Que Distinct Subsequences

ex S1 = "babgbag" S2 = "bag" $\xrightarrow{ans}$ 5

Que-1 :- why do we apply recursion & why not simple matching? or simple comparing?

⇒ S1 = "babgbag" S2 = "bag"

suppose "babgag" → first matching

ba bgag → 2nd "

→ In these two matching we can take two different 'g' also we can take two or more different 'b's

so we have Different methodolies of comparing

and This is type of **Trying All Ways**

So we apply recursion

Que-2 :- How to write Recurrance

Rules

1. express everything in terms of index (i, j)
2. explore all possibilites
3. Return the submission of all possibilities
4. Base case

expand rules

(1)

n → s1 = "babgbag" ← j

m ← s2 = "bag" ← j

$f(n-1, m-1) \rightarrow$ No. of distinct subsequence of s2[0------j] in s1[0--- i]

$\underset{i}{\uparrow}$ $\underset{j}{\uparrow}$; j → m-1 ; i → n-1

(2) All possibilites

two possibilites

Matched

"bagbgbag" , "bag"
↑ i ↑ j

Consider matched character (take)
↓
if (take)
↳ call for
f(i-1, j-1),

Not consider matched character (Not take)
↓
if (Not take)
↓
call for
f(i-1, j)

Not consider means we are looking for another occorance of 'g'

Not Matched

"bagbgbag" "bag"
↑ i ↑ j
↓
call for
f(i-1, j)

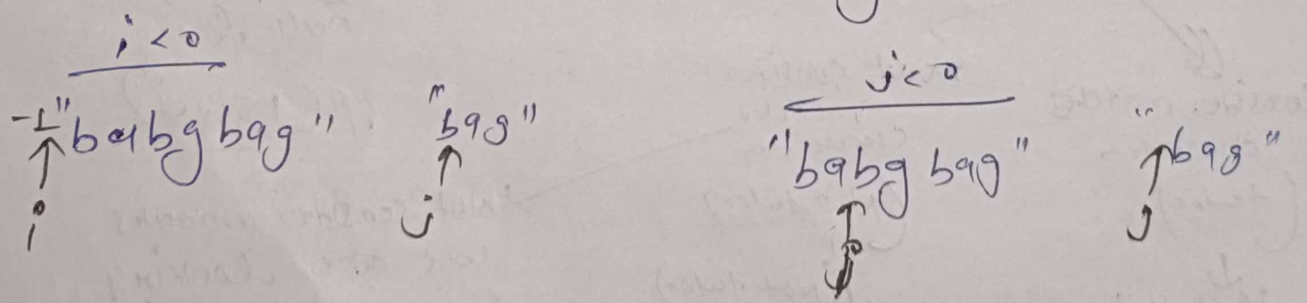
```
f( i, j)
{
    //Base case
    if(j<0) return 1;
    if(i<0) return 0;

    //all possibilities
    if(s1[i] == s2[j])
        return f(i-1, j-1) + f(i-1, j)
    else
        return f(i-1, j)

}
```

T.C. = exponential
S.C. = O(N+M)

④ Base cases.

(i) if j<0 that means s2 is matched in s1 then return 1

if i<0 that means s1 is exeguted but s2 is not matched completely.



Ovrlaping Subproblem. So draw recursion Tree

Memoization

$$\begin{tikzcd} i \arrow[d] & j \arrow[d] \\ N & M \end{tikzcd}$$

dp[N][M]

Tabulation

① Write down the Base case

② write down the changing parameters in opposite fashoin

i =
j =

③ Copy paste recurrance

① Write down the Base case

In memoization base case for $i == -1 \,||\, j == -1$
i.e. call for $f(-1, j)$ or $f(i, -1)$ possible
but in tabulation we can't write base case for -1 indep so we shift index by 1

1 based indexing

```
i    j
0    0
↓    ↓
-1   -1
```

```
f(i,j)
{ if(j==0) return 1;
  if(i==0) return 0;

  if(dP[i][j] != -1) ret dP[i][j])

  if(s1[i-1] == s2[j-1]
      ret dP[i][j] = f(i-1,j-1) +f(i-1,j]))

  else ret dP[i][j] = f(i-1,j));

}
```

Base case in tabulation

```
dP[n+1] [m+1]
for( i=0 to n)  dP[i][0]=0
for( j=0 to m)  dP[0][j]=1
```

```
S1     S2
↑0     ↑0
i      j
```

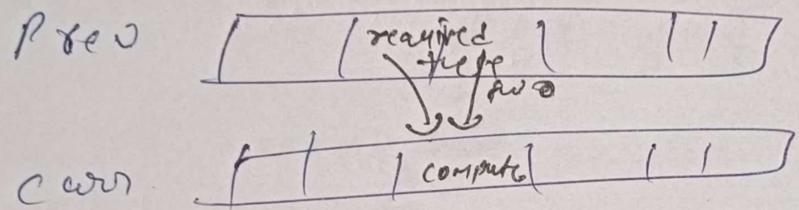
(2) Write down the changing parameters

```
i = 1 to n
j = 1 to m
```

(3) Copy paste the reccurance

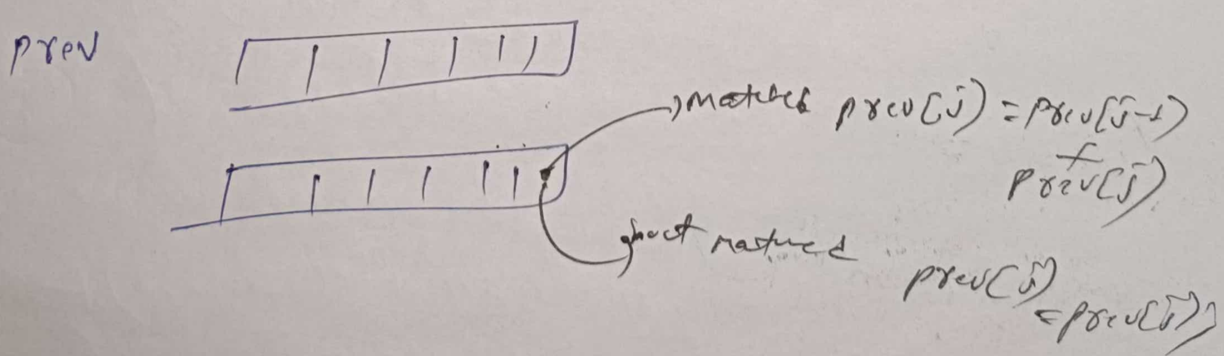
1-D Space Optimization

```
i = 1 to n
j = ~~m to 1~~ 1 to m

if (matched)   curr[j] = prev[j-1] + prev[j]
else   curr[j] = prev[j]
```



Now change → Knapsack 1-D array space Optimization

```
i = 1 to n
j = m to 1

if (matched)   prev[j] = prev[j-1] + prev[j]
else   prev[j] = prev(j)
```


prev
matched prev[j] = prev[j-1] + prev[j]
not matched prev[j] = prev[j]