Data Structures

1. **Perform Analysis on time complexity of insertion sort algorithm in best case. Can you suggest few modification in order to reduce time complexity of this algorithm from O(n2) to some lower order term**

Submitted By

Name : Harsh Soni

Roll No.: 11912082

Branch: IT

Q No

DS ASSIGNMENT

(SORTING AND BST)

Q1

```
INSERTION-SORT (arr)
1 for i ← 2 to arr.length:   //considering 1 indexed.
2     key = arr[i]
3     j = i-1
4     while (j>0) && (arr[j] > key):
5         arr[j+1] = arr[j]
6         j --=1
7     arr[j+1] = key.
```

BEST CASE TIME COMPLEXITY (when arr sorted).

eg:

1	2	3	4	5	6
5	6	1	8	3	2.

$\longrightarrow$

1	2	3	4	5	6.
1	2	3	5	6	8

sorted arr.

i	2	3	4	5	6.
j	1	2	3	4	5
comparisons	1	1	1	1	1
swaps.	1	1	1	1	1.

If cost of line 4 and 7 be c_1 and c_2.
then

$T(6) = c_1 \times 6 + c_2 \times 6 = (c_1+c_2)6 = c'6.$

Generalizing,

$T(n) = c'n. \approx O(n).$

* Enhancing efficiency of insertion sort :-

(a.) Using binary search instead of normal loop to search the correct place for "key". But then again the shifting will take $O(n)$ for each pass and hence the total worst case time complexity becomes $O(n^2)$.

(b.) Using doubly linked list. Using doubly linked list takes care of the shifting problem as we can directly insert in doubly linked list by manipulating pointers in $O(1)$ time. But still searching for particular element can take $O(n)$ for each pass. Hence making the worst case time complexity $O(n^2)$.

[Above implementations can be found in github repo].

Q2