CSE 326: Data Structures

Topic 3: Priority Queues and Binary Heaps

Ashish Sabharwal
Autumn, 2003

Today’s Outline

• Questions on Sound Blaster? (check updates!)
• Finish Asymptotic Analysis
• Trees Review
  • Priority Queues
  • (Binary) Heaps
  • d-Heaps

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.
  \[ T(n) = 2T(n/2 + 5) + 1 \to T(n) \geq 2T(n/2) + 1 \]

The One Page Cheat Sheet

• Calculating series:
  e.g. \[ \sum_{i=1}^{n} \frac{n(n+1)}{2} \]

• Solving recurrences:
  e.g. \[ T(n) = T(n/2) + 1 \]

1. Brute force (Section 1.2.3)
2. Induction (Section 1.2.5)
3. Memorize simple ones!
4. General proofs (Section 1.2.5)
   e.g. \[ \text{How many edges in a tree with } n \text{ nodes?} \]
   1. Counterexample
   2. Induction
   3. Contradiction

Tree Review

root(T):
leaves(T):
children(B):
parent(H):
siblings(E):
ancestors(F):
descendants(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?

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

A New ADT!

Let’s create a new ADT to solve this problem!

G(9) insert D(100) A(4) B(6)

F(7) E(5) deleteMin C(3)

What do we need to define this ADT?

Priority Queue ADT

1. **PQueue data**: collection of data with priority
2. **PQueue operations**
   - create
   - insert
   - deleteMin
   - is_empty
   Note: Often represented as collection of priorities, with data implicit
3. **PQueue 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
- Simulate events with explicit priorities
- Select most frequent symbols for compression
- Sort numbers, picking minimum first
- Anything greedy

Naïve Priority Q Data Structures

- **Unsorted array**:
  - insert:
  - deleteMin:
- **Sorted array**:
  - insert:
  - deleteMin:

Of the two, which is likely to be better?
Another Priority Q Data Structure:
Binary Search Tree

Average performance
insert:

deleteMin:

Problems
1.
2.

A Better Priority Q Data Structure:
Binary Heap

1. Heap-order property
   - parent’s key is less than
     children’s keys
   - result: minimum is always
     at the top

2. Structure property
   - complete tree with fringe
     nodes packed to the left
   - result: depth is always
     \( \log n \); next open
     location always known

How do we find the minimum?

Nifty Storage Trick: Array

- Index calculations:
  - child:
  - parent:
  - root:
  - next free:  

DeleteMin

```
pqueue.deleteMin()
```

Percolate Down – Basic

```
   5
  / \
 6 / \
9 7
```

Percolate Down – Optimized

```
   5
  / \
 6 / \
9 7
```
DeleteMin Code (Optimized)

```java
void deleteMin() {
    assert(!isEmpty());
    size--;
    newPos = percolaDown(1, Heap[size+1]);
    Heap[newPos] = Heap[size+1];
    return returnVal;
}
```

Insert

```java
pq_queue.insert(3)
```

Percolate Up

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

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?

More Priority Queue Operations

- **remove**
  - given a pointer to an object in the queue, remove it
  Solution: set priority to negative infinity, percola up to root and deleteMin

- **buildHeap**
  Naive solution:
  Running time:
  Can we do better?
Build Heap: Floyd’s Method

Add elements arbitrarily to form a complete tree. Pretend it’s a heap and fix the heap-order property!

Finally…

runtime:

A Solution: d-Heaps

- Each node has \( d \) children
- Still representable by array
- Good choices for \( d \):
  - choose a power of two for efficiency
  - fit one set of children in a cache line
  - fit one set of children on a memory page/disk block
  - optimize performance based on # of inserts/removes

Operations on \( d \)-Heap

- Insert: runtime =
- deleteMin: runtime =

Does this help insert or deleteMin more? Is this good or bad?
One More Operation

- Merge two heaps. Ideas?

Can do in $O(\log n)$ worst case time.
Next lecture!

To Do

- Assignments: Project 1 – check updates!
- Reading: Chapter 6
- Admin: Sign up for class email list