Assignment 1

Bhanu Prakash J - EE18BTECH11022

Download all Codes from

https://github.com/Bhanuprakash072/Insertion-In-Sorted-Linked-List/tree/main/Assignment-01/codes

Download all latex-tikz codes from

https://github.com/Bhanuprakash072/Insertion-In-Sorted-Linked-List/blob/main/Assignment-01/assignment1.tex

1 Problem

(Q 16) What is the worst case time complexity of inserting n elements into an empty linked list, if the linked list needs to be maintained in sorted order ?
(A) $\theta(n)$
(B) $\theta(nlog(n))$
(C) $\theta(n^2)$
(D) $\theta(1)$

2 Solution

Answer : (C)
Explanation This question is ambiguous: "needs to be maintained in sorted order", there are two possible cases:

Needs to be maintained in sorted order on each step (after each insertion). When we are inserting an element in to empty linked list and to perform sorted order list of every element will take O(n2). Each Insertion into a sorted linked list will take $\theta(n^2)$ and hence the total cost for n operations is $\theta(n^2)$.

Needs to be maintained in sorted order on final step (only after all insertion). When we are inserting all elements into an empty linked list and to perform a sorted list (using merge sort) after inserting all elements will take O(n log n) time.

```
// C Code for printing output in form of array
/* Program to insert in a sorted list */
#include <stdio.h>
#include <stdlib.h>

struct Node {
        int data;
        struct Node* next;
};

void sortedInsert(struct Node** head_ref,struct
    Node* new_node)
{
        struct Node* current;
        if (*head_ref == NULL || (*head_ref)
            ->data>= new_node->data) {
                new_node->next = *head_ref;
                *head_ref = new_node;
        }
        else {
                current = *head_ref;
                while (current->next != NULL
                    && current->next->data <
                    new_node->data) {
                        current = current->next
                            ;
                }
                new_node->next = current->
                    next;
                current->next = new_node;
        }
}

struct Node* newNode(int new_data)
{
        struct Node* new_node = (struct Node*)
            malloc(sizeof(struct Node));

        /* put in the data */
        new_node->data = new_data;
        new_node->next = NULL;

        return new_node;
}

/* Function to print linked list */
```

```
void printList(struct Node* head)
{
        struct Node* temp = head;
        while (temp != NULL) {
                printf("%d ", temp->data);
                temp = temp->next;
        }
}

/* Driver program to test count function*/
int main()
{
        /* Start with the empty list */
        struct Node* head = NULL;
        int n;
        printf("Enter the number of nodes. n=");
        scanf("%d",&n);

        for(int i=0;i<n;i++){
                int temp;
                printf("Enter the value of %d-th
                      node: ", i);
                scanf("%d",&temp);
                struct Node* new_node =
                     newNode(temp);
                sortedInsert(&head, new_node);
        }

        printf("Created Linked List: ");
        struct Node* temp = head;
        while (temp != NULL) {
                printf("%d ", temp->data);
                temp = temp->next;
        }
        printf("\n");

        return 0;
}
```