Priority Queues
(Today: Binary Min Heaps)

Chapter 6 in Weiss

Simplifying Recurrences

Given a recursive equation for the running time,
can sometimes simplify it for analysis.

- For an upper-bound analysis, can optionally simplify to something larger, e.g.

  $T(n) = T(\text{floor}(n/2)) + 1$ to $T(n) \leq T(n/2) + 1$

- For a lower-bound analysis, can optionally simplify to something smaller, e.g.

  $T(n) = 2T(n/2 + 5) + 1$ to $T(n) \geq 2T(n/2) + 1$

Priority Queue ADT

1. **PQueue <u>data</u>** : collection of data with priority

2. **PQueue <u>operations</u>**
   - insert
   - deleteMin

   (also: create, destroy, is_empty)

3. **PQueue <u>property</u>**: for two elements in the queue, *x* and *y*, if *x* has a **<u>lower</u>** priority value than *y*, *x* will be deleted before *y*

Applications of the Priority Q

- Select print jobs in order of decreasing length
- Forward packets on network routers in order of urgency
- Select most frequent symbols for compression
- Sort numbers, picking minimum first

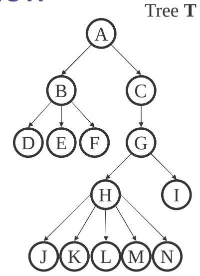
- **Anything *greedy***

Implementations of Priority Queue ADT

	insert	deleteMin
Unsorted list (Array)		
Unsorted list (Linked-List)		
Sorted list (Array)		
Sorted list (Linked-List)		
Binary Search Tree (BST)		

Tree Review

Tree **T**



root(**T**):
leaves(**T**):
children(B):
parent(H):
siblings(E):
ancestors(F):
descendents(G):
subtree(C):

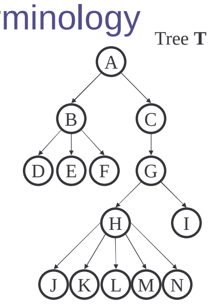
More Tree Terminology

depth(**T**):

height(G):

degree(B):

branching factor(**T**):

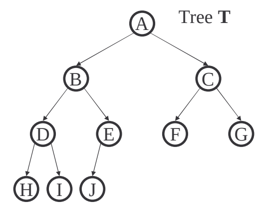


Some More Tree Terminology

T is *binary* if …

T is *n-ary* if …

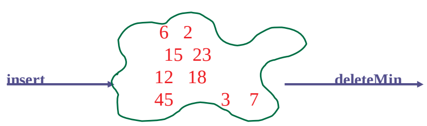
T is *complete* if …



How deep is a complete tree with *n* nodes?

Priority Queue ADT

- Processor scheduling example
- Printer queues ???
- Subtask of other algorithms
- **operations**: insert, deleteMin



Binary Heap Properties

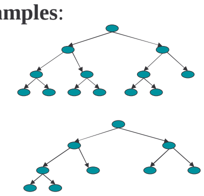
1. Structure Property
2. Ordering Property

Heap <u>**Structure**</u> Property

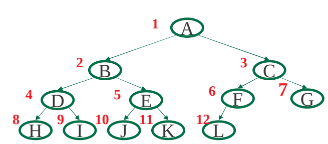
- A binary heap is a <u>***complete***</u> binary tree.

<u>**Complete binary tree**</u> – binary tree that is completely filled, with the possible exception of the bottom level, which is filled left to right.

Examples:



Representing Complete Binary Trees in an Array



From node **i**:

left child:
right child:
parent:

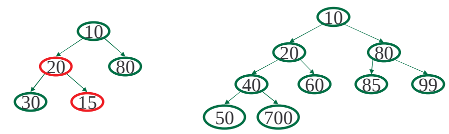
implicit (array) implementation:

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	
0	1	2	3	4	5	6	7	8	9	10	11	12	13

Why better than tree with pointers?

Heap **Order** Property

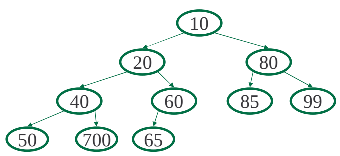
Heap order property: For every non-root node X, the value in the parent of X is less than (or equal to) the value in X.



not a heap

Heap Operations

- findMin:
- insert(val): percolate up.
- deleteMin: percolate down.

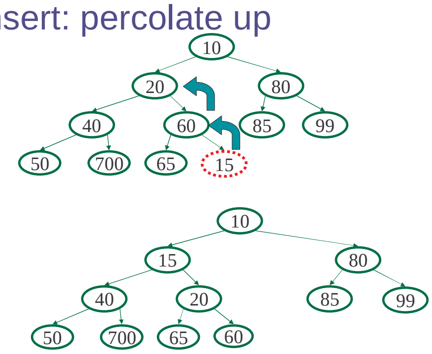


Heap – Insert(val)

Basic Idea:

1. Put val at “next” leaf position
2. Repeatedly exchange node with its parent if needed

Insert: percolate up



Insert pseudo/C++ Code (optimized)

```
void insert(Object o) {
  assert(!isFull());
  size++;
  newPos =
    percolateUp(size,o);
  Heap[newPos] = o;
}
```

```
int percolateUp(int hole,
                Object val) {
  while (hole > 1 &&
         val < Heap[hole/2])
    Heap[hole] = Heap[hole/2];
    hole /= 2;
  }
  return hole;
}
```

runtime:

(Java code in book)

Heap – Deletemin

Basic Idea:

1. Remove root (that is always the min!)
2. Put “last” leaf node at root
3. Find smallest child of node
4. Swap node with its smallest child if needed.
5. Repeat steps 3 & 4 until no swaps needed.

DeleteMin: percolate down