

Heapsort, Priority Queue

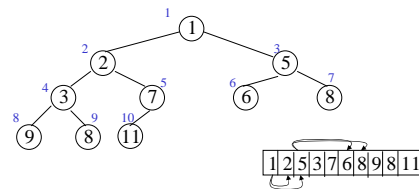
Outline

- Heap data structure
- Extract min
- Insert
- Priority queue
- Heapsort

Binary Min-heap

- Nearly complete binary tree that satisfies the heap property
 - tree completely filled on all levels except lowest level, which is filled from the left
- Heap property:
 - $A[\text{parent}(x)] \leq A[x]$

Array Representation



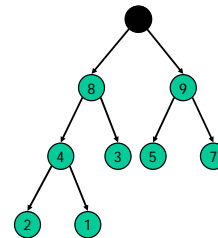
Left(i) = 2i
Right(i) = 2i+1
Parent(i) = $\lfloor i/2 \rfloor$

Extract Min

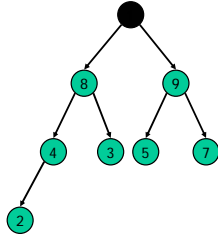
```
EXTRACT-MIN(A)
1  min ← A[1]
2  A[1] ← A[heap-size[A]]
3  heap-size[A] ← heap-size[A] - 1
4  MIN-HEAPIFY(A,1) Δ maintain heap property
5  return min
```

- Extract the min element for priority queue
- Simple way to sort - repeatedly extract-min.

Remove the max item



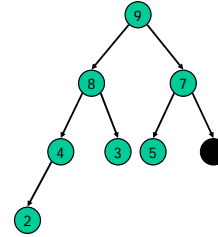
Re-establish heap property



Recitation 3: Heapsort, Priority Queue

7

Re-establish heap property



bubble down

Recitation 3: Heapsort, Priority Queue

8

Min-Heapify

MIN-HEAPIFY(A, i)

```

1  if  $2i \leq \text{size}[A]$  and  $A[2i] < A[i]$   $\Delta$  left child
2      then  $\text{smallest} \leftarrow 2i$ 
3      else  $\text{smallest} \leftarrow i$ 
4  if  $2i+1 \leq \text{size}[A]$  and  $A[2i+1] < A[\text{smallest}]$   $\Delta$  right child
5      then  $\text{smallest} \leftarrow 2i+1$ 
6  if  $\text{smallest} \neq i$ 
7      then swap  $A[i] \leftrightarrow A[\text{smallest}]$ 
8      MIN-HEAPIFY( $A, \text{smallest}$ )
    
```

- What is the running time of MIN-HEAPIFY?

Recitation 3: Heapsort, Priority Queue

9

Insert

INSERT(key, A)

```

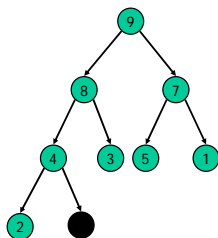
1  heap-size( $A$ )  $\leftarrow$  heap-size( $A$ ) + 1
2   $i \leftarrow$  heap-size( $A$ )
3  while  $i > 1$  and  $A[\lfloor i/2 \rfloor] > key$ 
4      do  $A[i] \leftarrow A[\lfloor i/2 \rfloor]$ 
5          $i \leftarrow \lfloor i/2 \rfloor$ 
6   $A[i] \leftarrow key$ 
    
```

- Start at new leaf, move parent down until we meet a parent smaller than key.
- Running time: $O(\lg n)$

Recitation 3: Heapsort, Priority Queue

10

Insert an item

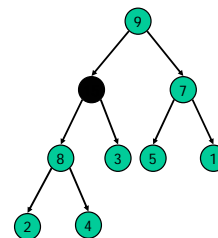


0	12
1	8
2	9
3	4
4	3
5	5
6	7
7	2
8	1
9	

Recitation 3: Heapsort, Priority Queue

11

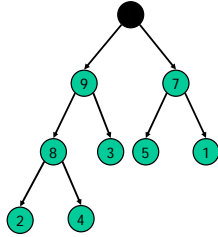
Re-establish heap property



Recitation 3: Heapsort, Priority Queue

12

Re-establish heap property



bubble up

Recitation 3: Heapsort, Priority Queue

13

Priority Queue

- Data structure for maintaining a set of elements, each with an associated value called a key.
- Example: scheduling jobs on a shared computer
 - When job finished, highest priority job is selected to be executed.
- One of the most popular application of heap
- Insert new element in $O(\lg n)$
- Extract element with smallest (largest) key in time $O(\lg n)$.

Recitation 3: Heapsort, Priority Queue

14

Heapsort

HEAPSORT(A)

```

1  BUILD-MAX-HEAP(A)
2  for i ← heap-size(A) downto 2
3      do swap A[1] ↔ A[i]
4      heap-size(A) ← heap-size(A) - 1
5      MAX-HEAPIFY(A,1)
    
```

- Use min-heap to get decreasing seq
- Sorts in place
- Running time = $O(n \log n)$ + Build-max-heap time.

Recitation 3: Heapsort, Priority Queue

15

Build-Min-Heap

BUILD-MIN-HEAP(A)

```

1  for i ← ⌊heap-size(A)/2⌋ downto 1
2      do MIN-HEAPIFY(A,i)
    
```

- Correctness:
 - Invariant: All trees rooted at $m > i$ are heaps
- Running time:
 - Height of tree is $\lfloor \lg n \rfloor$
 - Number of nodes at height h is at most $\lceil n/2^{h+1} \rceil$
 - MIN-HEAPIFY runs in time $O(h)$ when the node is of height h .

Recitation 3: Heapsort, Priority Queue

16

- Running time:

$$\sum_{h=0}^{\lfloor \lg n \rfloor} \left[\frac{n}{2^{h+1}} \right] O(h) = O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h} \right) = O(n)$$

- because

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \quad |x| < 1$$

$$\sum_{i=0}^{\infty} ix^{i-1} = \frac{1}{(1-x)^2} \quad \text{differentiation}$$

$$\sum_{i=0}^{\infty} ix^i = \frac{x}{(1-x)^2} \quad \text{mult both sides by } x$$

$$= 2 \quad \text{when } x = 1/2$$

Recitation 3: Heapsort, Priority Queue

17

Summary

- Heap data structure
 - nearly complete binary tree
 - $A[\text{parent}(x)] \leq A[x]$
- Extract min, insert
 - $O(\lg n)$
- Heapsort
 - $O(n \lg n)$
 - in place sort
- Priority queue
 - popular application of heap - $O(\lg n)$ insert and extract min

Recitation 3: Heapsort, Priority Queue

18