



## LECTURE-16

# Priority Queue

Priority Queue Introduction

## CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan  
Department of Computer Science, SBASSE



For Poll Ev

# Agenda

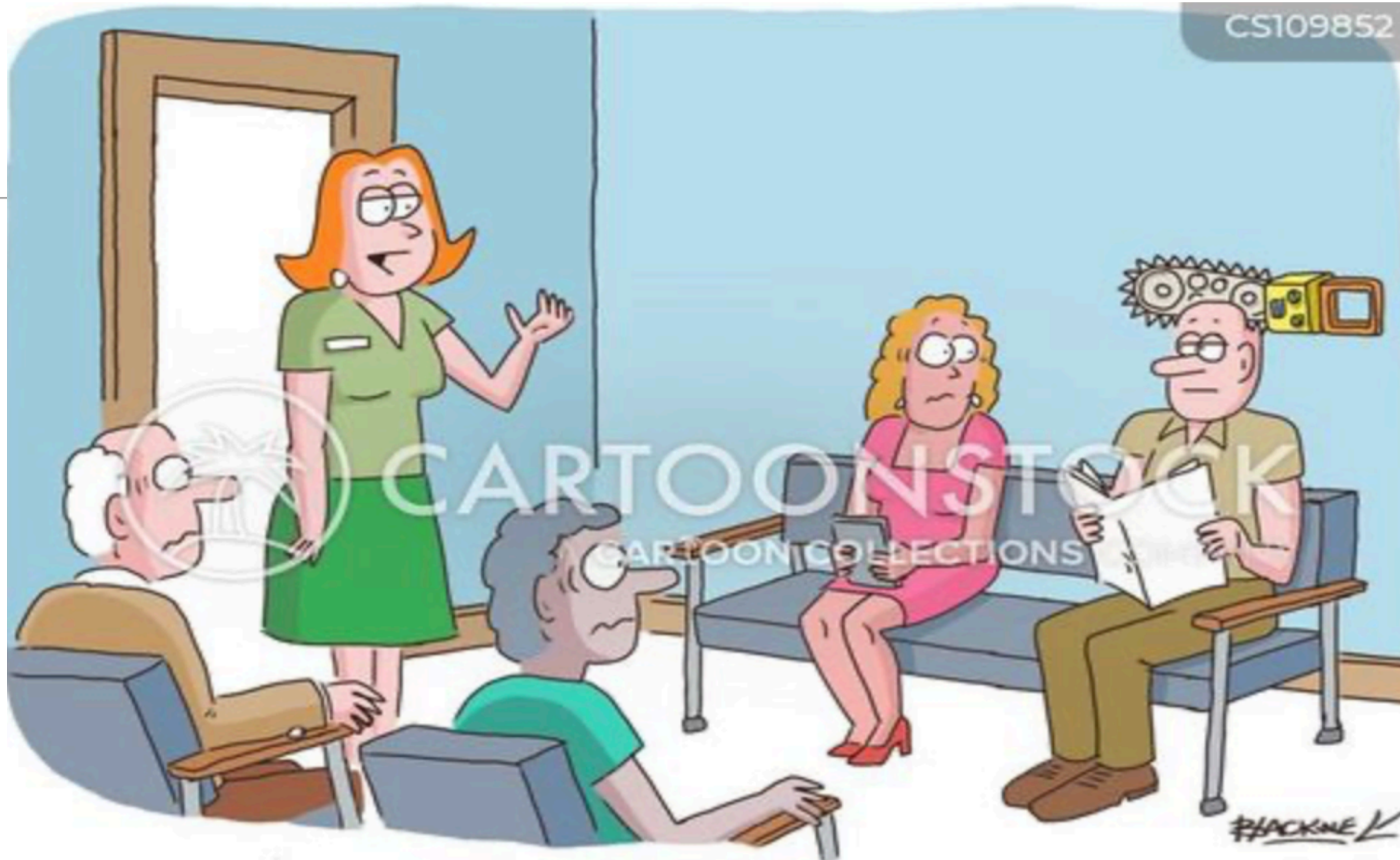
---

- 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
- Android mobile operating system
- In other algorithms

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






What process needs urgent attention?

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!?






07:56

< **Running services** :

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







07:56

< **Running services** :

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






07:55

< **Running services** :

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

07:55

< **Running services** :

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



# 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!

---

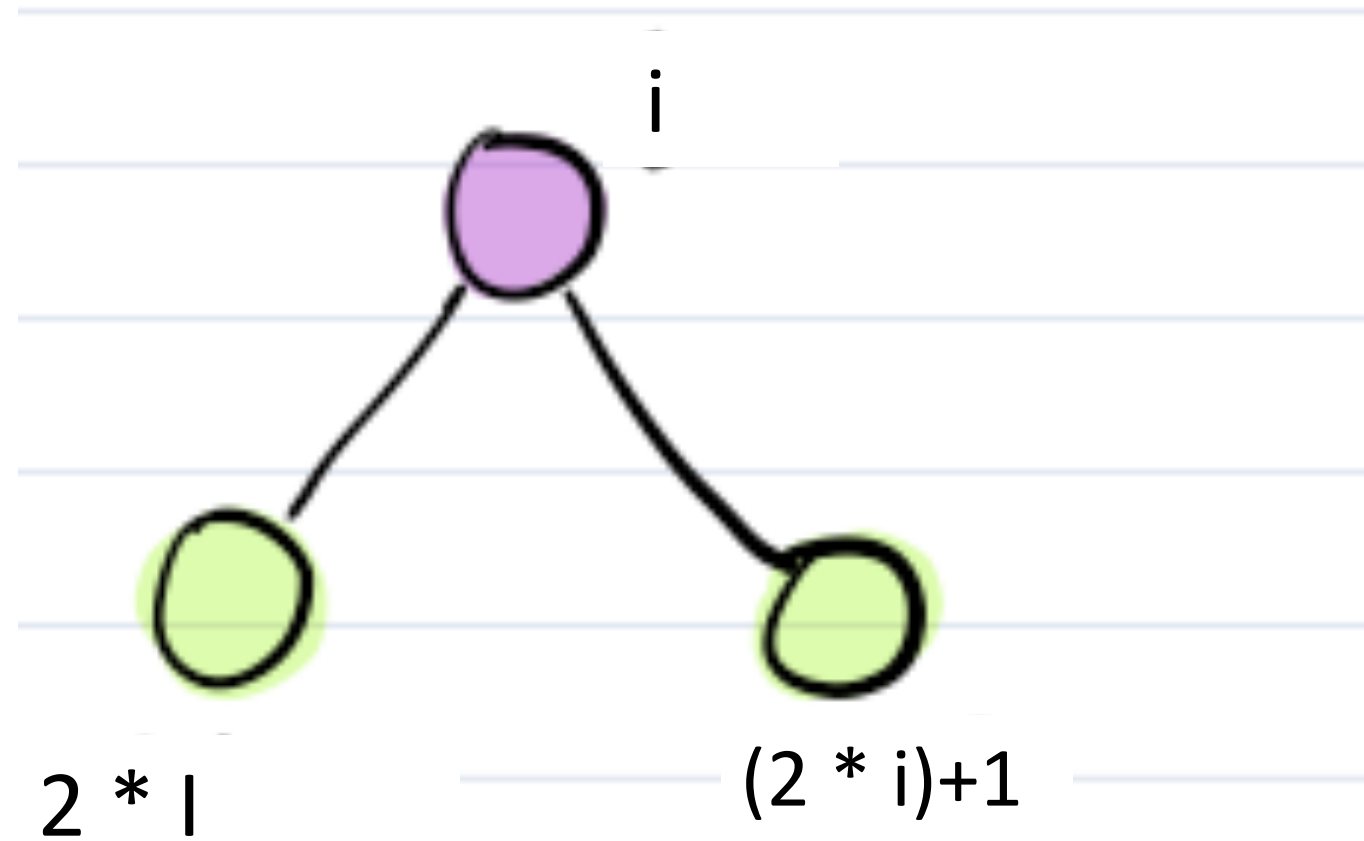
We need to do better for both insert() and removeMin() both!

## 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**

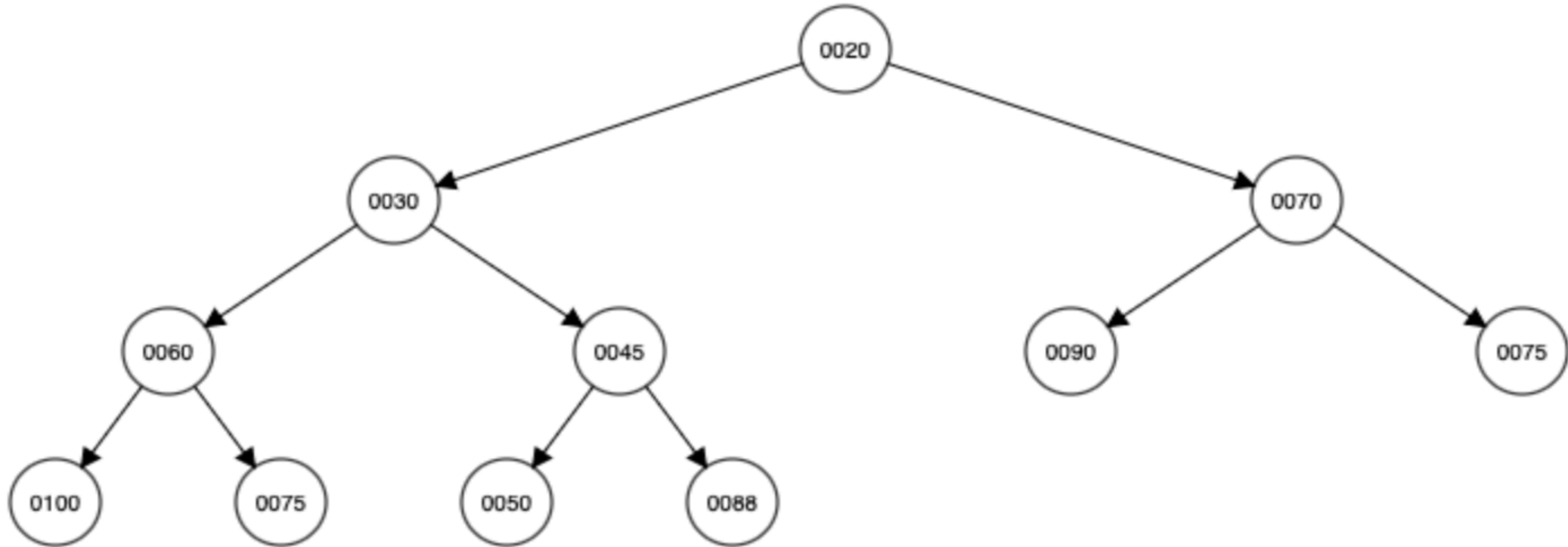
# Finding Child/Parent Indices

---



# A PQ of values using heap

x	0020	0030	0070	0060	0045	0090	0075	0100	0075	0050	0088
0	1	2	3	4	5	6	7	8	9	10	11



# 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!