LECTURE-16

Priority Queue

Priority Queue Introduction

For Poll Ev

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan

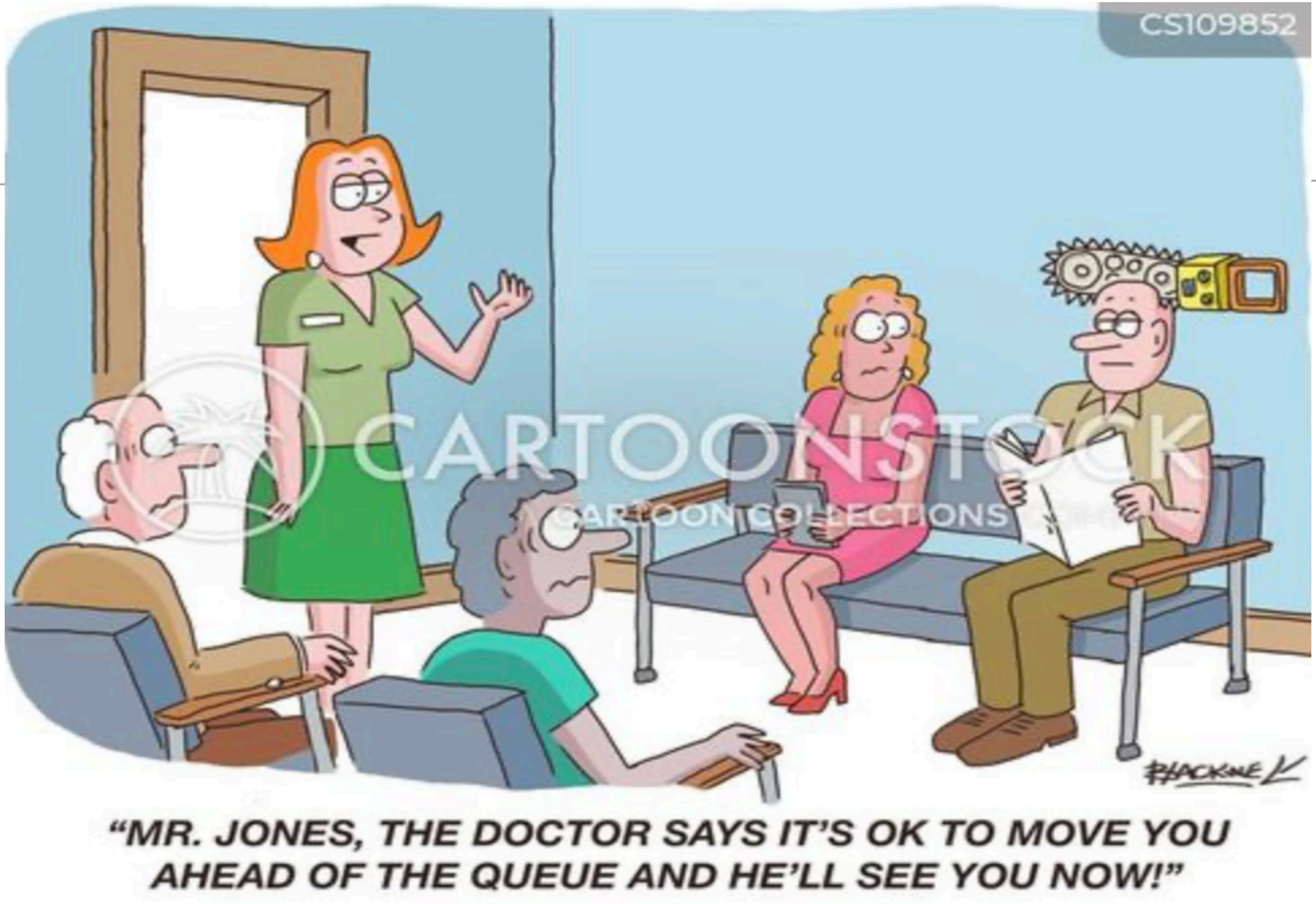
Department of Computer Science, SBASSE

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


CS109852
CARTOONSTOCK
CARTOON COLLECTIONS
BLACKNEY
"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?

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

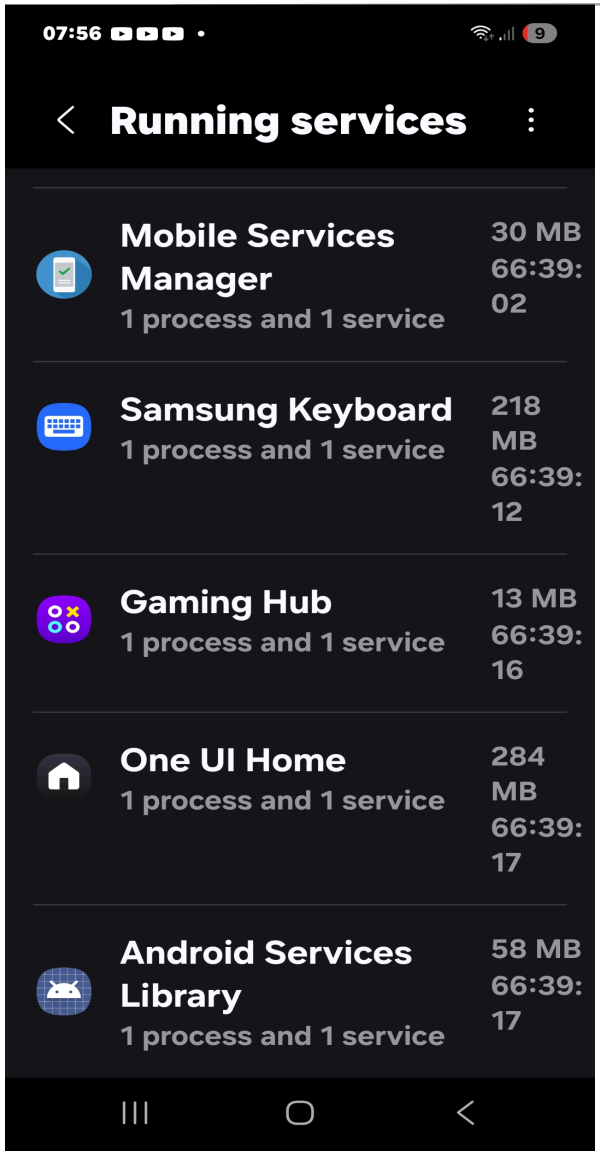
- Multi-user environments
- Android mobile operating system

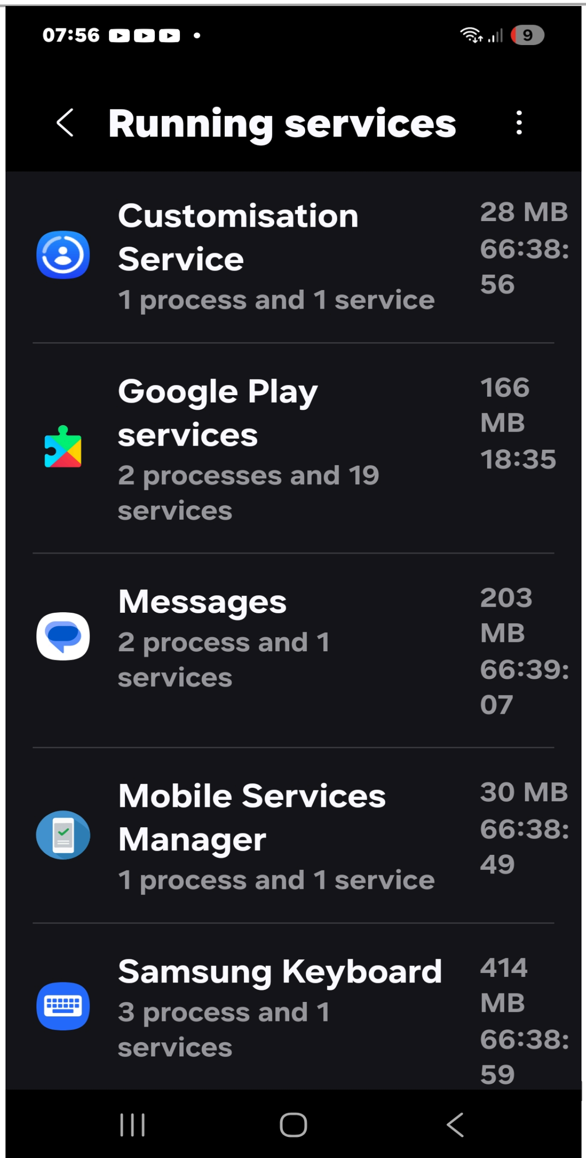
What process needs urgent attention?

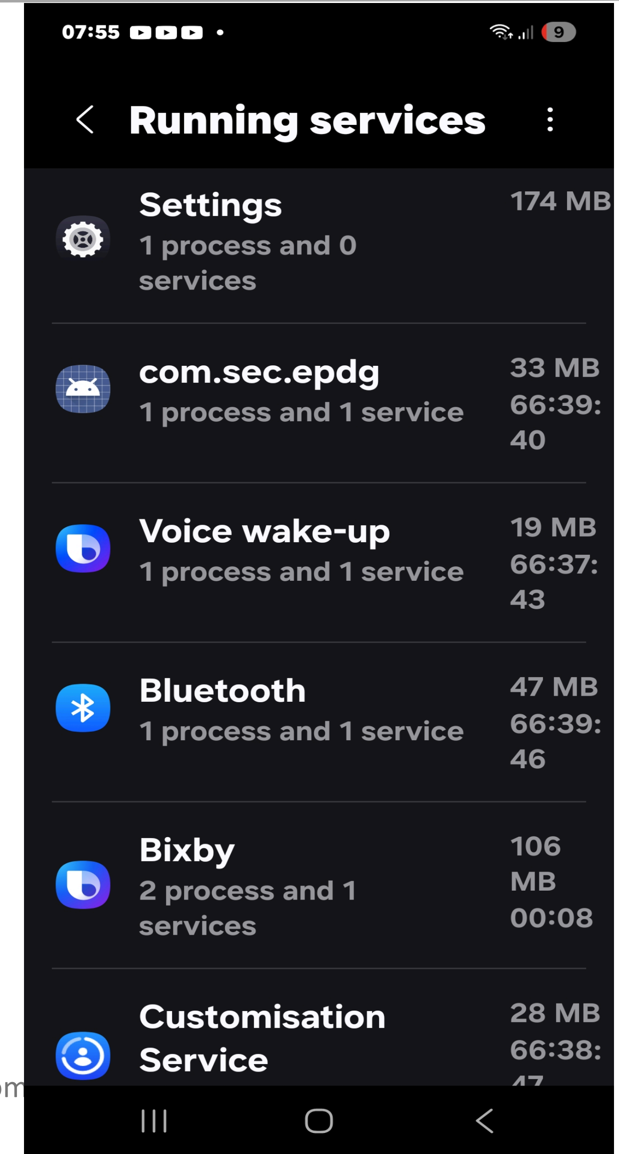
- In other algorithms

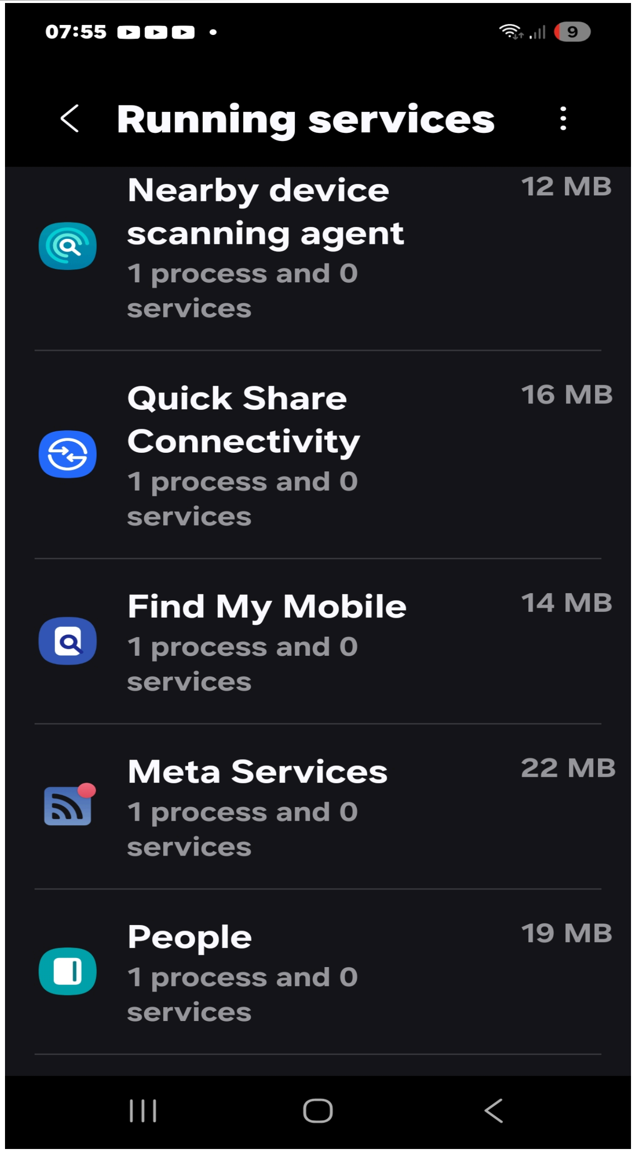
Some algorithms that need to discover data items based on high priority! Eg. Graph algorithms

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









How do we implement PQs?

- Option-1: Unsorted Linked List

	Unsorted LL
insert	Θ(1)
removeMin	Θ(N)
Min	Θ(N)

- What if we keep items in sorted order?

	Unsorted LL	Sorted LL
insert	Θ(1)	Θ(N)
removeMin	Θ(N)	Θ(1)
Min	Θ(N)	Θ(1)

How do we implement PQs?

- Option-2: Arrays

	Unsorted LL	Sorted LL	Array
insert	Θ(1)	Θ(N)	Θ(1)
removeMin	Θ(N)	Θ(1)	Θ(N)
Min	Θ(N)	Θ(1)	Θ(N)

  - What if the array is sorted?

	Unsorted LL	Sorted LL	Array	Sorted Array
insert	Θ(1)	Θ(N)	Θ(1)	Θ(N)
removeMin	Θ(N)	Θ(1)	Θ(N)	Θ(1)
Min	Θ(N)	Θ(1)	Θ(N)	Θ(1)

How do we implement PQs?

- Option-3: Balanced BSTs

	Unsorted LL	Sorted LL	Array	Sorted Array	AVL
insert	Θ(1)	Θ(N)	Θ(1)	Θ(N)	Θ(logN)
removeMin	Θ(N)	Θ(1)	Θ(N)	Θ(1)	Θ(logN)
Min	Θ(N)	Θ(1)	Θ(N)	Θ(1)	Θ(logN)

- Option-4: Hash Table

	Unsorted LL	Sorted LL	Array	Sorted Array	AVL	Hash Table
insert	Θ(1)	Θ(N)	Θ(1)	Θ(N)	Θ(logN)	Θ (1).
removeMin	Θ(N)	Θ(1)	Θ(N)	Θ(1)	Θ(logN)	Θ(N)
Min	Θ(N)	Θ(1)	Θ(N)	Θ(1)	Θ(logN)	Θ(N)

We need a better performance!

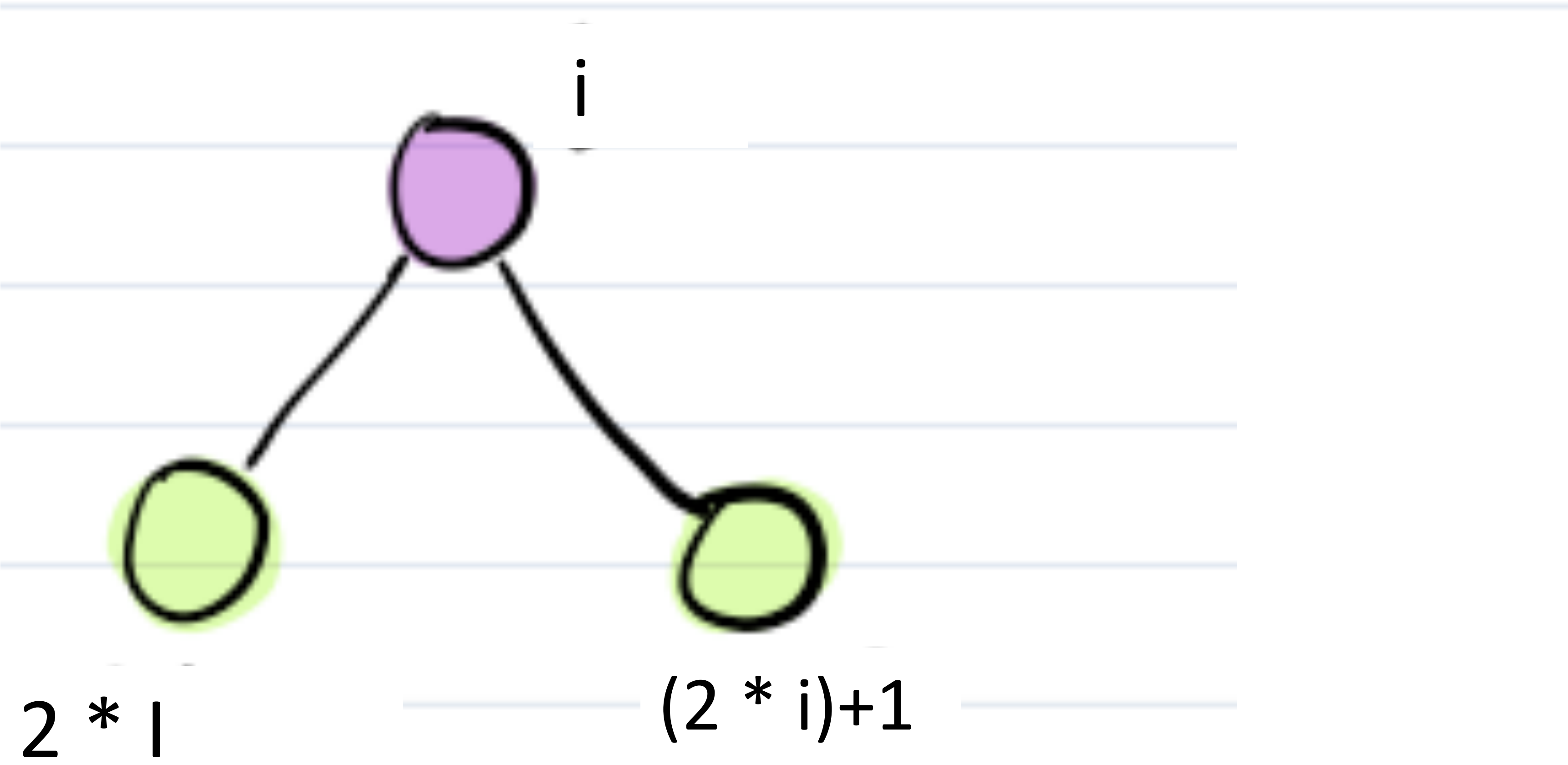
We need to do better for both insert() and removeMin() both!

Introducing a heap!

- Structural constraint!
  - The values must be modelled as a COMPLETE TREE.

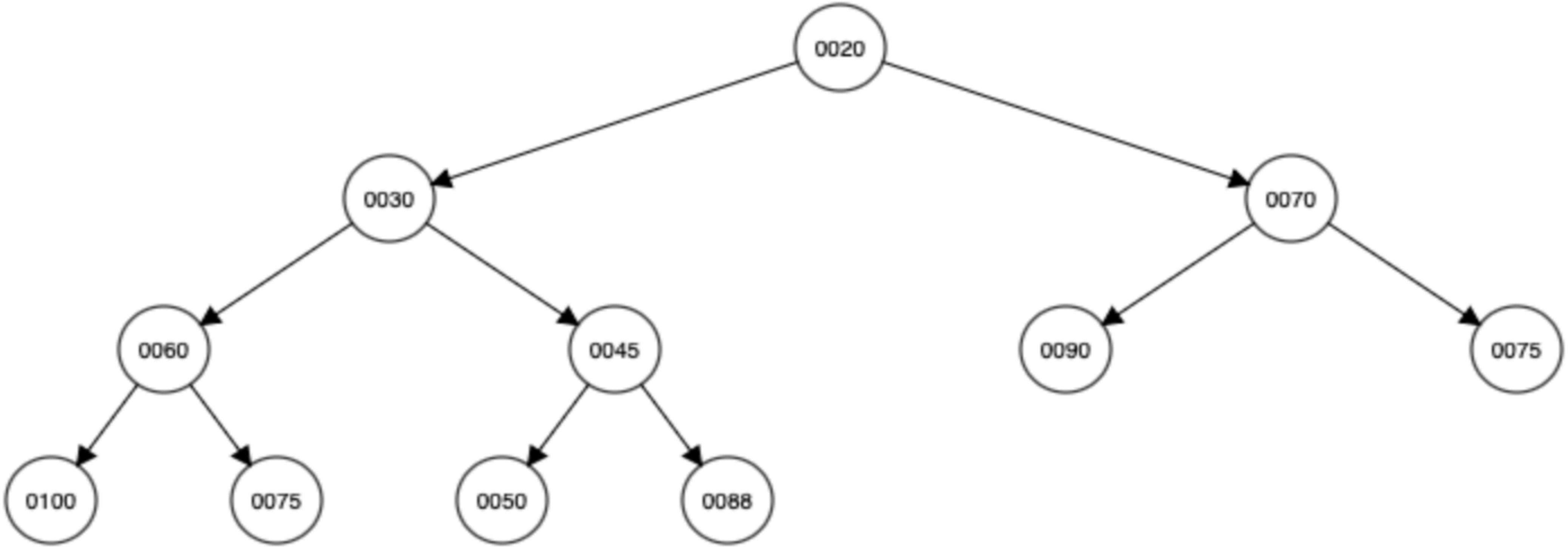
- Value Constraint
  - A child node must not have a key value that is smaller (larger) than its parent's key value to form a min (max) **heap**

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!