CSE 326: Data Structures

Priority Queues and Binary Heaps

Neva Cherniavsky
Summer 2006

Administration

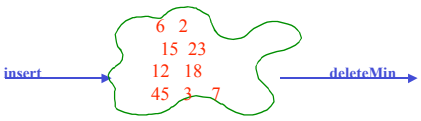
- Due tonight: Project 1
- Released today: Project 2, phase A
- Due Wednesday: Homework 1
- Released Wednesday: Homework 2
- Gary has office hours tomorrow

A New Problem…

- Application: Find the smallest (or highest priority) item quickly
- Operating system needs to schedule jobs according to priority
- Doctors in ER take patients according to severity of injuries

Priority Queue ADT

- Security line at the airport ???
- Printer queues ???
- operations: insert, deleteMin



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***

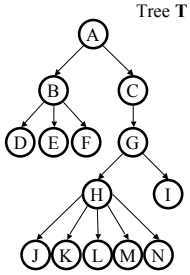
Implementations of P Queue ADT

	insert	deleteMin
Unsorted list (Array)	O(1)	O(N)
Unsorted list (Linked-List)	O(1)	O(N)
Sorted list (Array)	O(N)	O(1)
Sorted list (Linked-List)	O(N)	O(1)
Binary Search Tree (BST)	Don't worry	
Binary heap	O(log N)	O(log N)

Tree Review

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

Tree **T**



More Tree Terminology

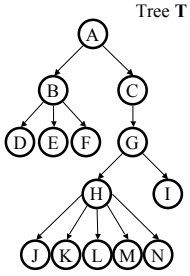
depth(**T**):

height(G):

degree(B):

branching factor(**T**):

Tree **T**



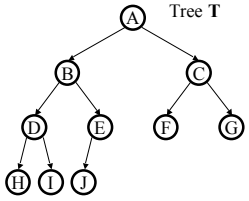
Some More Tree Terminology

T is *binary* if …

T is *n-ary* if …

T is *complete* if …

Tree **T**



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

Binary Heap Properties

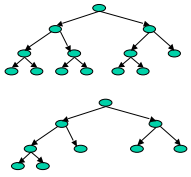
1. Structure Property
2. Ordering Property

Heap **Structure** Property

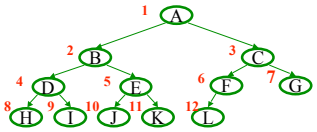
- A binary heap is a ***complete*** binary tree.

Complete binary tree – 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:

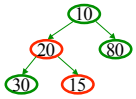
implicit (array) implementation:

	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

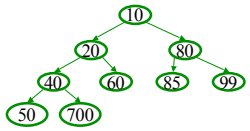
Why better than tree with pointers?

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.

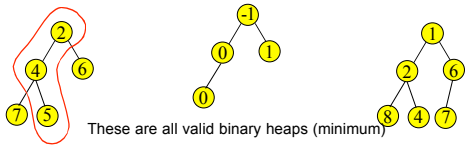


not a heap



Heap order property

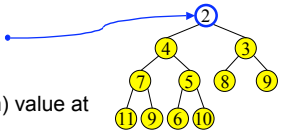
- A heap provides limited ordering information
- Each *path* is sorted, but the subtrees are not sorted relative to each other
  - A binary heap is NOT a binary search tree



These are all valid binary heaps (minimum)

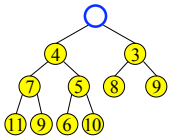
Heap Operations

- FindMin: Easy!
  - Return root value A[1]
  - Run time = ?
- DeleteMin:
  - Delete (and return) value at root node
- Insert(val):
  - Insert value into heap



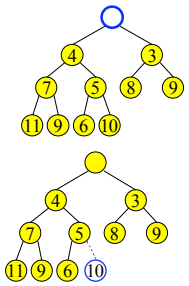
DeleteMin

- Delete (and return) value at root node



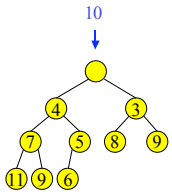
Maintain the Structure Property

- We now have a "Hole" at the root
  - Need to fill the hole with another value
- When we get done, the tree will have one less node and must still be complete

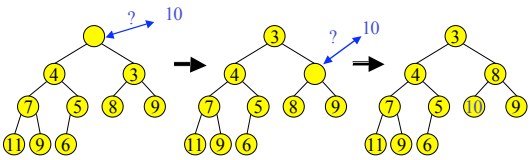


Maintain the Heap Property

- The last value has lost its node
  - we need to find a new place for it
- We can do a simple insertion sort operation to find the correct place for it in the tree



DeleteMin: Percolate Down



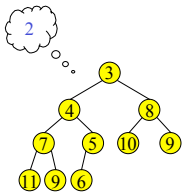
- Keep comparing with children A[2i] and A[2i + 1]
- Copy smaller child up and go down one level
- Done if both children are ≥ item or reached a leaf node
- What is the run time?

DeleteMin: Run Time Analysis

- Run time is O(depth of heap)
- A heap is a complete binary tree
- Depth of a complete binary tree of N nodes?
  - height = $\lceil \log_2(N) \rceil - 1$
- Run time of DeleteMin is O(log N)

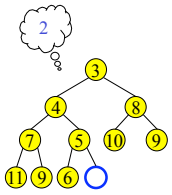
Insert

- Add a value to the tree
- Structure and heap order properties must still be correct when we are done



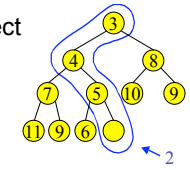
Maintain the Structure Property

- The only valid place for a new node in a complete tree is at the end of the array
- We need to decide on the correct value for the new node, and adjust the heap accordingly

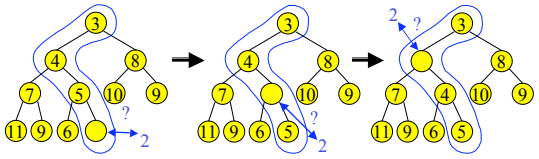


Maintain the Heap Property

- The new value goes where?
- We can do a simple insertion sort operation to find the correct place for it in the tree

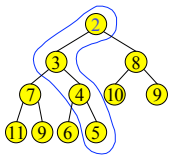


Insert: Percolate Up



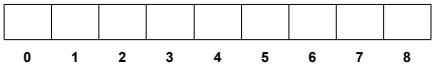
- Start at last node and keep comparing with parent A[i/2]
- If parent larger, copy parent down and go up one level
- Done if parent ≤ item or reached top node A[1]
- Run time?

Insert: Done



- Run time?

Insert: 16, 32, 4, 69, 105, 43, 2



Other Priority Queue Operations

- **decreaseKey**
  - given a pointer to an object in the queue, reduce its priority value

  Solution: change priority and _____
- **increaseKey**
  - given a pointer to an object in the queue, increase its priority value

  Solution: change priority and _____

Why do we need a *pointer*? Why not simply data value?

Other Heap Operations

decreaseKey(objPtr, amount): raise the priority of a object, percolate up

increaseKey(objPtr, amount): lower the priority of a object, percolate down

remove(objPtr): remove a object, move to top, them delete.
1) decreaseKey(**objPtr**, ∞)
2) deleteMin()

Worst case Running time for all of these:

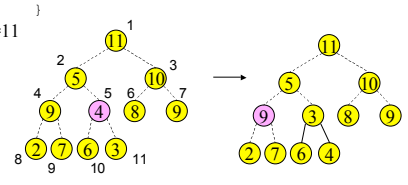
FindMax?

ExpandHeap – when heap fills, copy into new space.

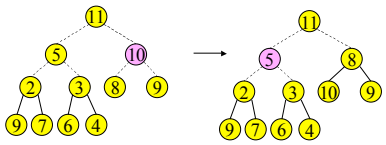
Build Heap

```
BuildHeap {
for i = N/2 to 1 by -1 PercDown(i,A[i])
}
```

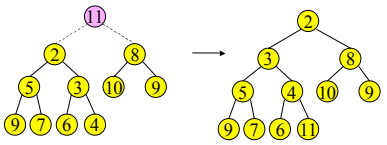
N=11



Build Heap



Build Heap



Analysis of Build Heap

- Assume $N = 2^K - 1$
  - Level 1: k -1 steps for 1 item
  - Level 2: k - 2 steps of 2 items
  - Level 3: k - 3 steps for 4 items
  - Level i : k - i steps for 2^{i-1} items

$$\text{Total Steps} = \sum_{i=1}^{k-1} (k-i)2^{i-1} = 2^k - k - 1$$
$$= O(N)$$

Binary Min Heaps (summary)

- **insert**: percolate up. O(log N) time.
- **deleteMin**: percolate down. O(log N) time.

- **Next time**: Even more priority queues??