Distinct Subsequences

s1 = "babgbag"

s2 = "bag"

babgbag

We have to count the no. of ways?

→ Recursion.

babg bag

i

bag

j

Recursion Approach:

- We have 2 strings. so,

match and non match concept

$f(ind1, ind2) \longrightarrow$ Express in Index

Base case

$\text{if } (\underline{j<0}) \text{ return } \underline{1};$

$\text{if } (i<0) \text{ return } 0;$

match

$\text{if } (s1[i] == s2[j]) = \underbrace{f(i-1, j-1) + f(i-1, j)}$

Summation of ways

Explore

Not match else

$f(i-1, j)$

Base Case

When do I return 1? → When all char of S2 ~~are~~ then I return 1.

matching
non-matching → common subsequence

but we have to find subsequence also

Not Take

Take

2^N

Recursion Tree:

0-based index $\underline{\text{ing}}$

b a bg bag ~~g~~

bag

f(6, 2) → match

take

not

f(5, 1)

f(5, 2)

match

3

match

f(4, 0)

f(4, 1)

1

2

f(3, -1)

f(3, 0) not match

1+1

1

f(2, 0) match

1

f(1, -1)

f(1, 0)

1 + 0

1

f(0, 0)

1 0

$f(-1,01)$ $f(-1,0)$

There would be overlapping sub problems.

→ So, we use Memoization

$dp \longrightarrow dp[n][m];$

Tabulation:

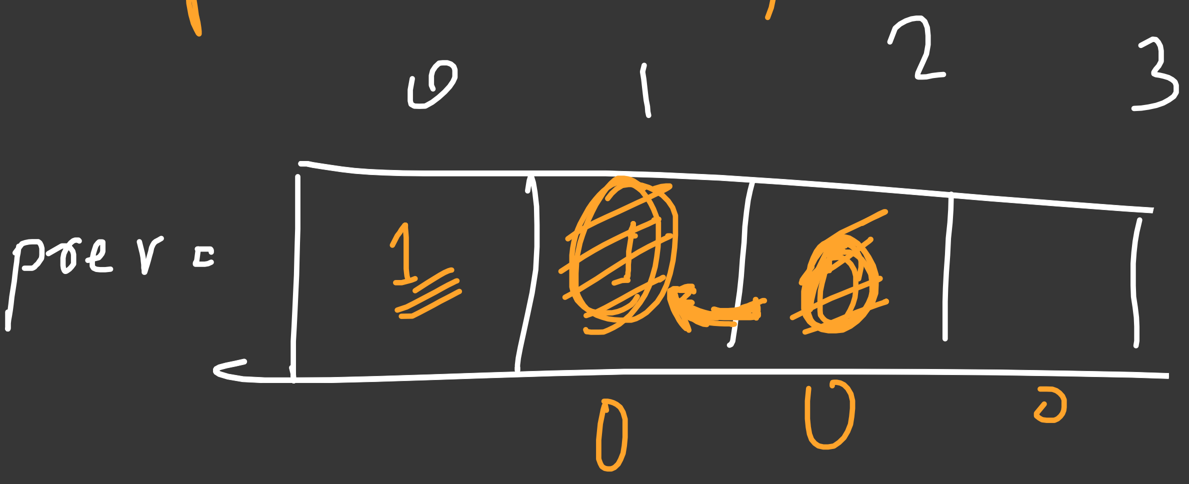
Base Case: if (i==0) → means for every i if 0 we have 0

if (j==0)

Space Optimization

In 1D array.

Observe Space Optimization



$i = 0$ prev $\quad j = 0$ prev

prev [0] = 1

$j = 1$

dp [

babgbag

bag

if (

dp[i-1] [j-1] + dp[i-1] [j]

prev

prev

prev[i] [j] = p[i] [j]

curr