**Year 2**

## Data Structures and Algorithms – IT2070

**BSc (Hons) in Information Technology**



**Tutorial 7 – Divide and Conquer**

# Semester 2, 2022

Question1

The pseudo codes for the *merge sort* are given below

**MERGESO RT** (A,*l*,*r*)

1. **if** *l* < *r*

2. *q* = ⎣(*l+r* )/2⎦

3. **MERGESORT** (A,*l*,*q*)

4. **MERGESORT** (A,*q*+1, *r*)

5. **MERGE** (A,*l*,*q*,*r*)

**MERGE** (A,*l*,*q*,*r*)

1. *i* = l

2. *j* = *q*+1

3. *k* = 0

4. **while** (*i* ≤ *q*) **and** ( *j* ≤ *r*) do

5. *k* = *k*+1

6. **if** A[*i*] ≤ A[*j*] **then**

7. TEMP [*k*] = A [*i*]

8. *i* = *i* +1

9. **else**

10. TEMP [*k*] = A [*j*]

11. *j* = *j* +1

12. **if** *j* > *r* **then**

13. **for** *t* = 0 **to** *q* – *i* do

14. A [*r-t*] = A [*q-t*]

15. **for** *t* = 0 **to** *k*-1 do

16. A [*l* +*t*] = TEMP [*t* +1]

* 1. Illustrate operation of *merge* *sort* on the array A=( 6,4,8,1,7,2,5,3).
  2. What is the purpose of the Temp array in the *merge sort* algorithm?
  3. Why do it execute line no 13 and 14?
  4. Modify the above **MERGESORT** *(A, l, r)* algorithm to sort the numbers in descending order. [Only the modified line should be described]

1. Two algorithms for the Maximum Sub Sequence Sum Problem are given below. Which algorithm will be the faster one? Justify your answer with Big O notation.

***Algorithm 01***

1. **for** (j=0;j< n ;j++)
2. { ThisSum=0;
3. **for** (k=j;k< n ;k++)
4. { ThisSum += A[k];
5. **if** (ThisSum>MaxSum)
6. MaxSum=ThisSum;
7. }

}

***Algorithm 02***

1. **for** (j=0;j< n ;j++)
2. {
3. ThisSum += A[j];
4. **if** (ThisSum>MaxSum)
5. MaxSum=ThisSum;
6. **else if** (ThisSum < 0)
7. ThisSum = 0;
8. }
9. Fill the table entries giving the time complexity in Big O notation.

|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | |
| Best Case | Worst Case |
| Insertion sorting |  |  |
| Quick sort |  |  |
| Merge sort |  |  |

Additional Exercises:

Question1

Below Partition algorithm is used in quick sort algorithm. Find out how it works and state if there are any drawback.

**PARTITION**(A,p,r)

*x* = A[*p*]

*i* = *p* -1

*j* = *r* +1

**while** TRUE

**repeat**

*j* = *j*  - 1

**until** A[*j*] ≤ *x*

**repeat**

*i* = *i* + 1

**until** A[*i*] ≥ *x*

**if**  *i* < *j*

exchange A[*i*] withA[*j*]

**else return** *j*

Question2

Analyze the below Merge algorithm used in merge sort. Find out how it works .

Procedure **MERGE** (A,p,q,r)

1. i = p

2. j = q+1

3. k = 0

4. **while** (i ≤ q) **and** ( j ≤ r)

5. k = k+1

6. **if** A[i] ≤ A[j]

7. TEMP [k] = A [i]

8. i = i +1

9. **else**

10. TEMP [k] = A [j]

11. j = j +1

12. **if** j > r

13. **for** t = 0 **to** q – i

14. A [r-t] = A [q-t]

15. **for** t = 0 **to** k-1

16. A [p+t] = TEMP [t+1]