

LECTURE-16

Priority Queue

Priority Queue Introduction



For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan

Department of Computer Science, SBASSE

Agenda

- Introducing Priority Queue ADT as a general Queue ADT
- Operations in the PQ interface
- Contrasting all known collections as candidates to implement PQ
- PQ invariants
- PQ insert()

Typically in what order are patients shown in to the doctor if the clinic does not take appointments?

FIFO



**"MR. JONES, THE DOCTOR SAYS IT'S OK TO MOVE YOU
AHEAD OF THE QUEUE AND HE'LL SEE YOU NOW!"**

Priority Queues

- A priority queue is an ADT for storing a **collection of items with priorities**. It is generalization of the Queue ADT
- A PQ supports “**arbitrary**” insertions but removals must always be in order of priority
- We can view a PQ as a (key,value) store where the **key represents the priority** of the value (or data)

Priority Queue Operations

A PQ supports three basic operations just like a Queue ADT:

- enqueue(Data)
- dequeue()
- peek()

PQ Applications

Why do we use PQs?

- Multi-user environments

For Each of These Priorities come into play either for scheduling or taking turns for items in a set.

- Android mobile operating system

What process needs urgent attention?

- In other algorithms

Some algorithms that need to discover data items based on high priority! Eg. Graph algorithms

Can you spot processes that might require urgent attention as opposed to those that can be dealt with less urgency!?

Running services		
 Mobile Services Manager	30 MB	66:39:02
1 process and 1 service		
 Samsung Keyboard	218 MB	66:39:12
1 process and 1 service		
 Gaming Hub	13 MB	66:39:16
1 process and 1 service		
 One UI Home	284 MB	66:39:17
1 process and 1 service		
 Android Services Library	58 MB	66:39:17
1 process and 1 service		

Running services		
 Customisation Service	28 MB	66:38:56
1 process and 1 service		
 Google Play services	166 MB	18:35
2 processes and 19 services		
 Messages	203 MB	66:39:07
2 process and 1 services		
 Mobile Services Manager	30 MB	66:38:49
1 process and 1 service		
 Samsung Keyboard	414 MB	66:38:59
3 process and 1 services		

Running services		
 Settings	174 MB	
1 process and 0 services		
 com.sec.epdg	33 MB	66:39:40
1 process and 1 service		
 Voice wake-up	19 MB	66:37:43
1 process and 1 service		
 Bluetooth	47 MB	66:39:46
1 process and 1 service		
 Bixby	106 MB	00:08
2 process and 1 services		
 Customisation Service	28 MB	66:38:47
1 process and 0 services		

Running services		
 Nearby device scanning agent	12 MB	
1 process and 0 services		
 Quick Share Connectivity	16 MB	
1 process and 0 services		
 Find My Mobile	14 MB	
1 process and 0 services		
 Meta Services	22 MB	
1 process and 0 services		
 People	19 MB	
1 process and 0 services		

How do we implement PQs?

- Option-1: Unsorted Linked List

	Unsorted LL
insert	$\Theta(1)$
removeMin	$\Theta(N)$
Min	$\Theta(N)$

- What if we keep items in sorted order?

	Unsorted LL	Sorted LL
insert	$\Theta(1)$	$\Theta(N)$
removeMin	$\Theta(N)$	$\Theta(1)$
Min	$\Theta(N)$	$\Theta(1)$

How do we implement PQs?

- Option-2: Arrays

	Unsorted LL	Sorted LL	Array
insert	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$
removeMin	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$
Min	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$

- What if the array is sorted?

	Unsorted LL	Sorted LL	Array	Sorted Array
insert	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$
removeMin	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$
Min	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$

How do we implement PQs?

- Option-3: Balanced BSTs

	Unsorted LL	Sorted LL	Array	Sorted Array	AVL
insert	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(\log N)$
removeMin	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$
Min	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$

- Option-4: Hash Table

	Unsorted LL	Sorted LL	Array	Sorted Array	AVL	Hash Table
insert	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(\log N)$	$\Theta(1)$.
removeMin	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$	$\Theta(N)$
Min	$\Theta(N)$	$\Theta(1)$	$\Theta(N)$	$\Theta(1)$	$\Theta(\log N)$	$\Theta(N)$

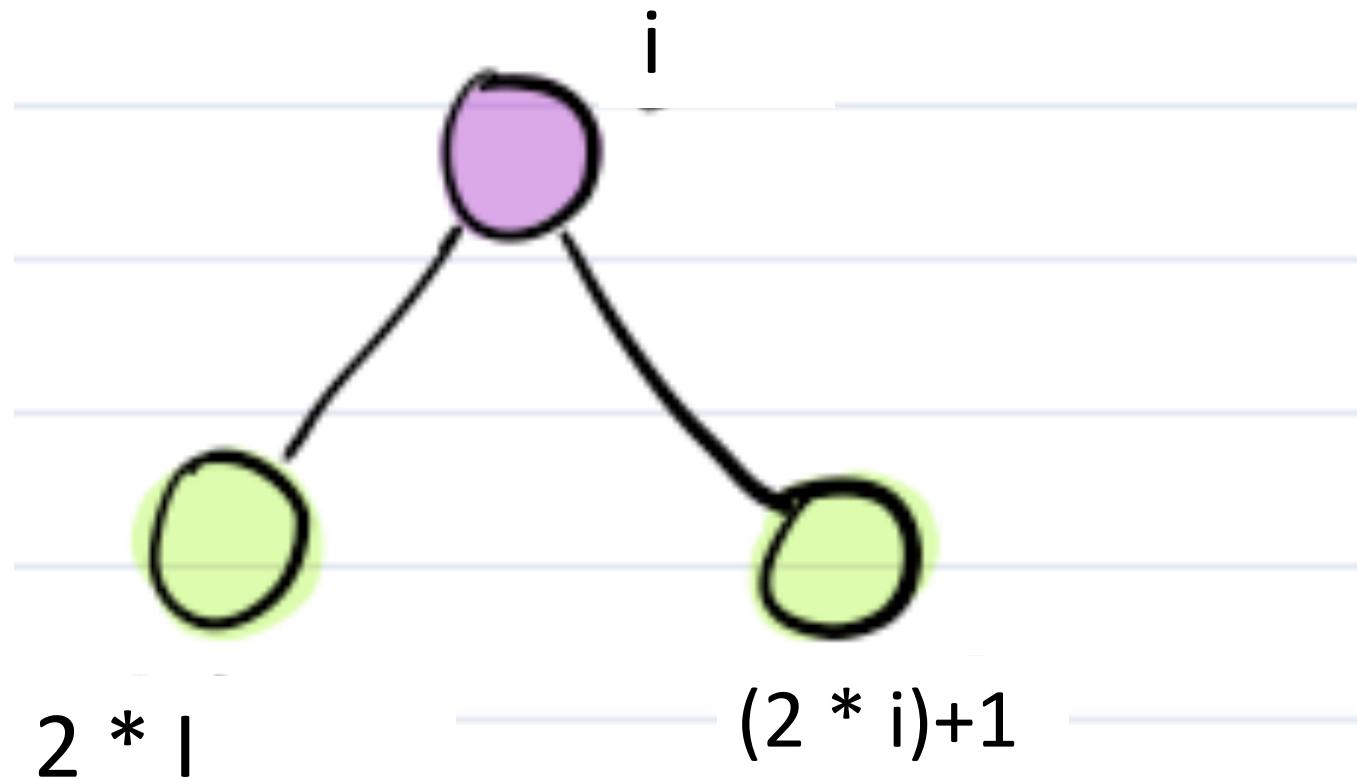
We need a better performance!

We need to do better for both insert() and removeMin() both!

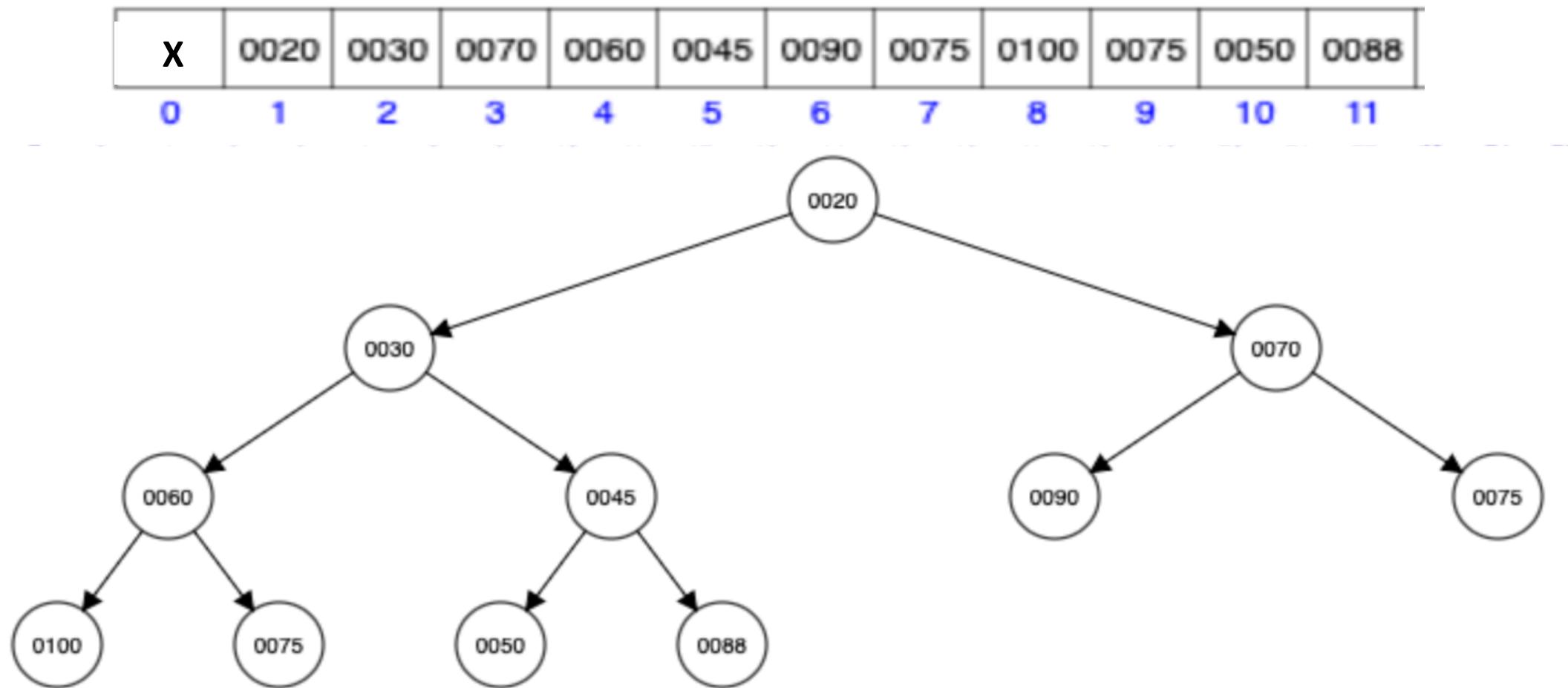
Introducing a heap!

- Structural constraint!
 - The values must be modelled as a COMPLETE TREE.
- Value Constraint
 - A child node must not have a key value that is smaller (larger) than its parent's key value to form a min (max) **heap**

Finding Child/Parent Indices



A PQ of values using heap



Insert()

1. Add the new item to the last of the array.
2. Compare value of key with that of its parent, if the order is reversed swap the data items of the child and the parent.
3. Repeat 2 till you reach the root!