# Lecture 17

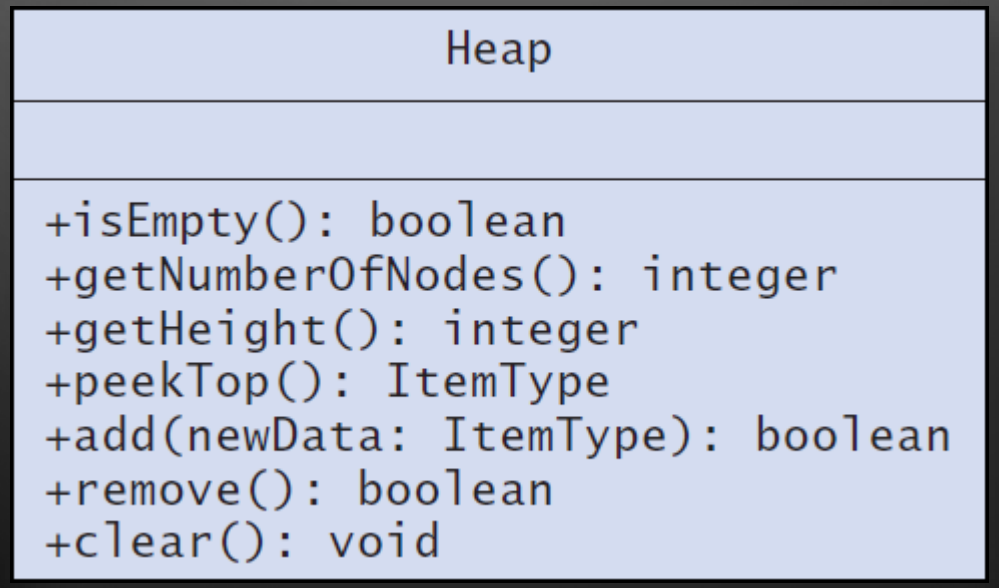
HEAPS

# Outline

- The ADT Heap
- Heap Implementation of the ADT Priority Queue
- Array-based Implementation of a Heap
- Heap Sort

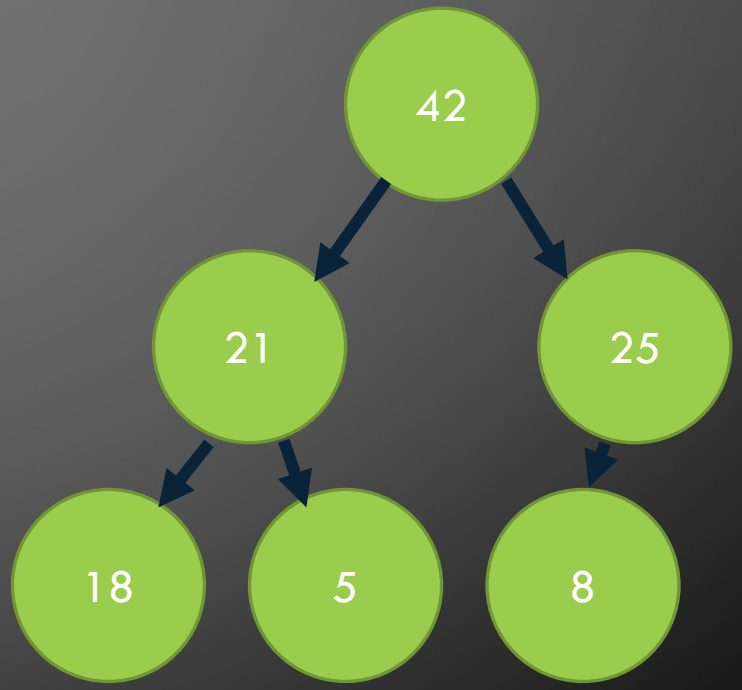
# The ADT Heap

- A heap is a complete binary tree that either is
  - Empty or …
  - Whose root contains a value ≥ each of its children and has heaps as its subtrees
- It is a special binary tree … different in that
  - It is ordered in a weaker sense
  - it will always be a complete binary tree



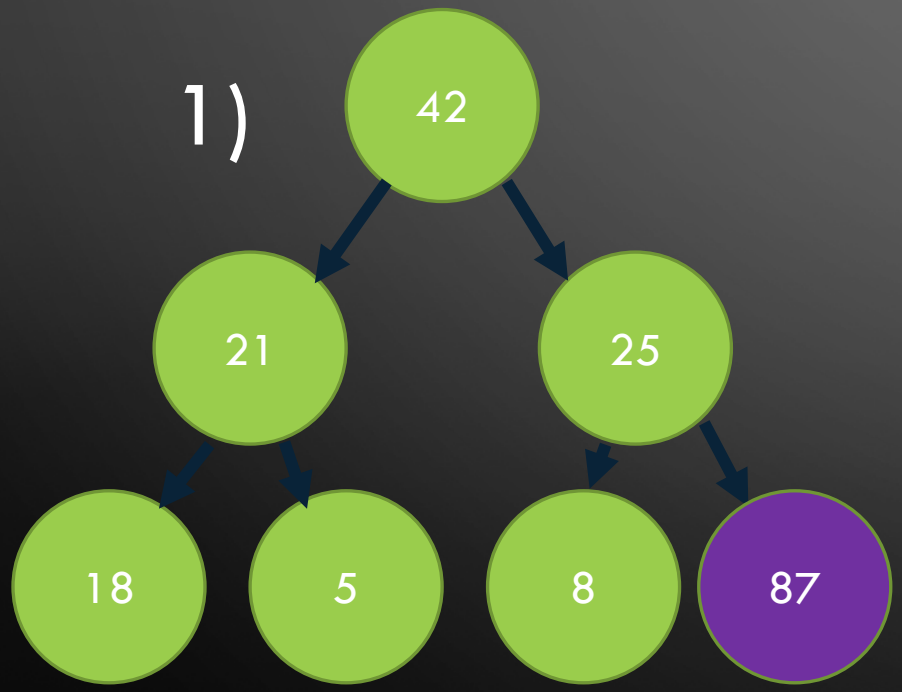
# Max Heap

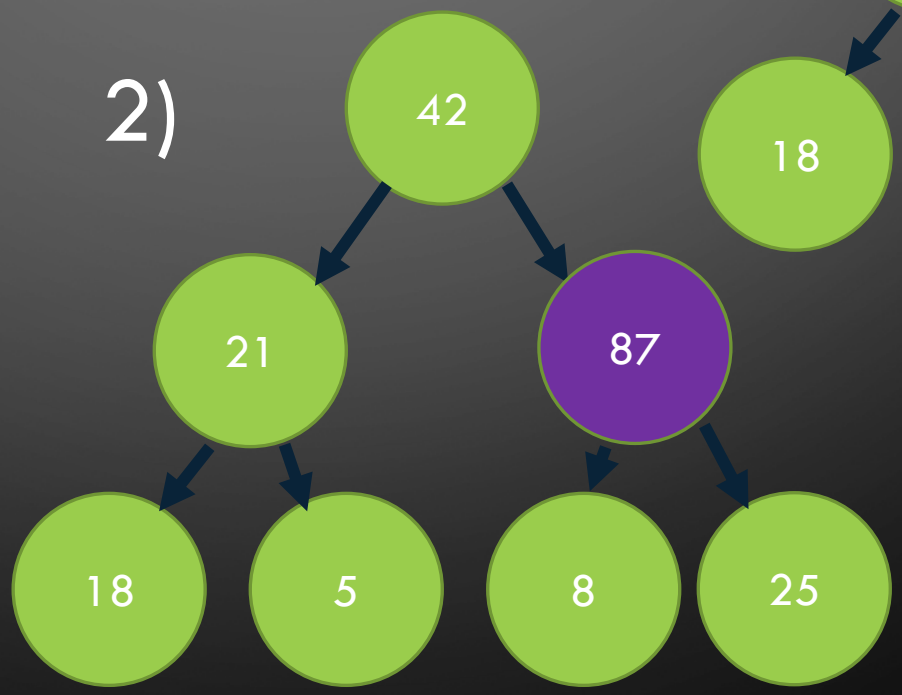
- Each parent node is larger than its children
- Only the next-in-line can be removed
- Min Heap has the opposite rule: each parent is smaller than its children

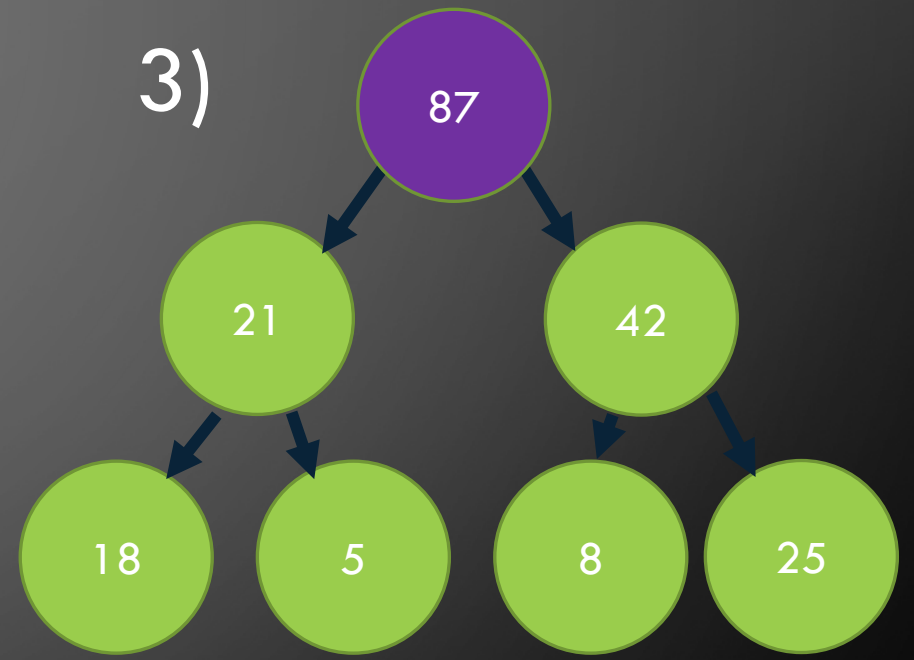


# Max Heap: Push (insert)

1. Add the node to the next empty spot in the structure
2. Bubble up, exchanging with parent if it is larger than the parent
3. Repeat Step 2 as needed…



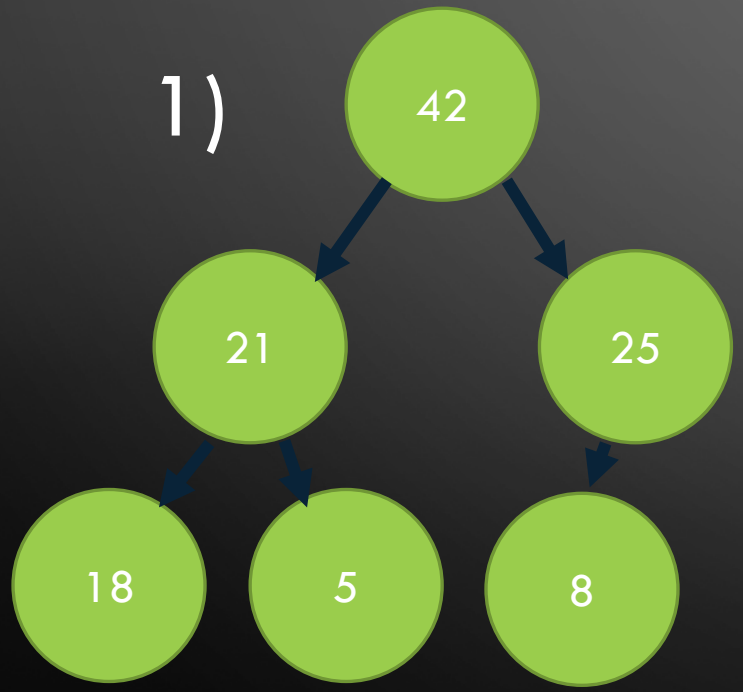



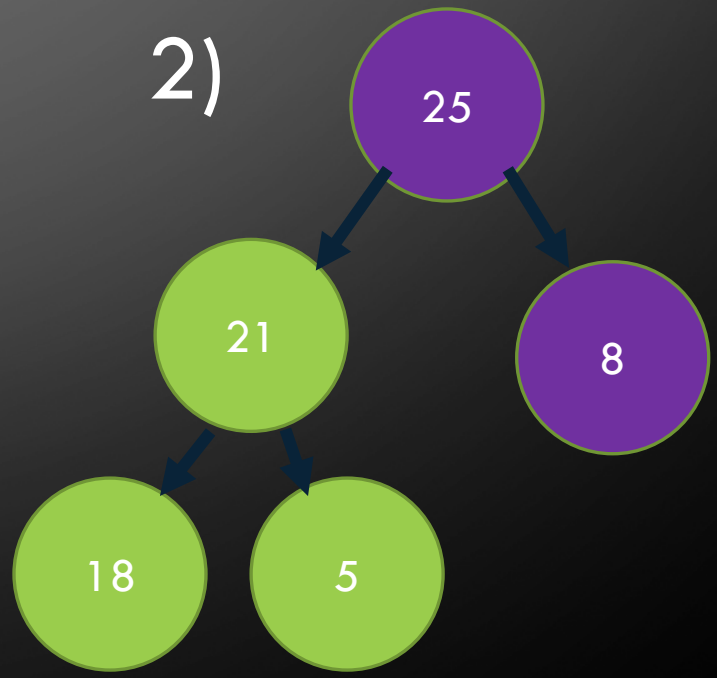
# Max Heap: Pop

- Remove root
- Replace root with “last node”
- Exchange nodes with largest child as much as possible

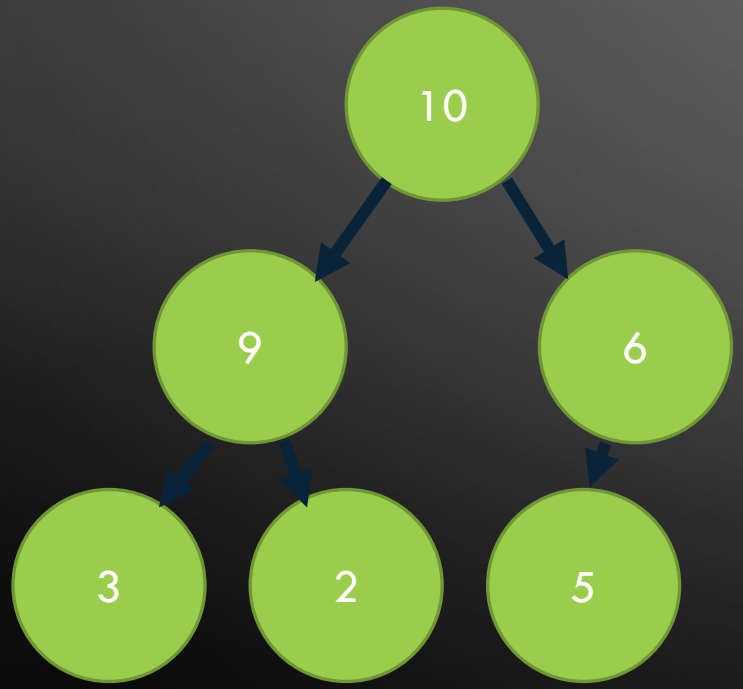
1)

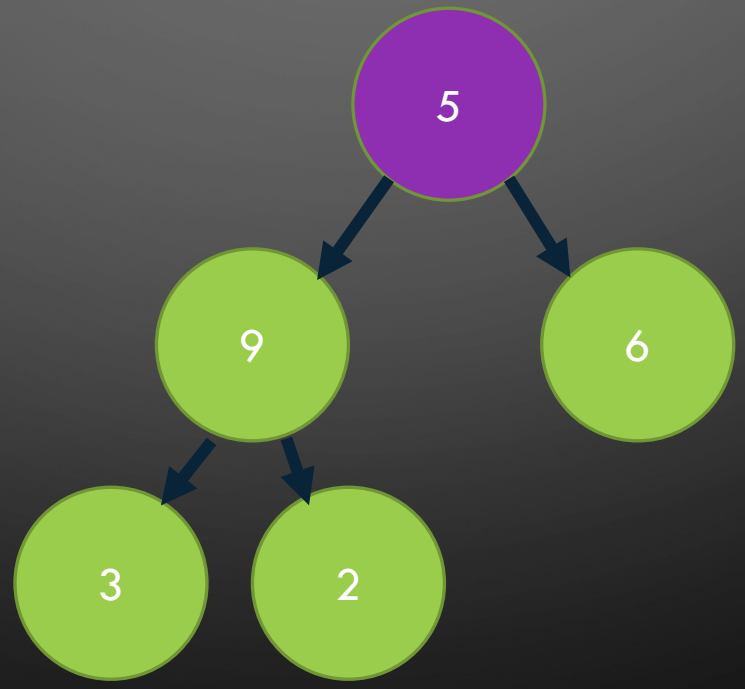


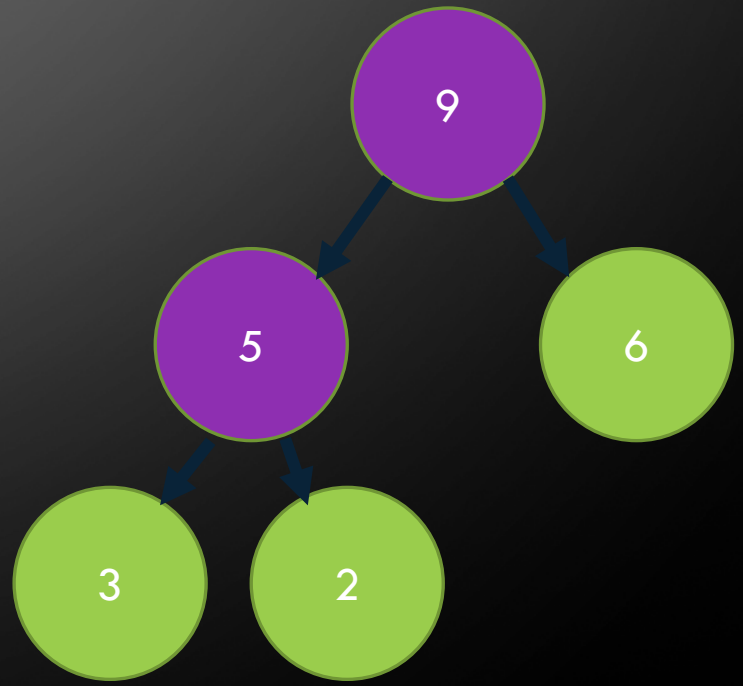

2)



# Second Pop Example



# Heap ADT

- A Heap is an ordered tree-based structure where the max (or min) object is stored at the root
  - Is this like anything we have seen?

| PriorityQueue |
| --- |
| items |
| PriorityQueue(); //constructor<br>~PriorityQueue(); //destructor<br>isEmpty();<br>push(item);<br>pop();<br>top(); //look at next record |

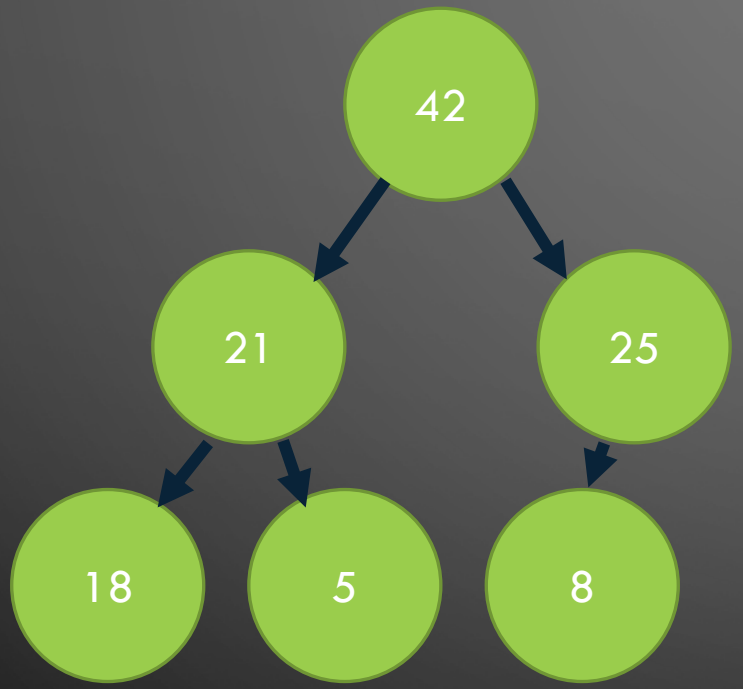
- A Heap is not a priority queue…but
- A heap can represent the most efficient implementation of a priority queue
- Sometimes priority queues are simply called heaps

# Heap Complexity

- Heap top() and isEmpty() are O(1)
- What is the complexity of push(item)?
- What is the complexity of pop()?

# Representing a Heap: Array

- Because a heap is a complete binary tree, it can be represented using an array




| 42 | 21 | 25 | 18 | 5 | 8 | | |
|---|---|---|---|---|---|---|---|

# Insert Psuedocode

- The parent of position k can be represented by k >>1

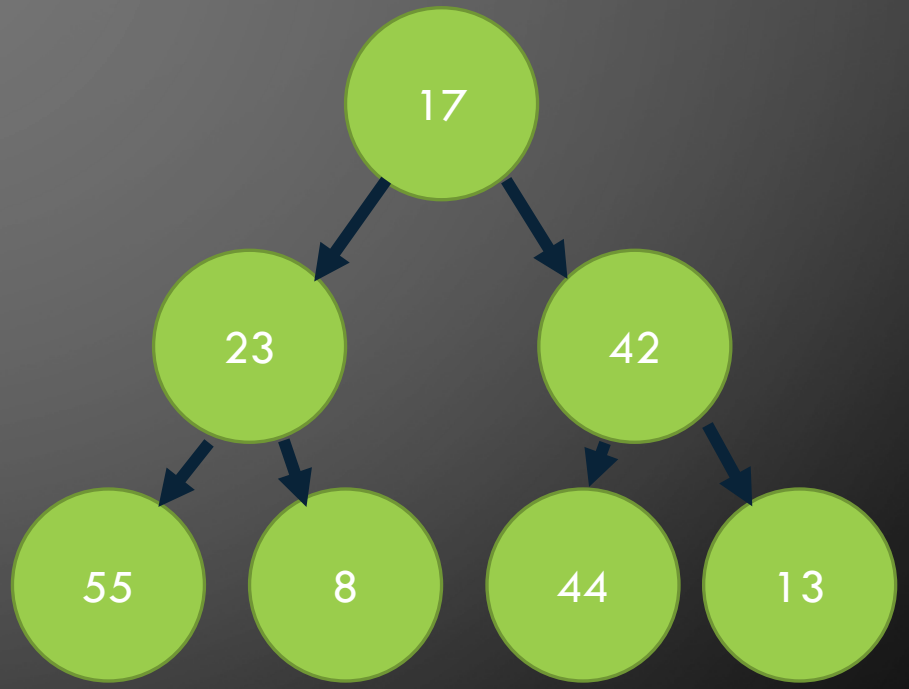
```
insert(ItemType item)
{
    k = heapsize + 1;
    j = k >> 1; //j is the parent
    //while the item is larger than its parent
    while( (j >= 1) and (array[j] < item) )
    {
        //replace current position with its parent
        array[k] = array[j];
        k = j;
        j = j >> 1; //move to next parent
    }
    array[k] = item;
    heapsize = heapsize + 1;
}
```

# Heap Sort

- Uses our new heap structure to sort an array

1. Start with an array, and then consider it an unsorted heap

| 17 | 23 | 42 | 55 | 8 | 44 | 13 |
|---|---|---|---|---|---|---|



# Heap Sort

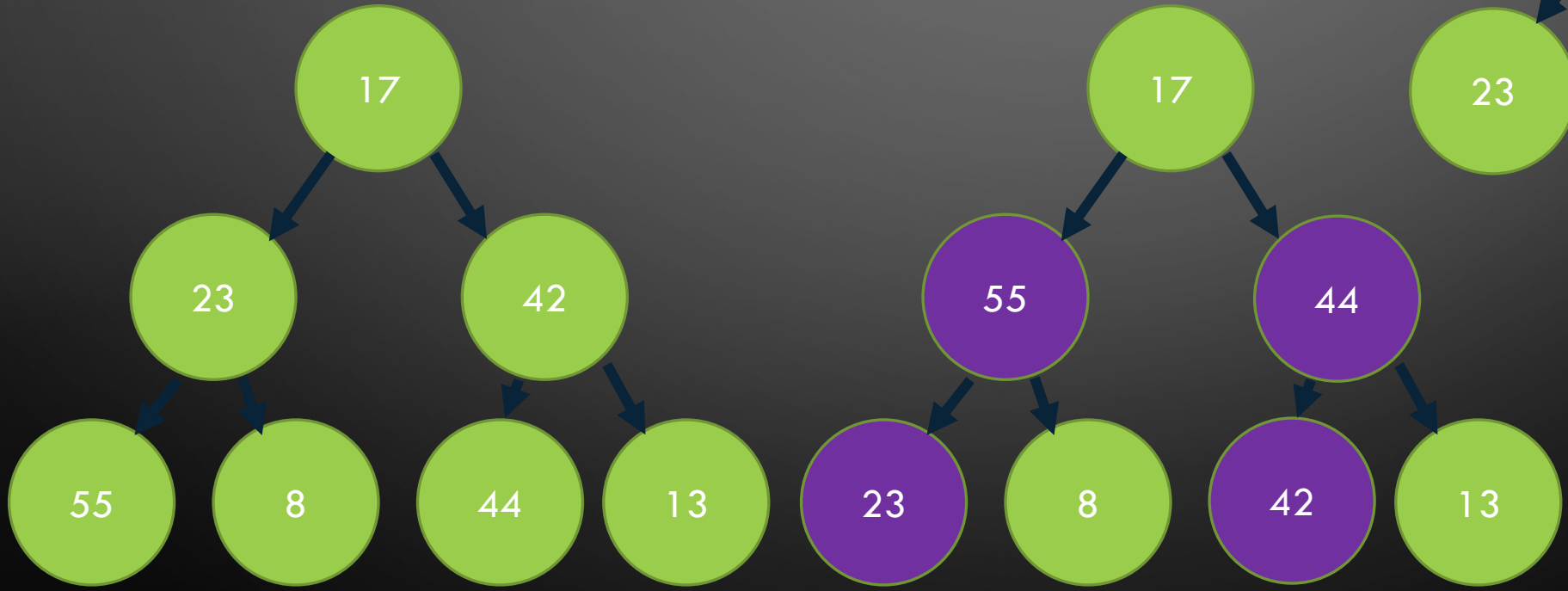
| 55 | 17 | 44 | 23 | 8 | 42 | 13 |
|---|---|---|---|---|---|---|

- Uses our new heap structure to sort an array

2. Sort (partially) into a max/min heap

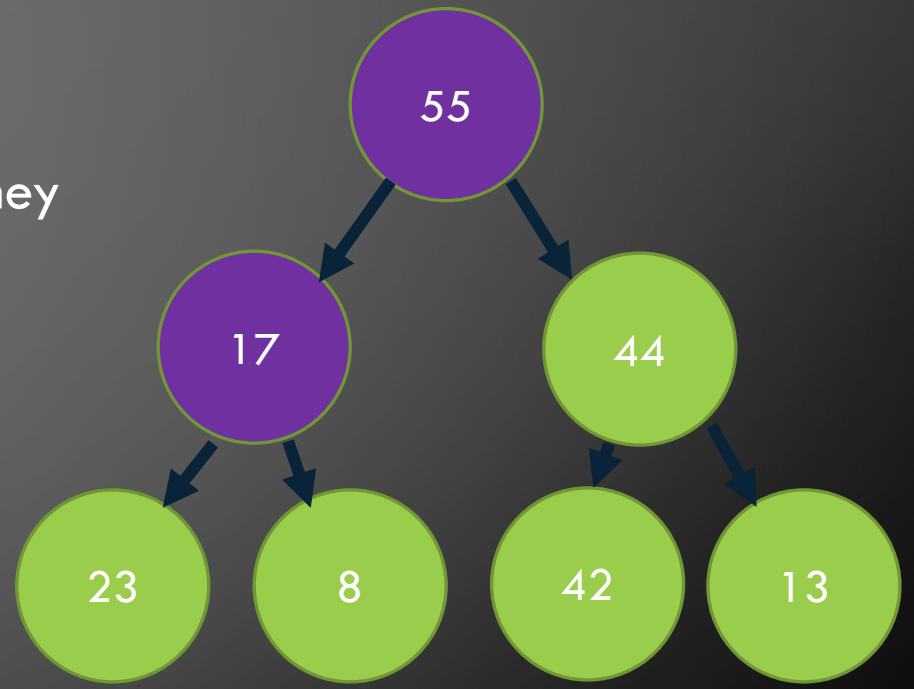
This is done by swapping each element with its children if they are out of place.

***Note that at the end, 17 is not sorted correctly***







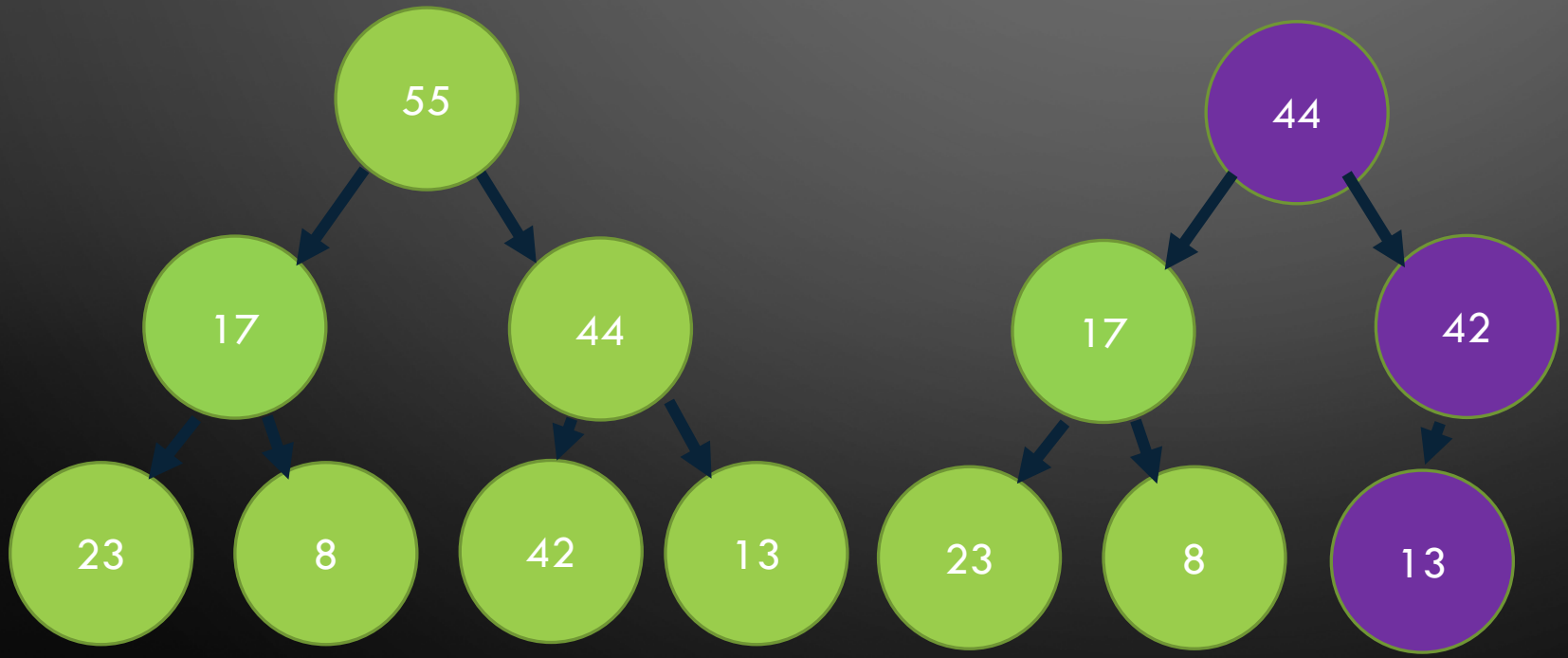


# Heap Sort


- Uses our new heap structure to sort an array

3. Remove root, keep in separate sorted array

***Promote largest child***

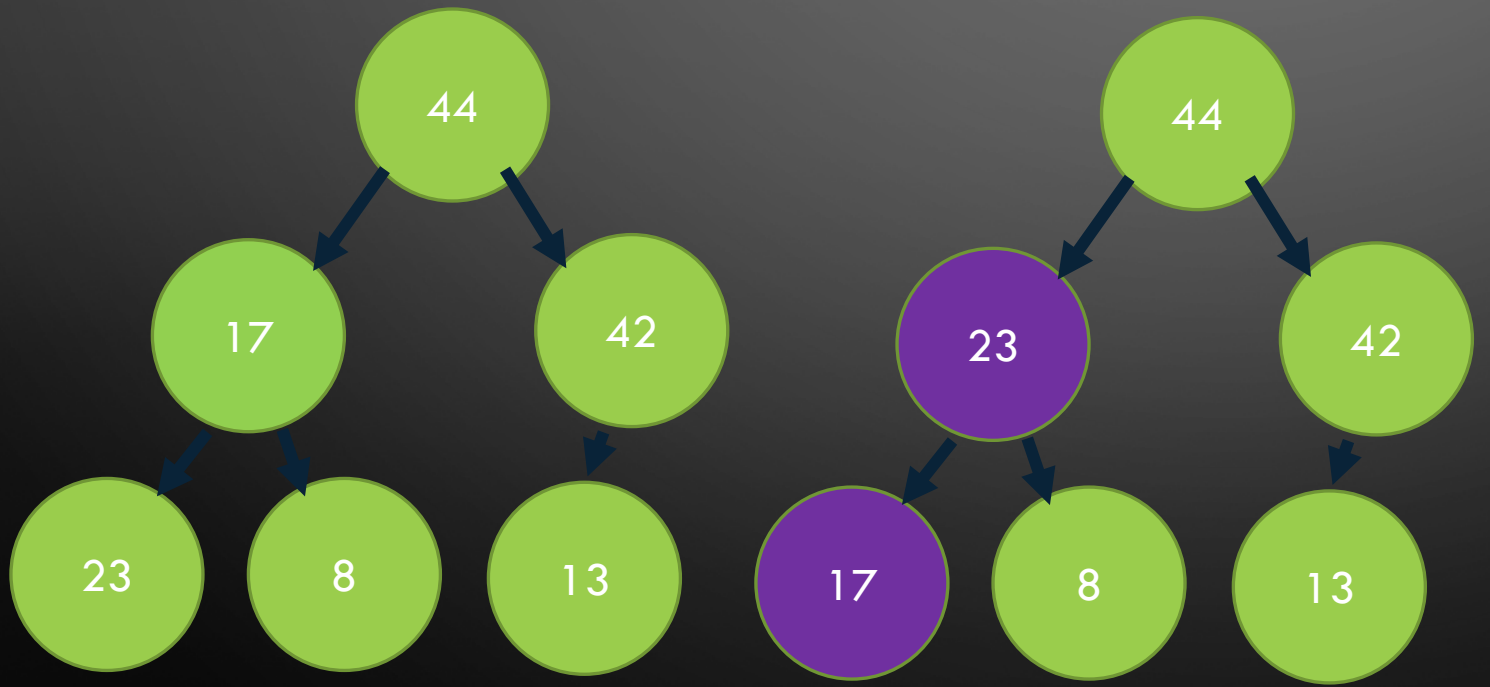


# Heap Sort

| 44 | 23 | 42 | 17 | 8 | 13 | 55 |
|---|---|---|---|---|---|---|

- Uses our new heap structure to sort an array

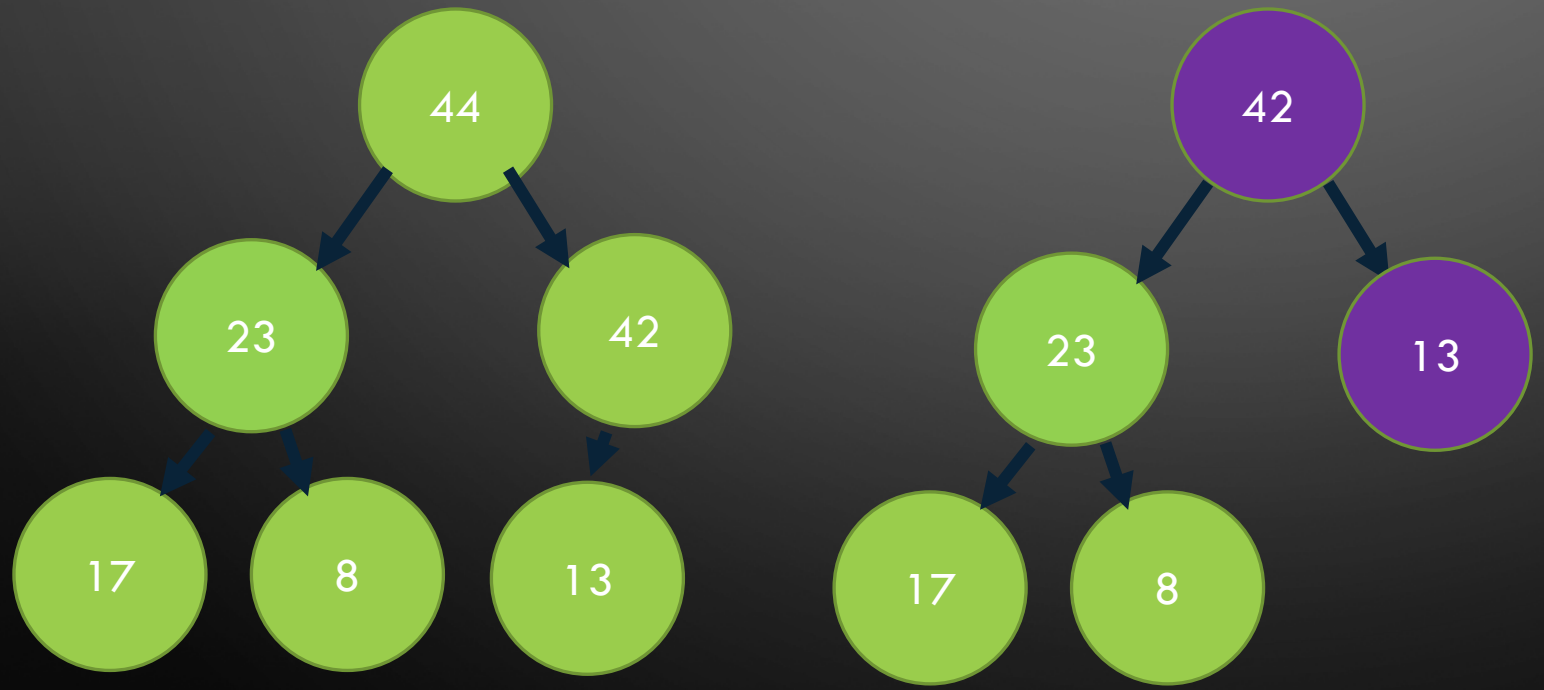
4. Repeat steps 2-3 until sorted



# Heap Sort

| 42 | 23 | 13 | 17 | 8 | 55 | 44 |
|---|---|---|---|---|---|---|

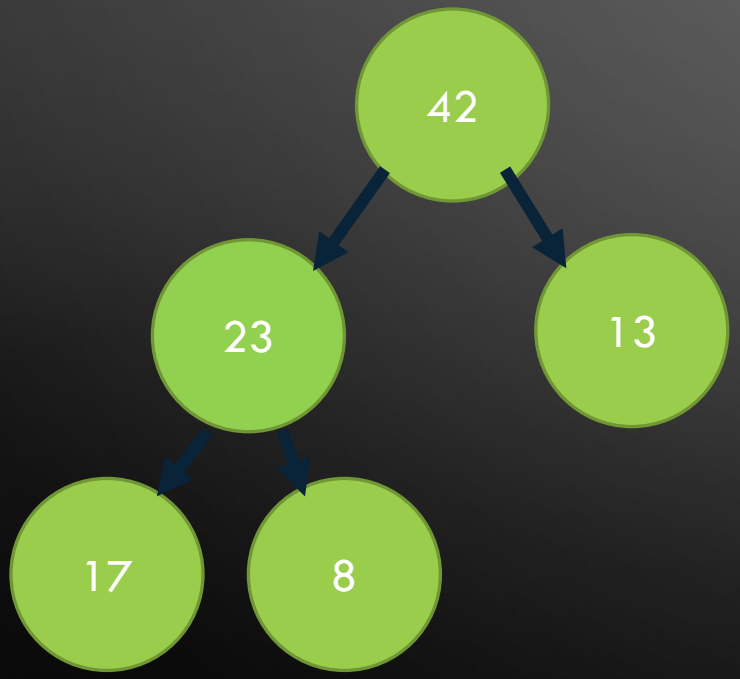
- Uses our new heap structure to sort an array

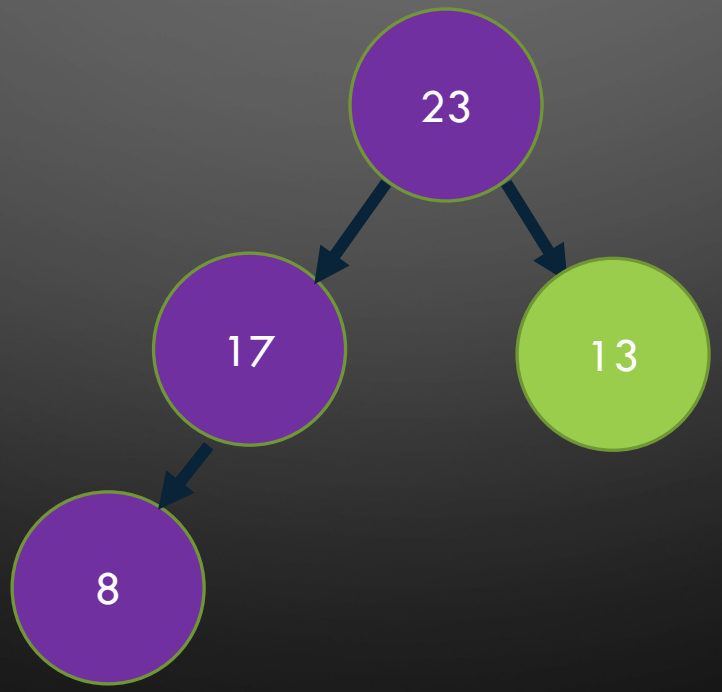


# Heap Sort

| 23 | 17 | 13 | 8 | 55 | 44 | 42 |
|---|---|---|---|---|---|---|

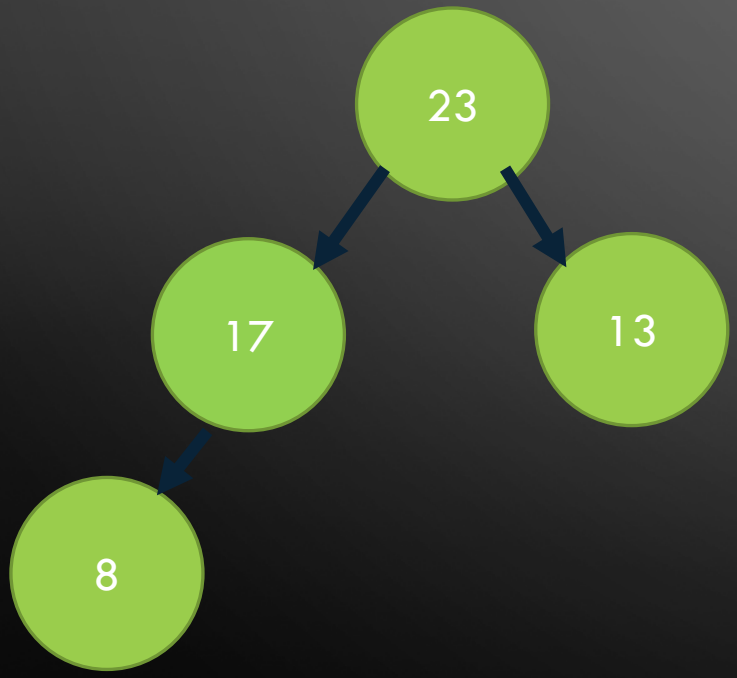
- Uses our new heap structure to sort an array

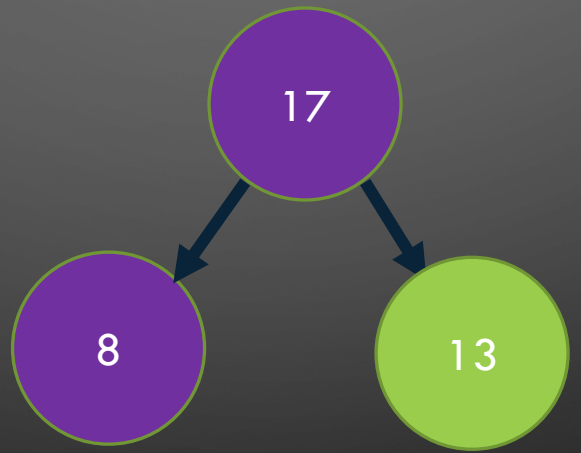


# Heap Sort


- Uses our new heap structure to sort an array



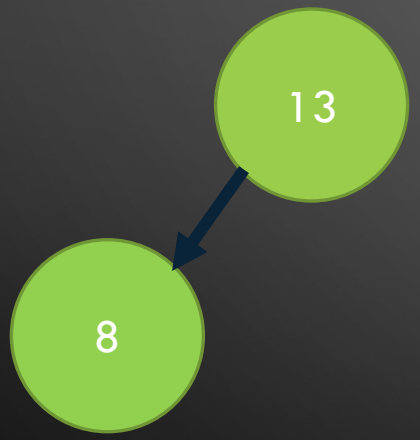
# Heap Sort


- Uses our new heap structure to sort an array

# Heap Sort

| 8 | 55 | 44 | 42 | 23 | 17 | 13 |
|---|---|---|---|---|---|---|

- Uses our new heap structure to sort an array




# Heap Sort

| 55 | 44 | 42 | 23 | 17 | 13 | 8 |
|---|---|---|---|---|---|---|

- Uses our new heap structure to sort an array

# Heap Sort Complexity

- **What is the time complexity of heap sort?**
- Average Case: O(n log n)
- Worst Case: O(n log n)
- **What is the space overhead of heap sort?**
- O(1)

# Assignment/Homework

- Reading pp. 537 -545, 547-587
- P4 (Courseweb) due on Today.
- ICE 11: BST (Courseweb) due on Thursday.
- Homework 8: BST (Gradescope) due on Thursday.