NEVRA GÜRSES
161044071

CSE 321 HOMEWORK 4

1. QUESTION:

a) We know that a special array is:

A[i,j] + A[k,l] ≤ A[i,l] + A[k,j] where **1 ≤ i < k ≤ m** and **1 ≤ j < l ≤ n**

I will prove that array is special if and only if :

A[i,j]+A[i+1,j+1] ≤ A[i,j+1]+A[i+1,j] for **i = 1, 2, ..., m − 1** and **j = 1, 2, ., n – 1**

This equation can be writed as:

A[i,j]+A[k,j+1]≤A[i,j+1]+A[k,j] where **i < k** and I assume that **k=i+1**

I will prove correctness of this equation by induction.Base case is k=i+1 where I find above. For inductive step, I assume it holds for k=i+n and I want to prove it for k+1=i+n+1.

A[i,j]+A[k,j+1] ≤ A[i,j+1]+A[k,j] => Assumption by base case (k= i + 1 where i<k)

A[k,j]+A[k+1,j+1] ≤ A[k,j+1]+A[k+1,j] => According to induction rules , if k is true by assumption ,then k+1 also true. So I will prove this:

If I sum this two equations:

A[i,j]+A[k,j+1]+A[k,j]+A[k+1,j+1] ≤ A[i,j+1]+A[k,j]+A[k,j+1]+A[k+1,j]

When simplified this equation:

A[i,j]+A[k+1,j+1]≤A[i,j+1]+A[k+1,j] . So I prove the correctness of given equation by induction.

b) **Pseudo code is:**

```
function convert2DtoSpecial (array)
    for i=0 to length of rows do
        for j=0 to length of colums do
            if (array[i][j] + array[i+1][j+1] > array[i][j+1] + array[i+1][j] then
                array[i][j+1] += (array[i][j] + array[i+1][j+1]) - (array[i][j+1] + array[i+1][j])
            end if
        end for
    end for
  return array
end
```

Explanation: In my algoritm , I use rule of special array that is:

A[i,j] + A[i + 1,j + 1] ≤ A[i,j + 1] + A[i + 1,j]

I control 2d array as quartet. If array[i][j] + array[i+1][j+1] > array[i][j+1] + array[i+1][j] then I change one block to create special array by making

array[i][j+1] += array[i][j] + array[i+1][j+1] - array[i][j+1] + array[i+1][j] . So given 2d array converted to special array.

c) In my algorithm I use merge sort algorithm that is a divide and conquer algorithm for finding leftmost minimum element in each row.I sorted each row according to small element to large element and after that I find leftmost min element in this way.Actually ,leftmost min element is first element at each row after sorting by using merge sort algorithm. Merge sort algorithm is a divide a conquer algoritm so my algorithm of finding leftmost min element also divide a conquer algorithm.

d) In my algorithm I used merge sort for finding leftmost element in each row.So I write recurrence relation for merge sort for running time of my algorithm.

In merge sort function:

If length of array > 1

- I find the middle point to divide the array into two halves:
  - middle = length/2
- I make recursive call for first half:
  - Call mergeSort(left)
- I make recursive call for second half:
  - Call mergeSort(right)
- I call merge function for merging the two halves sorted in step 2 and 3:
  - Call merge(array,left,rigth)

That is code in python file in my homework:

```
def mergeSort(arr):
    if(len(arr)>1):
        middle=(int)(len(arr)/2) #middle point
        left=arr[0:middle]  #left part of array.
        right=arr[middle : len(arr)]  #left part of array.
        mergeSort(left)
        mergeSort(right)
        merge(arr,left,right)
    return arr
```

NEVRA GÜRSES
161044071

Running time of below merge part of merge sort is **Θ(n)** because all elements is controlling.

```
def merge(arr,leftArr,rightArr):   #For merging arrays.
        i = j = k = 0
        while i < len(leftArr) and j < len(rightArr):
            if leftArr[i] < rightArr[j]:
                arr[k] = leftArr[i]
                i+=1
            else:
                arr[k] = rightArr[j]
                j+=1
            k+=1
        # Checking whether there are any element was left or not.
        while i < len(leftArr):
            arr[k] = leftArr[i]
            i+=1
            k+=1
         # Checking whether there are any element was rigth or not.
        while j < len(rightArr):
            arr[k] = rightArr[j]
            j+=1
            k+=1
```

General recurrence relation for merge sort:

Merge Sort is a recursive algorithm .Length is divided two equal part in each step and than called merge function that is running rime **Θ(n) .**

T(n) = 2T(n/2) + Θ(n)

After sorted each row, leftmost min element is first element of row.So this operation is O(1) time.

Summary for recurrence relation of my algorithm is:

T(n) = 2T(n/2) + Θ(n)

This relation is solving by Master Theorem :

a=2 b=2 c=1

c= $\log_b a$ **=> 1=** $\log_2 2$ ** So time complexity is: Θ (nlogn)of my algorithm.**

NEVRA GÜRSES
161044071

2. QUESTION:

Firstly,I find median elements and median indices of array A and array B.Then if k is bigger than the sum of median indices of array A and array B and if median element of array A is bigger than median element of array B, I make recursive call and in this recursive calling first half of array B doesn't include k.So I extract first half of array B. On the other hand , if k is bigger than the sum of median indices of array A and array B and if median element of array B is bigger than median element of array A, I make recursive call and in this recursive calling first half of array A doesn't include k.So I extract first half of array A in that case. For another cases that are:

If k is smaller than the sum of median indices of array A and array B and if median element of array A is bigger than median element of array B, I make recursive call and in this recursive calling second half of array A doesn't include k. On the other hand , if k is smaller than the sum of median indices of array A and array B and if median element of array B is bigger than median element of array A, I make recursive call and in this recursive calling second half of array B doesn't include k.

Actually in each step, problem is diveded into 2 sub problems by recursively.

So by these recursive callings according to explained conditions, kth element of the merged array of these two sorted arrays is found without merge first after find kth elemet later.

Worst Case of my algorithm:

In my algorithm there is a binary searching in each recursive callings.In each step problem is divided into 2 sub problems. We know that size of first array is m and size of second array is n.

m + n = 2^K -1 . There are k comparisions in worst case complexity.

So , **k = $\log_2(m+n)$**

Briefly worst case is O ($\log_2(m+n)$)

NEVRA GÜRSES
161044071

3. QUESTION:

Firstly, I find the middle point of array and I divide the array into 2 parts.Then I make recursive calls for left part of array and right part of array.After I return the maximum of following three :

- Maximum subarray sum in left half by making a recursive call
- Maximum subarray sum in right half by making a recursive call
- Maximum subarray sum such that the subarray crosses the midpoint.

For finding crossing sum, I find the maximum sum starting from mid point and ending at some point on left of middle point, then I find the maximum sum starting from mid + 1 and ending with sum point on right of midpoint + 1. Finally, I combine these two and return.

Recurrence relation of my algorithm is:

T(n) = 2T(n/2) + Θ(n) because there are 2 recursive call by dividing array into 2 part and also calling for crossing left and right parts function that is linear time Θ(n) complexity.

Worst-case running time is:

This relation is solving by Master Theorem :

a=2 b=2 c=1

c= $\log_b a$ => 1= $\log_2 2$ So worst-case running time is: Θ (nlogn)of my algorithm.

4. QUESTION:

In my algorithm I apply DFS by recursively for coloring section. In coloring section,there are 2 colors that for first color I assign number 1 for node ,for second color I assign number 0 for node. Nodes that are consequtively must be different color in DFS . If all nodes can be colored in this way, this is a bipartite graph so this graph can be divided into two disjoint and independent sets U and V such that every edge connects a vertex in U to one in V by making coloring operation.DFS is decrease and conquer algorithm so my algorithm is also decrease and conquer algorithm.

Worst-Case Running Time:

Actually ,the worst-case running time should be sum of V+ E where V is vertex number and E is edge number if adjajency list is used . But in my algorithm I used adjacency matrix and in adjajency matrix E is close to V^2 so the worst case complexity is **O(V^2)**

NEVRA GÜRSES
161044071

5. QUESTION:

In my algorithm, firstly I find gain array according to cost array and price array.After that I call merge sort function for sorting gain array .After sorting, the best day is last element of gain array. Merge sort function is divide and conquer algorithm.So my algorithm is also divide and conquer algorithm.

Recurrence relation is:

T(n) = 2T(n/2) + Θ(n)

Worst-case running time is:

This relation is solving by Master Theorem :

a=2 b=2 c=1

c= $\log_b a$ => 1= $\log_2 2$ So worst-case running time is: Θ (nlogn)of my algorithm.