Top 10 Algorithms in Interview

LINKED LIST

GEEKS FOR GEEKS

Sunil016@Yahoo.com

Given a linked list which is sorted, how will you insert in sorted way

Quite Straight Forward Approach, we need to seep the track of Previous Node and NextNode while traversing on to each Node in the Linked List.
We will travers the Linked List Until we have a condition - insertNode < Node.Value
We always consider the primary case for the head, if we have Head.Value > insertedNode.
We will Simply apply logic -

```
LLNode TNode = head;
  LLNode PrevNode = head;
  while (head != null && head.val < node.val) {
        PrevNode = head;
        head = head.next;
  }
PrevNode.next = node;
node.next = head;
```

Complexity -> Time - O(n) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/SortedInsert.java

Delete a given node in Linked List under given constraints

Travers the Linked List with maintaining previous node and whenever we find a node has to be deleted, we simple remove the next of previous to next of current node, that is bypassing the current node.
We will Simply apply logic -

```
while (head != null) {
  LLNode successor = head.next;
  if (head.val == delNode) {
      PrevNode.next = successor;
      return;
  }
  PrevNode = head;
  head = head.next;
}
```

Complexity -> Time - O(n) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/DeleteNode.java

Compare two strings represented as linked lists

We have given a two string in the form of linked list, in such case we will traverse both the linked list at same time and if it any point we find the char different, we simply compare their difference and returns the desired value.
We will Simply apply logic -

```
if (strOne == null && strTwo == null) {
    return 0;
}
if (strOne.c != strTwo.c) {
    if (strOne.c > strTwo.c) {
        return 1;
    } else {
        return -1;
    }
}
```

Complexity -> Time - O(max(m,n)) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/CompareString.java

Add two numbers represented by linked lists

Question seems easy but there is a trick of adding two numbers inside the linked list from the end, and that makes it completed. So here we will be using Two Stack for storing the element of two list and there after we will be making Addition of each pops, once done we will generate a linked list.
We will Simply apply logic -

```
while (!OneStack.isEmpty() && !TwoStack.isEmpty()) {
    int Sum = OneStack.pop().val + TwoStack.pop().val + carry;
    carry = 0;
    if (Sum > 9) {
        ResultStack.add(Sum % 10);
        carry = Sum / 10;
    } else {
        ResultStack.add(Sum);
    }
}
```

Complexity -> Time - O(n+m) & Space - O(n+m)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/AddTwoNumbers.java

Merge a linked list into another linked list at alternate positions

This is game of pointers, that will be travelling both the linked lists and connecting dots with each other.

We will Simply apply logic -

```
while (One_Current != null && Two_Current != null) {
    One_Next = One_Current.next;
    Two_Next = Two_Current.next;

    One_Current.next = Two_Current;
    Two_Current.next = One_Next;

    One_Current = One_Next;
    Two_Current = Two_Next;

}
```

Complexity -> Time - O(n+m) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/MergeLinkedListAlternative.java

Reverse a Linked List in groups of given size

Complicated and most famous interview question, the basic approach to keep in mind is to make a function, that will reverse the linked list for k nodes, and after that we will append all the reversed component into one making it a resultant.

We will Simply apply logic -

```
while (counter < K && head != null) {
    sNode = head.next;
    head.next = pNode;
    pNode = head;
    head = sNode;
    counter++;
}
```

Complexity -> Time - O(n) & Space - O(n) - HeapStack

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/ReverseLinkedList.java

Union and Intersection of two Linked Lists

HashSet is the Best Option to Store the Unique element inside, that will solve our Union problem and for intersection we can use two additional HashSets, once for storing all and other for storing only repeating once i.e. intersection.
We will Simply apply logic -

```
while (Two != null) {
    unionSet.add(Two.val);
    Two = Two.next;
}
for (int x : unionSet) {
    resultList.next = new ListNode(x);
    resultList = resultList.next;
}
```

Complexity -> Time - O(n) & Space - O(n)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/UnionIntersection.java

Detect and Remove Loop in a Linked List

Two Pointers is the Approach to solve this problem, Slow pointer one step at a time, Fast pointer two step at a time, if every they meet, it means we have a loop, and the point where they meet we have to traverse the slow pointer till it meets fast and once found we will make prevNode.next = null.
We will Simply apply logic -

```
while (fast != null && slow != fast) {
    slow = slow.next;
    fast = fast.next.next;
}while (true) {
    Two = slow;
    while (Two.next != slow && Two.next != One) {
        Two = Two.next;
    }
    if (Two.next == One) {
        break;
    }
    One = One.next;
}
 Two.next = null;
```

Complexity -> Time - O(n) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/DetectAndRemoveLoop.java

Merge Sort for Linked Lists

This is divide and conquer rule, in this we will find the middle of a linked list and divide until they are fragmented to a single unit, once they are fragmented, we will merge then on the basic of their order.
We will Simply apply logic -

```
ListNode middle = getMiddle(head);
ListNode next2Middle = middle.next;
middle.next = null;
ListNode left = mergeSort(head);
ListNode right = mergeSort(next2Middle);
ListNode x = sortedList(left, right);
```

Complexity -> Time - O(nlgn) & Space - O(n)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/MergeSort.java

Select a Random Node from a Singly Linked List

First find the length of the linked list and then apply probability of 1/N for each outcome using Math.Random

We will Simply apply logic -

```
int probability = (int) (Math.random() * nodeCount);
 for (int i = 0; i < probability; i++) {
     pHead = pHead.next;
 }
```

Complexity -> Time - O(n) & Space - O(1)

Java Solution @Git ::- https://github.com/Ysunil016/Coding_Challenges/blob/master/GeeksForGeeks/src/_60Days/LinkedList/PickRandomNode.java