CSE 321 HW_4 REPORT

YAGIZ DÖNER
141044062

1)

a) $i = 1, 2, \ldots, m-1 \qquad j = 1, 2, \ldots, n-1$

$$A[i,j] + A[i+1, j+1] \leq A[i, j+1] + A[i+1, j]$$

(k) (l) (l) (k)

Let; $j+1 = l$ and $i+1 = k$

for $l \Rightarrow 1, 2, \ldots, n$ and $\geq j$

for $k \Rightarrow 1, 2, \ldots, m$ and $\geq i$

So: $1 \leq j \leq l \leq n \quad , \quad 1 \leq i \leq k \leq m$

AND $A[i,j] + A[k,l] \leq A[i,l] + A[k,j]$ is a special array.

b) pseudocode

```
for i ← 0 to length of array - 1 :
    for j ← 0 to len of array[i] - 1 :
        if condition is not special array:
            convert to special array
```

→ Two for loops are searching the array and convert to special array. In example, I give an array which can be converted to special array with one move.

Part C)

I write a function that calculates the leftmost minimum in the given row. This function is looking to the left and right part of the row and then returns the minimum elemenet.

I give the first row to this function and then call the remainig rows recursively (Decrease & Conquer). For this, I use a helper function.

Part D)

For findMin function: $T(n) = T(n/2) + 1$ -> From Master The. -> $T(n) = O(\lg n)$

For helper Function : $T(n) = T(n-1) + \lg n$ -> lgn is from findMin function.

$T(n-1) = T(n-2) + \lg(n-1) + \lg(n)$ -> $T(n) = T(n-k) + \lg(n * (n-1) * \ldots * (n-k))$

Let , k= n-1 and T(1) =1 $T(n) = T(1) + \lg(n!)$

Using Stirling's approximation, O(logn!) = O(n logn)

2) the middle points of two arrays are m1, m2

According to m1 + m2, I call the functions recursively.

In every iteration, I call the function in length / 2.
(0 ... mid or mid ... last etc.)

$T(n) = T(n/2) + 1$ → From m.T

$a = 1$
$b = 2$
$d = 0$ ⇒ $\underline{O(\lg n)}$

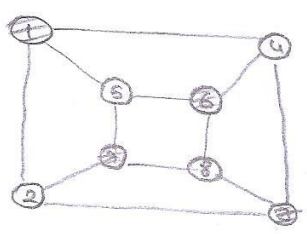
3) The list is divided in two parts (left and right).

$T(n) = T(n/2) + 1$ → From m.T ⇒ $O(\lg n)$

Also, There is combine operation. For n element, this operation takes $O(n)$ time.

So, the complexity is $O(n \lg n)$.

4) Ex:

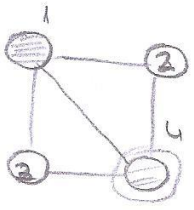
1) 

V = [1, 2, 3, 4, 5, 6, 7, 8]

E = [[2,4,5] ..... [4, 5, 8] ...]

I give the input like this.

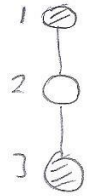
Return True.

2) 

return False.

V = [1, 2, 3, 4]

E = [[2, 3, 4], [1, 4], [1, 4], [1, 2, 3]]

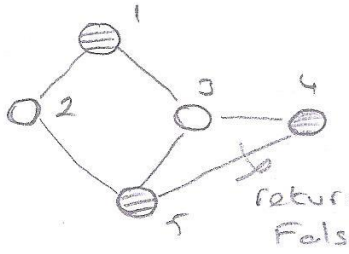
3) 

V = [1, 2, 3]

E = [[2], [1, 3], [2]]

return True

Ex: 4)



$V = [1, 2, 3, 4, 5]$

$E = [[2,3], \ldots [2,3,4]]$

→ For this part, I create the color array with lenght for vertex array. In initial, I give (-1) for "not visited."

I started from first index. I make it zero (black) and also its neighbour are 1 (white).

And then, I call the function recursively without first index. This is the decrease & conquer part.

In next iterations, if there is similer colors in order (I mean, vertex is 1 and its neighbour is also one.) zero.

, return false.

At the end, if the list is empty, return true.

Running Time => Worst Case :

$T(n) = T(n-1) + 1$

$T(n-1) = T(n-2) + 1 + 1$

$T(n) = T(n-k) + k$ → $k = n-1$, $T(1) = 1$

$T(n) = T(1) + n - 1 \Rightarrow T(n) \in O(n)$

PART 5) For this part, I give the arrays. For (-) place, I give the -1 in examples. I trim this -1. and use helper function.

I return the maximum profit and their index.

For this, I divide the array from middle point. and search the maximum in right and left part.

$T(n) = T(n/2) + 1$ → From M.T => $\Theta(\lg n)$

Yağız DÖNER
141044062