

# CSE 12 – Basic Data Structures and Object-Oriented Design Lecture 20

Greg Miranda & Paul Cao, Winter 2021

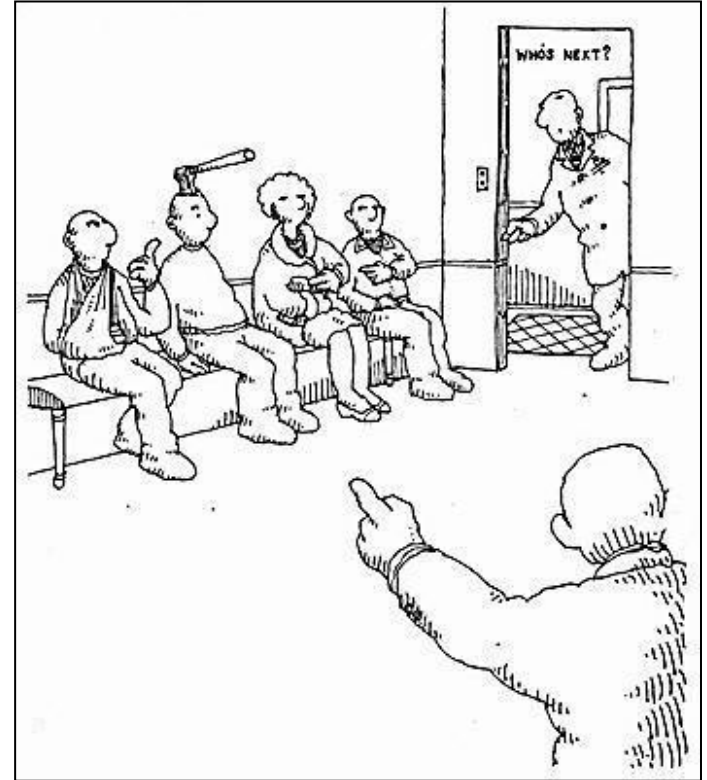
This lecture is being recorded

# Announcements

- Quiz 20 due Monday @ 8am
  - No class on Friday, only the exam
- Survey 8 due Friday @ 11:59pm
- PA7 due next Tuesday (3/2) @ 11:59pm
- Exam 2 – Released Friday @ 8am, due Saturday @ 10am
  - 90 minutes once you start - see Piazza post for more details

# Priority Queue ADT

- Emergency Department waiting room operates as a priority queue
- Patients sorted according to seriousness, NOT how long they have waited



# Implementing a Priority Queue

Which of the following best describes what is true about the implementation of a Priority Queue?

- A. There is only one correct way to implement a Priority Queue
- B. There are many correct ways to implement a Priority Queue, and they are all equally good (assuming they implement the correct behavior)
- C. There are many correct ways to implement a Priority Queue, but they vary in their efficiency (about how long it takes to do basic operations).

# Priority queue implementation options

- **Sorted array**

- Always insert based on the priority sorted order
- Remove from the front

- **Unsorted linked list**

- Insert new element in front
- Remove by searching list for highest-priority item

- **Sorted linked list**

- Always insert new elements where they go in priority-sorted order
- Remove from front

# Sorted Array

- **Add**
  - Need to step through the array to find where item goes in priority-sorted order. Also need to shift things back
- **Remove/peek**
  - Easy to find item you are looking for (first in list)



from towardsdatascience.com

# Unsorted linked list

- **Add**
  - Just throw it in the list at the front
- **Remove/peek**
  - Hard to find item the highest priority item—could be anywhere

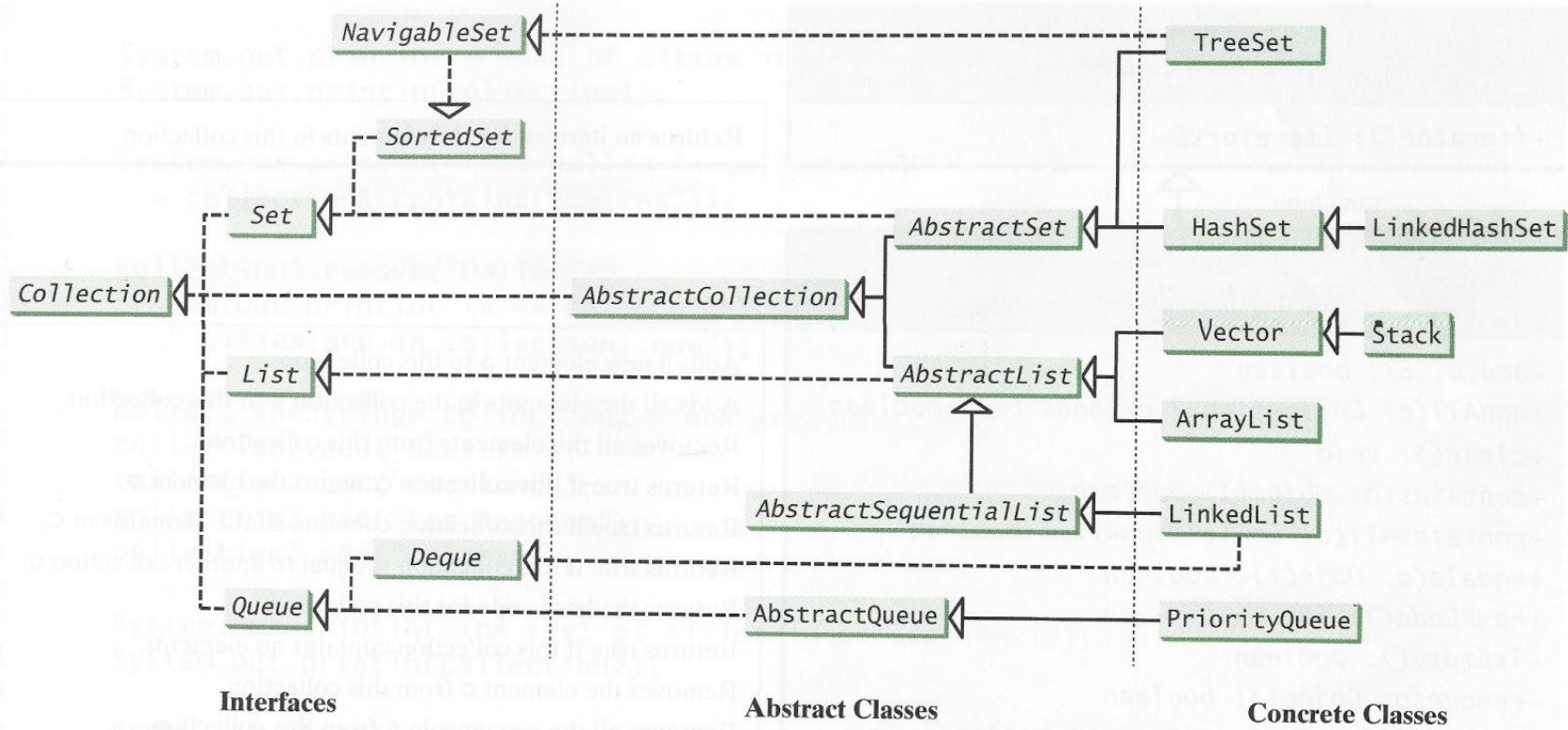


# Sorted linked list

- **Add**
  - Need to step through the list to find where item goes in priority-sorted order
- **Remove/peek**
  - Easy to find item you are looking for (first in list)







## Priority queue implementations

# We want the best of all

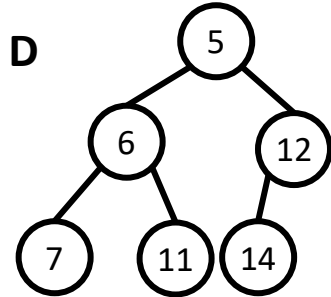
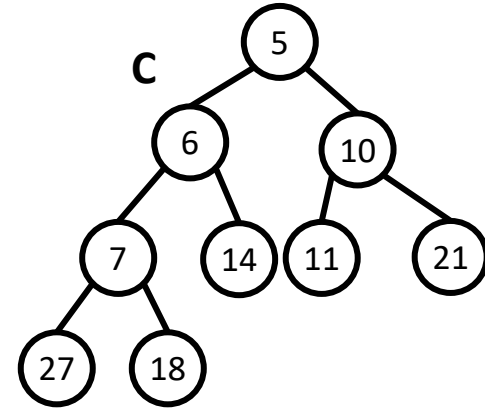
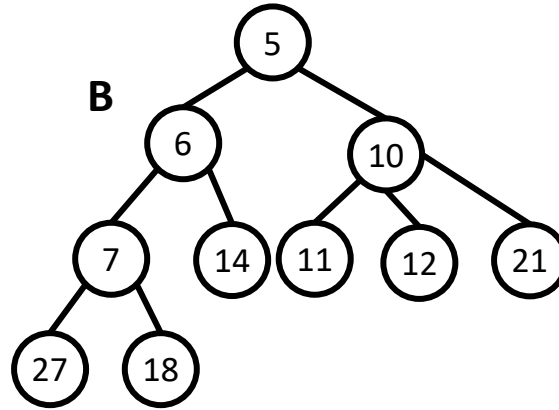
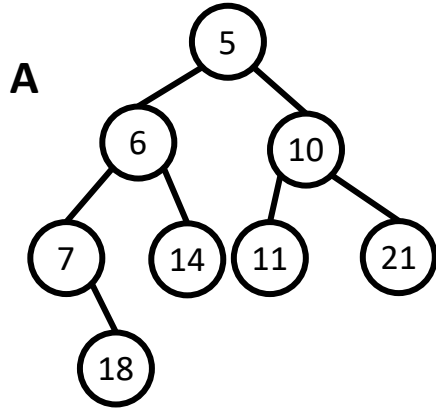
- Fast add AND fast remove/peek
- We will investigate a new data structure called a "heap" as a way to do this

Worksheet: Simulation of all implementations

# Heaps

- Heaps are **one kind** of binary tree
- They have a few special restrictions, in addition to the usual binary tree requirements:
  - Must be **complete**
  - Ordering of data must obey **heap property**
    - Min-heap version: a parent's data is always  $\leq$  its children's data
    - Max-heap version: a parent's data is always  $\geq$  its children's data

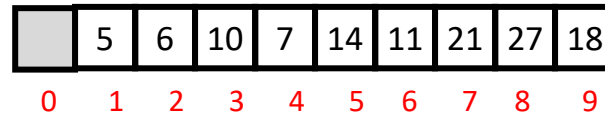
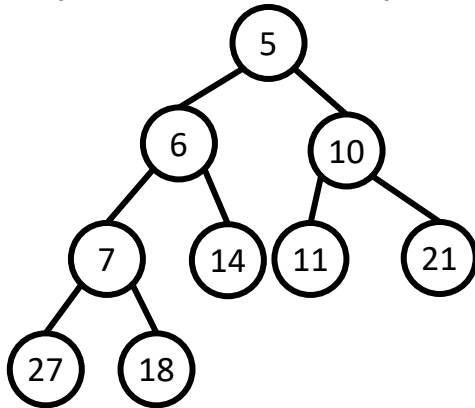
Which of the following are valid heaps?



**E.** More than one is valid

# Heap in an array (vs a linked structure)

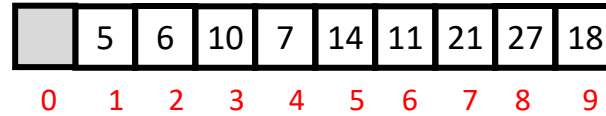
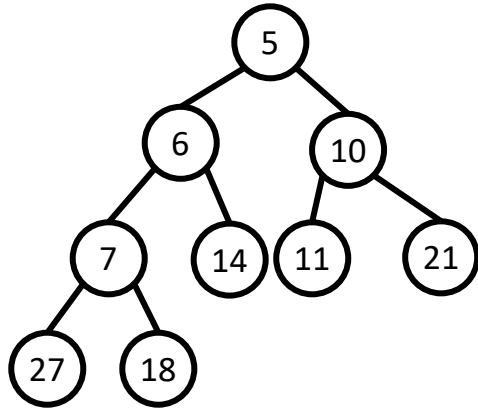
- We actually do NOT typically use a node object to implement heaps
- Because they must be **complete**, they fit nicely into an array, so we usually do that



size 9

Why are arrays better than linked lists?

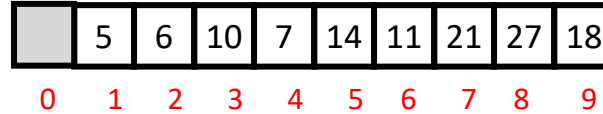
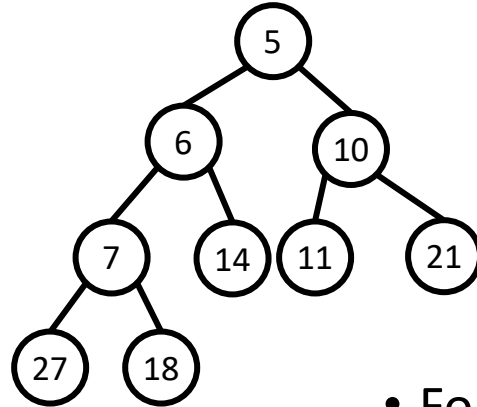
# Heap in an array, starting at index 1



size 9

- For a node in array index  $i$ :
  - Parent is at array index:
    - A.  $i - 2$
    - B.  $i / 2$
    - C.  $(i - 1) / 2$
    - D.  $2i$

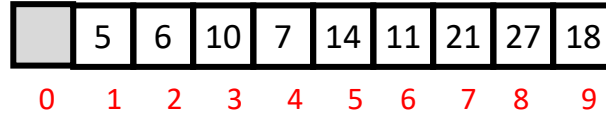
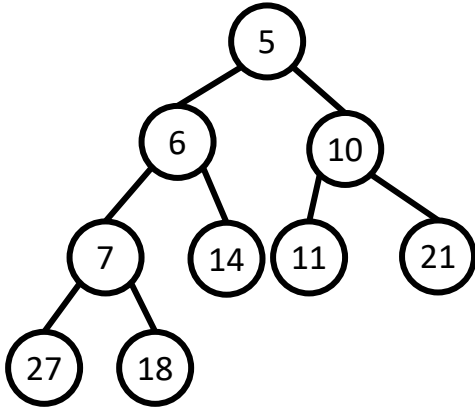
# Heap in an array, starting at index 1



size 9

- For a node in array index  $i$ :
  - Left child is at array index:
    - A.  $i + 1$
    - B.  $i + 2$
    - C.  $2i$
    - D.  $2i + 1$

# Heap in an array, starting at index 1



size 9

- For a node in array index  $i$ :
  - Parent is at array index:  $i / 2$
  - Left child is at array index:  $2i$
  - Right child is at array index:  $2i + 1$

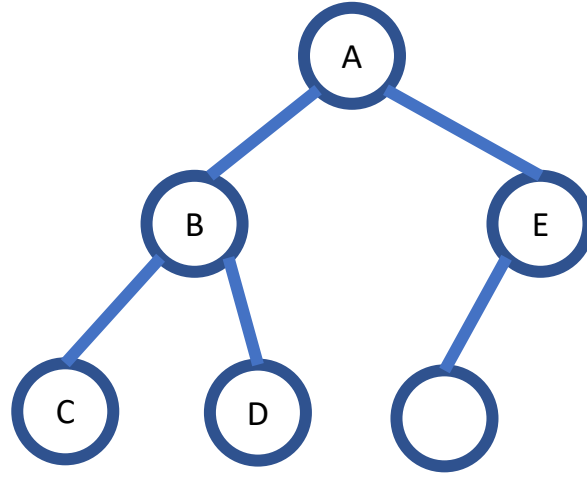


# Heap

Can you draw the heap structure based on the following array representation?

0	1	2	3	4	5	6	7	8	9
	3	6	5	8	9	12	14	11	10

Given the heap structure, which node corresponds to the red location in the array?



0	1	2	3	4	5	6

Who is the direct parent of the red cell?

0	1	2	3	4	5	6	7	8	9	10
			C	A	B	D		E		

Who are the children of the red cell?

0	1	2	3	4	5	6	7	8	9	10

- A. cell 9 and 10
- B. cell 10 only
- C. cell 6 and 7
- D. cell 8 and 9
- E. none of the above

# Max Heap (draw the picture and array)

- Assume the key and value are identical for this example
- Draw the picture and the array for the following:
  - Add the following elements to the max heap (in this order):
    - 5, 10, 15, 25, 30, 35, 40
  - Call poll() twice
    - What elements were returned?

Questions on Lecture 20?