Wild Card Matching

? = matches any single character

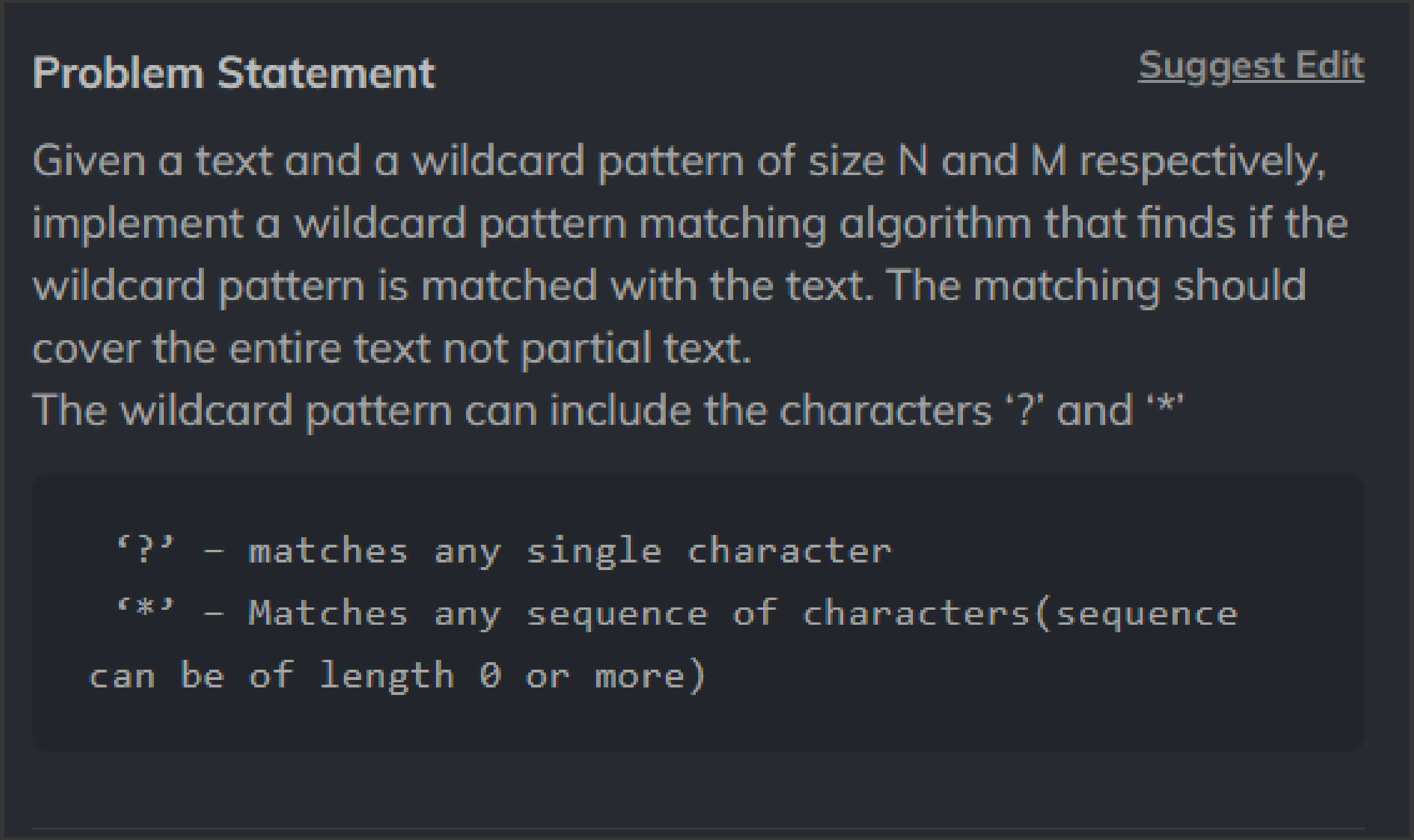
* = matches any sequence of character of length 0 or more

S1 = "?ay"
S2 = "Tay"
} True

S1 = "~~a~~~~b~~*~~c~~~~d~~"
S2 = "~~a~~~~b~~def~~c~~~~d~~"
} True

S1 = "~~a~~~~b~~? d"
S2 = "~~a~~~~b~~~~c~~c"
} false.

S1 = "* * abcd"
S2 = "abcd"
} True



Problem Statement

Suggest Edit

Given a text and a wildcard pattern of size N and M respectively, implement a wildcard pattern matching algorithm that finds if the wildcard pattern is matched with the text. The matching should cover the entire text not partial text.

The wildcard pattern can include the characters '?' and '*'

```
   '?' – matches any single character
   '*' – Matches any sequence of characters(sequence
  can be of length 0 or more)
```

return false || true.

Appoooach!

String matching Recursion

ab*cd $\uparrow$ f(n-1, m-1)

abdefcd $\uparrow$ f(4,6)

$\Downarrow$

ia s[0....4] = ~~t~~ [0.....6]

- Express f(i, j)
- Try all ways
- True || false

Recursion.

$\overset{i}{ab * cd}$

ab def c $\overset{j}{d}$

f(i, j)

if(i<0 && j<0) return true;

if (i<0 && j>=0) return false;

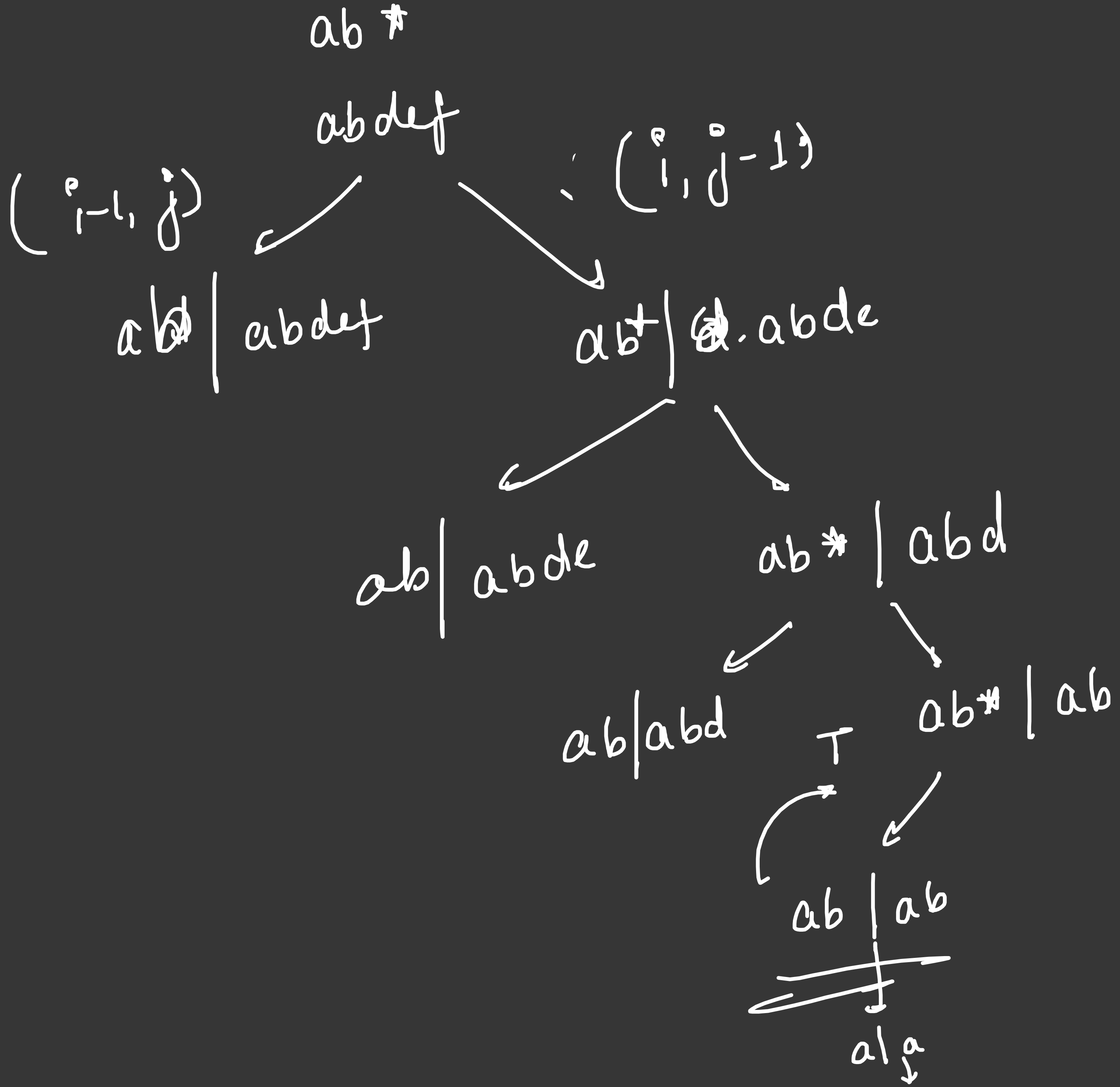
if(j<0 and i>=0) for ——

if(s[i] == t[j] || s1[i] == '?') return f(i-1, j-1);

if (s[i] == '*')

f(i-1, j) || f(i, j-1)

return false


ab*
abdef
(i-1, j)
(i, j-1)
ab | abdef
ab* | abde
ab | abde
ab* | abd
ab | abd
T
ab* | ab
ab | ab
a | a

// Base Case:

if S1 get exhausted

```
i<0 && j<0
    return True

if(i<0 && j>=0) return false
```

if S2 get exhausted

```
if (j<0 && i>=0)
    if (s[i] == '*')
    for ( k=0 , k<i )
        if(s[i] != "*")
            return false
    return true
```

Time Complexity : Exponential

S.C : O(N+M) Auxillary Space

Memoization

$dp[n][m]$

$dp[n-1][m-1]$

T.C: $O(N \times M)$

S.C. $O(N \times M) + \underbrace{O(N+M)}_{\text{Auxiliary Space}}$

Tabulation

if (i == 0 && j == 0) true

i == 0, j > 0

Base Case:

```
if dp[0][0] = 1;

i > 0

for(i=1; i<n; i++)
  if (s1[i-1] == '*')
    dp[i][0] = 1;
```

Space Optimised:

dp[0][0]

prev[0] = 1

j = 1; j < m; j++)

prev[j] = ~~t~~false

for curr