

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 assignment:

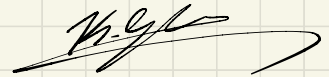
(i) Course Textbook

(ii) All material that is made available to me by 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.

Kerem Güvener



```
int count(char[][] mat)
```

```
result = 0
```

```
n = mat.length
```

```
int s;
```

```
int e;
```

```
for (row = 0 to n)
```

```
    s = 0
```

```
    e = n - 1
```

```
    result = result + rB(mat[row], s, e)
```

```
return result
```

```
// find first occurrence of 'b' using binary search -  $O(\log n)$ 
```

```
rB(row[], int s, int e)
```

```
col = s + (e - s) / 2
```

```
if (row[col] == 'b')
```

```
    if (col == 0) return 0;
```

```
    if (row[col - 1] == 'a') return col;
```

```
    if (row[col - 1] == 'b')
```

```
        e = col - 1;
```

```
        return rB(row[], s, e);
```

```
if (row[col] == 'a')
```

```
    if (col == n - 1) return n;
```

```
    s = col + 1;
```

```
    return rB(row[], s, e);
```

Big-O Complexity: For this algorithm, I have used a custom recursive binary search to locate the first occurrences of 'b' in each row in a 0 to n for loop. The recursive binary search part has Big-O Complexity of $O(\log n)$, and the for loop contributes a Big-O Complexity of $O(n)$. Therefore, the complete algorithm has a Big-O Complexity of $O(n \log n)$.