Welcome ☺

Agenda : DP on strings.
3 questions.

Q Find the min. cost to convert string A to string B.

Cost of → 1) insertion of any char → 2
2) deletion of any char → 2
3) update any char → 3

eg: A : a b $\overset{x}{\cancel{c}}$ $\cancel{d}$
B : a b x

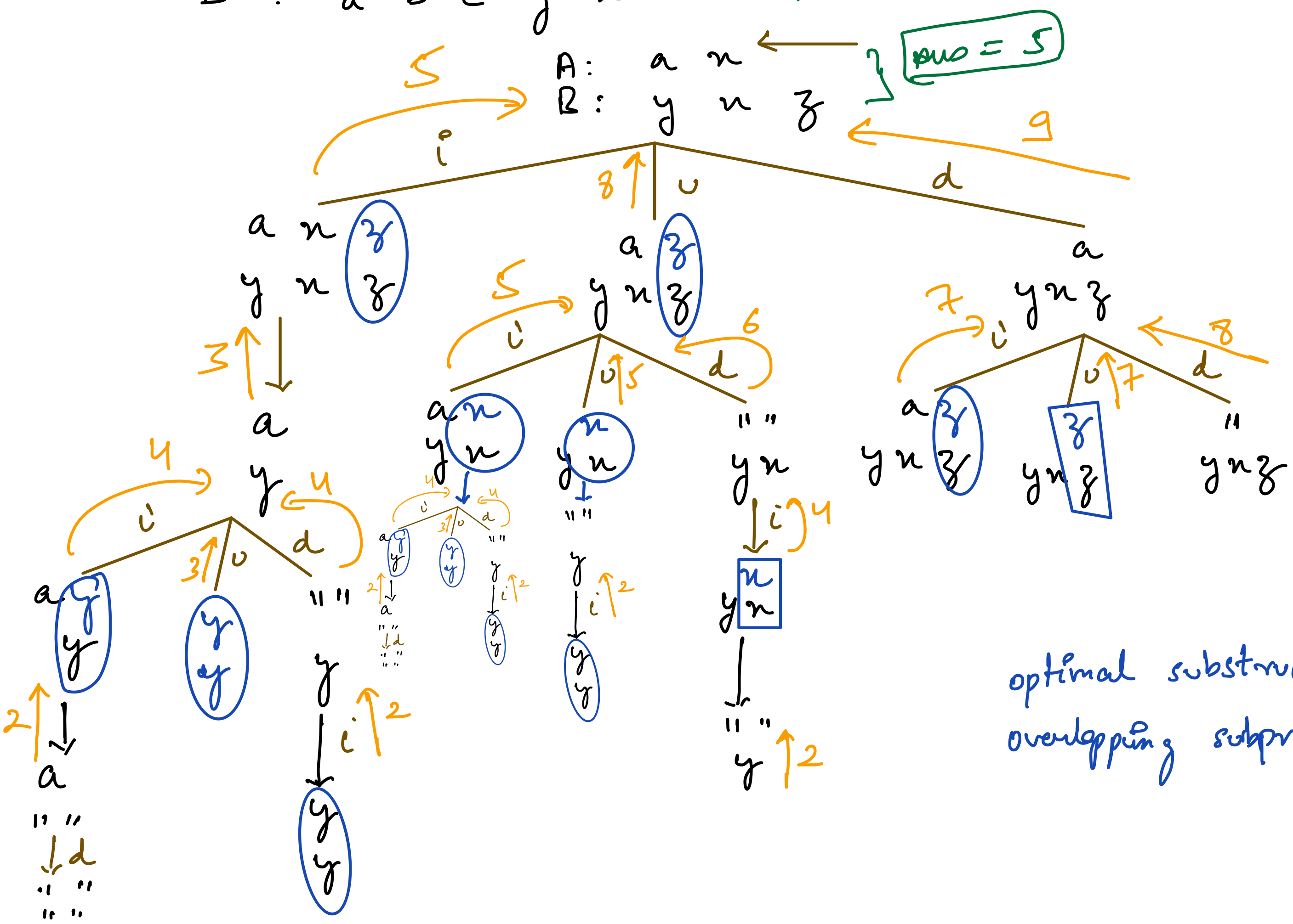
Ans = 3 + 2
= 5

A : a $\overset{b}{\wedge}$ c
B : a b c

Ans = 2

A : a $\overset{b}{\wedge}$ c $\overset{g}{\cancel{d}}$ x $\cancel{y}$
B : a b c g x

Ans = 3 + 2 + 2
= 7



optimal substructure. ✓
overlapping subproblems ✓

cost[i][j] ⟹ min cost to convert first i characters of A to first j characters of B

```
                 → if ( i==0 && j==0 )  → 0
                 → else if ( i==0 )     → 2*j
cost[i][j] ──────→ else if ( j==0 )     → 2*i
                 → else if ( A[i]==B[j] ) → cost[i-1][j-1]
                 → else   min ( 2 + cost[i][j-1]     // insertion
                                3 + cost[i-1][j-1]   // updation
                                2 + cost[i-1][j]     // deletion
```

Code:

```
for i → 0 to N
{
    for j → 0 to M
    {
        if ( i==0 || j==0 )  cost[i][j] = 2*(i+j)
        else if ( A[i]==B[j] )  cost[i][j] = cost[i-1][j-1]
        else cost[i][j] = min ( 2 + cost[i][j-1]
                                3 + cost[i-1][j-1]
                                2 + cost[i-1][j]
    }
}
```

T.C → O(N*M)

S.C → O(N*M) → O(2*M) ≃ O(M)

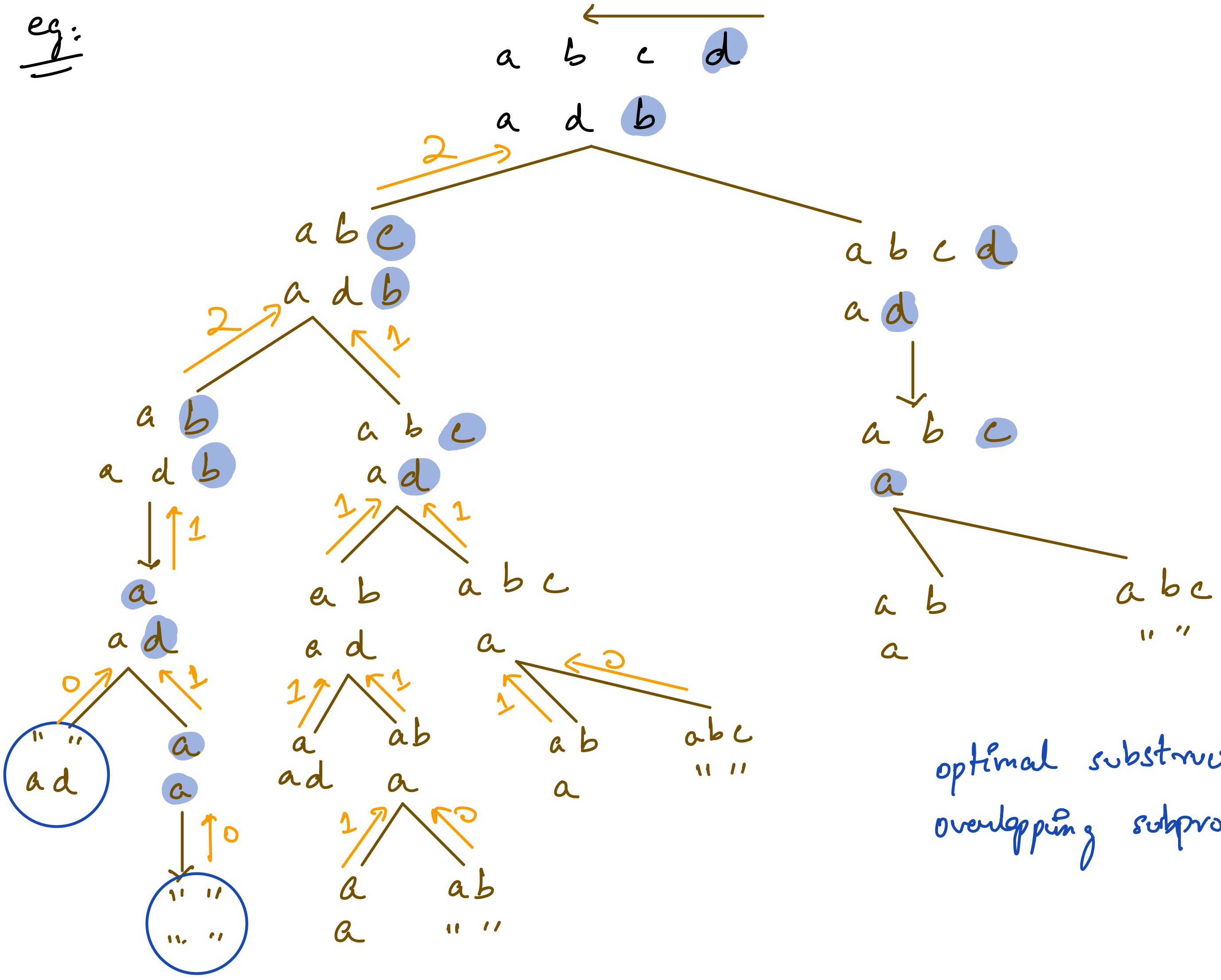
Q Find the length of longest common subsequence (LCS) among 2 strings.

Intuit

A : a b c d
B : a d b
Ans = 2

A : a b x y
B : b x a y
Ans = 3

eg:



optimal substructure. ✓

overlapping subproblems ✓

lcs [i][j] ⟹ length of longest common subsequence of first i characters, of A

& first j characters of B

```
lcs → if ( i==0 || j==0 ) → 0
      else if ( A[i] == B[j] ) → 1 + lcs[i-1][j-1]
      else   max ( lcs[i][j-1] , lcs[i-1][j] )
```

T.C → O(N*M)

S.C → O(N*M) → O(2*M) ≃ O(M)

Q Wildcard Matching / Regular Expression Match

Check if the string (regular exp) R matches with string S.

"?" → match with any 1 character (exactly 1)

"*" → match with any number of characters. (including empty sequence)

eg:

R : a*a

S : a b a ✓

S : a b b a ✓

S : a b a b ✗

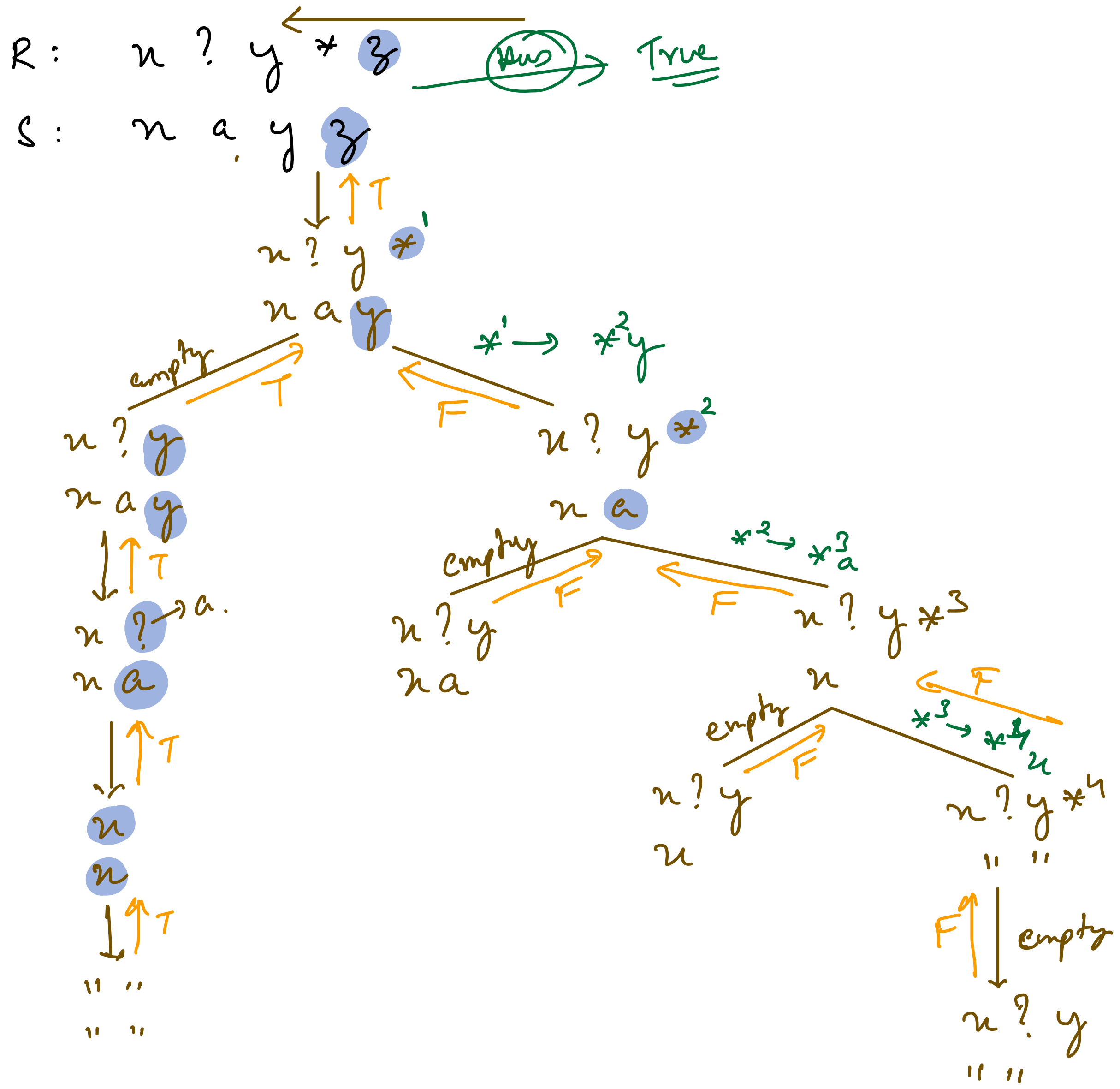
R : x ? y * z

S : x a y z ✓ ?→a *→""

S : x y z ✗

S : x x y y y z z ✓ ?→x *→yyz

S : x x x y y z z ✗



```
match[i][j] ⇒ check if first i characters of R
              matches with first j characters of S

matches[i][j] → if( i==0 && j==0) → True
                else if ( i==0) → false.
                else if ( j==0) {
                    if( R[i] == *) → match[i-1][j]
                    else → false.
                }
                else if ( R[i] == S[j] ) → match[i-1][j-1]
                else if ( R[i] == '?') → match[i-1][j-1]
```

else if (R[i] = = '*') → match[i-1][j]

||

match[i][j-1]

else → false.

T.C → O(N*M)

S.C → O(N*M) → O(2*M) ≃ O(M)
