CSE 326 Lecture 11: Heaps and Binomial Queues

✦ What’s on the menu today?
  ✓ **Heaps**: DeleteMin, Insert, DecreaseKey, BuildHeap…
  ✓ **Binomial Queues**: Merge, Insert, DeleteMin

✦ Covered in Chapter 6 in the text

---

**Flashback**

Definition of Heaps

✦ A binary heap is a binary tree that is:
  1. **Complete**: Tree completely filled except possibly the bottom level, which is filled from left to right
  2. **Satisfies the heap order property**: every node is smaller than (or equal to) its children

✦ Therefore, the root node is always the smallest in a heap

Array implementation

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

flashback icon

R. Rao, CSE 326
Last Time: Heap Operations

Basic Heap ADT Operations: FindMin, DeleteMin, Insert

FindMin:
Return Root

Fine but how do we DeleteMin?

DeleteMin using Percolate Down

- Replace with smaller child and go down one level
- Done if both children are $\geq$ item or reached a leaf node
- Maintains both completeness and Heap order
Heaps: Insert Operation

Uh... How would I Insert 1 into this heap?

N = 10

Insert: Percolate Up

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

✦ Every iteration of Insert needs to test:
  1. if it has reached the top node A[1]
  2. if parent ≤ item
✦ Can avoid first test if A[0] contains a very large negative value (denoted by -∞)
✦ Then, test #2 always stops at top ✗ - ∞ < item for all items
✦ Such a data value that serves as a marker is called a sentinel ➤ Used to improve efficiency and simplify code

Summary of Heap ADT Analysis: Space

✦ Consider a heap of N nodes
✦ Space needed: O(N)
  ➤ Actually, O(MaxSize) where MaxSize = size of the array
  ➤ One more variable to store the current size N
  ➤ With sentinel:
    ◆ Array-based implementation uses total N+2 space
    ◆ Pointer-based implementation: pointers for children and parent
      ♦ Total space = 3N + 1 (3 pointers per node + 1 for size)
Run Time Analysis of Heap ADT

✦ Consider a heap of N nodes
✦ FindMin: $O(1)$ time
✦ DeleteMin and Insert: $O(\log N)$ time
✦ **BuildHeap** from N inputs: What is the run time?
  ➔ $N$ Insert operations = $O(N \log N)$.
  ➔ Can we do better?

=N Insert operations = $O(N \log N)$.
Can we do better… $O(N)$: Treat input array as a heap and fix it using percolate down

<table>
<thead>
<tr>
<th>i</th>
<th>N/2 to 1, percolateDown(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why N/2? Nodes after N/2 are leaves!</td>
<td></td>
</tr>
</tbody>
</table>
See text for proof that this takes $O(N)$ time.

R. Rao, CSE 326

9

Run Time Analysis of Heap ADT

✦ Consider a heap of N nodes
✦ FindMin: $O(1)$ time
✦ DeleteMin and Insert: $O(\log N)$ time
✦ **BuildHeap** from N inputs: What is the run time?
  ➔ $N$ Insert operations = $O(N \log N)$.
  ➔ Actually, can do better… $O(N)$: Treat input array as a heap and fix it using percolate down
  ♦ for $i = N/2$ to 1, percolateDown(i)
  ♦ Why N/2? Nodes after N/2 are leaves!
  ♦ See text for proof that this takes $O(N)$ time.

R. Rao, CSE 326

10
Other Heap Operations

✦ **Find(X, H):** Find the element X in heap H of N elements
  ➤ What is the running time?

✦ **FindMax(H):** Find the maximum element in H
  ➤ What is the running time?

One More Operation

✦ **Find and FindMax:** $O(N)$

✦ **DecreaseKey(P, Δ, H):** Decrease the key value of node at position P by a positive amount Δ.
  ➤ E.g. System administrators can increase priority of important jobs.
  ➤ How?
    ♦ First, subtract Δ from current value at P
    ♦ Heap order property may be violated
    ♦ Percolate up or down?
    ♦ Running time?
Some More Ops…

- **DecreaseKey(P, Δ, H):** Subtract Δ from current key value at P and **percolate up**. Running Time: $O(\log N)$

- **IncreaseKey(P, Δ, H):** Add Δ to current key value at P and **percolate down**. Running Time: $O(\log N)$
  - E.g. Schedulers in OS often decrease priority of CPU-hogging jobs (sound familiar?)

- **Delete(P, H):** E.g. Delete a job waiting in queue that has been preemptively terminated by user
  - How (using above operations)?
  - Running Time?

One Last Operation: Merge

- **Delete(P, H):** E.g. Delete a job waiting in queue that has been preemptively terminated by user
  - Use DecreaseKey(P, $\infty$, H) followed by DeleteMin(H).
  - Running Time: $O(\log N)$

- **Merge(H1, H2):** Merge two heaps H1 and H2 of size $O(N)$. H1 and H2 are stored in two arrays. E.g. Combine queues from two different sources to run on one CPU.
  1. Can do $O(N)$ Insert operations: $O(N \log N)$ time
  2. Better: Copy H2 at the end of H1 and use **BuildHeap**
     - Running Time: $O(N)$

  Can we do even better? (i.e. Merge in $O(\log N)$ time?)
Say Hello to Binomial Queues

- Binomial queues support all three priority queue operations \texttt{Merge}, \texttt{Insert} and \texttt{DeleteMin} in $O(\log N)$ time
- Idea: Maintain a collection of heap-ordered trees
  - \textit{Forest of binomial trees}
- Recursive Definition of Binomial Tree (based on height $k$):
  - Only one binomial tree for a given height
  - Binomial tree of height 0 = single root node
  - Binomial tree of height $k = B_k = \text{Attach } B_{k-1} \text{ to root of another } B_{k-1}$
3 Steps to Building a Binomial Tree

✦ To construct a binomial tree $B_k$ of height $k$:
1. Take the binomial tree $B_{k-1}$ of height $k-1$
2. Place another copy of $B_{k-1}$ one level below the first
3. Join the root nodes

✦ Binomial tree of height $k$ has exactly $2^k$ nodes (by induction)

Definition of Binomial Queues

Binomial Queue = “forest” of heap-ordered binomial trees

Binomial queue $H_1$ 5 elements = $2^0 + 2^2$
i.e. Uses $B_0$ and $B_2$

Binomial queue $H_2$ 11 elements = $2^0 + 2^1 + 2^3$
i.e. uses $B_0 B_1 B_3$
Binomial Queue Properties

✦ Suppose you are given a binomial queue of N nodes

1. There is a unique set of binomial trees for N nodes (express N in binary to find out which trees are in the set)

2. What is the maximum number of trees that can be in an N-node queue?
   ⇒ 1 node 1 tree $B_0$; 2 nodes 1 tree $B_1$; 3 nodes 2 trees $B_0$ and $B_1$; 7 nodes 3 trees $B_0$, $B_1$, and $B_2$ …

Number of Trees in a Binomial Queue

✦ What is the maximum number of trees that can be in an N-node binomial queue?
   ⇒ 1 node 1 tree $B_0$; 2 nodes 1 tree $B_1$; 3 nodes 2 trees $B_0$ and $B_1$; 7 nodes 3 trees $B_0$, $B_1$, and $B_2$ …

✦ Trees $B_0$, $B_1$, …, $B_k$ can store up to $2^0 + 2^1 + … + 2^k = 2^{k+1} - 1$ nodes = N.

✦ Maximum is when all $k+1$ trees are used.

✦ So, number of trees in an N-node binomial queue is $\leq k+1 = (\log(N+1)-1)+1= O(\log N)$
Binomial Queues: Merge

✦ Main Idea: Merge two binomial queues by merging individual binomial trees
✦ Since $B_{k+1}$ is just two $B_k$'s attached together, merging trees is easy
✦ Creating new queue by merging:
  1. Start with $B_k$ for smallest $k$ in either queue.
  2. If only one $B_k$, add $B_k$ to new queue and go to next $k$.
  3. Merge two $B_k$’s to get new $B_{k+1}$ by making larger root the child of smaller root. Go to step 2 with $k = k + 1$.

Binomial Queues: Merge Exercise

✦ What do you get when you Merge H1 and H2?

H1:

```
8
9
```

```
5
6
```

```
7
```

H2:

```
2
```

```
1
```

```
4
```

```
7
```

```
2
```

```
1
```

```
8
```

```
1
```

```
5
```

```
6
```

Binomial Queues: Merge

What is the run time for Merge of two O(N) queues?

BQ after Merge

Binomial Queues: Merge and Insert

What is the run time for Merge of two O(N) queues?
- Keep connecting roots of trees
- Total Run Time = O(number of trees) = O(log N)

Uh...now how would I Insert “1” into this BQ?
Binomial Queues: Insert

✦ How would you insert a new item into the queue?
  ➤ Create a single node queue $B_0$ with new item and Merge with existing queue
  ➤ Again, $O(\log N)$ time
✦ Exercise: Insert 1, 2, 3, …, 7 into an empty binomial queue

Binomial Queues: DeleteMin

Insert is easy… how do we DeleteMin?

R. Rao, CSE 326
Binomial Queues: \text{DeleteMin}

✦ Steps:
1. Find tree $B_k$ with the smallest root
2. Remove $B_k$ from the queue
3. \textbf{Delete root} of $B_k$ (return this value); You now have a second queue made up of the forest $B_0, B_1, \ldots, B_{k-1}$
4. Merge this queue with remainder of the original (from step 2)

✦ \textbf{Run time analysis}: How much time do Steps 1 through 4 take for an $N$-node queue?

\[\text{Run time analysis: Step 1 is } O(\log N), \text{ steps 2 and 3 are } O(1), \text{ and step 4 is } O(\log N). \text{ Total time } = O(\log N)\]
Next Class:
From Heaps to Hashes

To Do:
Finish Chapter 6 and Start Chapter 5
Homework #3 has been assigned on the Web
Due Thursday, Feb 13. Start Early!!