CSE 332: Data Structures and Parallelism

Spring 2022

Richard Anderson

Lecture 5: Binary Heaps

Announcements

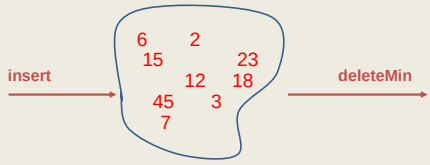
- Reading: Weiss
  - Today: Priority Queues, 6.1-6.5
  - Friday: Algorithm Analysis, 2.1-2.2

Today

- Binary Tree Heaps
  - Insert
  - Delete Min
- Array implementation
- Heap Initialization
  - Put n elements into a heap in O(n) time

Priority Queue ADT

- Operations: Insert an item,
  Remove the "Best" Item

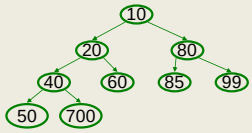


Binary Heap Properties

Store elements in a binary nodes of a binary tree

Maintain the two properties

1. Heap Order – parents are smaller than their children
2. Completeness – the tree is of minimum depth with all leaves as far to the left as possible

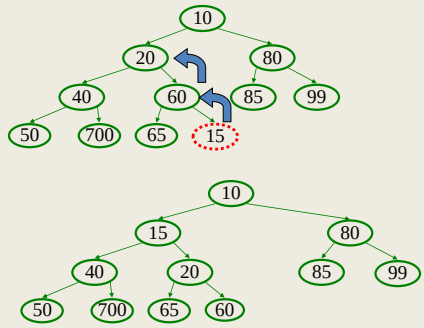


Heap Operations

- Operations: insert, deleteMin
- Maintain heap properties by propagating changes either up or down the tree
- The heap properties imply
  - If there are n values in the heap, the tree has height log n
  - The smallest value is at the root
  - When a value is added, a new leaf is created in the left most position

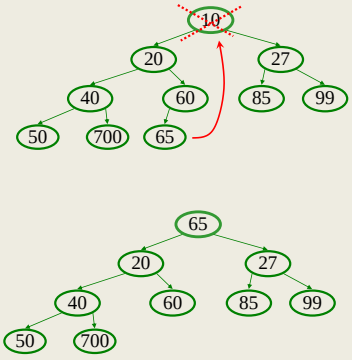
Heap Insert

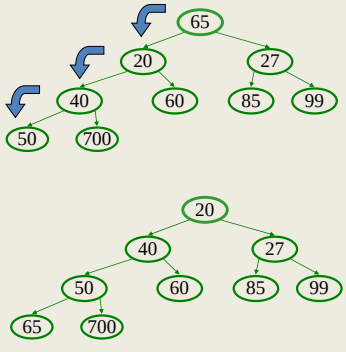
- Put value in first available leaf position
- Swap value up until heap condition is satisfied



Heap DeleteMin

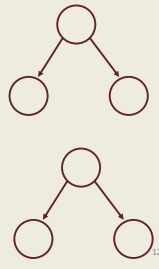
- Remove the root (minimum element)
- Promote the last leaf to the root
- Swap with smallest child until heap property is restored





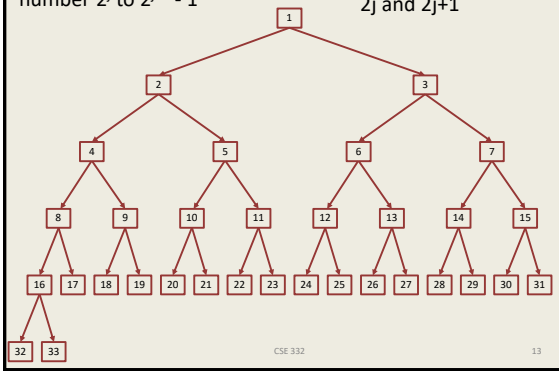
Correctness Proofs

- Show operations preserve heap properties
- Insert
  - Complete tree
  - Parent smaller than children
- DeleteMin
  - Return smallest element
  - Complete tree
  - Parent smaller than children



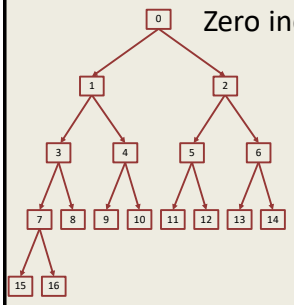
Nodes on level j are number 2^j to $2^{j+1} - 1$

Node j has children 2j and 2j+1



Mapping a binary tree to an array

Zero indexing

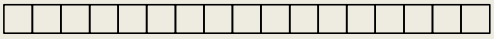


Node j has:

Left child 2j + 1

Right child 2j + 2

Parent $\lfloor (j-1)/2 \rfloor$



Why use an array

Insert Code

```
void insert(int v) {
  assert(!isFull());
  size++;
  newPos =
    percolateUp(size,v);
  Heap[newPos] = v;
}
```

```
int percolateUp(int hole,
                int val) {
  while (hole > 0 &&
         val < Heap[(hole-1)/2])
    Heap[hole] = Heap[(hole-1)/2];
    hole = (hole-1)/2;
  }
  return hole;
}
```

runtime:

(Java code in book)

DeleteMin Code

```
Object deleteMin() {
  assert(!isEmpty());
  returnVal = Heap[0];
  size--;
  newPos =
    percolateDown(0,
        Heap[size + 1]);
  Heap[newPos] =
    Heap[size + 1];
  return returnVal;
}
```

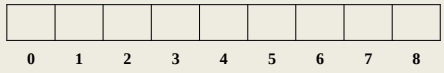
runtime:

(Java code in book)

```
int percolateDown(int hole,
                  int val) {
while (2*hole <= size) {
    left = 2*hole + 1;
    right = left + 1;
    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;
}
```


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



More Priority Queue Operations

decreaseKey(nodePtr, amount):
given a pointer to a node in the queue, reduce its priority

Binary heap: change priority of node and _____

increaseKey(nodePtr, amount):
given a pointer to a node in the queue, increase its priority

Binary heap: change priority of node and _____

More Priority Queue Operations

remove(objPtr):
given a pointer to an object in the queue, remove it

Binary heap: _____

findMax( ):
Find the object with the highest value in the queue

Binary heap: _____

Building a Heap

12	5	11	3	10	6	9	4	8	1	7	2

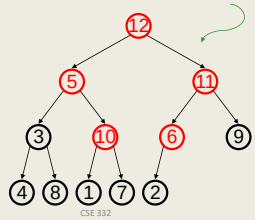
BuildHeap: Floyd's Method

12	5	11	3	10	6	9	4	8	1	7	2

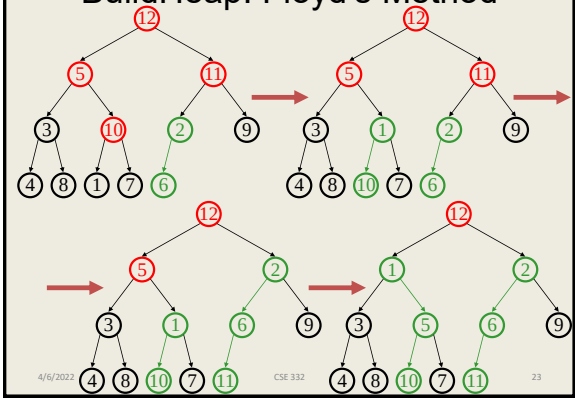
Add elements arbitrarily to form a complete tree.
Pretend it's a heap and fix the heap-order property!

Red nodes need to percolate down

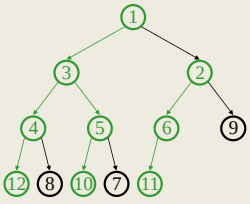
Key idea: fix red nodes from **bottom-up**



BuildHeap: Floyd's Method



Finally . . .



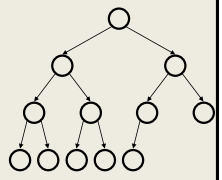
Buildheap pseudocode

```
private void buildHeap() {
  for ( int i = currentSize/2; i >= 0; i-- )
     percolateDown( i );
}
```

runtime:

Buildheap Analysis

n/4 nodes percolate at most 1 level
n/8 percolate at most 2 levels
n/16 percolate at most 3 levels
...



runtime:

The Math:

$$\sum_{i\geq 1} \frac{i}{2^i} = 2$$

$$\frac{n}{4}+\frac{2n}{8}+\frac{3n}{16}+\frac{4n}{32}+\cdots = \frac{n}{2}\left[\frac{1}{2}+\frac{2}{4}+\frac{3}{8}+\frac{4}{16}+\cdots\right] = \frac{n}{2}\sum_{i\geq 1}\frac{i}{2^i}$$

$$S = \sum_{i\geq 1}\frac{i}{2^i} = \sum_{i\geq 1}\frac{1}{2^i}+\sum_{i\geq 1}\frac{i-1}{2^i} = 1+\sum_{i\geq 1}\frac{i-1}{2^i} = 1+\frac{1}{2}\sum_{i\geq 1}\frac{i-1}{2^{i-1}}$$

$$= 1+\frac{1}{2}\sum_{i\geq 0}\frac{i}{2^i} = 1+\sum_{i\geq 1}\frac{i}{2^i} = 1+\frac{S}{2}$$