CSE 321 Introduction to Algorithm Design

Fall 2019 – HW4

Name:Berke Süslü

Number:161044076

2) In this algorithm, i divide the both array with k/2 and then recurse until k=1.When k is 1, the kth element is the minimum of first element of array1 or array2.

In the worst case, k is biggest element in union of arrays (k=m+n). So, the time complexity is O(log (m+n)).

3) This algorithm is similiar to the finding closest pair with divide and conquer method. The array is divided from middle(n/2). To evaluate maximum subarray sum, we have 3 case:

The maximum subarray can be found in first half.

The maximum subarray can be found in second half.

The maximum subarray can be found in between first and second half.

In first two cases, we can use recurrence.

In last case, we can evaluate the value directly.

The recurrence relation is

$T(n) = 2T(n/2) + \Theta(n)$

($\Theta(n)$ is evaluating the value.)

By using Master Theorem, complexity is $\Theta(n\log n)$ (Worst case = Best case)

4) In this algorithm, i colored the graph with DFS(Depth-First-Search) method. The algorithm colors the current vertices with c and colors the neighbor of current vertices with 1-c. If the 2 vertices has an edge and they has same color, function returns false.(-1 is means not colored.)

The complexity rely on how the graph is implemented.

If the graph implemented with adjacency list, the complexity is O(V+E).

If the graph implemented with adjacency matrix, the complexity is $O(V^2)$.

5) In this algorithm, i divided the array with n/2 until the array has 1 element.While dividing the array, select the maximum gain of halves.

The recurrence relation is:

$T(n)=2T(n/2)+\Theta(1)$

By using Master Theorem, the complexity is $\Theta(n)$.