1. What is the time complexity of insert at any position in a singly linked list?

A) O(1)

**B) O(N)**

C)O(logN)

D) O(N\*N)  
  
**Explanation:**  একটি singly linked list-এ যখন কোনো নির্দিষ্ট পজিশনে ইনসার্ট করতে হয়, তখন প্রথমে সেই পজিশনটি খুঁজে বের করতে হয়। এই কাজের জন্য লিস্টটি head থেকে শুরু করে target position-এ পৌঁছানো পর্যন্ত traverse করতে হবে।

এটি linear traversal হওয়ায় O(N) সময় লাগে, যেখানে N হলো লিস্টের দৈর্ঘ্য।

2. What is the time complexity of insert at head in a singly linked list?

**A) O(1)**

B) O(N)

C)O(logN)

D) O(N\*N)

**Explanation:** Insert at head খুব দ্রুত করা যায় কারণ নতুন node যোগ করার জন্য শুধু head পয়েন্টারটিকে নতুন node-এর দিকে পয়েন্ট করতে হয়।এখানে কোনো traversal লাগবে না।

তাই, Time Compleity O(1)।

3. What is the time complexity of insert at tail in a singly linked list? We have only one pointer(head) for tracking.

A) O(1)

**B) O(N)**

C)O(logN)

D) O(N\*N)

**Explanation:** যদি head পয়েন্টার দিয়ে tail-এ node যোগ করতে হয়, তবে tail-এ পৌঁছানো পর্যন্ত linked list-টি traverse করতে হবে।

এটি O(N) সময়ে সম্পন্ন হয় কারণ N সংখ্যক nodes-এর মধ্য দিয়ে যেতে হয়।

4. What is the time complexity of insert at tail in a singly linked list? We have two pointers(head and tail) for tracking.

**A) O(1)**

B) O(N)

C)O(logN)

D) O(N\*N)

**Explanation:** যদি tail pointer থাকে, তাহলে tail-এ সরাসরি node যোগ করা সম্ভব।

tail->next-এ নতুন node যোগ করা হয় এবং tail-কে নতুন node-এ পয়েন্ট করা হয়।

এই কাজটি O(1) সময়ে হয় কারণ কোনো traversal প্রয়োজন হয় না।

5. What is the time complexity of delete head in a singly linked list??

**A) O(1)**

B) O(N)

C)O(logN)

D) O(N\*N)

**Explanation:** Singly linked list-এ head node delete করা খুব সহজ এবং দ্রুত একটি কাজ।

প্রথমে head পয়েন্টারটি বর্তমান head node-এর next node-এর দিকে পয়েন্ট করা হয়।

তারপর পুরাতন head node-টিকে delete করা হয়।

6. What is the time complexity of delete at tail in a singly linked list? We have two pointers(head and tail) for tracking.

A) O(1)

**B) O(N)**

C)O(logN)

D) O(N\*N)

**Explanation:** যদি শুধু head এবং tail পয়েন্টার থাকে, তবে tail-এর ঠিক আগের node খুঁজে বের করতে linked list-টি traverse করতে হয়। কারন singly linkedlist এ আগের নোড এ ব্যাক করা যায় না।

এটি O(N) সময়ে হয়।

7. Which is better to use if we always want to delete from tail?

**A) Array**

B) Linked list.

C)Both are equal.

D) None.

**Explanation:** Array-এর ক্ষেত্রে শেষ থেকে delete করা খুব দ্রুত হয় কারণ এটি O(1) সময়ে করা সম্ভব।

কিন্তু singly linked list-এ tail-এর আগের node খুঁজে বের করতে O(N) সময় লাগে।

তাই delete from tail করার জন্য array ভালো।

8. What will the following code snippet do?

Node \*tmp = head;

for (int i = 1; i <= pos - 1; i++)

{

tmp = tmp->next;

}

Node \*deleteNode = tmp->next;

tmp->next = tmp->next->next;

delete deleteNode;

* 1. Insert a node at head
  2. Insert a node at any position
  3. **Delete a node from any position**
  4. Delete the head node

**Explanation:** code snippet-টি একটি নির্দিষ্ট পজিশনে node delete করার কাজ করে।

প্রথমে tmp দিয়ে target position-এর আগের node-এ পৌঁছানো হয়।

তারপর tmp->next-কে tmp->next->next-এ পয়েন্ট করা হয়, যাতে target node লিস্ট থেকে বাদ পড়ে।

পরে delete deleteNode দিয়ে সেই node মেমোরি থেকে ডিলিট করা হয়।

9. What is the time complexity of sorting a singly linked list using selection sort?

A) O(1)

B) O(N)

C)O(logN)

**D) O(N\*N)**

**Explanation:** Selection Sort-এ প্রতিটি node-এর জন্য বাকি nodes-এর মধ্যে সর্বনিম্ন বা সর্বোচ্চ মান খুঁজতে হয়।

এর জন্য প্রতিটি iteration-এ O(N) সময় লাগে।

মোট N iterations থাকায় সময় O(N\*N) হয়।

10. What will the following code snippet do?

Node \*deleteNode = head;

head = head->next;

delete deleteNode;

* 1. Insert a node at head
  2. Insert a node at any position
  3. Delete a node from any position
  4. **Delete the head node**

**Explanation:**  code snippet-টি head node ডিলিট করে।

head পয়েন্টারকে head->next-এ সেট করা হয়।

তারপর পুরানো head node-কে delete করা হয়।

এটি O(1) সময়ে সম্পন্ন হয়।