CSE 326: Data Structures
Priority Queues

Hannah Tang and Brian Tjaden
Summer Quarter 2002

Today’s Outline

- Introduction to Trees
- Priority Queues
- Heaps

Chapter 6 in Weiss

Trees

- Family Trees
- Organization Charts
- Classification trees
  - what kind of flower is this?
  - is this mushroom poisonous?
- File directory structure
  - folders, subfolders in Windows
  - directories, subdirectories in UNIX or WWW

Tree Terminology

- root:
- leaf:
- child:
- parent:
- sibling:
- ancestor:
- descendant:
- subtree:

More Tree Terminology

- depth:
- height:
- branching factor:

One More Tree Terminology

- binary:
- n-ary:
- complete:
  - size???
Tree Calculations

Find the height of the tree...

Back to Queues

- Some applications
  - ordering CPU jobs
  - simulating events
  - picking the next search site
- Problems?
  - short jobs should go first
  - earliest (simulated time) events should go first
  - most promising sites should be searched first

Remember ADTs?

Priority Queue ADT

- Priority Queue operations
  - insert
  - deleteMin
  - is_empty
- Priority Queue property: for two elements in the queue, \( x \) and \( y \), if \( x \) has a lower priority value than \( y \), \( x \) will be deleted before \( y \)

Applications of the Priority Q

- Hold jobs for a printer in order of length
- Store packets on network routers in order of urgency
- Order web search results by the “rank” of the web page
- Sort numbers
- Anything greedy

Naïve Priority Q Data Structures

- Unsorted list:
  - insert:
  - deleteMin:
- Sorted list:
  - insert:
  - deleteMin:

Binary Search Tree Priority Q Data Structure (that’s a mouthful)

- insert:
- deleteMin:
Binary Heap Priority Q Data Structure

- Structure property
  - complete binary tree (fringe nodes packed to the left)
  - result: depth is always \( O(\log n) \); next open location always known
- Heap-order property
  - parent’s key is less than children’s keys
  - result: minimum is always at the top

How do we find the minimum?

Array Implementation

- Calculations...
  - left child:
  - right child:
  - parent:
  - root:
  - next free:

Array Implementation

Insert

\texttt{pqueue.insert(3)}

Percolate Up

Insert Code

```java
void insert(Object o) {
    // Heap cannot be full!
    size++; // Heap[0]++;
    newPos = percolateUp(size, o); // Heap[newPos] = o;
}
```

DeleteMin

```java
pqueue.deleteMin()
```

runtime:
Percolate Down

Finally…

DeleteMin Code

```java
Object deleteMin() {
    // Heap cannot be empty
    returnVal = Heap[1];
    size--; // Heap[0]--;
    newPos = percolateDown(1, Heap[size+1]);
    Heap[newPos] = Heap[size + 1];
    return returnVal;
}
```

```java
int percolateDown(int hole, Object val) {
    while (2*hole <= size) {
        left = 2*hole;
        if (right <= size && Heap[right] < Heap[left])
            target = right;
        else
            target = left;
        if (Heap[target] < val) {
            Heap[hole] = Heap[target];
            hole = target;
        } else
            break;
    }
    return hole;
}
```

runtime:

Coming Up

- Leftist heaps
- Skew heaps
- d-heaps

• No section on July 4!!!