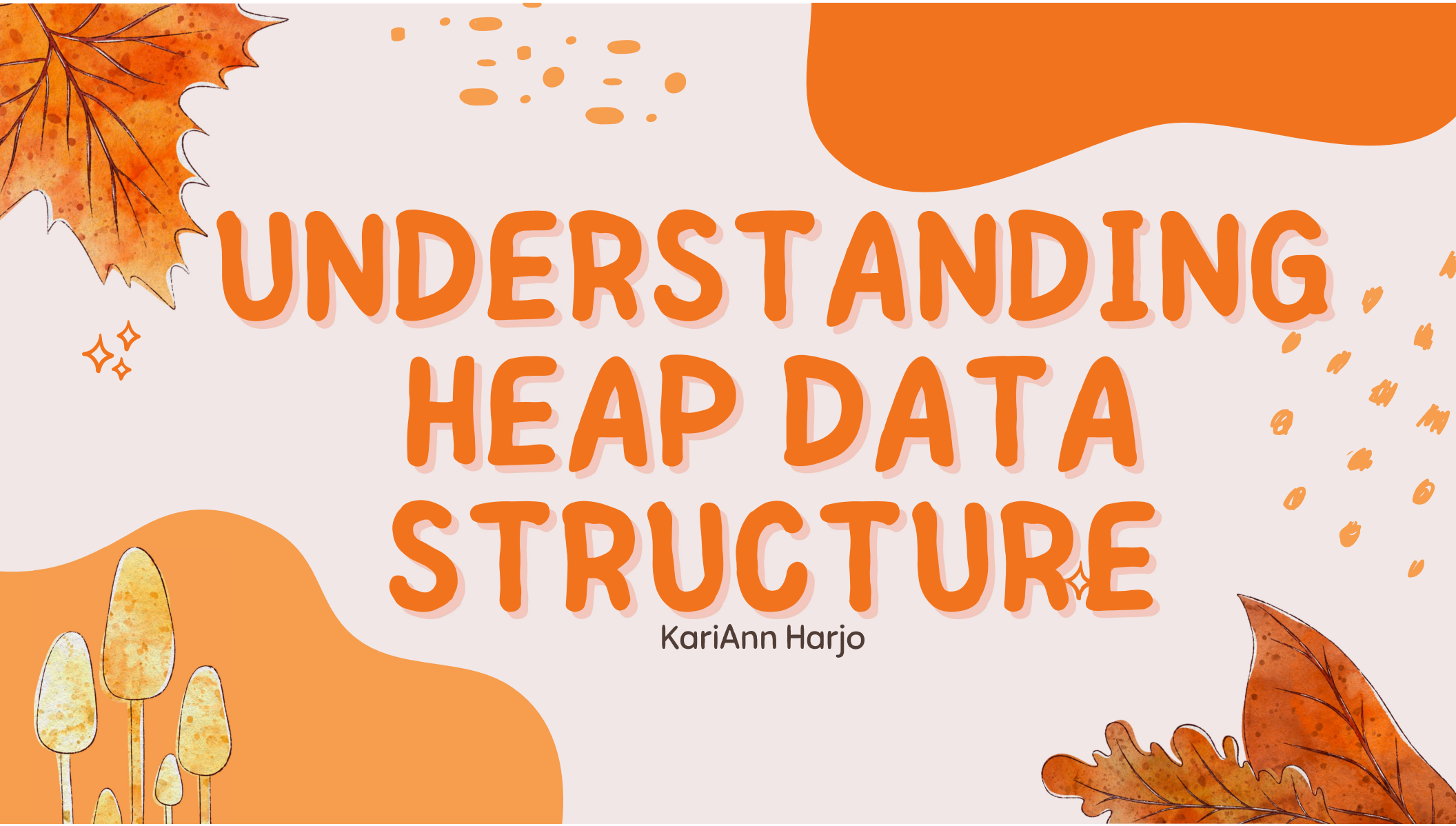
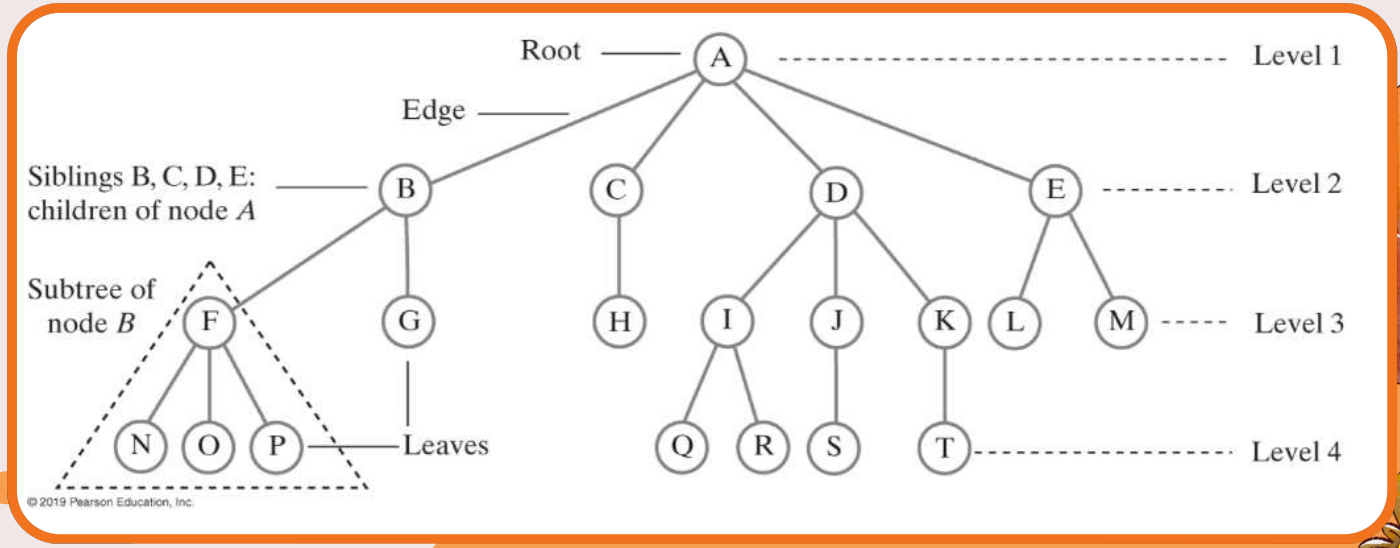

UNDERSTANDING
HEAP DATA
STRUCTURE
KariAnn Harjo

WHAT IS A BINARY TREE?

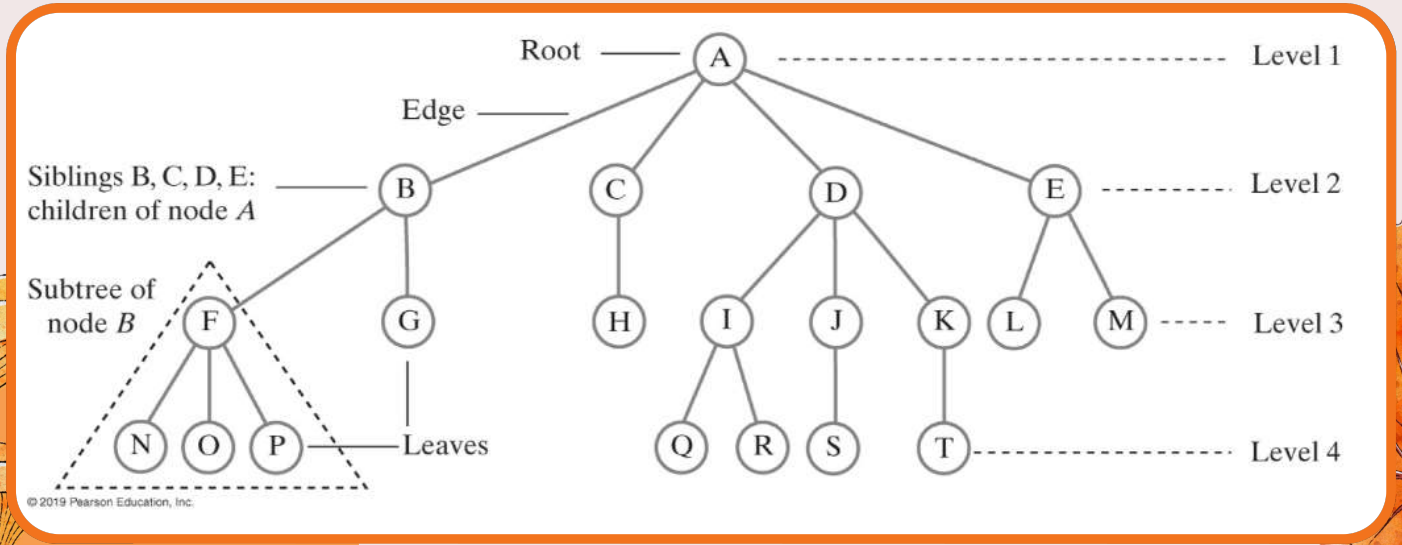
A Binary Tree is a data structure made up of nodes connected by edges, where each node contains a value and two references to two other nodes, representing the subtree to the left and right.



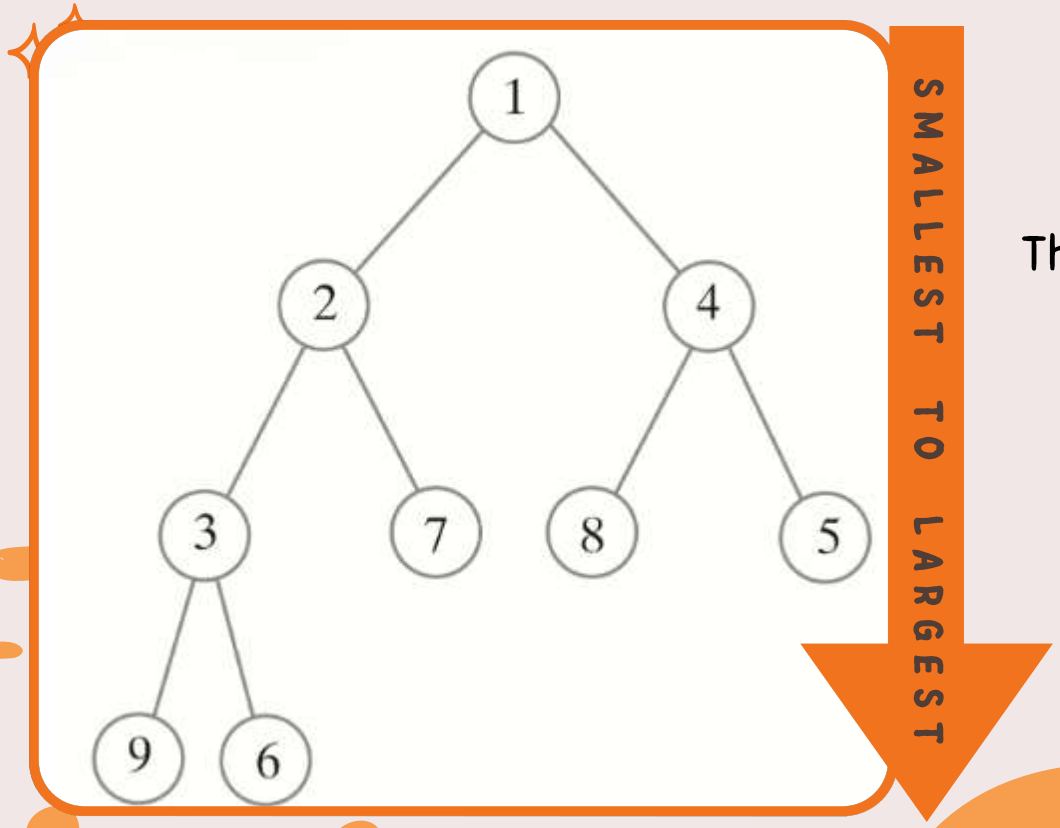
DEFINITION OF HEAP

A heap is a specialized, complete binary tree hat satisfies the heap property–max or min.

Complete Binary Tree: All levels of the tree are fully filled except possibly for the last level, which is filled from left to right.



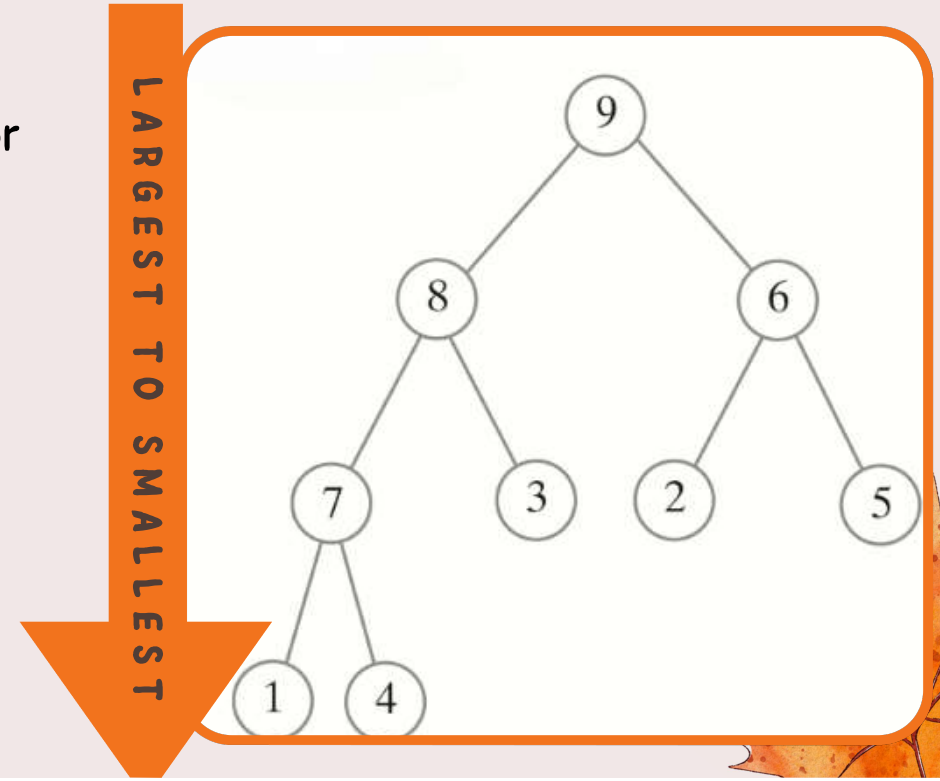
MIN-HEAP



The parent node is always less than or equal to its child nodes.

MAX-HEAP

The parent node is always greater than or equal to its child nodes.



OPERATIONS ON A HEAP

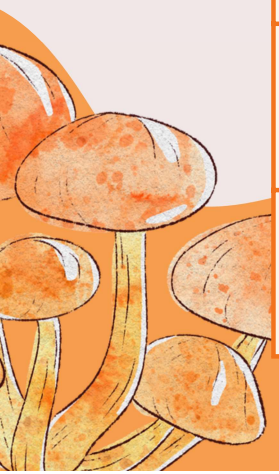
- Insert: Add an element to the heap while maintaining the heap property.
- Delete: Remove an element from the heap while maintaining the heap property.
- Extract Max/Min: Remove and return the maximum/minimum element from a max/min heap.
- Peek Max/Min: Return the maximum/minimum element from a max/min heap without removing it.
- Heapify: Transform a complete binary tree into a heap.

APPLICATIONS OF HEAPS

- Priority Queue: A common application of a heap is to implement a priority queue, where elements are retrieved in order of priority, rather than in a first-in, first-out manner like in a regular queue.
- Heap Sort: A heap can be used to sort an array in O(n log n) time.
- Selection Algorithms: Heaps are useful for algorithms that need to quickly find the kth smallest (or largest) element in a sequence.

PROS AND CONS OF HEAPS

Pros	Cons
Efficient Operations: Heaps provide efficient implementations of operations such as insertion, deletion, and retrieval of the maximum or minimum element, all of which can be done in O(log n) time.	Unordered Nature: Other than the root, there is no guarantee on the order of the elements. So, finding an arbitrary element, or searching for an element with a specific value, could take O(n) time in the worst case.
Memory Efficiency: Heaps are complete binary trees, which means they have a minimal possible height, and this reduces the space needed to store pointers.	Complexity in Deletion: While deleting the root (maximum/minimum) is easy and efficient, deleting an arbitrary element can be complex and time-consuming.
Dynamic: Heaps can efficiently accommodate dynamically changing data, as insertions and deletions can be handled efficiently.	



CONCLUSION

Heaps are a powerful data structure when you need quick and efficient access to the extreme elements (maximum or minimum), especially in the context of dynamic and changing datasets. However, they might not be the best choice if you need sorted data or have to perform a lot of search operations for arbitrary elements.

THANK YOU