Heaps

A Tree-Based Data Structure

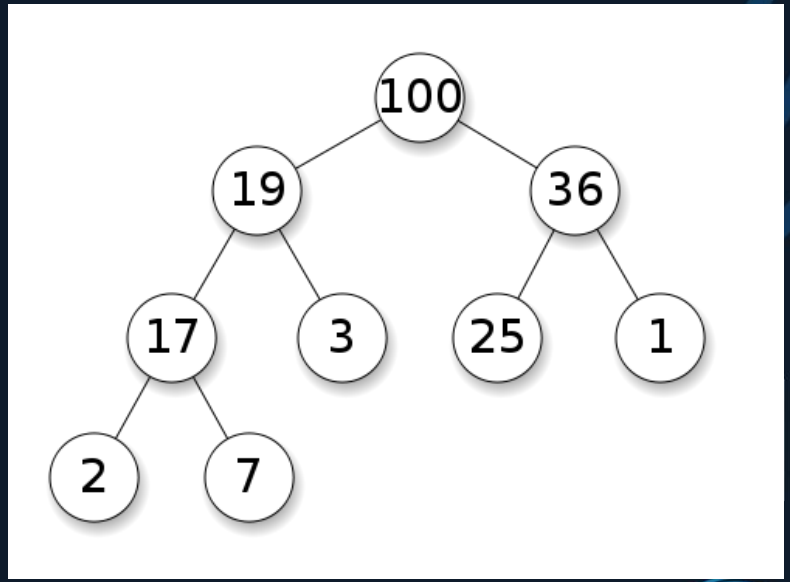
Contents

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

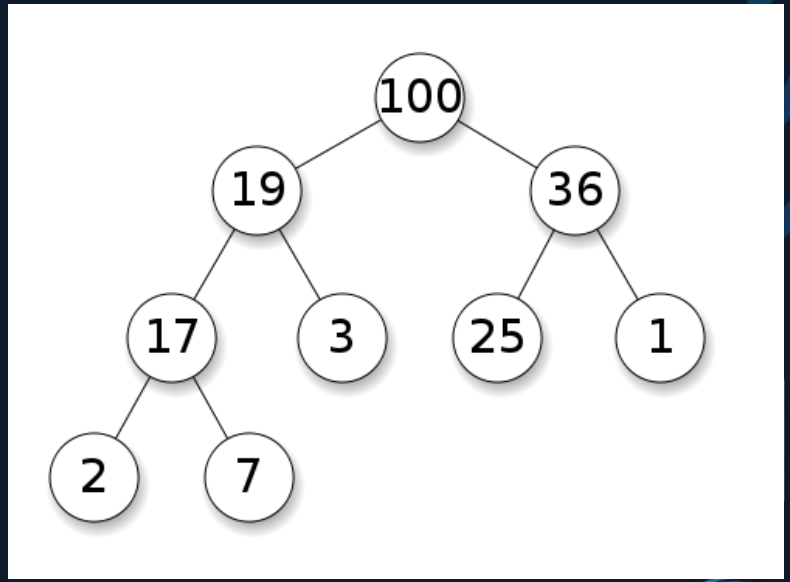
What is a heap?

- A tree-based data structure
  - All nodes are ordered based on the following rule:
    - the *min-heap property*: the value of each node is greater than or equal to the value of its parent, with the minimum-value element at the root.
    - the *max-heap property*: the value of each node is less than or equal to the value of its parent, with the maximum-value element at the root.
- Often stored in an array
  - Fixed size or Dynamic
  - Different types of heaps store information differently in the arrays



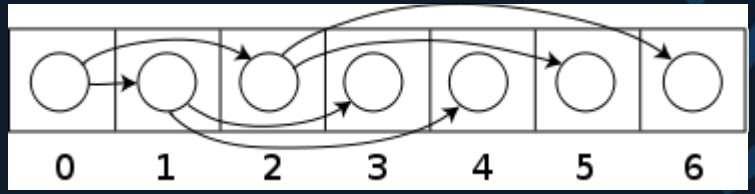
The Binary Heap

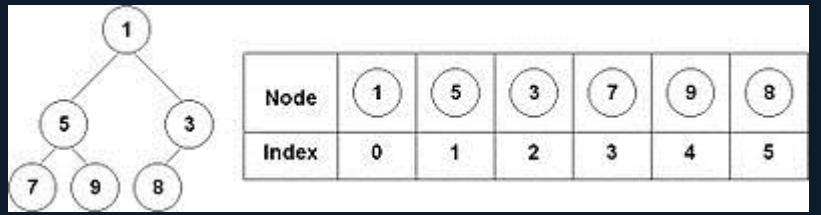
- The most common form of heap.
- Follows the following constraints:
  - No more than two children per node
  - All levels of the tree, except possibly the last one (deepest) are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right
  - All nodes are *either* ***greater than or equal to*** *or* ***less than or equal to*** *each of its children*



Binary Heap – How are they stored?

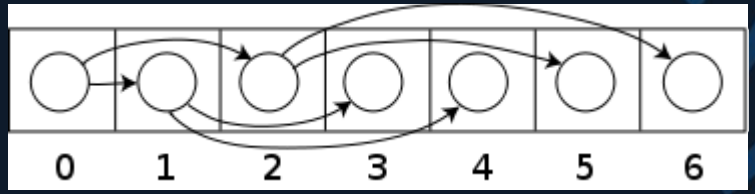
- Usually stored in an array for efficiency reasons.
  - Could also be stored in a tree.
- First index holds the first node.
  - Next two hold the children of the first node
  - Next four hold the children of the second and third nodes.
  - And so on….
- Allows for quickly calculating the location of nodes in the array.

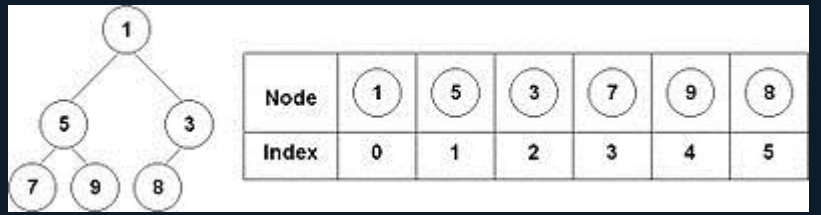




Binary Heap – How are they stored?

- When stored in an array, the parent and children of a node can be found at:
  - Parent:
    - $floor(\frac{(i-1)}{2})$
  - Children:
    - $1^{st}: 2i + 1$
    - $2^{nd}: 2i + 2$
- Where i is the index of the node in the array





aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Common uses for a heap?

- Great for storing data where the smallest (or largest) value needs to be easily accessible.

- Examples:
  - Calculating and storing the costs/benefits of different actions an AI could do.
    - Can then get the “best” action quickly from the top of the heap.
  - Proven most efficient implementation of a Priority Queue

Common operations on a Heap:

- Add/Insert/Push
  - Adds a new value to the heap
  - Requires the heap to be “rebalanced”
- Remove
  - Removes a value from the heap
  - Requires the heap to be “rebalanced”
- Clear
  - Removes all values from the heap
- Peek
  - Get the value of the root element of the tree
    - Either the lowest or highest value, depending on how the tree is sorted
- Pop
  - Remove the root element from the tree
    - Either the lowest or highest value, depending on how the tree is sorted

Common operations on a Heap:

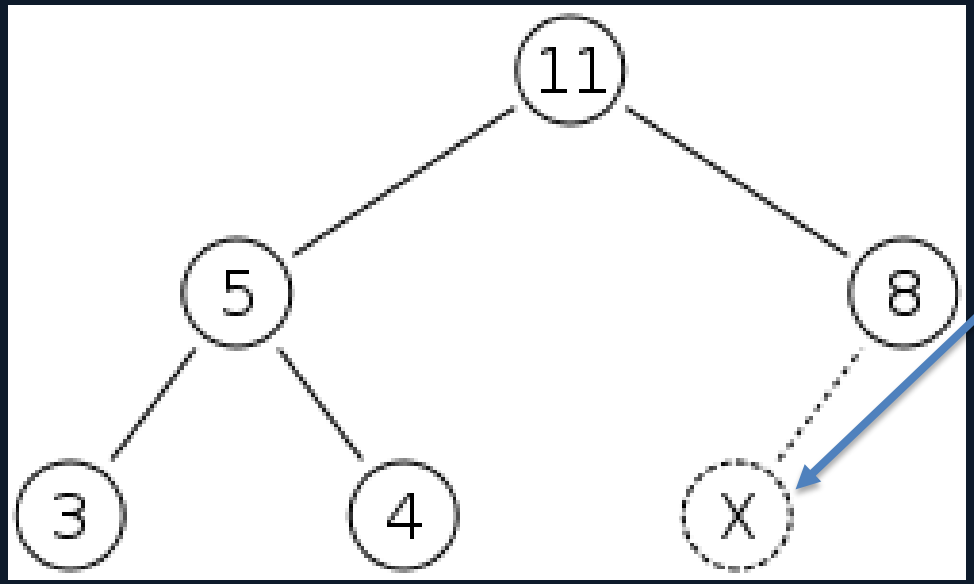
- Find
  - Searches the tree for a specific value. If an object is associated with the value, returns the object
- Size
  - Returns the number of values stored in the tree
- Iterators/Step functions
  - Some way in order to step through every value in correct order, and perform operations

Implementing the Push Function

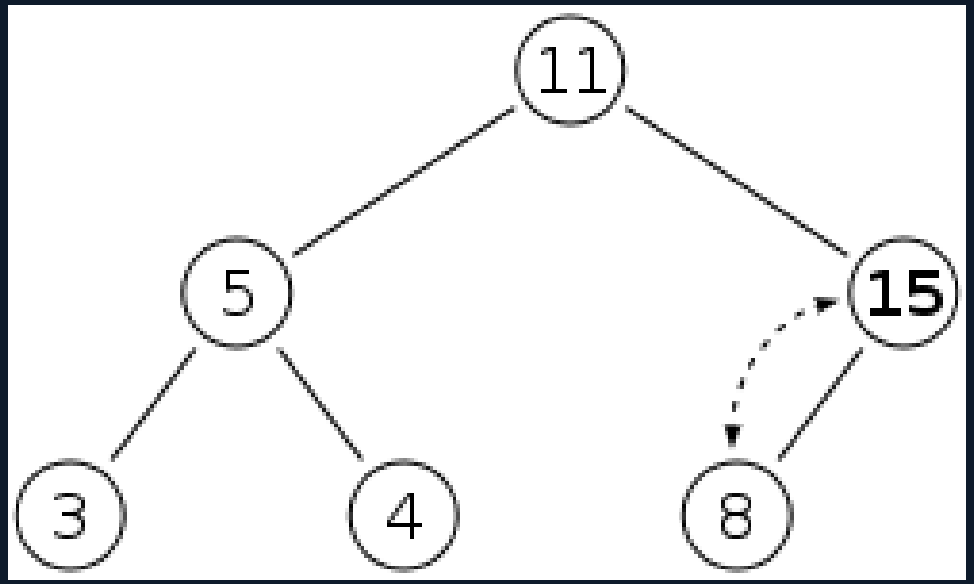
- Uses the “up-heap” operation

- Three steps required:
  - Add the value to the bottom of the tree.
  - Swap the value with its parent if the parent is smaller than the value (or larger, for a min-heap)
  - Repeat until no more swaps are required!

Implementing the Push Function



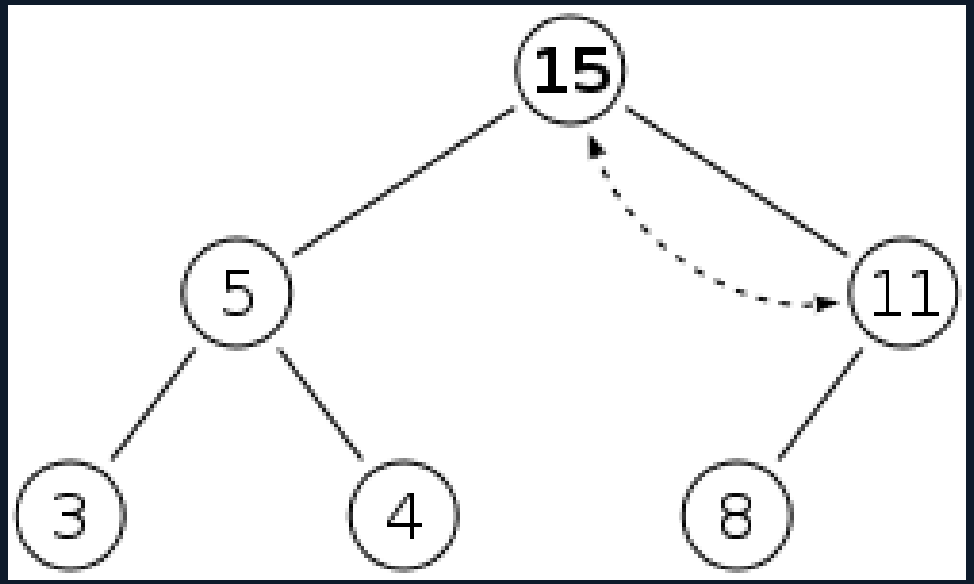
Implementing the Push Function



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Implementing the Push Function



Canberra Institute of Technology

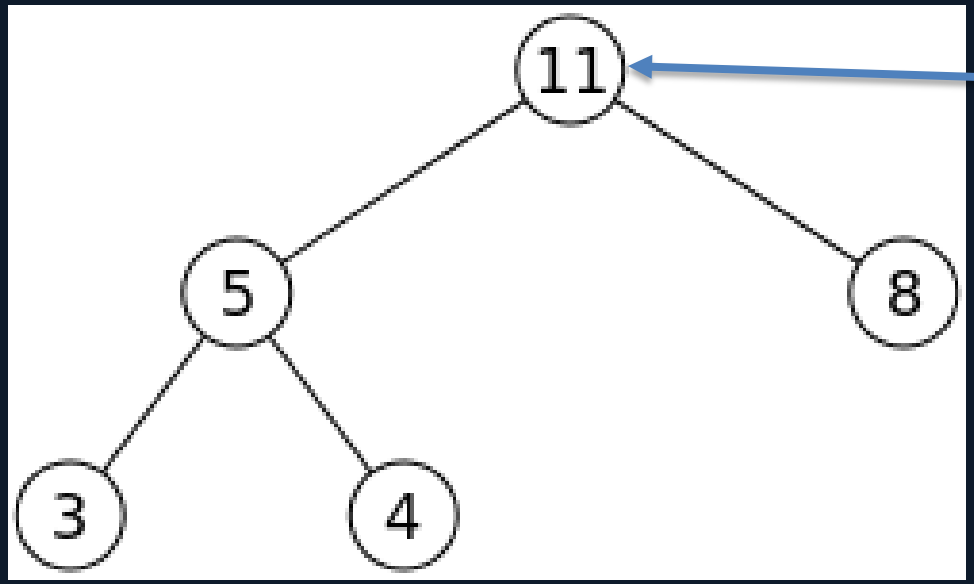
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Implementing the Remove/Pop Function

- Uses the “down-heap” operation

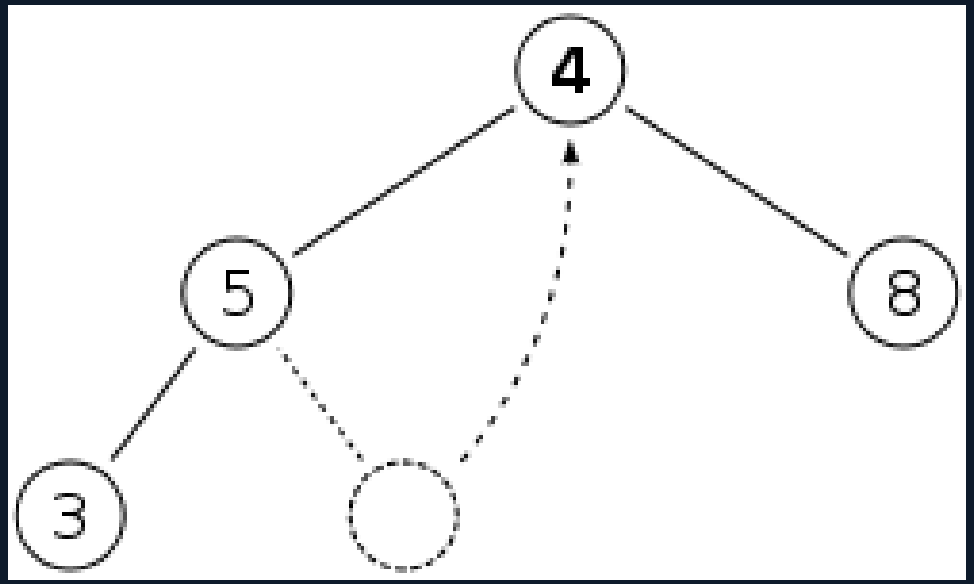
- Three steps required:
  - Replace the value to be removed with the last value stored in the tree
  - Swap the new value with its child if the parent is larger than the new value(or smaller, for a min-heap)
  - Repeat until no more swaps are required!

Implementing the Remove/Pop Function



Calling Pop to remove the 11

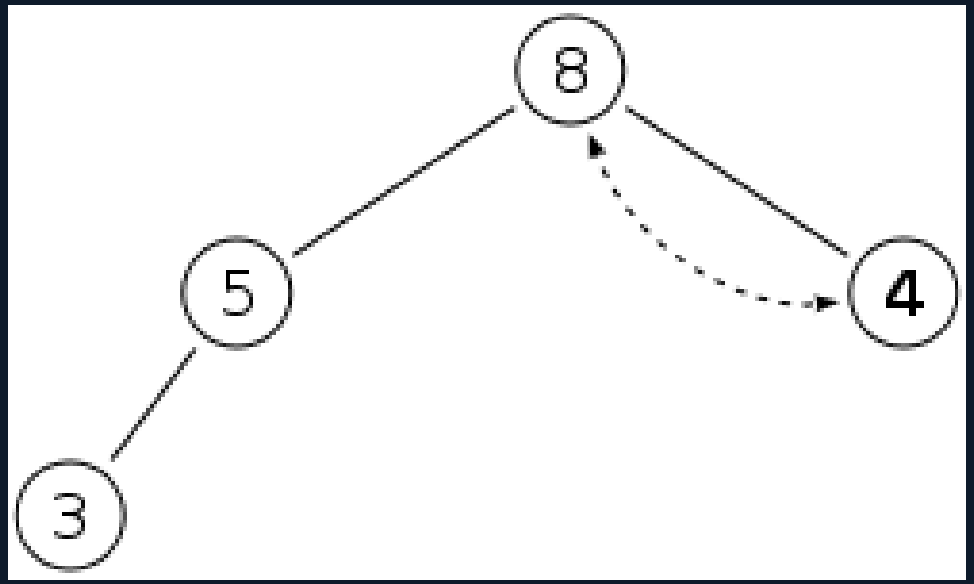
Implementing the Remove/Pop Function



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Implementing the Remove/Pop Function



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Speed of Common Operations

- Fast to find the smallest/largest value
  - O(1)
- Slow for searching, O(n)
- Quite good times for insert/delete
  - Optimal data structure for a priority queue

Binary Heap		
Type	Tree	
	Time complexity in big O notation	
	Average	Worst case
Space	O(n)	O(n)
Search	O(n)	O(n)
Insert	O(1)	O(log n)
Delete	O(log n)	O(log n)

Summary

- There are many different types of Heaps
  - Binary heaps are common, but not the only type!
- Heaps are a great way to store data if you only need it partially ordered.
- They can be complex to set up, due to how information is stored.

References

- Allen Sherrod, 2007. *Data Structures and Algorithms for Game Developers (Charles River Media Game Development)*. 1 Edition. Charles River Media.

- Thomas H. Cormen, 2001. *Introduction to Algorithms, Second Edition*. 2nd Edition. The MIT Press.

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX