Name : Kiran Tulsulkar NUID : 001057882

EECE 7205 : Fundamentals of Computer Engineering

Assignment 2

Question 1 :

Write a C++ program for sorting an array *A* of *n* integers. First you need to implement both the MERGESORT and MERGE algorithms .

Solution :

Merge Sort :

```
-bash-4.2$ g++ assignment2_mergesort.cpp -o as21 -std=c++11
-bash-4.2$ ./as21
Enter the value of n 10
Randomly Generated Numbers are :
36      49      60      17      96      75      69      51      99      31
Sorted Numbers are :
17      31      36      49      51      60      69      75      96      99
-bash-4.2$
```

Output of Merge Sort when n =10

Question 2:

In the above MERGE algorithm and to avoid having to check whether either list is empty in each basic step, a sentinel value of infinity is placed at the end of each list.

Solution :

Modified Algorithm :

MERGE(A,p,q,r)

n1 = q - p + 1
n2 = r – q
let L [1 .. n1] and R [1 .. n2] be in newarrays
for i = 1 to n1
    L[i] = A[p + i -1] **for** j =
1 to n2
    R[j] = A[q + j]

i = 1
j = 1

```
k = p
while i < n1 and j <n2
        if L[i] <=R[j]
           A[k] = L[i]
           i = i + 1
        else
            A[k] = R[j]
           j = j + 1
        k = k + 1

while i  to  n1
        A[k] = L[i]
        i = i + 1
        k = k + 1
while j to n2
        A[k] = R[j]
        j = j + 1
        k = k + 1

MERGE-SORT(A,p,r)
        if p < r
           q = [ (p+r) / 2]
           MERGE-SORT(A, p, q)
           MERGE-SORT(A, q + 1, r)
           MERGE(A, p, q, r)
```

Name : Kiran Tulsulkar NUID : 001057882

Question 3:

```
ProcedureX (A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

The above code is for **ProcedureX** that takes list *A* of integers as an input parameter.

Solution :

a) In 70 words or less, explain the purpose of ProcedureX and how it achieves that purpose.
Answer :
Purpose is to sort array in ascending order. First 'for' loop marks A[1] element in array than second for loop starts comparison from last element of array and adjacent element till element marked by first for loop. After completion of first for loop, first element in array is smallest in array. First for loop than marks next element in array, comparison continues till all elements in array are sorted.

b) If *n* = *A.length*, determine the worst-case running time formula of ProcedureX. Write the formula as a function of *n* (show your steps).

Answer :

Step	Cost	Number of times
1	C1	n
2	C2	n(n-1)/2
3	C3	n(n-1)/2
4	C4	n(n-1)/2

Running Time = C1*n + C2*n(n-1)/2 + C3*n(n-1)/2 + C4*n(n-1)/2

= n^2 *(C2 + C3 + C4)/2 + n*(C1 + (C2 + C3 + C4)/2)

= A*n^2 + B*n

First term n^2 is of interest. Worst Case running time is given as O(n^2)

Name : Kiran Tulsulkar NUID : 001057882

Question 4:

Following the same approach that we used with analyzing the merge sort algorithm, analyze your recursive insertion sort algorithm to find its running time recurrence equation. Solve the recurrence equation to find the asymptotic notation of the running time.

Solution :

Algorithm:

1. Pass A[] and number of elements through InsertionSort
2. If n>1 then
3. Recursively Sort A[n-1]
4. Insert A[n] in sorted array

Pseudo Code :

```
InsertionSort(int A[] , int n)

If ( n<=1)

        Return

InsertionSort(A[] , n-1)

Int copy = A[n-1]

Int i = n -2

While ( i>=0 And A[i]>copy)

        A[i+1] = A[i]

        i=i-1

A[i] = copy
```

Recurence Equation :

$T(n) = 1 \quad \text{for } n = 1$
$= T(n - 1) + n \quad \text{for } n > 1$

$T(n) = T(n-1) + n$
$T(n) = T(n-2) + (n-1) + n$
$T(n) = T(n-3) + (n-2) + (n-1) + n$
$T(n) = T(n - x) + (n - (x - 1)) + (n - (x - 2)) + ... + (n - 1) + n$
Let $n - x = 0$,
$n = x$
$T(n) = T(0) + 1 + 2 + 3 + ... + (n - 2) + (n - 1) + n$
$T(n) = [n (n + 1) / 2]$
$T(n) = O(n^2)$