LINKED LIST-2

Solutions

1. Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?

(a) Deleting a node whose location in given
(b) Searching of an unsorted list for a given item
(c) Inverting a node after the node with given location
(d) Traversing a list to process each node

Solution: Option (a)

Explanation:

The presence of previous pointer make the deletion of node easier in doubly link list. This feature is missing in singly link list.

2. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head and tail pointer. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. Insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the last node of the linked list

(a) I and II
(b) I and III
(c) I, II and III
(d) I, II and IV

Solution: Option (c)

Explanation:

Insertion at the beginning of the list can be done as (let temp represent the newly created node)

temp ---> ptr=start; start=temp;

Insertion at the end of the list can be done as (let temp represent the newly created node)

rear-----> ptr=temp; temp=rear;

Deletion of the front node can be done as

```
struct node * temp= start;

start=start---> prt;

free(temp);
```

However deletion cannot be done at O(1) time as we need to have a pointer to the previous node of tail for which we need to traverse the entire list.

3. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. Insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the last node of the linked list

(a) I and II
(b) I and III
(c) I, II and III
(d) I, II and IV

Solution: Option (b)

4. Consider an implementation of unsorted doubly linked list. Suppose it has its representation with a head pointer and tail pointer. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. Insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the end node of the linked list

(a) I and II
(b) I and III
(c) I, II and III
(d) I, II, III and IV

Solution: Option (d)

Explanation:

A doubly link list has all the features of a singly link list along with that it points to the node previous to it. In case of singly link list for opertation (iv) we didnot know how to go to the nodes just previous of rear, but in this case the previous pointer of rear node will point to the

second last node, hence this operation can be done in O(1) .

5. Consider an implementation of unsorted doubly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. Insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the end node of the linked list

(a) I and II
(b) I and III
(c) I, II and III
(d) I, II, III and IV

Solution: Option (b)

6. Consider an implementation of unsorted circular linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. Insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the end node of the linked list

(a) I and II
(b) I and III
(c) I, II, III and IV
(d) None

Solution: Option (d)

7. Consider an implementation of unsorted circular doubly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

i. Insertion at the front of the linked list
ii. insertion at the end of the linked list
iii. Deletion of the front node of the linked list
iv. Deletion of the end node of the linked list

(a) I and II
(b) I and III
(c) I, II and III
(d) I, II, III and IV

Solution: Option (d)

8. In linked list each node contain minimum of two fields. One field is data field to store the data second field is?

(a) Pointer to character
(b) Pointer to integer
(c) Pointer to node
(d) Node

Solution: Option (c)

9. What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?

(a) O(1)
(b) O(n)
(c) θ (n)
(d) θ (1)

Solution: Option (c)

10. What would be the asymptotic time complexity to add an element in the linked list?

(a) O(1)
(b) O(n)
(c) $O(n^2)$
(d) None

Solution: Option (b)

11. What would be the asymptotic time complexity to find an element in the linked list?

(a) O(1)
(b) O(n)
(c) $O(n^2)$
(d) None

Solution: Option (b)

12. What would be the asymptotic time complexity to insert an element at the second position in the linked list?

(a) O(1)
(b) O(n)
(c) $O(n^2)$
(d) None

Solution: Option (a)

13. The concatenation of two list can performed in O(1) time. Which of the following variation of linked list can be used?

(a) Singly linked list
(b) Doubly linked list
(c) Circular doubly linked list
(d) Array implementation of list

Solution: Option (c)

14. Consider the following definition in c programming language

```
struct node
{
int data;
struct node * next;
}
typedef struct node NODE;
NODE *ptr;
```

Which of the following c code is used to create new node?

(a) ptr=(NODE*)malloc(sizeof(NODE));
(b) ptr=(NODE*)malloc(NODE);
(c) ptr=(NODE*)malloc(sizeof(NODE*));
(d) ptr=(NODE)malloc(sizeof(NODE));

Solution: Option (a)

15. A variant of linked list in which last node of the list points to the first node of the list is?

(a) Singly linked list
(b) Doubly linked list
(c) Circular linked list
(d) Multiply linked list

Solution: Option (c)

16. In doubly linked lists, traversal can be performed?

(a) Only in forward direction
(b) Only in reverse direction

(c) In both directions (d) None

Solution: Option (c)

17. What kind of linked list is best to answer question like "What is the item at position n?"

(a) Singly linked list (b) Doubly linked list
(c) Circular linked list (d) Array implementation of linked list

Solution: Option (d)

18. A variation of linked list is circular linked list, in which the last node in the list points to first node of the list. One problem with this type of list is?

(a) It waste memory space since the pointer head already points to the first node and thus the list node does not need to point to the first node.

(b) It is not possible to add a node at the end of the list.

(c) It is difficult to traverse the list as the pointer of the last node is now not NULL

(d) All of above

Solution: Option (c)

19. A variant of the linked list in which none of the node contains NULL pointer is?

(a) Singly linked list (b) Doubly linked list
(c) Circular linked list (d) None

Solution: Option (c)

20. In circular linked list, insertion of node requires modification of?

(a) One pointer (b) Two pointer
(c) Three pointer (d) None

Solution: Option (b)