Lecture 11: Priority Queues

(Chapter 9)

Nov 15, 2016

Data Structures (CS 102)

Overview

Next...

I. Warmup, Review

II. Heaps

III. More on Heaps

I. Warmup, Review

Quiz of Last Thursday

Discussions

Minute Quiz

Two questions on recursion

Q1: Write a recursive method with this header

```
int height(BinNode u);
```

– returns height of binary tree rooted at `u`

– May assume the method `int max(int x, int y)`

Q2: Write a recursive method

```
Node reverse(Node u);
```

– returns head of the reverse of the singly-linked list beginning at `u`.

Minute Quiz

SOLUTION 1:

```
height height(BinNode u){
   if (u == null) return -1;
   return 1+max(height(u.left), height(u.right));
}
```

– Note: the height of the empty tree is −1.

Minute Quiz

SOLUTION 2:

```
Node reverse(Node u){
   if (u == null || u.next == null) return u;
   Node v = u.next;
   Node w = reverse(u.next); // w is non-null
   v.next = u;
   u.next = null; // (*)
   return w;
}
```

– Q: Isn't line (*) a bit redundant?

– A: Yes, except for the last call.

Minute Quiz

Therefore: we can have a 20% speedup if we

make reverse an internal method,

omit (*), and

use an external method to call reverse.

Next...

I. Warmup, Review

II. Heaps

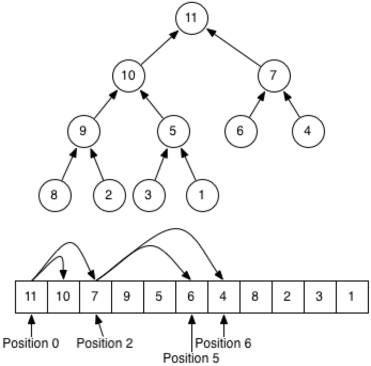
III. More on Heaps

[Start] [End]

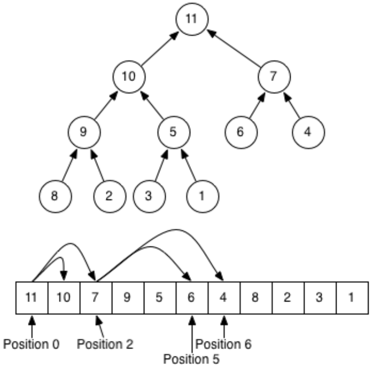
II. Heaps

Heaps

View an array as a binary tree

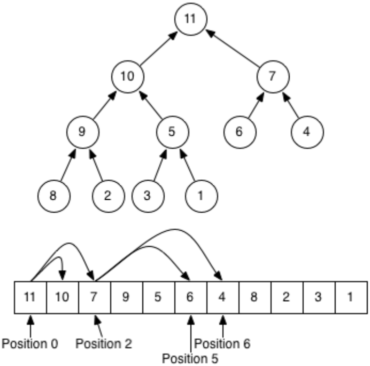


Heaps



This is a complete binary tree

Heaps



Definition: if height is h, then for each $i = 0, \ldots, h-1$,

– level i is filled with 2^i nodes, while

– level h is filled from left to right.

Heaps

(Min) Heap Property

At each non-root u:

$$\texttt{u.Key} \leq \texttt{u.Parent.Key}$$

REMARK: compare to BST Property

Heaps

How to Enqueue

reheapUp(`k`)

- There is only ONE place to place key `k`.
- Place key there and "reheapUp"
- Blackboard Lecture

Heaps

How to Dequeue

reheapDown(`k`)

– There is only ONE place to remove the node

– Remove that node, put it at the root, and “reheapDown”

– Blackboard Lecture

Heaps

Code Review

PQeueue.java (in Piazza Resources)

Next...

I. Warmup, Review

II. Heaps

III. More on Heaps

III. More on Heaps

Hand Simulations

Binary Heap Demo

Hand Simulations

Blackboard Demo:

* Enqueue these items into an empty P-Queue:

`3, 1, 4, 1, 5; 9, 2, 6, 5, 3`

* Dequeue until empty

Hand Simulations

Blackboard Demo:

* The following array is not a heap: but fix it with one swap:

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12
Keys:	13	8	10	7	9	6	2	1	6	0	3	5	4

Hand Simulations

Swap 8 and 9!

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12
Keys:	13	8	10	7	9	6	2	1	6	0	3	5	4

Hand Simulations

Put it all together

Code Review (eclipse)

Complexity

Let $ht(n)$ be the height of the complete binary tree of size n.

$\boxed{\text{LEMMA: } \lg(n+1) \leq ht(n) \leq \lg(n+1)+1}$

* Proof: Let $h = ht(n)$. Then

$$n \geq \sum_{i=0}^{h-1} 2^i = 2^h - 1$$

$$\lg(n+1) \geq h$$

$$n \leq \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

$$\log(n+1) \geq h+1$$

* reheapUp(k) takes $\log(n)$ time

* reheapDown(k) takes $\log(n)$ time

Complexity

Let $ht(n)$ be the height of the complete binary tree of size n.

$\boxed{\text{LEMMA: } \lg(n+1) \leq ht(n) \leq \lg(n+1)+1}$

* Proof: Let $h = ht(n)$. Then

$$n \geq \sum_{i=0}^{h-1} 2^i = 2^h - 1$$

$$\lg(n+1) \geq h$$

$$n \leq \sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

$$\log(n+1) \geq h+1$$

* reheapUp(k) takes $\log(n)$ time

* reheapDown(k) takes $\log(n)$ time

Complexity:

* enqueue $O(\log n)$

* dequeue $O(\log n)$

Complexity

What is height when `n=10`?

Answer: 3

... when `n=100`?

NEED exact formula for height: $\boxed{ht(n) = \lceil \lg(1+n) \rceil - 1}$

$\boxed{ht(n) = \lceil \lg(1+n) \rceil - 1}$

Answer: $ht(100) = \lceil \lg(101) \rceil - 1 = 7 - 1 = 6$

... when `n=1000`?

Answer: $ht(1000) = \lceil \lg(1001) \rceil - 1 = 10 - 1 = 9$

... when `n=1000,000`?

Answer: $ht(1000,000) = 20 - 1 = 19$

... when `n=1000,000,000`?

Answer: $ht(1000,000,000) = 29$

Complexity

Proof of $\boxed{ht(n) = \lceil \lg(n+1) \rceil - 1}$

Level i has 2^i nodes

So with height h, we have at most $1+2+4+\cdots+2^h = 2^{h+1}-1$ nodes

Thus $n \leq 2^{h+1}-1$

Or, $n+1 \leq 2^{h+1}$

Or, $\lg(n+1) \leq h+1$

By the minimality of h, we conclude that $\lceil \lg(n+1) \rceil = h+1$

Thus $ht(n) = \lceil \lg(n+1) \rceil - 1$

Heap Sort

How does priority Queues lead to a sorting algorithm?

1. Insert all the keys into a priority queue
2. Dequeue all the keys in the Priority queue

Heap Sort

How does priority Queues lead to a sorting algorithm?

1. Insert all the keys into a priority queue
2. Dequeue all the keys in the Priority queue

Heapsort:

Suppose the input keys are in an array

Organize so that we do not need a second array!

Heap Sort

Heapsort:

Suppose the input keys are in an array

Organize so that we do not need a second array!

Two ideas:

1. Buildheap (by repeated enqueue-ing)
2. Dequeue but keep items in queue

RESULT: we can sort n items in $O(n \log n$ time.

Blackboard Demo.

[Web Animation]

Heap Sort

Heapsort Code Review

BuildHeap (Advanced Topic)

How to do Builheap

* Do this level-by-level

$\boxed{\text{RESULT:}}$ Can do Buildheap in $O(n)$ instead of $O(n \log n)$

BuildHeap (Advanced Topic)

Complexity?

Level h: 0

Level $h-1$: $(n/2)\cdot 1$

Level $h-2$: $(n/4)\cdot 2$

$\vdots$

Level $h-i$: $(n/2^i)\cdot i$

$\vdots$

Level $h-h=0$: $(n/2^h)\cdot h = h$

BuildHeap (Advanced Topic)

CLAIM: $\boxed{\text{Buildhead(n) takes } O(n) \text{ time}}$

Proof:

$$T = n\sum_{i=0}^{h-1} \frac{i}{2^i}$$

But $\frac{i}{2^i} < \frac{1}{2^{i/2}}$ for $i \geq 4$

So $T = O(n) + n\sum_{i\geq 4} \frac{1}{2^{i/2}}$

$= O(n) + n\sum_{i\geq 4} c^i$ (where $c = 2^{-1/2}$)

$= O(n)$

Thanks for Listening!

"Algebra is generous,
she often gives more than is asked of her."

— Jean Le Rond D'Alembert (1717-83)