Habib University - Algorithms: Design and Analysis (CS/CE 412/471) – Spring 2025

Quiz 02 Solution

Name: ____ **ID:** __________ **Section:** ________

Q1. Consider the following divide and conquer algorithm for the given problem:

Problem: Given an array A[1..n], find the maximum sum of a contiguous sequence in the array.

e.g. 1

1	2	3	4	5	6	7	8
-2	-5	6	-2	-3	1	5	-6

Maximum Sum = 7 (i.e. A[3]+A[4]+..A[7])

e.g. 2

1	2	3	4	5	6	7	8
1	-3	2	1	-1	3	-2	-4

Maximum Sum = 5 (i.e. A[3] +A[4]+..A[6])

```
FIND-MAXIMUM-CONTIGUOUS-SEQUENCE(A, left, right)
    if left = right
        return A[left]

    mid ← ⌊(left + right) / 2⌋

    left_max_sum ← FIND-MAXIMUM-CONTIGUOUS-SEQUENCE(A, left, mid)
    right_max_sum ← FIND-MAXIMUM-CONTIGUOUS-SEQUENCE(A, mid + 1, right)
    bridge_max_sum ← FIND-BRIDGE-SUM(A, left, mid, right)

    return max(left_max_sum, right_max_sum, bridge_max_sum)
```

a) **[1 point] State the input(s):** *An array A[1..n] of n numbers (positive and negative).*

b) **[1 point] State the output(s):** *Maximum sum of sub-array A[n..m] where n<=m.*

c) **[5 points] Write down the recurrence for the above algorithm assuming procedure FIND-BRIDGE-SUM takes linear time:**

$$T(n) = 2\,T\left(\frac{n}{2}\right) + \theta(n)$$

d) **[5 points] Find out the time complexity of the given algorithm using Iterative/backward substitution . (back side)**

T(n) = 2T(n/2) + cn

*T(n) = 2 [2T(n/4) + c * (n/2)] + cn*
= 4T(n/4) + cn + cn
= 4T(n/4) + 2cn

*T(n) = 4 [2T(n/8) + c * (n/4)] + 2cn*
= 8T(n/8) + cn + 2cn
= 8T(n/8) + 3cn

…

*T(n) = 2^k T(n / 2^k) + k * cn*

The recursion stops when n / 2^k = 1:

n / 2^k = 1 -> 2^k = n -> k = log_2 n

*T(n) = 2^(log_2 n) * T(1) + cn * log_2 n*
*= n * Θ(1) + cn log n*
= Θ(n) + cn log n

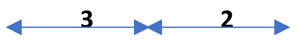
The dominating term is cn log n, so:

T(n) = Θ(n log n)

e) **[8 points]** The basic idea of the algorithm given above is to recursively find the maximum subarray in the left half and the right half, and then find the maximum subarray that crosses the midpoint. Finally, the solution is the maximum of the three subarrays.

Design the procedure **FIND-BRIDGE-SUM** which computes the maximum sum of a contiguous sequence that crosses the midpoint of the array i.e. the subarray that spans both sides from middle of the array. Consider e.g.2 again:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	**-3**	**2**	**1**	**-1**	**3**	**-2**	**-4**



bridge_max_sum = 3+2 = 5

```
FIND-BRIDGE-SUM(A, left, mid, right)
  max_left_sum ← -∞, current_sum ← 0
  for i ← mid down to left
    current_sum ← current_sum + A[i]
    max_left_sum ← max(max_left_sum, current_sum)

  max_right_sum ← -∞, current_sum ← 0
  for j ← mid + 1 to right
    current_sum ← current_sum + A[j]
    max_right_sum ← max(max_right_sum, current_sum)

  return max_left_sum + max_right_sum
```