Linked List Theory Assignment

Name:- Sumit sunil koundanya PRN:-2019BTEIT00023

Q.1) What are Linked List explain with example?

A linked list is a linear data structure. Nodes make up linked lists. Nodes are structures made up of data and a pointer to another node. Usually the pointer is called next.

NULL

Q.2) Differentiate between Array and linked Lists.

|  |  |
| --- | --- |
| Arrays | Linked List |
| Fixed size: Resizing is expensive | Dynamic size |
| Insertions and Deletions are inefficient: Elements are usually shifted | Insertions and Deletions are efficient: No shifting |
| Random access i.e., efficient indexing | No random access Not suitable for operations requiring accessing elements by index such as sorting |
| No memory waste if the array is full or almost full; otherwise may result in much memory waste. | Since memory is allocated dynamically (acc. to our need) there is no waste of memory. |
| Sequential access is faster [Reason: Elements in contiguous memory locations] | Sequential access is slow [Reason: Elements not in contiguous memory locations] |

Q.3) Explain different types of Linked List.

Depending on the way in which the links are used to maintain adjacency, several different types of linked lists are possible.

1) Linear singly-linked list (or simply linear list)

In this, first node is referred as head and last node next pointer contain NULL

2) Circular Linked list

In CLL, The pointer from the last element in the list points back to the first element.

3) Doubly Linked list

Pointers exist between adjacent nodes in both directions. The list can be traversed either forward or backward. Usually two pointers are maintained to keep track of the list, *head* and *tail*.

Q.4) Write an algorithm to inserting the node in a SLL at front, rare and specified location or middle

i) Algorithm to insert at front in SLL

Step 1: IF PTR = NULL

Write OVERFLOW  
     Go to Step 7  
    [END OF IF]

Step 2: SET NEW\_NODE = PTR

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW\_NODE → DATA = VAL

Step 5: SET NEW\_NODE → NEXT = HEAD

Step 6: SET HEAD = NEW\_NODE

Step 7: EXIT

ii) Algorithm to insert at specified location

STEP 1: IF PTR = NULL

WRITE OVERFLOW  
    GOTO STEP 12  
   END OF IF

STEP 2: SET NEW\_NODE = PTR

STEP 3: NEW\_NODE → DATA = VAL

STEP 4: SET TEMP = HEAD

STEP 5: SET I = 0

STEP 6: REPEAT STEP 5 AND 6 UNTIL I<loc< li=""></loc<>

STEP 7: TEMP = TEMP → NEXT

STEP 8: IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"  
     GOTO STEP 12  
    END OF IF  
 END OF LOOP

STEP 9: PTR → NEXT = TEMP → NEXT

STEP 10: TEMP → NEXT = PTR

STEP 11: SET PTR = NEW\_NODE

STEP 12: EXIT

iii) Algorithm to insert at end

Step 1: IF PTR = NULL Write OVERFLOW  
    Go to Step 1  
   [END OF IF]

Step 2: SET NEW\_NODE = PTR

Step 3: SET PTR = PTR - > NEXT

Step 4: SET NEW\_NODE - > DATA = VAL

Step 5: SET NEW\_NODE - > NEXT = NULL

Step 6: SET PTR = HEAD

Step 7: Repeat Step 8 while PTR - > NEXT != NULL

Step 8: SET PTR = PTR - > NEXT  
[END OF LOOP]

Step 9: SET PTR - > NEXT = NEW\_NODE

Step 10: EXIT

Q.5) Write an algorithm to Deleting a node in SLL at front , rare and specified location or middle.

i) Delete node at front

* Step 1: IF HEAD = NULL

Write UNDERFLOW  
     Go to Step 5  
    [END OF IF]

* Step 2: SET PTR = HEAD
* Step 3: SET HEAD = HEAD -> NEXT
* Step 4: FREE PTR
* Step 5: EXIT

ii) Delete node after Specified location

* STEP 1: IF HEAD = NULL

WRITE UNDERFLOW  
    GOTO STEP 10  
   END OF IF

* STEP 2: SET TEMP = HEAD
* STEP 3: SET I = 0
* STEP 4: REPEAT STEP 5 TO 8 UNTIL I<loc< li=""></loc<>
* STEP 5: TEMP1 = TEMP
* STEP 6: TEMP = TEMP → NEXT
* STEP 7: IF TEMP = NULL

WRITE "DESIRED NODE NOT PRESENT"  
    GOTO STEP 12  
    END OF IF

* STEP 8: I = I+1

END OF LOOP

* STEP 9: TEMP1 → NEXT = TEMP → NEXT
* STEP 10: FREE TEMP
* STEP 11: EXIT

iii) Delete node at end/Rare

* Step 1: IF HEAD = NULL

Write UNDERFLOW  
   Go to Step 8  
  [END OF IF]

* Step 2: SET PTR = HEAD
* Step 3: Repeat Steps 4 and 5 while PTR -> NEXT!= NULL
* Step 4: SET PREPTR = PTR
* Step 5: SET PTR = PTR -> NEXT

[END OF LOOP]

* Step 6: SET PREPTR -> NEXT = NULL
* Step 7: FREE PTR
* Step 8: EXIT

Q.6) Searching a SLL.

* Step 1: SET PTR = HEAD
* Step 2: Set I = 0
* STEP 3: IF PTR = NULL

  WRITE "EMPTY LIST"  
  GOTO STEP 8  
  END OF IF

* STEP 4: REPEAT STEP 5 TO 7 UNTIL PTR != NULL
* STEP 5: if PTR → data = item

  Write i+1  
 End of IF

* STEP 6: I = I + 1
* STEP 7: PTR = PTR → NEXT

[END OF LOOP]

* STEP 8: EXIT

Q.7) Reversing a Single Linked List

Algorithm:-

1. Initialize three pointers prev as NULL, curr as head and next as NULL.
2. Iterate through the linked list. In loop, do following.   
   // Before changing next of current,   
   // store next node   
   next = curr ->next  
   // Now change next of current   
   // This is where actual reversing happens   
   curr->next = prev   
   // Move prev and curr one step forward   
   prev = curr   
   curr = nex

Q.8) Write complexity of various operations in array and SLL compare the DLL and SLL.

DLL compared to SLL

Advantages:

* Can be traversed in either direction (may be essential for some programs)
* Some operations, such as deletion and inserting before a node, become easier

Disadvantages:

* Requires more space
* List manipulations are slower (because more links must be changed)
* Greater chance of having bugs (because more links must be manipulated)

Complexities of Various operations in array and SLL

|  |  |  |
| --- | --- | --- |
| Operation | Array Complexity | SLL Complexity |
| Insert at beginning | O(n) | O(1) |
| Insert at end | O(1) | O(1) if the list has tail reference O(n) if the list has no tail reference |
| Insert at middle | O(n) | O(n) |
| Delete at beginning | O(n) | O(1) |
| Delete at end | O(1) | O(n) |
| Delete at middle | O(n): O(1) access followed by O(n) shift | O(n): O(n) search, followed by O(1) delete |
| Search | O(n) linear search  O(log n) Binary search | O(n) |
| Indexing: What is the element at a given position k? | O(1) | O(n) |

Q.9) What are the application of Linked list.

1) Applications that have an MRU list (a linked list of file names).

2) The cache in your browser that allows you to hit the BACK button (a linked list of URLs)

3) Undo functionality in Photoshop or Word (a linked list of state)

4 A stack, hash table, and binary tree can be implemented using a doubly linked list.

Q.10) Write an algorithm to inserting the node in DLL at front, rear and specified location or middle.

A) Insert at Front

* Step 1: IF PTR = NULL

  Write OVERFLOW  
 Go to Step 9  
 [END OF IF]

* Step 2: SET NEW\_NODE = ptr
* Step 3: SET PTR = PTR -> NEXT
* Step 4: SET NEW\_NODE -> DATA = VAL
* Step 5: SET NEW\_NODE -> PREV = NULL
* Step 6: SET NEW\_NODE -> NEXT = START
* Step 7: SET head -> PREV = NEW\_NODE
* Step 8: SET head = NEW\_NODE
* Step 9: EXIT

B) Insert at specified location

* Step 1: IF PTR = NULL

   Write OVERFLOW  
   Go to Step 15  
 [END OF IF]

* Step 2: SET NEW\_NODE = PTR
* Step 3: SET PTR = PTR -> NEXT
* Step 4: SET NEW\_NODE -> DATA = VAL
* Step 5: SET TEMP = START
* Step 6: SET I = 0
* Step 7: REPEAT 8 to 10 until I<="" li="">
* Step 8: SET TEMP = TEMP -> NEXT
* STEP 9: IF TEMP = NULL
* STEP 10: WRITE "LESS THAN DESIRED NO. OF ELEMENTS"

   GOTO STEP 15  
   [END OF IF]  
 [END OF LOOP]

* Step 11: SET NEW\_NODE -> NEXT = TEMP -> NEXT
* Step 12: SET NEW\_NODE -> PREV = TEMP
* Step 13 : SET TEMP -> NEXT = NEW\_NODE
* Step 14: SET TEMP -> NEXT -> PREV = NEW\_NODE
* Step 15: EXIT

C) Insert at end

* Step 1: IF PTR = NULL

 Write OVERFLOW  
  Go to Step 11  
 [END OF IF]

* Step 2: SET NEW\_NODE = PTR
* Step 3: SET PTR = PTR -> NEXT
* Step 4: SET NEW\_NODE -> DATA = VAL
* Step 5: SET NEW\_NODE -> NEXT = NULL
* Step 6: SET TEMP = START
* Step 7: Repeat Step 8 while TEMP -> NEXT != NULL
* Step 8: SET TEMP = TEMP -> NEXT

[END OF LOOP]

* Step 9: SET TEMP -> NEXT = NEW\_NODE
* Step 10C: SET NEW\_NODE -> PREV = TEMP
* Step 11: EXIT

Q.11) Write an algorithm to deleting a node in DLL at front, rare and specified location or middle.

A) Delete node at Front

* STEP 1: IF HEAD = NULL

WRITE UNDERFLOW  
GOTO STEP 6

* STEP 2: SET PTR = HEAD
* STEP 3: SET HEAD = HEAD → NEXT
* STEP 4: SET HEAD → PREV = NULL
* STEP 5: FREE PTR
* STEP 6: EXIT

B) Delete at Specified location

* Step 1: IF HEAD = NULL

    Write UNDERFLOW  
   Go to Step 9  
  [END OF IF]

* Step 2: SET TEMP = HEAD
* Step 3: Repeat Step 4 while TEMP -> DATA != ITEM
* Step 4: SET TEMP = TEMP -> NEXT

   [END OF LOOP]

* Step 5: SET PTR = TEMP -> NEXT
* Step 6: SET TEMP -> NEXT = PTR -> NEXT
* Step 7: SET PTR -> NEXT -> PREV = TEMP
* Step 8: FREE PTR
* Step 9: EXIT

C) Delete at end

* Step 1: IF HEAD = NULL

Write UNDERFLOW  
Go to Step 7  
[END OF IF]

* Step 2: SET TEMP = HEAD
* Step 3: REPEAT STEP 4 WHILE TEMP->NEXT != NULL
* Step 4: SET TEMP = TEMP->NEXT

[END OF LOOP]

* Step 5: SET TEMP ->PREV-> NEXT = NULL
* Step 6: FREE TEMP
* Step 7: EXIT

Q.12) Write an algorithm to inserting the node in CLL at front, rear and specified location or middle.

A) Insert at Front in CLL

* Step 1: IF PTR = NULL

  Write OVERFLOW  
 Go to Step 11  
 [END OF IF]

* Step 2: SET NEW\_NODE = PTR
* Step 3: SET PTR = PTR -> NEXT
* Step 4: SET NEW\_NODE -> DATA = VAL
* Step 5: SET TEMP = HEAD
* Step 6: Repeat Step 8 while TEMP -> NEXT != HEAD
* Step 7: SET TEMP = TEMP -> NEXT

[END OF LOOP]

* Step 8: SET NEW\_NODE -> NEXT = HEAD
* Step 9: SET TEMP → NEXT = NEW\_NODE
* Step 10: SET HEAD = NEW\_NODE
* Step 11: EXIT

B) Insert at end in CLL

* Step 1: IF PTR = NULL

  Write OVERFLOW  
   Go to Step 1  
  [END OF IF]

* Step 2: SET NEW\_NODE = PTR
* Step 3: SET PTR = PTR -> NEXT
* Step 4: SET NEW\_NODE -> DATA = VAL
* Step 5: SET NEW\_NODE -> NEXT = HEAD
* Step 6: SET TEMP = HEAD
* Step 7: Repeat Step 8 while TEMP -> NEXT != HEAD
* Step 8: SET TEMP = TEMP -> NEXT

[END OF LOOP]

* Step 9: SET TEMP -> NEXT = NEW\_NODE
* Step 10: EXIT

Q.13) Write an algorithm to deleting a node in CLL at front, rare and specified location or middle.

A) Deleting node at front

* Step 1: IF HEAD = NULL

Write UNDERFLOW  
  Go to Step 8  
 [END OF IF]

* Step 2: SET PTR = HEAD
* Step 3: Repeat Step 4 while PTR → NEXT != HEAD
* Step 4: SET PTR = PTR → next

[END OF LOOP]

* Step 5: SET PTR → NEXT = HEAD → NEXT
* Step 6: FREE HEAD
* Step 7: SET HEAD = PTR → NEXT
* Step 8: EXIT

B) Delete node at end

* Step 1: IF HEAD = NULL

  Write UNDERFLOW  
   Go to Step 8  
  [END OF IF]

* Step 2: SET PTR = HEAD
* Step 3: Repeat Steps 4 and 5 while PTR -> NEXT != HEAD
* Step 4: SET PREPTR = PTR
* Step 5: SET PTR = PTR -> NEXT

[END OF LOOP]

* Step 6: SET PREPTR -> NEXT = HEAD
* Step 7: FREE PTR
* Step 8: EXIT

Q.14) What are advantages and disadvantages of CLL.

* Advantages:  
   1. If we are at a node, then we can go to any node. But in linear linked list it is not possible to go to previous node.  
  2. It saves time when we have to go to the first node from the last node. It can be done in single step because there is no need to traverse the in between nodes. But in double linked list, we will have to go through in between nodes.
* Disadvantages:  
  1. It is not easy to reverse the linked list.  
  2. If proper care is not taken, then the problem of infinite loop can occur.  
  3. If we at a node and go back to the previous node, then we can not do it in single step. Instead we have to complete the entire circle by going through the in between nodes and then we will reach the required node.