69361

Özlem ŞERİFOĞULLARI

I have completed this assignment individually, without support from anyone else. I hereby accept that only the below listed sources are approved to be used during this assingnment:

(i) Course textbook,

(ii) All material that is made available to me by the professor (e.g., via Blackboard for this course, course website, email from professor / TA),

(iii) Notes taken by me during lectures.

I have not used, accessed or taken any unpermitted information from any other source. Hence, all effort belongs to me.

Pseudo Code:

```
func(array[][], int row, int lowerbound, int upperbound){
    if row is equal to n for nxn matrix:
        return 0
    if upperbound ~~don't~~ equal to lowerbound:
        midpoint = lowerbound + (upperbound-lowerbound)/2
        if array[row][midpoint] is char'a':
            lowerbound will be midpoint+1
            return func(array[][], row, lowerbound, upperbound)
        else:
            upperbound will be midpoint
            return func(array[], row, lowerbound, upperbound)
    go to next row
    return lowerbound + func(array[],[], row, upperbound =n-1, lowerbound =0)
```

in same row

```
acount ( char[][] mat) {
    int result = 0;

    if n=1 for nxn matrix:
        if mat[0][0] is char 'a':
            result = 1

    Otherwise:
        result = func(mat, 0, 0, (n-1))

    return result
```

Complexity analysis

1) Time

For each row the algorithm's comperxity is the same as binary search. For a $n \times n$ matrix the complexity of each row is $O(\log n)$. There are n rows so the overall complexity is $n \cdot O(\log n)$. Hence the time complexity is $O(n \log n)$

For each row → binary search algorithm

$T(1) = 2$

$T(n) = 2 + T(n/2) = 2 + (2 + T(n/4)) = 4 + (2 + T(n/8))$

$= 2k + T\left(\frac{n}{2^k}\right)$

$n = 2^k$ $\quad k = \log n$

$2 \log n = O(\log n)$

2) Space

array → $N \times N \times 1$ byte

row → 4 bytes

lowebound → 4 bytes

upperbound → 4 bytes

midpoint → 4 bytes

Auxillary space → 4 bytes

\+

$(N^2 + 20 \text{ bytes}) \times N \log N$ (recursion)

$= N^3 \log N + 20 \times N \log N$

$= O(N^3 \log N)$