Data Structures

An array is a fixed size contiguous block of memory where elements are stored in consecutive locations allowing fast random access to elements via indices. Once an array's size is defined it cannot be changed without creating a new array. A list on the other hand is a more flexible structure often implemented as a linked list or dynamic array where elements can be added or removed easily and its size can grow or shrink during runtime.

Question One

Why would you prefer to implement a linked list over an array? What are the problems which may be encountered when implementing a linked list over an array

Advantages of Linked List over Arrays:

Dynamic Size: A linked list can dynamically adjust its size during runtime. Arrays, on the other hand, have a fixed size once declared, which could either waste memory (if over allocated) or cause overflow issues (if under allocated).

Efficient Insertions and Deletions: Linked lists allow for efficient insertion and deletion of elements. In an array, these operations are typically slower because they may require shifting elements to maintain the array's integrity.

Problems Encountered when Implementing Linked List:

Memory Overhead: Each element in a linked list requires extra memory to store the pointer to the next node, which can be inefficient compared to arrays where elements are stored contiguously.

Random Access Limitation: Linked lists do not support random access to elements. To access an element, we must traverse the list from the beginning, whereas arrays allow direct access via an index.

Complexity of Implementation: Linked lists are more complex to implement and manage compared to arrays, especially when handling edge cases such as adding or removing nodes in various positions.

Question Two

Discuss the advantages and disadvantages of stacks and queues implemented using a linked list.

Advantages:

Dynamic Size: Both stacks and queues can grow and shrink dynamically, with no pre-allocation of memory needed, unlike arrays which have fixed sizes.

Efficient Operations:

- Stack: Both push and pop operations can be done in O(1) time.
- Queue: Both enqueue and dequeue operations can be done in O(1) time.

Disadvantages:

Memory Overhead: Linked list nodes require extra memory for storing pointers, leading to higher memory usage compared to array based implementations.

Pointer Management: You need to manage pointers explicitly which increases the complexity of the implementation and potential for errors.

Question Three

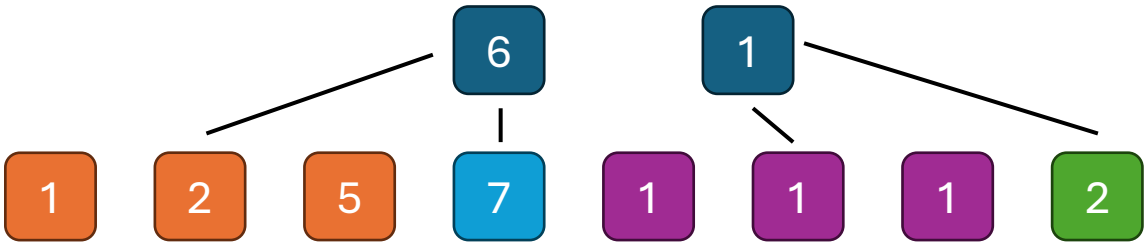
The multiway search tree is used for disc access. Construct a B tree of order 5 for the following data:

1 6 7 2 11 5 10 13 12 20

A tree is a general data structure made up of nodes, where each node has a value and can have children. In the case of a B-tree, it is a self-balancing search tree designed to efficiently handle large amounts of data, especially in situations where data is stored on disk or in databases. The number of keys and children per node is determined by the order of the tree. In this case a B-tree of order 5, each node can have up to 4 keys and 5 children.

Construction Process:

- Start with an empty tree and insert keys.
- If a node becomes full, split it. The middle key moves up to the parent, and the remaining keys are divided into two child nodes.
- If the root splits, a new root is created, maintaining the balanced structure.
- Repeat the process for all keys, ensuring that every node remains within the B-tree properties



Question Four

Suppose you are doing a sequential search of the list below:

[15, 18, 2, 19, 18, 0, 8, 14, 19, 14]

How many comparisons would you need to do to find the key 18? Show the steps.

A sequential search starts at the beginning and looks at each element in turn. To find 18 it would take 2 steps.

15 - False

15, 18 - True

Question Five

Suppose you have following list of numbers to sort:

[19, 1, 9, 7, 3, 10, 13, 15, 8, 12]

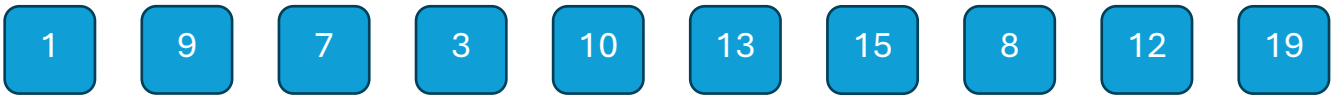
Show the partially sorted list after three complete phases of bubble sort.

Phase 1:

Compare 19 and 1, swap: [1, 19, 9, 7, 3, 10, 13, 15, 8, 12]
Compare 19 and 9, swap: [1, 9, 19, 7, 3, 10, 13, 15, 8, 12]
Compare 19 and 7, swap: [1, 9, 7, 19, 3, 10, 13, 15, 8, 12]
Compare 19 and 3, swap: [1, 9, 7, 3, 19, 10, 13, 15, 8, 12]
Compare 19 and 10, swap: [1, 9, 7, 3, 10, 19, 13, 15, 8, 12]
Compare 19 and 13, swap: [1, 9, 7, 3, 10, 13, 19, 15, 8, 12]
Compare 19 and 15, swap: [1, 9, 7, 3, 10, 13, 15, 19, 8, 12]
Compare 19 and 8, swap: [1, 9, 7, 3, 10, 13, 15, 8, 19, 12]
Compare 19 and 12, swap: [1, 9, 7, 3, 10, 13, 15, 8, 12, 19]

After Phase 1:

[1, 9, 7, 3, 10, 13, 15, 8, 12, 19]

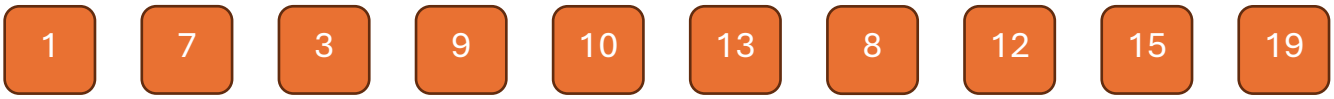


Phase 2:

Compare 9 and 7, swap: [1, 7, 9, 3, 10, 13, 15, 8, 12, 19]
Compare 9 and 3, swap: [1, 7, 3, 9, 10, 13, 15, 8, 12, 19]
Compare 15 and 8, swap: [1, 7, 3, 9, 10, 13, 8, 15, 12, 19]
Compare 15 and 12, swap: [1, 7, 3, 9, 10, 13, 8, 12, 15, 19]

After Phase 2:

[1, 7, 3, 9, 10, 13, 8, 12, 15, 19]



Phase 3:

Compare 7 and 3, swap: [1, 3, 7, 9, 10, 13, 8, 12, 15, 19]
Compare 13 and 8, swap: [1, 3, 7, 9, 10, 8, 13, 12, 15, 19]
Compare 13 and 12, swap: [1, 3, 7, 9, 10, 8, 12, 13, 15, 19]

After Phase 3:

[1, 3, 7, 9, 10, 8, 12, 13, 15, 19]

Question Six

Draw a binary search tree given that the keys were inserted in the following order 5, 30, 2, 40, 25, 4.

A binary tree is a hierarchical data structure where each node has at most two children. The Node, Left and Right.