DP Concepts & Questions

Video 22

भाषण
(Motivation)

" Your parents have worked really hard to give you a good life. ♡ work hard to let them know that their hard work didn't go in vain "

[cswithMIK → Twitter
Facebook, Instagram → codestorywithMIK
whatsapp → codestorywithMIK]

Done
- 1-D based DP
- 2-D based DP
- Progress String based DP
- Grid based DP
- Game Strategy

We'll do :-

(·) RECURSION + MEMOIZATION (Top Down)

(·) Bottom Up.

(·) Time & Space

DP on Strings :-

→ Longest Common Subsequence (LCS)

→ Print LCS

→ Edit Distance

→ Shortest common Supersequence. (SCS)

→ Print SCS

→ Longest Palindromic Subsequence

→ More coming ..... :)

(Hard Qns → Easy) many more...

Google amazon Microsoft

72. Edit Distance

Medium | Topics | Companies

Given two strings `word1` and `word2`, return *the minimum number of operations required to convert* `word1` *to* `word2`.

You have the following three operations permitted on a word:

- Insert a character
- Delete a character
- Replace a character

Example :- word1 = "horse" → rorse → rose → ros

word2 = "ros"

Output = 3

word1 = "intention"

ex c

word2 = "execution"

1 + 1 + 1 + 1 + 1

Output = 5

Thought Process ...

Options – Recursion (Tree)

"horse", "ros"

1 (insert r) (i, j+1) → horse, ros

1 delete (h) (i+1, j) → orse, ros

replace h with r (1) (i+1, j+1) → rorse, ros

0 (i+1, j+1) → rorse, ros

insert s (1) (i, j+1) → rorse, ros

3

(m) (n)

if (j == n)

return m-i ; // deletions

if (i == m)

n-j ; // insert

delete (i+1, j) → rorse, ros

i+1, j+1

rorse, ros

3

5

a b c d

wo 1 1

a b c d

a 2.

$\underbrace{xyz}$

↓

delete 2

x

Story To Code :-

```
Solve( s1, s2, 0,0);

int Solve( s1, s2, i, j) {
    if (i == s1.length()) {
        return n-j ; //insert
    }
    else if (j == s2.length()) {
        return m-i ; //deletions
    }

    if (s1[i] == s2[j]) {
        return solve(s1, s2, i+1, j+1);
    }

    int insert = 1 + solve(s1,s2, i, j+1);
```

ab$\cancel{c}^{i}$ ab^{j}

```
int delete = 1 + solve(s1, s2, i+1, j);
int replace = 1 + solve(s1, s2, i+1, j+1);

return min({insert, delete, replace});
}
```

Memoization:- t[ ][ ];

→ i=0, j=0 start

→ i=m, j=n start

i=m, j=n

m=2
n=3

"ab" , "cbe"
(m) (n)

s1[m-1]
s2[n-1]
insert C
delete C
replace C

1+ solve(m, n-1)

$\boxed{ab}$, $\boxed{cb}e$

m=2 n=2

solve(m-1, n-1)

$\boxed{a}b$, $\boxed{c}be$

m=1 n=1

replace

1+ solve(m-1, n-1)

cb , cbe

m=0 n=0

"cb a" , "a"

m=3 n=1

solve(m-1, n-1)

$\boxed{cb}a$, "a"

m=2 n=0

cb "a" , "cba"

m=1 n=3

```
if(n==0) {
    return m; // deletions.
```

solve(m-1, n-1)

m=0 , n=2

} else if (m==0) {

return n // insertions

}

if (m==0 || n==0) {

return m+n ;

}

Solve (S1, S2, m, n);

Why I prefer right to left :-

i=m, j=n

```
int solve(string& s1, string& s2, int i, int j) {

    if(i == m) {
        return n-j; //insert in s1
    } else if(j == n) {
        return m-i; //delete from s1
    }

    if(t[i][j] != -1) {
        return t[i][j];
    }

    if(s1[i] == s2[j]) {
        return t[i][j] = solve(s1, s2, i+1, j+1);
    } else {
        int insertC  = 1 + solve(s1, s2, i, j+1);
        int deleteC  = 1 + solve(s1, s2, i+1, j);
        int replaceC = 1 + solve(s1, s2, i+1, j+1);

        return t[i][j] = min({insertC, deleteC, replaceC});
```

H.W.

```
int solve(string& s1, string& s2, int m, int n) {

    if(m == 0 || n == 0) {
        return m + n;
    }

    if(t[m][n] != -1) {
        return t[m][n];
    }

    if(s1[m-1] == s2[n-1]) {
        return t[m][n] = solve(s1, s2, m-1, n-1);
    } else {
        int insertC  = 1 + solve(s1, s2, m, n-1);
        int deleteC  = 1 + solve(s1, s2, m-1, n);
        int replaceC = 1 + solve(s1, s2, m-1, n-1);

        return t[m][n] = min({insertC, deleteC, replaceC});
```

i j

```
    return t[i][j] = min(insert, delete, replace);
    }

    return -1;
}
```

```
    return t[m][n] = min(insert, delete, replace);
    }

    return -1;
}
```

// State Definition:-

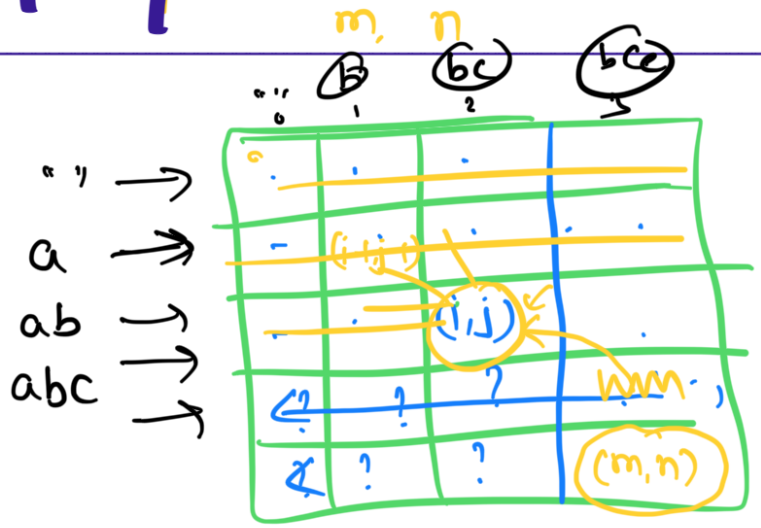
$t[i][j]$ = min opns for making S1 of length i & S2 length j equal

m, n

return t[m][n];

abc

bce



(m+1)(n+1)

t

Bottom up :-

m ⇐ for (i=0 ; i<=m ; i++) { → i

n ⇐ for (j=0; j<=n ; j++) { → j

if (i==0 || j==0) {

$t.c = O(m*n)$

$s.c = O(m*n)$

```
t[i][j] = 1+j;
} else if ( s1[i-1] == s2[j-1]) {
    t[i][j] = t[i-1][j-1];
} else {
    t[i][j]=1+min{ t[i][j-1]),
                   t[i-1][j]),
                   t[i-1][j-1]; }
}

}

return t[m][n];
}
```