Assignment 1.

Page No.
Date

1) What are data structure?

Data structure is a specific way to store and organise the data in computers memory so that it can be used efficiently later.

2) Why create data structure?

i) To organize the data in computers memory ii) Data structure gives different level of organization data. iii) It tells how data can be stored and accessed in its elementary level. iv) Provide operation on group of data v) Provide a means to manage huge amount of data efficiently vi) Provide fast searching and sorting

~~What are some applications of data structure?~~

3) Describe the types of data structure:

Mainly there are 2 types of data structure

i) linear data structure.

ii) Non linear data structure.

i) linear data structure - It is the type of data structure in which elements combine to form any specific order. Data is arranged in linear form in continuous memory location.

e.g. Array, linked list, stacks, queues.

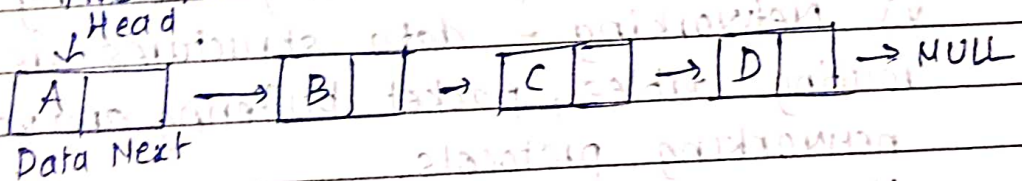
ii) Non linear data structure :- A kind of data

structure in which data elements are not arranged in sequential order. There is hierarchial relationship between individual data items.
e.g. Tree, Graphs

> What is an algorithm?
Algorithm is a step by step procedure which defines a set of instructions to be executed in a certain order to get the desired output.

> What is data abstraction?
→ Data abstraction is the concept of hiding complex implementation details of data providing a simplified interface for users to interact with that data.

> What are dynamic data structure?
→ In dynamic data structure, the size of structure is not fixed and can be modified during the operations performed on it. Dynamic data structures are designed to facilitate change of data structure in the run time.



e.g. Linked list

Assignment 2.

What is linear search?

Linear search is a searching method to check each element one by one in sequence. In this method, linear search() searches the target value in given array and returns its index.

How does a selection sort work for an array?

What are the applications of data structure?

i) Database Management :- Data structures like B-trees and hash tables are used to organize and manage data in database efficiently.

ii) Algorithms :- Data structures are the building blocks of designing and implementing algorithms.

iii) Operating systems :- Data structures are essential for the tasks like memory manage file systems and process scheduling.

iv) Compiler design - Data structures are used in parsing and syntax analysis during compilat

v) Networking - data structures are used for routing tables, packet buffering and various networking protocols.

vi) Game development - Data structure are used for managing game state, collision detection and rendering.

Page No.
Date

2) How does selection sort work for an array?

→ selection sort works by first finding the biggest element in an array and then placing it to the first position. What element is already at the first position will be moved to the position of biggest this is then repeated for an array starting from 2nd, 3rd position etc. consider the following array as an example:-

14	33	27	10	35	19	42	44

we search the whole list sequentially and find the lowest value i.e. 10.

14	33	27	10	35	19	42	44
↑			↑				

so we replace 14 with 10: after one iteration 10, which happens to be the minimum, appears at the first.

10 33 27 14 35 19 42 44

from the second value i.e 33 we will scan the list again to find lowest value.

10 33 27 14 35 19 42 44
↑ ↑ ↑

we find that 14 is the second lowest value we swap these two.

10 14 27 33 35 19 42 44

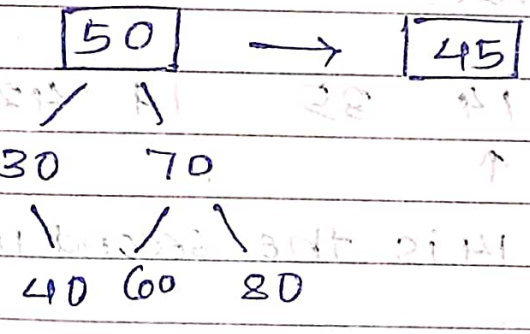
for the rest of the array we will follow same procedure.

How do you insert a new item in a binary search tree?

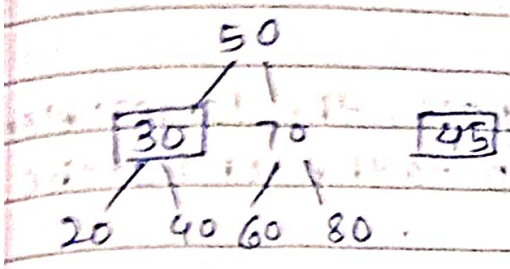
To insert a new item in binary search tree the steps are:-

1. Start at the root:- Begin at the root of BST
2. Compare values:- compare the value of the new item you want to insert with the value of current node.
3. Choose a direction:- if the new value is less than the current node's value move to the left subtree. if it's greater move to the right subtree.
4. Repeat:- Repeat steps 2 and 3 until you reach a node i.e leaf node or a null pointer.
5. Insert the new Node:- create a new node with the value you want to insert and attach it to the left or right, depending on whether it is less than or greater than nodes value.

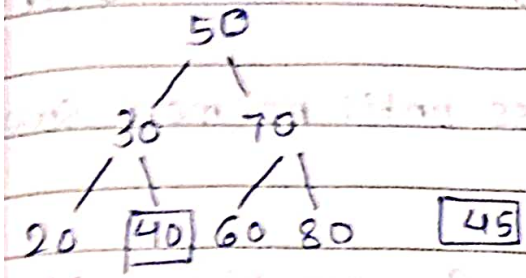
e.g.



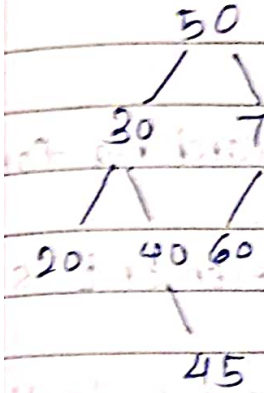
since 45 < 50 move left



45 > 30, move right.



45 > 40 move right
end of the node.
insert 45



What is a bubble sort and how do you perform it?

Bubble sort is a simple sorting algorithm that repeatedly steps through the list of elements to be sorted, compares adjacent elements and swap them if they are in wrong order.

1. Start from the beginning of the list.
2. Compare the first two elements. if the first element is greater than the second, swap them.
3. move to the next pair of elements.

Continue comparing and swapping as you move through the list.

4. Repeat steps 2 and 3 for each pair of adjacent elements in the list until you reach the end of the.

list in the first pass.
5. After the first pass, the largest element will have bubbled up to the end of the list.
6. Repeat step 1-5 for the remaining unsorted portion of the list.
7. continue the process until no more swaps are needed in a pass

5) Give a basic algorithm for searching a binary search tree.

→ 1. Start at the root Node.
2. Compare the value you're searching for with the current node's value.
3. If the values match, the element is found.
4. If the value is smaller, move to the left subtree.
5. If the value is larger, move to the right subtree.
6. Repeat steps 2-5 until you find the element or reach a null node.

Compare all searching methods based on the following.

1. Linear Search:-

i) Time complexity : O(n) (linear time)

ii) Space complexity :- O(1)

iii) Process for searching :- Sequentially go through each element in the list until the target element is found or the entire list is searched.

2. Binary search :-

i) Time Complexity : O(logn)
ii) Space Complexity : O(1)
iii) process of searching :- Divide the list in half repeatedly and compare the target element with middle element. Continue searching in the half where the target might be until the target is found or the search range becomes empty.

3. Hash Table (Hashing) :-

i) Time complexity : O(1)
ii) Space complexity : O(n)
iii) Process for searching : Calculate the hash code of the target element, locate the correspondin bucket and perform a search within that bucket.

4. Fibonacci search :-

i) Time complexity :- O(logn)
ii) Space complexity :- O(1)
iii) process for searching :- Fibonacci search is uses fibonacci numbers to divide the list into segments to search. it compares the target element with the middle element of the curren

segment and narrow down the search range accordingly.

5. Sentinal Search :-

i) Time complexity :- O(n)
ii) Space complexity : O(1)
iii) Process for searching :- It involves adding a sentinal at the end of the list and then searching for the target element.

7. Explain in detail tim sort.

→ Tim sort is a hybrid sorting algorithm derived from merge sort and insertion sort. Tim sort works on divide and conquer technique. It is a stable sorting algorithm. Time complexity of time sort is O(nlogn). and space complexity is O(n). Tim sort is adaptive, stable and efficient. It divides the input into runs (subarrays with ascending or descending order), sorts and merges them efficiently and uses insertion sort for small segments.

8. What is collision? explain collision handling techniques.

> According to the hash function, two or more item would need in the same slot This is said to be called as collision.

There are 2 methods of handling collisions.
1. Open addressing
2. Chaining

1. Chaining - in this technique, each bucket of the hash table maintains a linked list or another data structure to store multiple elements that hash to the same location. When a collision occurs, the new element is simply added to the corresponding bucket.

2. Open addressing :- Instead of using separate data structures to handle collisions, open addressing involves searching for the next available ~~so~~ slot within the hash table when a collision occurs. This can be done using various probing techniques like linear probing, quadratic probing or double hashing

Compare all sorting methods.

1. Bubble Sort :-
   i) Time complexity : $O(n^2)$
   ii) Space complexity : $O(1)$
   iii) stability : stable
   iv) Suitable for small datasets or nearly sorted data.

2. Selection Sort :-
   i) Time complexity : $O(n^2)$
   ii) Space complexity : $O(1)$

pass - depends on ...

iii) Stability: Not stable (may change the order of equal elements).
iv) easy to implement, but inefficient for large datasets

3. Insertion Sort:-
i) Time complexity:- O(n²)
ii) Space complexity:- O(1)
iii) Stability:- stable
iv) efficient for small datasets or partially sorted data.

4. Merge Sort:-
i) Time complexity:- O(nlogn)
ii) Space complexity:- O(n)
iii) Stability:- stable
iv) Requires additional memory for merging, making it less memory efficient.

5. Quick Sort:-
i) Time complexity: O(nlogn)
ii) Space complexity: O(logn)
iii) Stability: Not stable
iv) efficient and widely used.

6. Explain in detail Hashing.

1. Purpose of Hashing :- Hashing is used for efficient data retrievel, data integrity verification and security in various applications.

2. Hash function:- A hash function tums input data into a fixed-size hash code with properties like speed, uniform distribution and collision resistence.

3. Hash table: Hash tables are data structure that use hashing to store and retrive values based on keys.

4. collision Handling:- collisions occue when two different keys produce the same hash code and are managed using techniques like seperate chaining and open addressing.

5. Applications of Hashing:-
   i) hash table
   ii) Password storage - Storing hashed passwords instead of plain text.
   iii) File systems- Hashing is used for efficie file storage and retrivel.
   iv) caching:- Hashing helps to manage and retrive cached data quickly.
   v) Cryptography:- Hash functions are essent for secure communications and digital signatures.