Assignment

Name :- Ruchir Dongre
Branch :- CSE
Semister :- 3rd III
Subject :- Data structure and Algorithom.

Q1) Why Quick Sort preffered for Arrays and Merge Sort for linked list?

Ans) • Quick sort in its general form is an in-place sort whereas merge sort requires O(N) extra storage, N denoting the array size which may be quite expensive. Allocating and de-allocating the extra space used for merge sort increases the running time of the algorithm.

• Comparing average complexity we find that both type of sorts have O(NlogN) average complexity but the constants differ. For arrays, merge sort ~~less~~ loses due to the use of extra O(N) storage space

• Quick sort is also a cache friendly sorting algorithm as it has good locality of reference when used for array.

• Quick sort is also tail recursive, therefore tail call optimization is done.

Now Some of the reasons why Merge sort is ~~for~~ prefered for Linked list

• In case of Linked list the case is different ~~mainly~~ mainly

due to difference in memory allocation of arrays and linked lists. Unlike arrays, linked list nodes may not be adjacent in memory.

- Unlike array, in linked list, we can insert items in the middle in O(1) extra space and O(1) time if we are given reference / pointer to the previous ~~Code~~ node. Therefore merge operation of merge sort can be implemented without extra space for linked lists
- In arrays, we can do random access as elements are continuous in memory. Let us say we have an integer (4-byte) array A and let the address of A[0] be x then the access A[i], we can directly access the memory at (x + i*4). (x + i*4). Unlike arrays, we can not do random access in linked list
- Quick sort requires a lot of this kind of access in linked list, to access i'th index, we have to travel each and every node from the head to i'th node ~~id~~ as we don't have continuous block of memory.

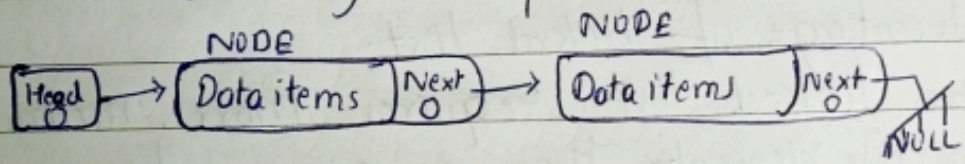
Qzy What is linked list. How will you represent a linked list in a graphical view?

Ans A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked ~~using~~ ~~using~~ using

Using Using pointers.. In simple words, a linked list Conto Consists of nodes, where each node. Contains a data field and a reference (link) to the next node in the list.

Linked List Reperesentation as ~~graph~~.

Linked list can be visualized as a chain of nodes, where every node points to the next node.



As per the above illustration, following are the important points to be considered

- Linked list contains a link element called first
- Each link carries a data field (s) and a link field called next
- Each link is linked with its next link using its next link.
- Last link carries a link as null to mark the end of the list

Q3) How many pointers are necessary to implement a simple linked list? Mention some drawbacks of the linked list.

Ans) There are generally three type of pointers required to implement a simple linked list:

- A 'head' pointer, which is used for pointing to the start of the record in a list.
- A 'tail' pointer, which is used for pointing to the last node. The key factor in the last node is that its subsequent pointer points to nothing (NULL)
- A pointer in every node which is used for pointing to the next node element.

Disadvantage of linked list.

1) The linked list requires more memory to store the elements than an array, because each node of the linked list points a pointer, due to which it requires more money.
2) It is very difficult to traverse the nodes in a linked list, in this, we cannot access randomly to any one node.
3) Reverse traversing in a linked list is a very difficult, because it requires more memory for the pointer.

Q4) How can we find the sum of two linked list using stack in Java?

```
import Java.util.stack;
public class Add Numbers linked list
{
    Private class ListNode
```

Ans) 1) Create stack 's1' by pushing all node values of the first linked list to a stack

2) Create stack 's2' by pushing all node values of the second linked list to a stack

3) Create an empty stack 's3' to store the result of addition

4) initialize sum and carry to 0.

5) Pop the top element from stack 's1'. Let this top element be 'value1'

6) Pop the top element from stack 's2'. Let this top element be 'value2'

7) Make 'sum' = (value1 + value2 + carry) % 10. and push this 'sum' to stack 's3' and make 'carry' = (value1 + value2 + carry)/10.

8) Repeat steps 5-7 till one of the stacks become empty. if both stacks are of same size, the both of the stacks should become empty at the same time.

9) if stack 's1' has elements left in it then.
   a) pop the top element from stack 's1'. Let this top element be 'value1'
   b) Make 'sum' = (value1 + carry) % 10. and push this 'sum' to stack 's3' and make 'carry' = (value1 + carry)/10.
   c) Repeat steps 9a and 9b until stack 's1' is not empty.

10) Similarly, if stacks 's2' has elements left left left left left

in. it then -

a. Pop the top element from stack 'S2'. Let this top element be 'value2'

b. Make 'Sum' = (value2 + (carry)) % 10. and push this 'sum' to stack 'S3'. And make 'carry' = (value 2 + carry) / 10.

c. Repeat steps 10a and 10b. Until stack 'S2' is not empty.

11) After all the above steps are executed. If 'Carry' is greater than 0, push it to the resultant stack 'S3'

12) Create an empty linked list 'result'. Now pop elements one by one from the stack 'S3', and keep appending them to the 'result' linked list Until stack 'S3' is not empty. Return the output as 'result' linked list.

Q5) Explain why stack is a recursive data structure

Ans) Recursive functions use something called "the call stack". When a program calls a function that function goes on top of the call stack. This similar to a stack of books. You add things one at a time. Then, when you are ready to take something off, you always take off the top item.

The factorial call stack in action with the factorial function. factorial(5) is written as

5!. and it is defined like this: $5! = 5*4*3*2*1$. Here is a recursive function to calculate the factorial of a number:

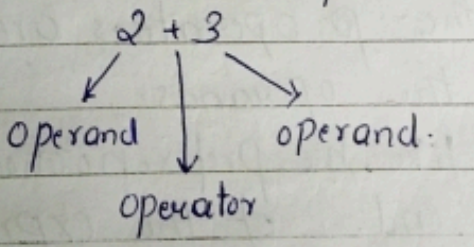
```
function fact(x) {
    if (x==1) {
        return 1;
    } else {
        return x * fact(x-1);
    }
};
```

The ~~top~~ topmost box in the stack tells you what call to fact. you're currently on.

Q6) Explain what are infix, prefix and postfix of Expression

Ans) infix: infix notation is seen as the "normal" way to create mathematical formulae.

eg:



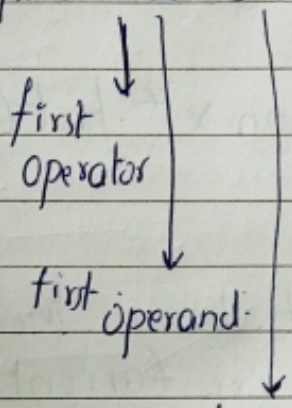
prefix:- Prefix notation puts the operator before all of the operands. This means that the precedence is always clear. We are going to compare the infix notation to this Prefix notation.

prefix. Example of 2+3

in prefix. equivalent we simply. take the operator then move from. left to right. reading. off the operands. in left to. right order.

Prefix: +23

Prefix: +2+34



postfix:- It. is expression in which. the operators are placed before. all the operands.

- This. is just like. the prefix. notation, but the. Operand. Comes. at the end. of the expression, and similarly. removes. the need. for. brackets. within any. expression
- So we take. the bottom two. operand: first from left to right. then append. the operator at the end.
- Similarly. when we have. two. expressive expressions. we. have. to. expression expressions

We have to remember the operators to append to the end.

Prefix :- 2 3 4 + +

Q7) Explain minimum spanning tree.

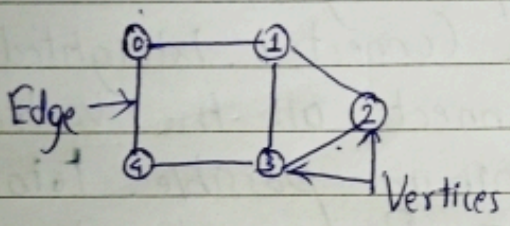
Ans)

- A Minimum Spanning Tree (MST) is a subset of edges of a Connected, Weighted, undirected graph that connects all the vertices together with the minimum possible total edge weight. To drive an MST, Prim's algorithm can be Used. Hence, We will discuss Prim's algorithm in this Chapter.
- As We have discussed, one graph may have more than one spanning tree. if there are n number of vertices, the spanning tree should have n-1 number of edges. In this Context, if each edge of the graph In this Context, if each of the graph is associated with a weight, and there exists more than one spanning tree, We need to find the minimum spanning tree of the graph.
- Moreover, if there exist any duplicate or Weighted edges, the graph may have multiple minimum spanning tree.
- In the above graph, We have shown a spanning tree though it's not the minimum spanning tree tree tree tree

Q8) . Mention the applications of graph and ~~map~~. heap.

Ans) i) Application of Graph

- . A graph is a non-linear data structure, which consist of vertices (or nodes) connected by edges (or arcs) where edges may be directed or Undirected.

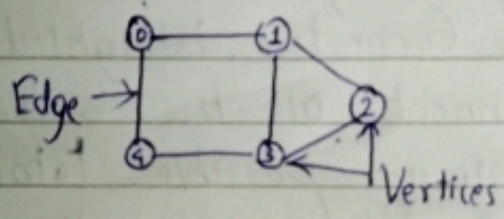


- In Computer Science graphs are used to represent the flow of Computation.
- Google maps uses graphs for building transportation systems, where ~~intersection~~ intersection of two (or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge.
- In World Wide Web, web pages are considered to be ~~the~~ the vertices. There is an edge from a page u to other page v if there is a link of page v ~~if~~ if there is a link of page v on page u
- In similar manner it is used in facebook.
- In operating system, we come across the Resource Allocation Graph where each process and resources ~~to~~ are considered to be vertices. Edges are drawn from ~~the~~ resources to the allocated ~~Process to the Resource~~ process, or from requesting Process

Q8) Mention the applications of graph and ~~map~~ heap.

Ans) i) Application of Graph

- A graph is a non-linear data structure, which consist of vertices (or nodes) connected by edges (or arcs) where edges may be directed or Undirected.



- In Computer Science graphs are used to represent the flow of computation.
- Google maps uses graphs for building transportation systems, where ~~intersec~~ intersection of two (or more) roads are considered to be a vertex and the road connecting two vertices is considered to be an edge.
- In World Wide Web, web pages are considered to be ~~the~~ the vertices. There is an edge from a page u to other page v. if there is a link of page v ~~if~~ if there is a link of page v on page u
- In similar manner it is used in facebook.
- In operating system, we come across the Resource Allocation Graph where each process and resources ~~to~~ are considered to be vertices. Edges are drawn from ~~to~~ resources to the allocated ~~Process to~~ the ~~requ~~ process, or from requesting Process

to the requested resource.

ii) Application of Heap Data Structure.

- Heap Data structure is generally taught with Heapsort. Heapsort algorithm has limited uses because Quicksort is better in practice. Nevertheless, the Heap data structure itself is enormously used other than Heapsort

Priority Queues:- Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreasekey() operations in O(logn) time.

Order Statistics:- The Heap data structure can be used to efficiently find the kth smallest (or largest) element in an array. ~~See method 4.~~

Binomial Heap and Fibonacci Heap are variation of Binary Heap