DSA-501

LECTURE #8

TOPIC: DP-4

Edit Distance →

autocomplete

str1 = abcde
str2 = abcg

convert

(I)

abcde

2 delete

abc

+1 add" ⇒ 3

abcg

opers

(1) insert" a char
(2) delete " "
(3) replace a char

(II)

abcde

1 delete

abcd

1 replace

abcg

Min edit dist = 2

Edit Distance

→ Min # operations on S1 to convert it to S2

S1 = "xyz"

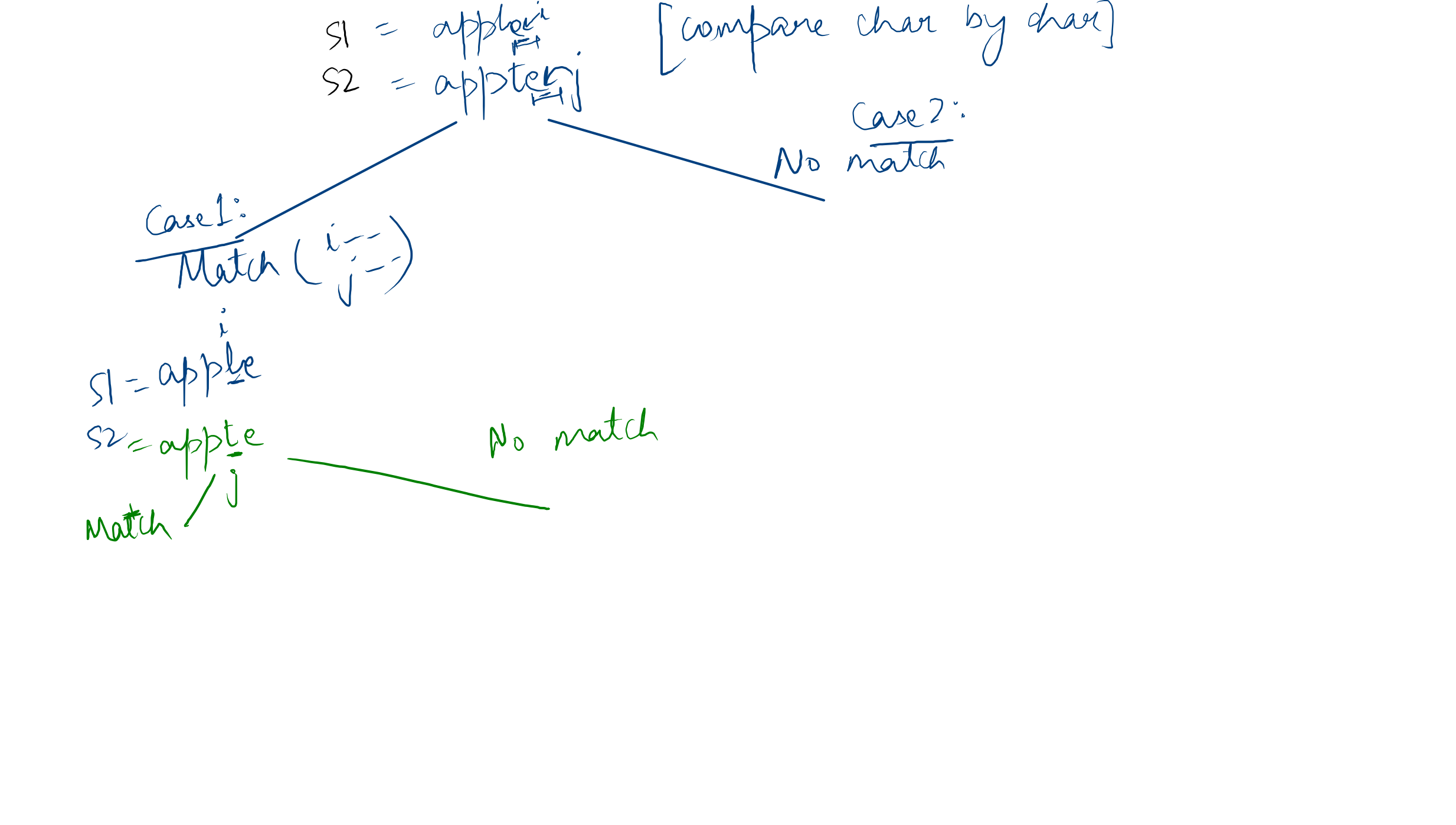
S2 = "xypq"

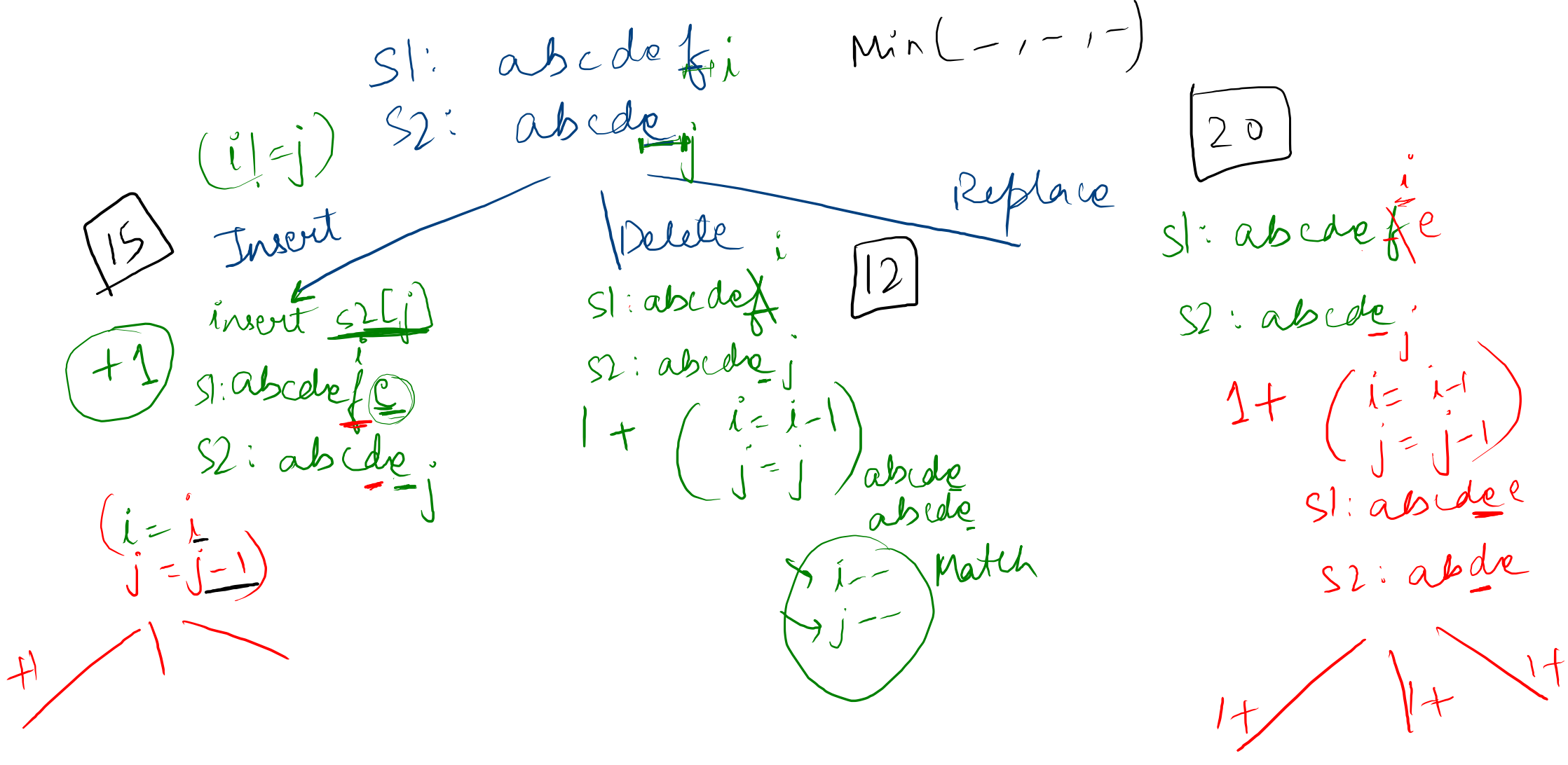
s1 = abcde

s2 = abxde

} edit dis = ? 1

op^ = replace


S1 = apple
S2 = apter
[compare char by char]
Case 2:
No match
Case 1:
Match (i--, j--)
i
S1 = apple
S2 = apte
j
No match
Match


S1: abcdef i
S2: abcde j
Min(-,-,-)
(i != j)
15
Insert
+1
insert S2[j]
S1: abcdef e
S2: abcde j
(i = i
j = j-1)
+1
1
Delete i
S1: abcde f
S2: abcde j
12
1 + (i = i-1, j = j)
abcde
abcde
Match
i--
j--
Replace
20
S1: abcdef e
S2: abcde j
1+ (i = i-1, j = j-1)
S1: abcdee
S2: abcde
1+
1+
1+

Recursive f^N

```
editDist ( str1 , str2 , i , j )
{
    // 1. base cond^N
        if(i==0)   ret j
        if(j==0)   ret i

    // 2. recursive calls
        // Match
        if(str1[i-1] == str2[j-1])
            ret editD(str1, str2, i-1, j-1)

        // No Match
        else Min ( 1 + editD(str1, str2, i , j-1) // insert
                   1 + editD(str1, str2, i-1, j) // delete
                   1 + editD(str1, str2, i-1, j-1) // replace)
}
```

S1: abc P Z

S2: P Q

edD = i

⇒ editDist (s1, s2, len1, len2)

TC = O(3^N)

Memoization

N → len 1

M → len 2

I) create Data Stru — vars = 2

dp[N+1][M+1] = -1 // all with -1

II) Fill dp array recursively

```
EditDist( str1, str2, N, M)
{
        if(N==0) return M;
        if(M==0) return N;

        if(dp[N][M] != -1)  return dp[N][M]

        //recursive calls
        //match
        if( str1[N-1] == str2[M-1])
                return dp[N][M]= EditDist(str1, str2, N-1, M-1)

        //no match
        else{
                return dp[N][M] = min( 1+ EditDist(str1, str2, N, M-1) //ins
                                , 1+ EditDist(str1, str2, N-1, M) //del
                                  , 1+ EditDist( str1, str2, N-1,M-1) )
        }

}
```

TC = O(N*M)

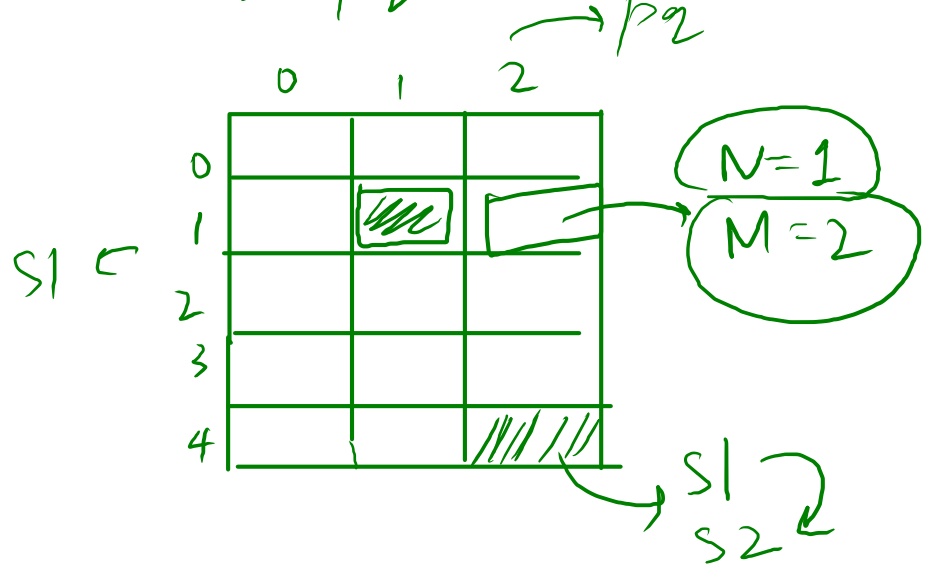
SC = O(N*M)

N+1 M+1

S1: abpq

S2: pq

dp[5][3]



TABULATION

I) create data struct

dp[N+1][M+1]

II) Initialize

III) Filling iteratively

```
//init
for(i=0;i<=M;i++)
        dp[0][i] = i
for(i=1;i<=N;i++)
        dp[i][0] = i

//filling rest of the matrix
for(i=1;i<=N;i++){
    for(j=1;j<=M;j++){
        if( s1[ i-1] == s2[j-1])
            dp[i] [j ] = dp [i-1][j-1]

        else
            dp[i][j] = 1 + min( dp[i][ j-1],
                                dp[ i-1][j] , dp[ i-1][j-1])
    }
}
return dp[N][M]
```

TC = O(N*M)

SC = O(N*M)

s1 a
s2 a

s1 = abcf

s2 = aec

eD = 2

N = 4
M = 3

dp[5][4]

i = 1 2 3 4

j = 1

s1[3] == s2[0]

f == a

j = 2

s1[3] == s2[1]

f == e

j = 3

s1[3] == s2[2]

f == c

	0	1	2	3
0	0	1	2	3
1	1	0	1	2
2	2	1	1	2
3	3	2	2	1
4	4	3	3	2

s1 abcf

s2 a

s2 = a e c

s1

dp