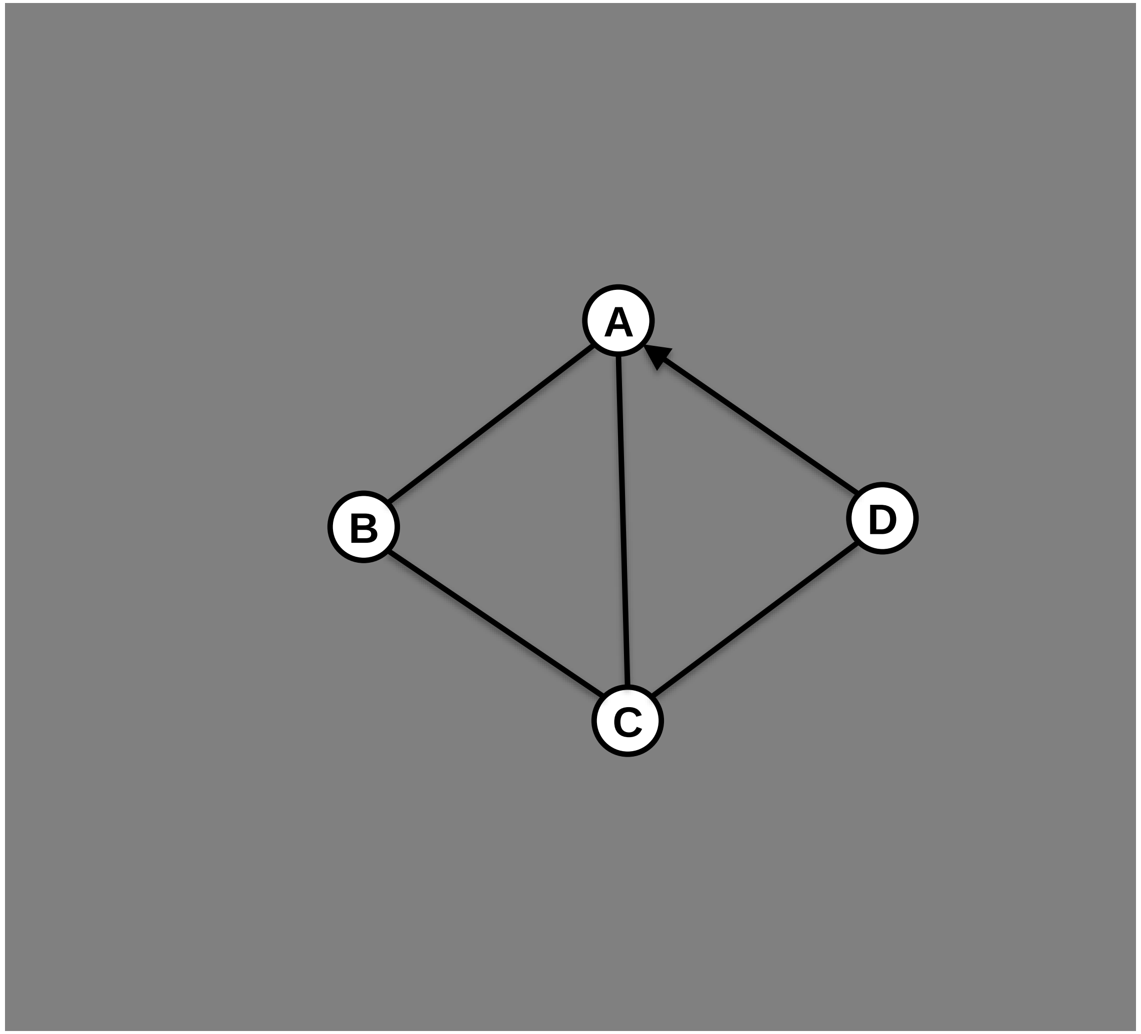
A: (B,C)

B: (A,C)

C: (A,B,D)

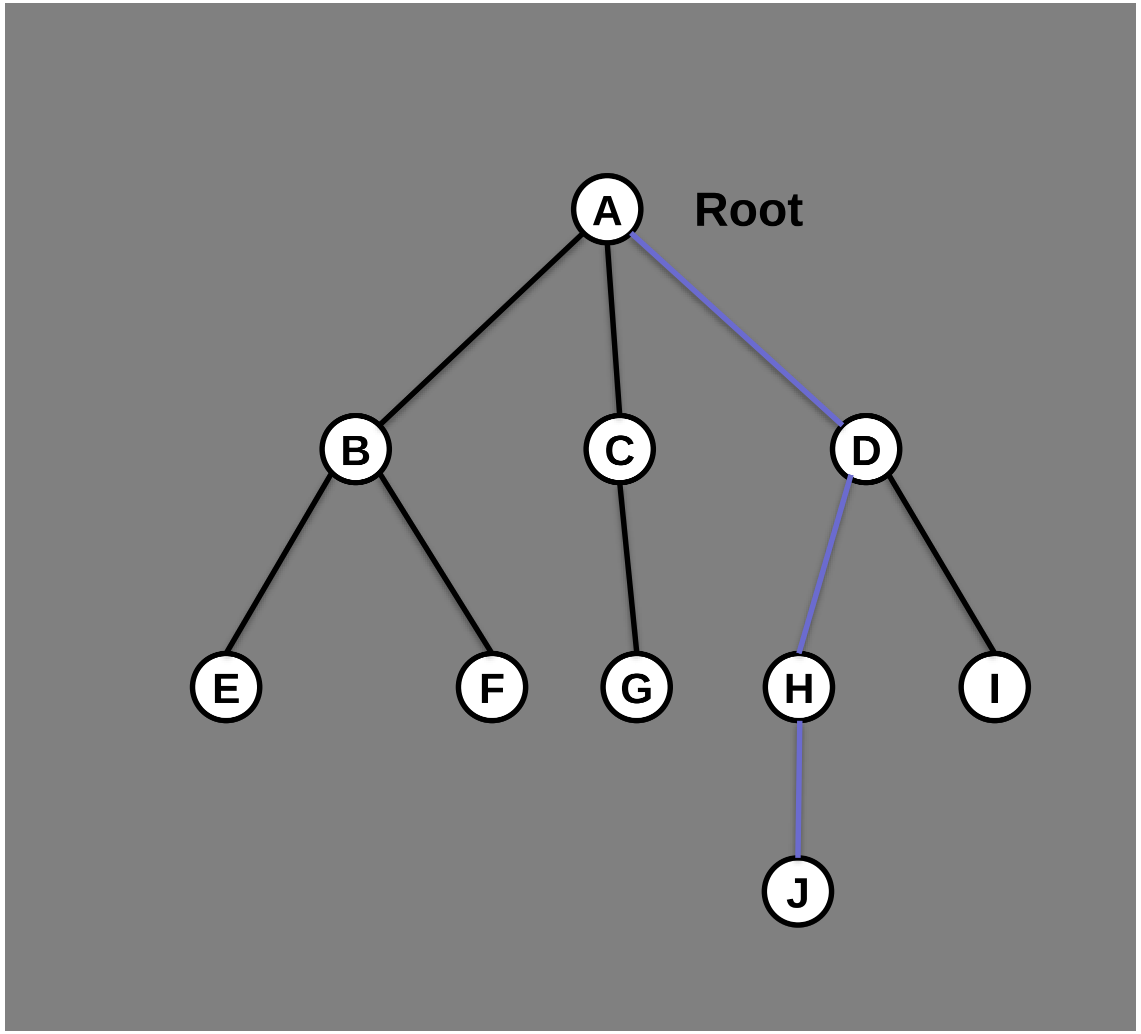
D: (A,C)

- The edge list is a list of pairs of nodes indicating where the edges are.
- [(A,B),(A,C),(B,A),(B,C),(C,A),(C,B),(C,D),(D,A),(D,C)]
- If the graph is known to be undirected, we need only list a pair once, i.e. we would list just (A,B) rather than (A,B),(B,A)
- More compact than adjacency matrix.



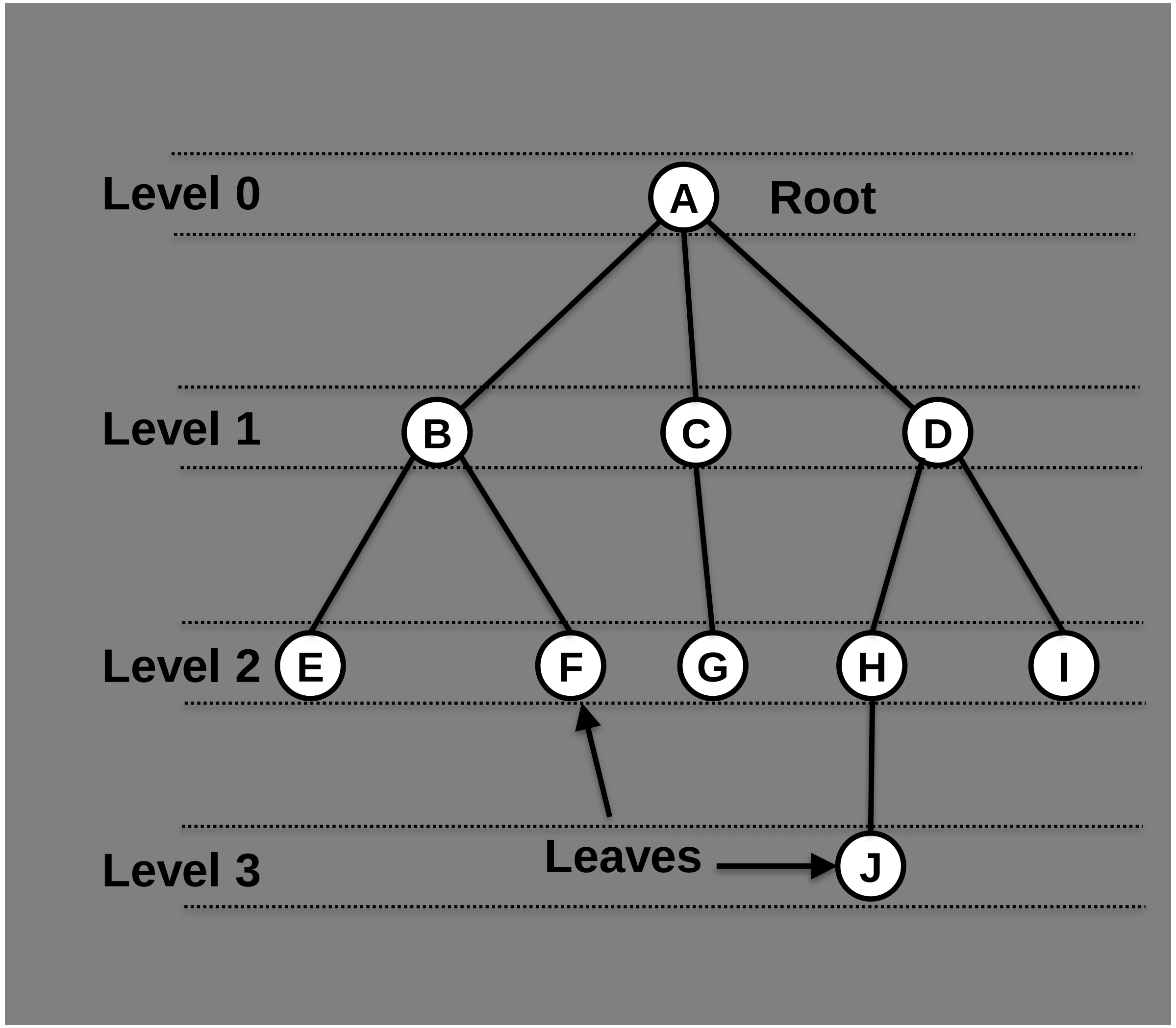
What is a Tree?

- A tree is a type of graph
  - It has vertices (nodes) and edges which connect vertices
- A tree is a graph which has no loops
- A rooted tree is a tree which has a specially designated vertex called the root
  - All our trees here are rooted and we will just call them trees
- There is a unique shortest path from every vertex to the root



Tree – Characteristics

- Two connected vertices are called a **Parent** and **Child**. The parent is closer to the root, the child further away. E is the child of B, B is the parent of E.
- The root has no parent.
- A leaf vertex is one with no children. E,F,G,I,J are leaves (and hence degree 1).
- The height of a vertex is the length of the path back to the root. J has height 3

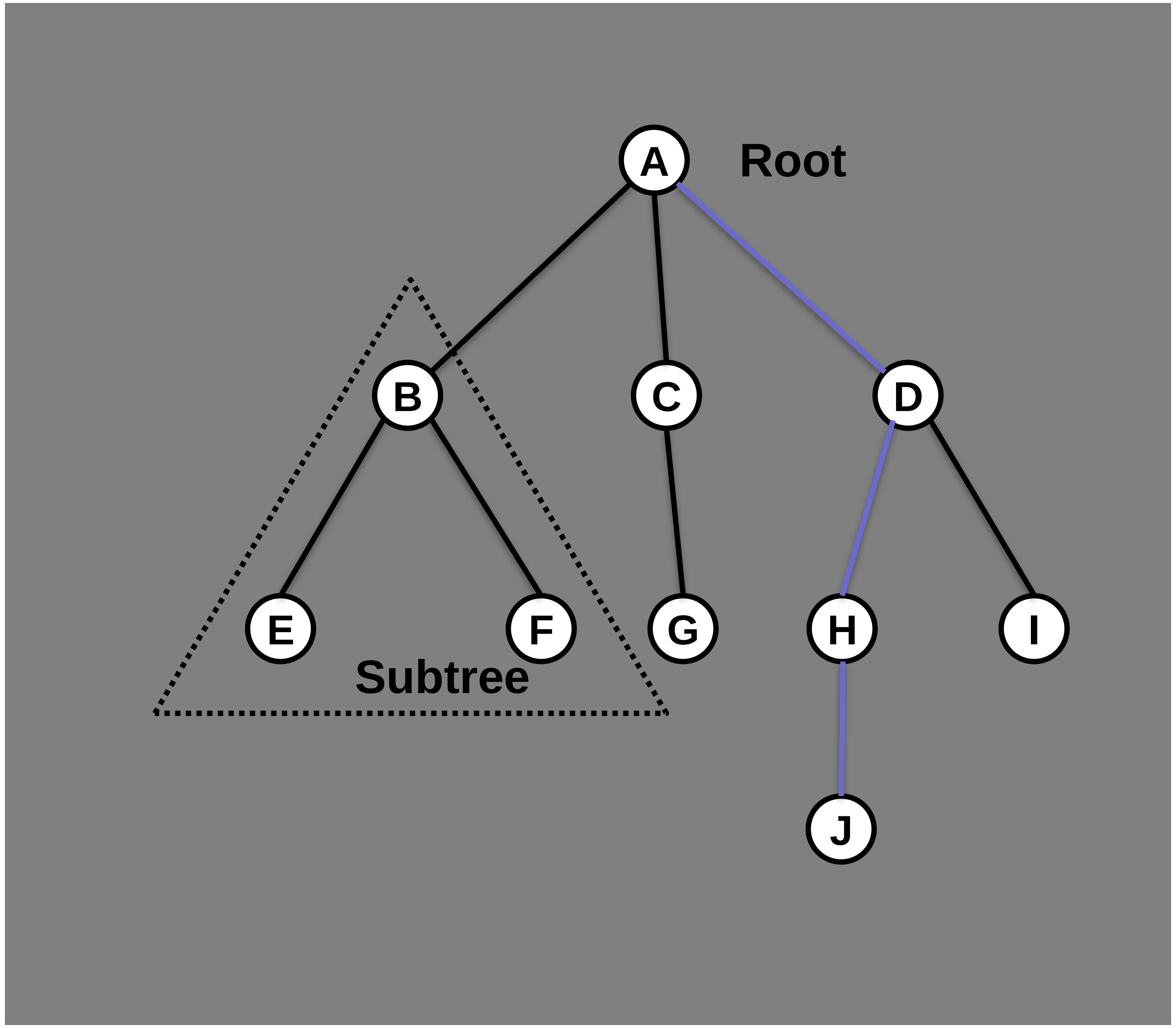


Tree – Characteristics (2)

- Two vertices which share a parent are called siblings e.g. H and I

- If two vertices X and Y are on the same path to the root, then X is an ancestor of Y (Y is a descendant of X) if X is closer to the root.
- D is an ancestor of J.

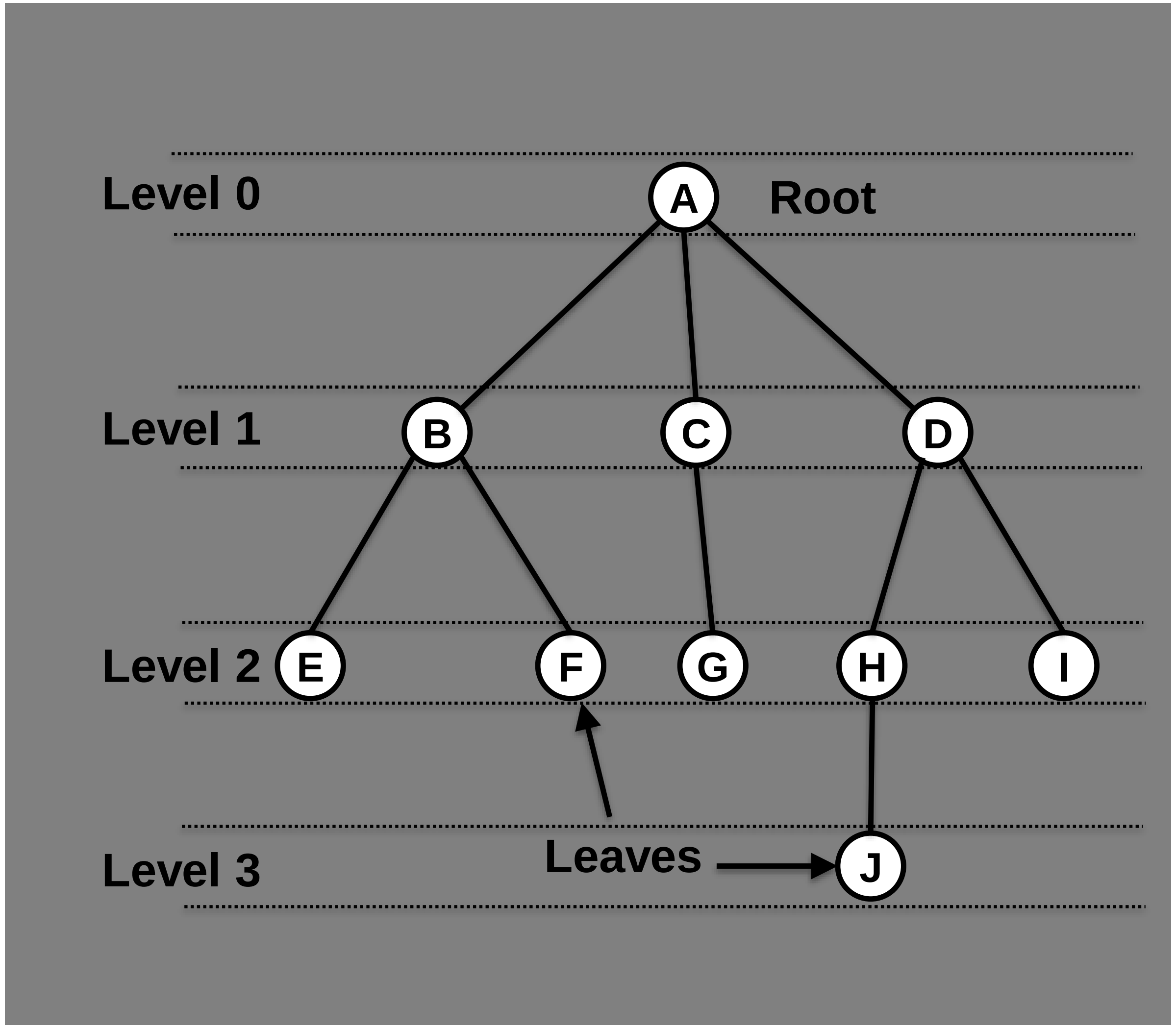
- A subtree is any collection of vertices and edges from the tree which remains a tree.



Vertex order relation

- The structure of a tree allows us to define a vertex order using the height.
- A≤B because height(A)≤height(B)
- This is only a **partial order;** it does not allow comparisons of things at the same level.
- To get a total order (and sort the vertices) we must arbitrarily define the order of the children of a vertex (e.g. left to right).

- The partial order means it is non-linear



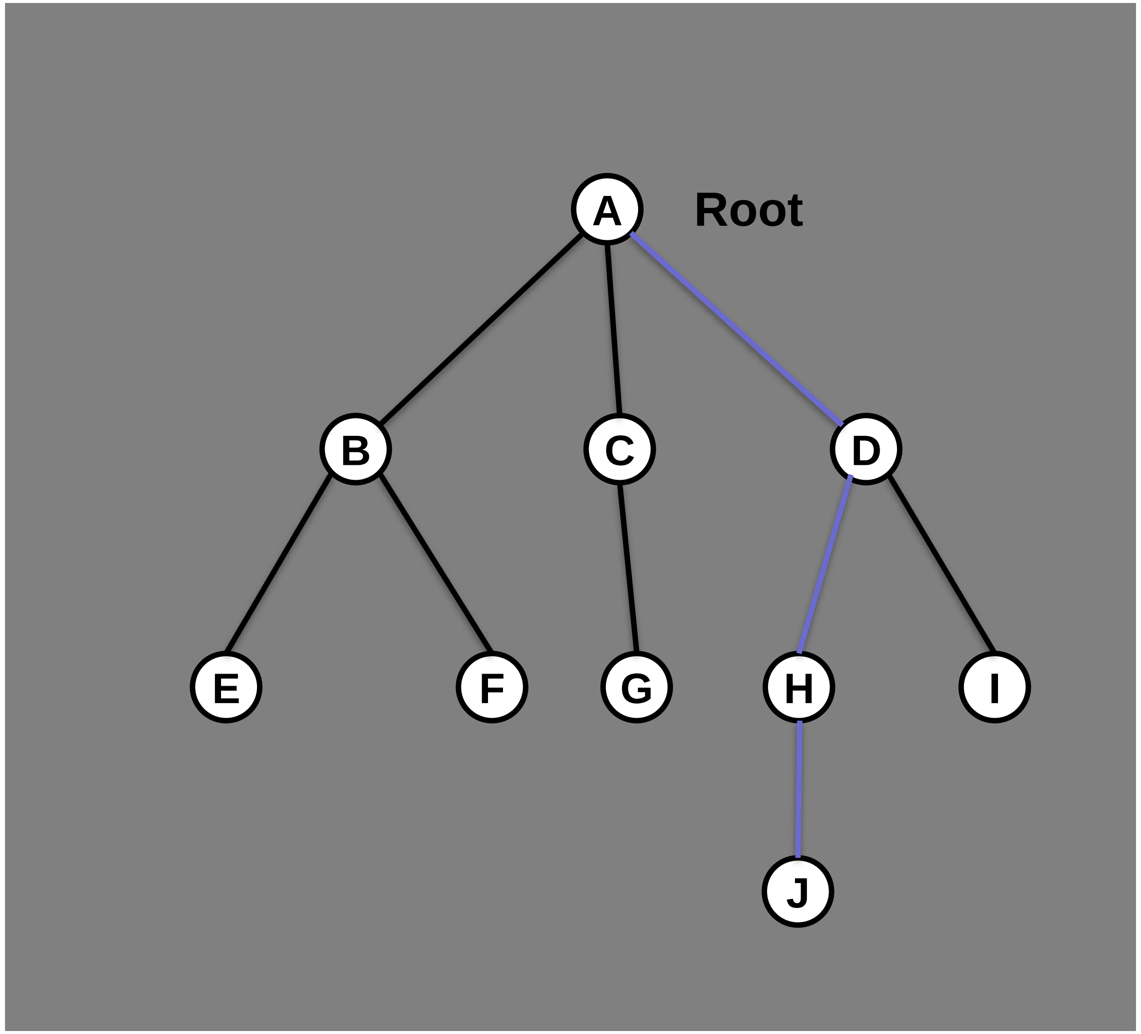
Number of edges

- A tree with n vertices has n-1 edges.

Proof:

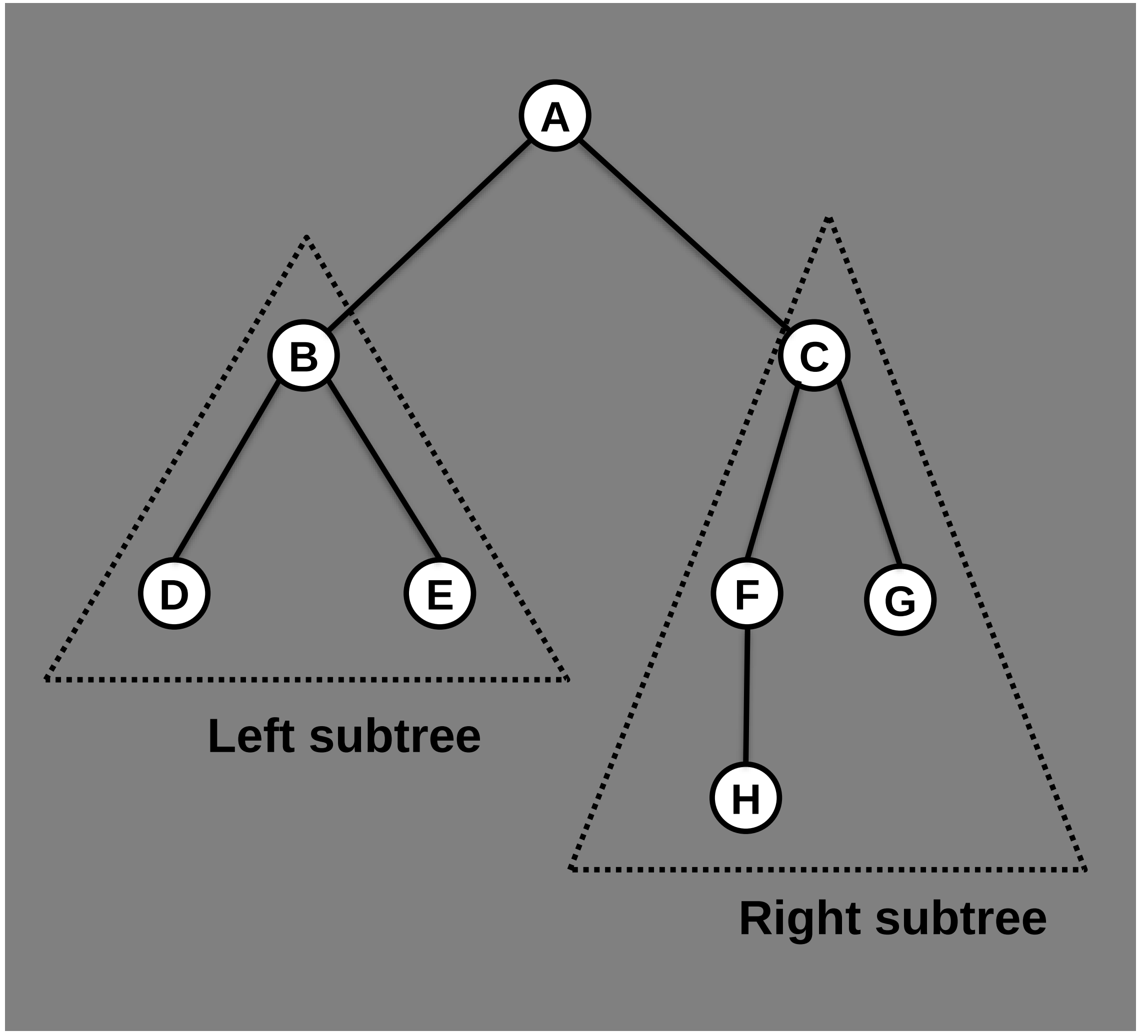
- Every edge joins a parent to one child. Take the edge joining parent to child and associate it with the child.
- Each vertex, except the root, has exactly one edge associated. Hence there are n-1 edges.

- Trees are **sparse**. They have an average of less than 1 edge per vertex.



Binary Tree

- A binary tree is a tree where each vertex has at most 2 children.
- The 2 children are referred to as the left child and right child.
- B and all its descendants is called the left subtree.
- Similarly, the tree starting at C is the right subtree.



Tree Representation – Adjacency List

- You have already looked at the representation of graphs in THE1
  - Adjacency matrix and edge list
- Trees are sparse. Since there are n vertices and n-1 edges, the data is of size $O(n)$.
- The adjacency matrix is of size n^2 and so highly redundant and not often used for trees.
- The adjacency list is a map from the vertices to the children of that vertex.

Adjacency List

- Example:

A (B, C, D)

B (E, F)

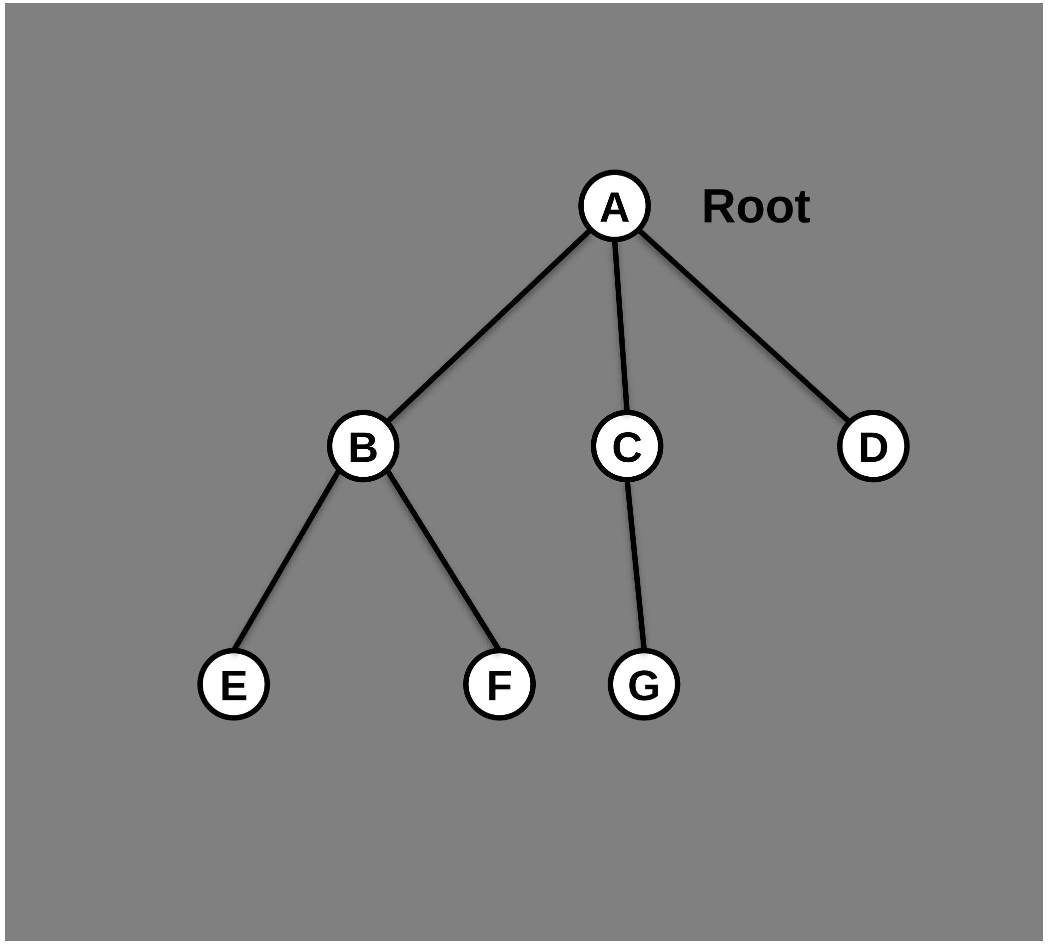
C (G)

D ()

E ()

F ()

G ()



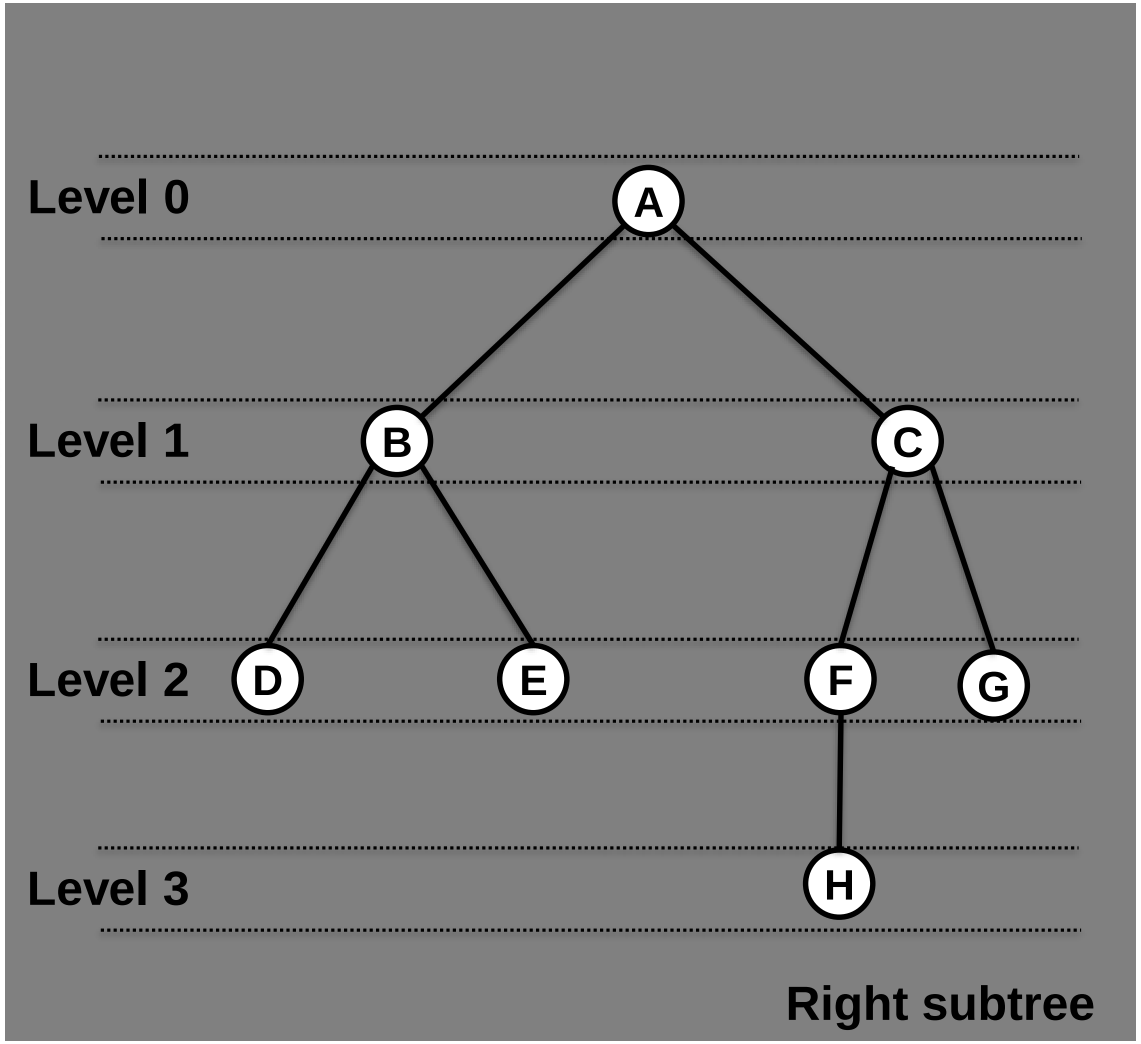
- List of lists.
- As with a doubly-linked list, can also link each vertex to its parent.

Binary Tree Representation

- For a binary tree we can have a compact array representation. At each level k, there are at most 2^k vertices

A	B	C	D	E	F	G	Ø	Ø	Ø	Ø	H	Ø	Ø	Ø
0	1		2				3							

- Requires array size $2^{h+1} - 1$ where h is the height of the tree.



Heaps and Priority Queues

- In some problems we have data where the ordering $\leq$ represents a priority. The highest priority item is the min with regards to (w.r.t.) $\leq$.
- We would like to represent this data with an efficient data structure that allows us to recover the minimum item (i.e. highest priority) and insert new data.
- This is the purpose of a **heap**

Heap Data Structure

Organization

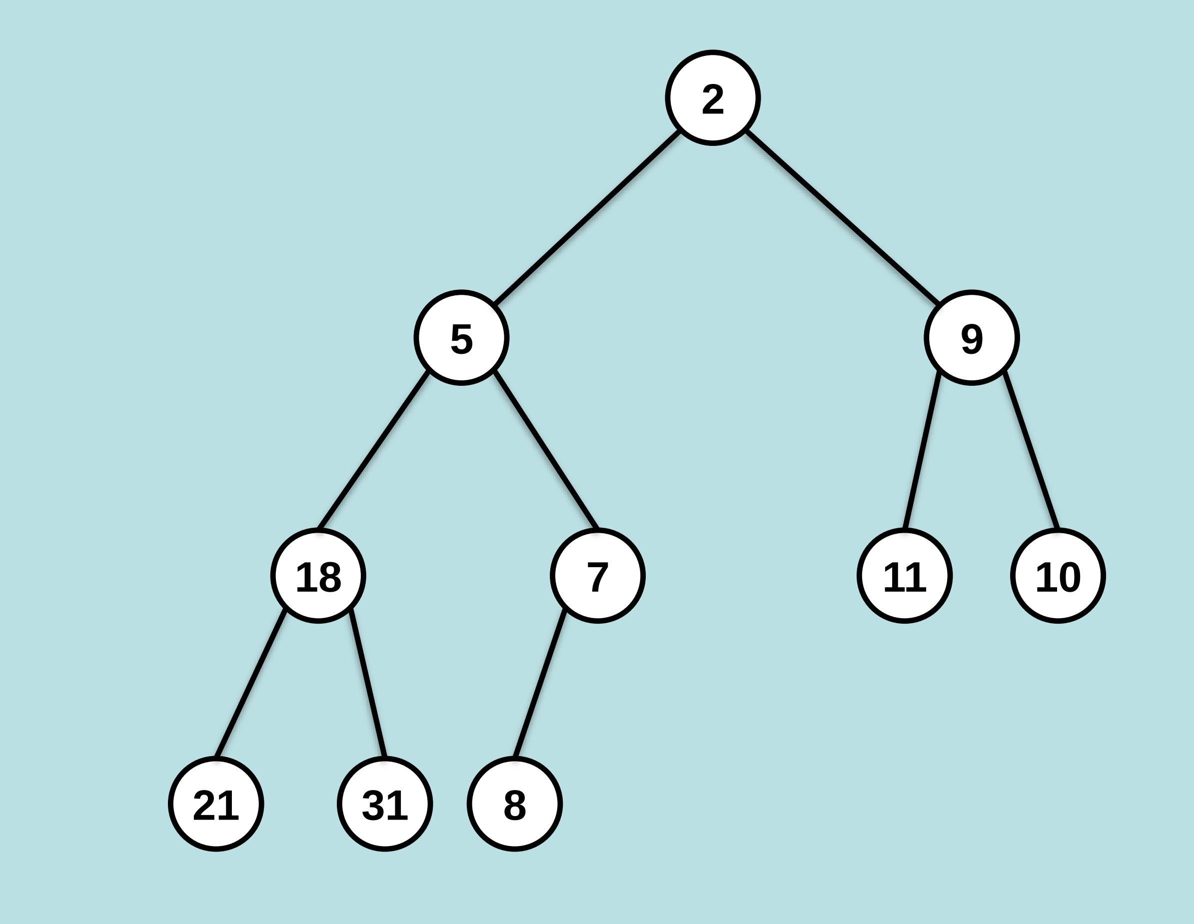
Binary Tree

Common operations

`Insert(v)`	Insert element v
`ExtractMin()`	Remove and return the minimum element

Heaps as Binary Tree

- We can represent a heap efficiently by a special binary tree.
- The tree has the following structure
- It is **complete** – every level is full except for (possibly) the last one. Items on the last row are left-justified.
- Each vertex represents one item of data
- The value on a vertex is no more than that of any of its descendants
- Therefore, any subtree is also a heap

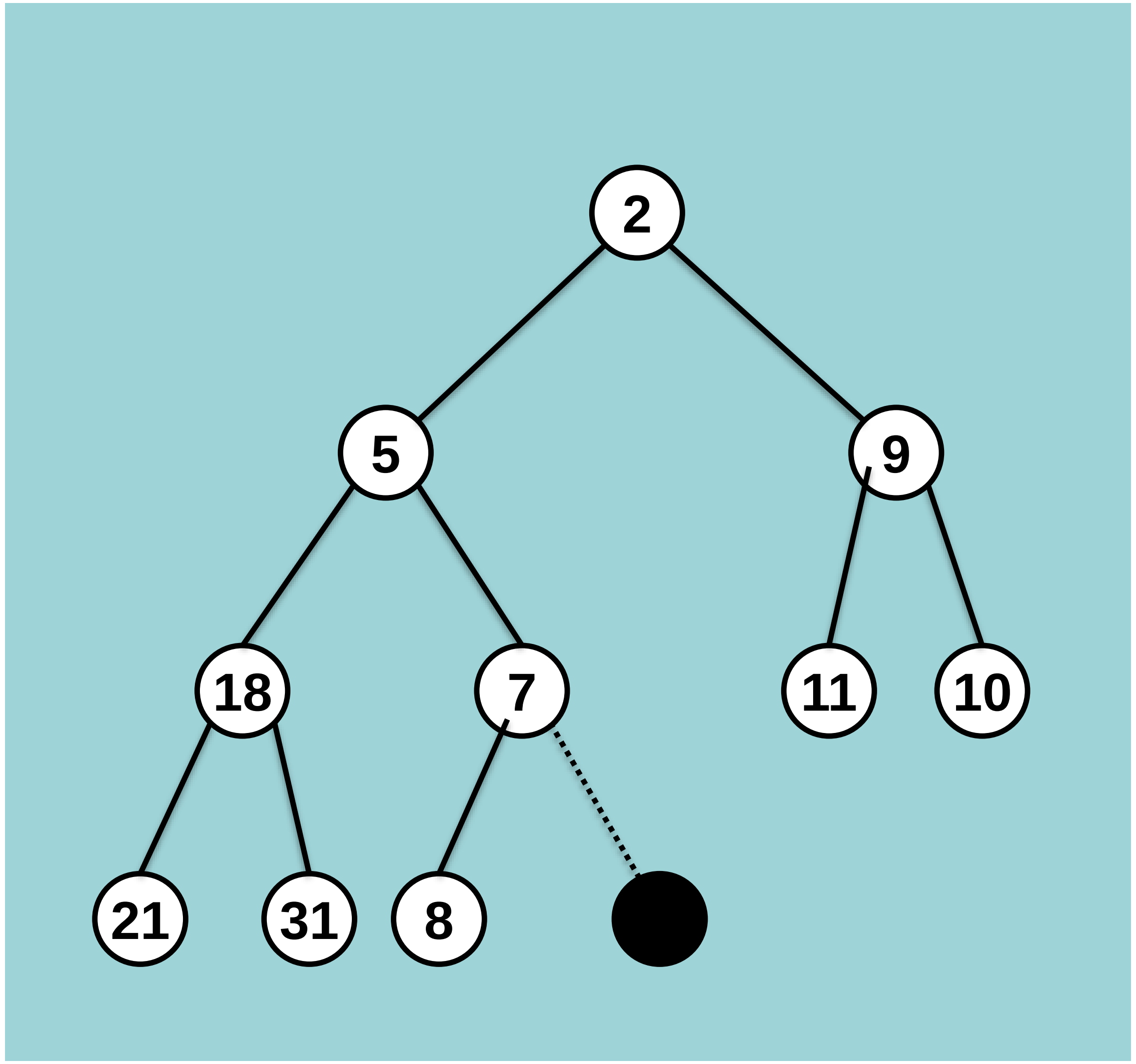


Properties of Binary Heaps

Since the tree is binary and complete, we can represent it efficiently with an array with $2^{h+1} - 1$, with $h = O(\log_2 n)$ (the number of items)

The root is the smallest element (highest priority). We can find this value in O(1) time.

A new item is inserted on the next position in the final row. Using an array, this can be done in O(1) time.



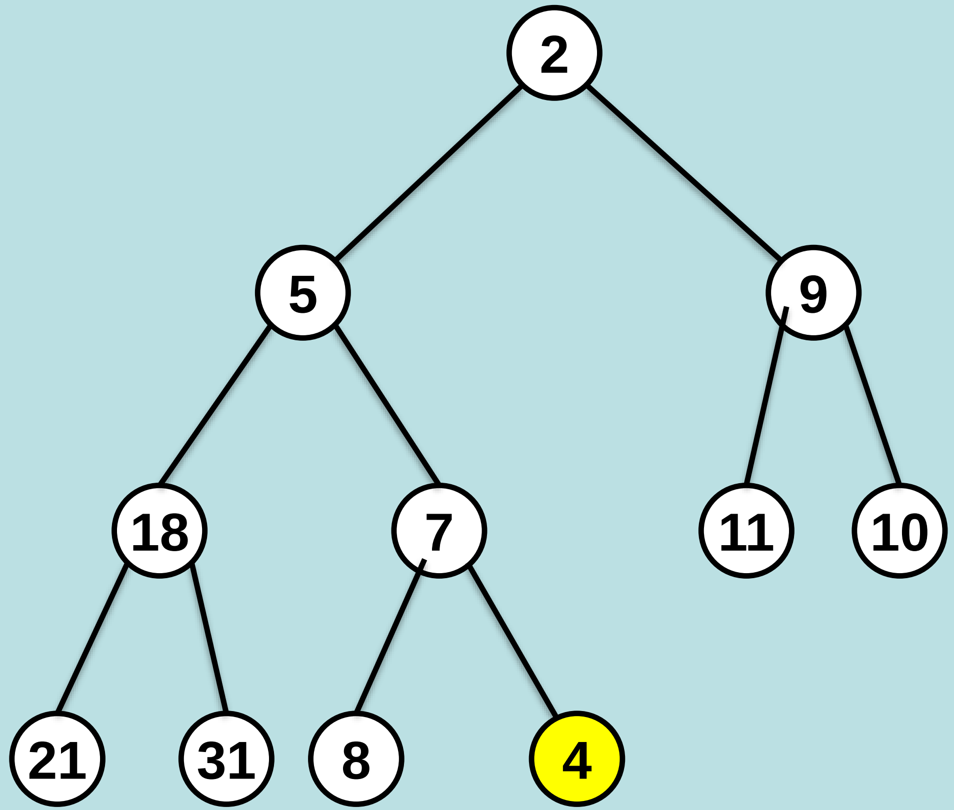
Insertion(1)

When we insert a new item, the tree loses its heap property, and we must restore it.

Compare the new item with its parent.

If it is less, swap them.

$4 < 7$, so swap them.

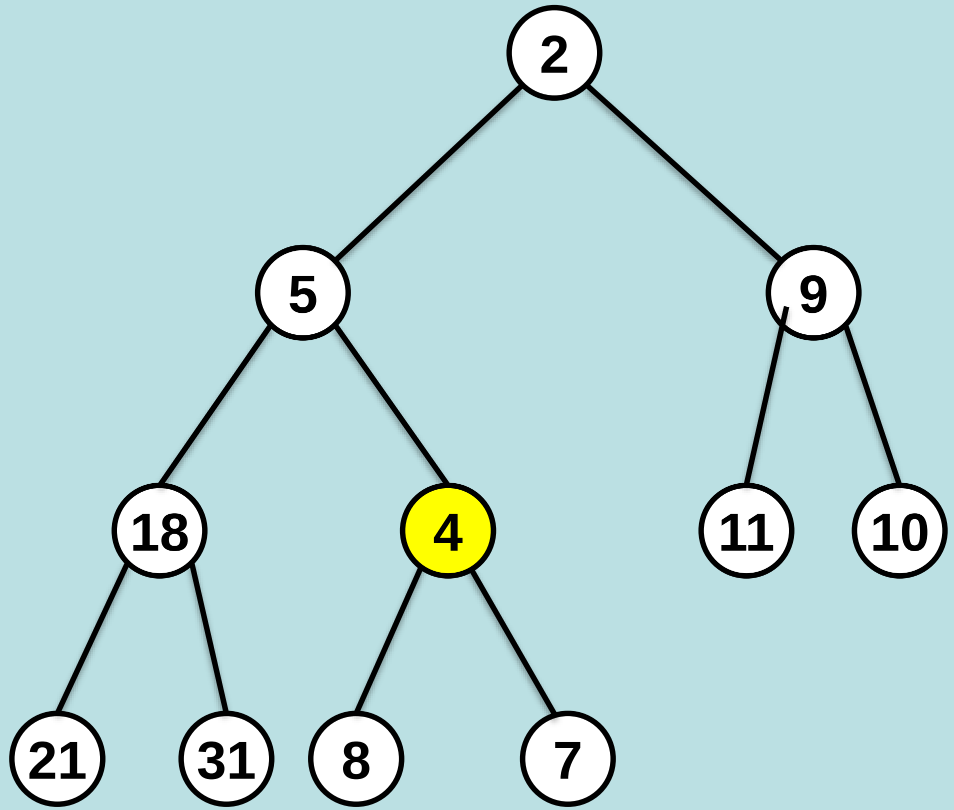


Insertion(3)

When we insert a new item, the tree loses its heap property, and we must restore it.

Continue comparing the new item with its parent

$4<5$, so swap them.

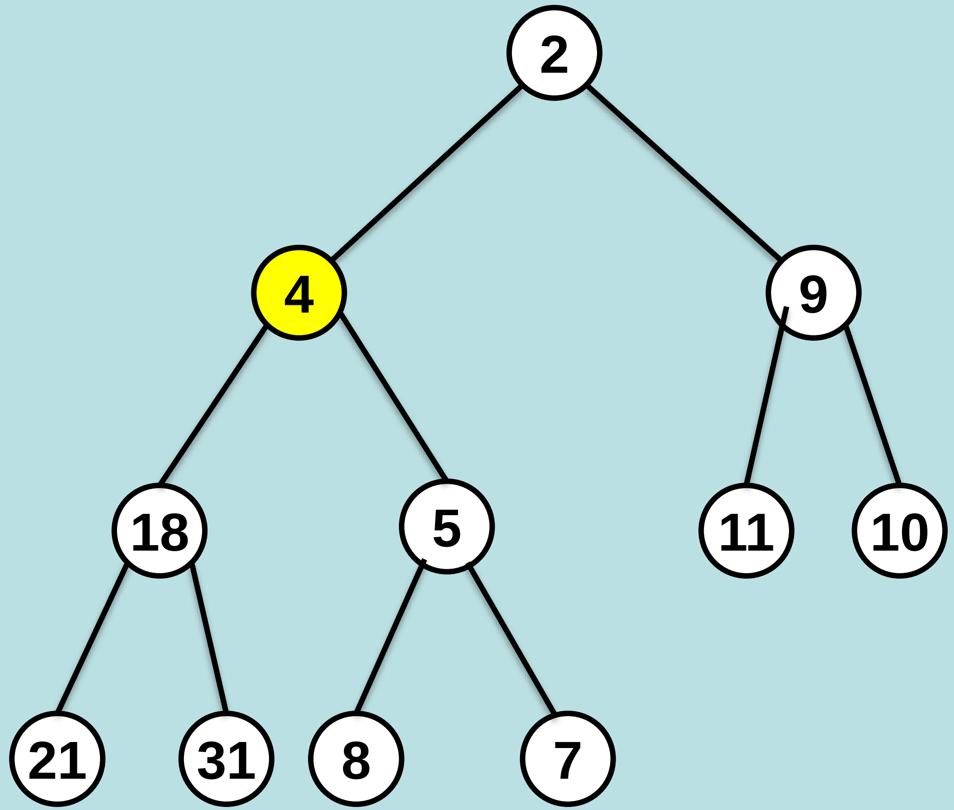


Insertion(3)

When we insert a new item, the tree loses its heap property, and we must restore it.

Continue comparing the new item with its parent

$4<5$, so swap them.



Insertion(4)

When we insert a new item, the tree loses its heap property, and we must restore it.

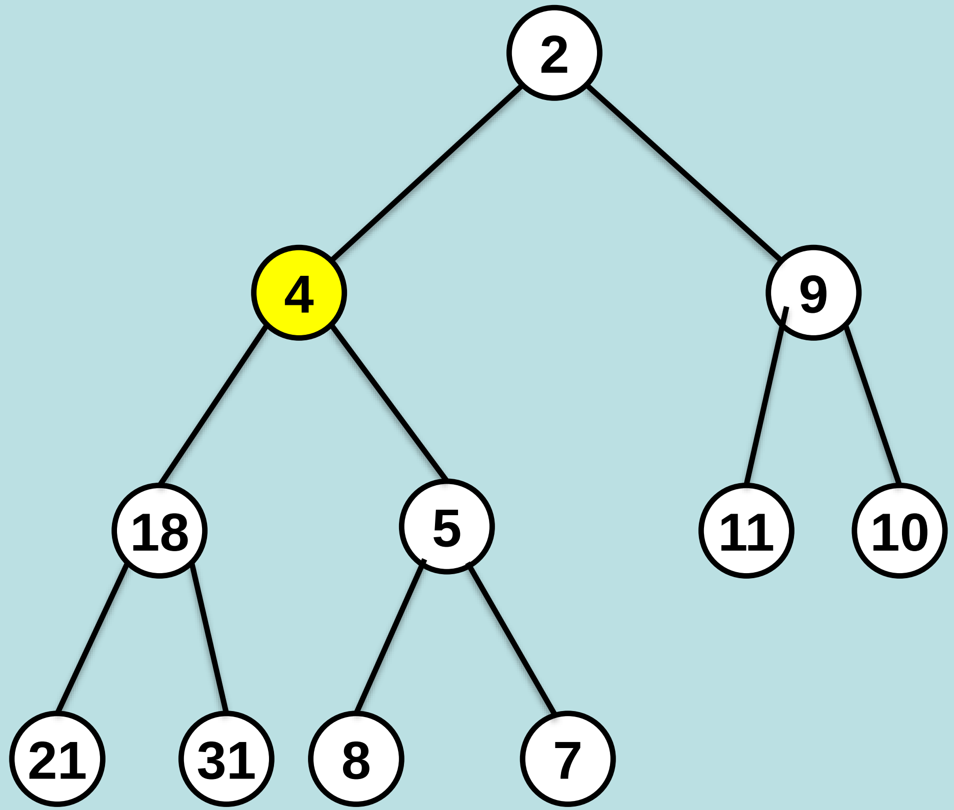
$4 > 2$, so the heap property is restored.

Adding the item initially takes O(1)

Each swap takes O(1)

The maximum number of swaps is the tree height, so O(log *n*)

Insertion takes O(log *n*)

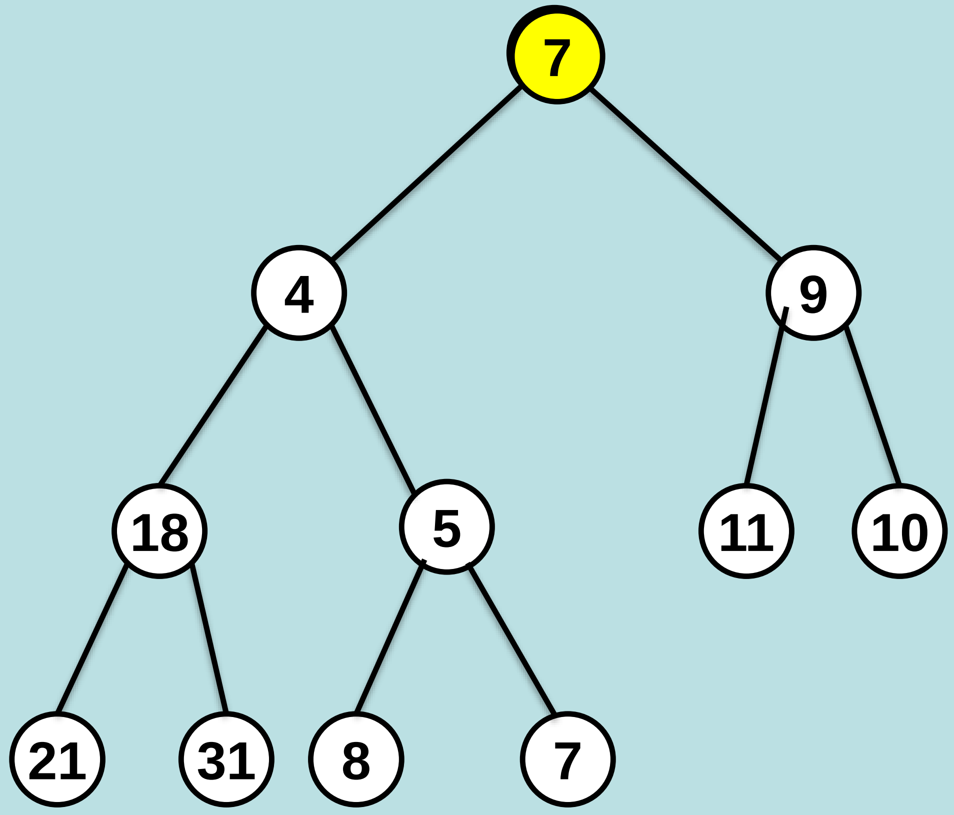


Extract Minimum(1)

The minimum item is at the top of the tree. Access is O(1). To remove, we replace it with the last item in the tree.

Again, the heap property must be restored, this time downwards.

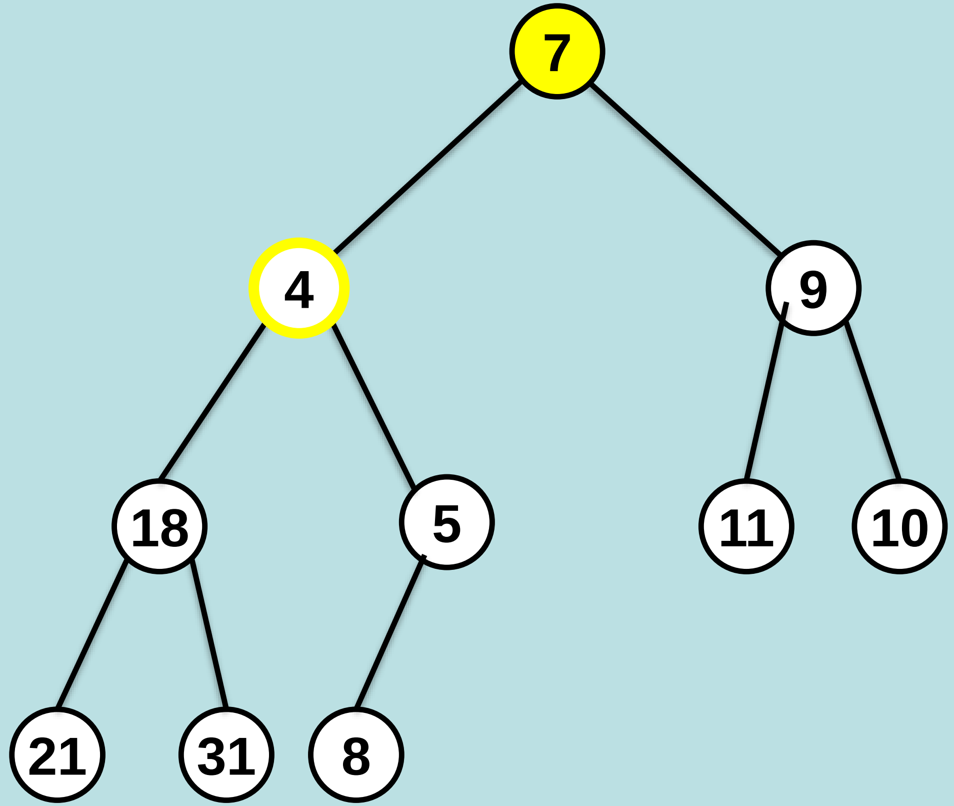
Compare with the smallest child and swap if necessary.



Extract Minimum(2)

4 is the smallest child.

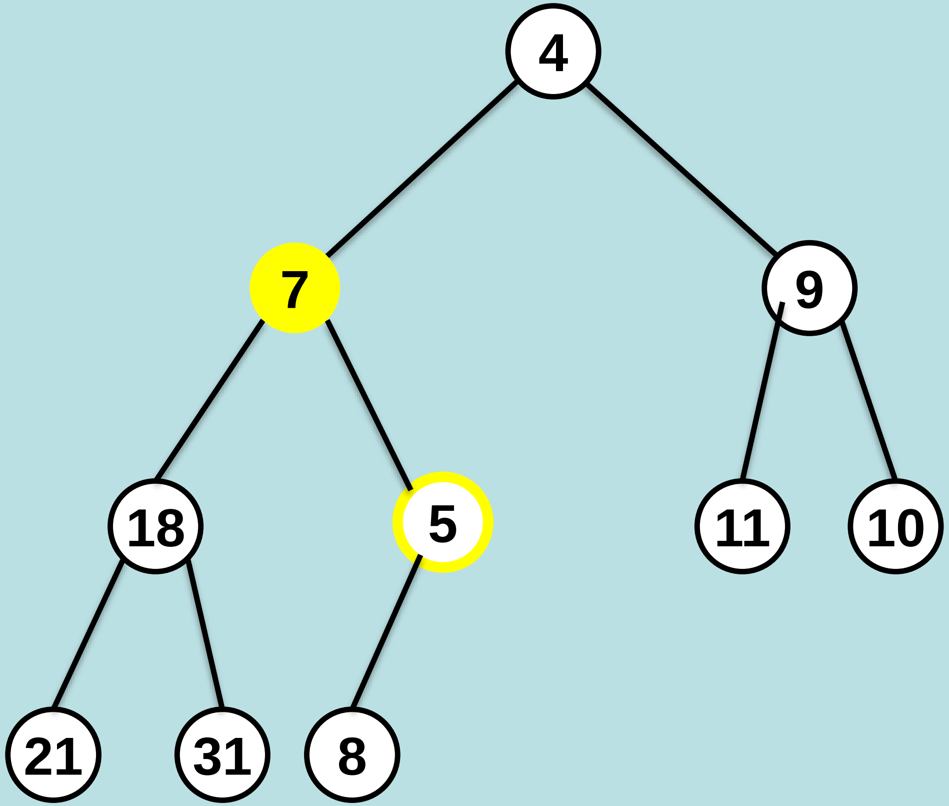
4<7, so swap



Extract Minimum(3)

5 is the smallest child

5<7, so swap



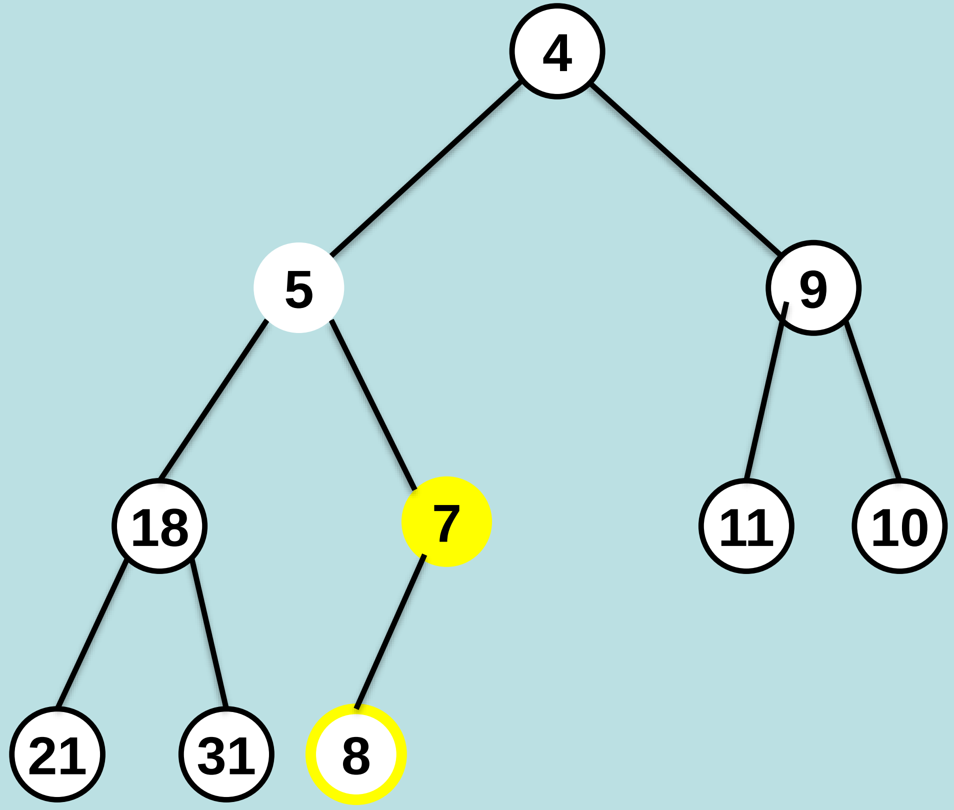
Extract Minimum(3)

8 is the smallest child

$8 \geq 7$, so procedure complete.

The maximum number of swaps is the tree height, so the ExtractMin operation is
$O(\log n)$

The heap data structure can represent a priority queue with both operations $O(\log n)$



Summary

- Understand:
    - Trees and their terminology
    - Two ways of representing trees
    - Binary trees
    - Heaps using binary trees
- Read
    - Revise SOF1 week 7
    - Skiena, Sections 3.5, 4.3.1-4.3.3
- Next
    - More on sorting

Menti questions

- The powerpoints on the VLE don't have any symbols (like lambda or pi) so they're very difficult to make notes from during the lecture. I can't type the symbols and need to screenshot the slides
  - I will post PDF from now on.
- Is it only rooted trees for which the statement 'trees with n vertices have n-1 edges?
  - It’s true for trees in general.
- Why are there empty spaces after the G vertex in the binary tree representation ?
  - Because nodes D and E have no children.
- What does w.r.t mean in the heaps and priority queues slide?
  - with regards to
- Why is it specified on slide 17 that h = O(log base 2 n). When talking about big O notation, does log not refer to the binary logarithm? Or does it still mean log base 10?
  - In general the log can be any base. But it is mostly base 2 in this module.