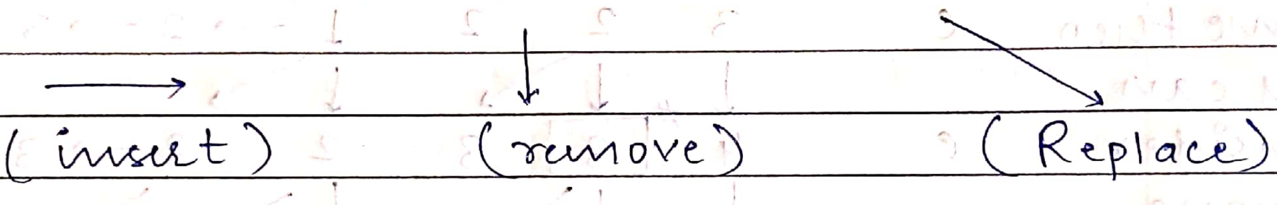
Problem 2. Minimum Edit distance.

Given two strings str1 and str2 and below operations that can be performed on str1. Find minimum number of edits (operations) required to convert 'str1' to 'str2'.

1. Insert
2. Remove
3. Replace

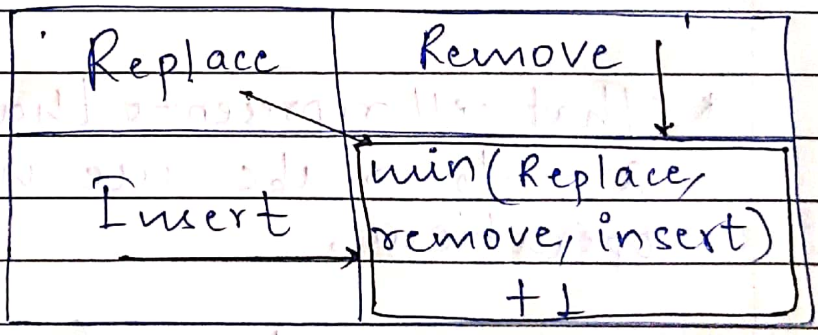
All of the above operations are of equal cost.

Solution. Notations to be remembered:



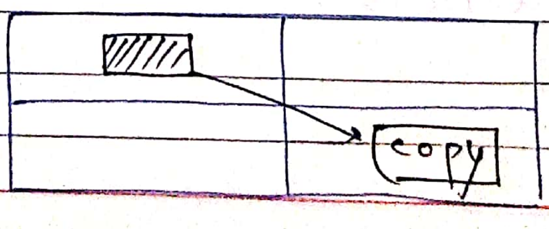
Steps to create memoization matrix.

1. If $r \neq c$



2. If $r == c$ $r \rightarrow$ row & $c \rightarrow$ column.

Just copy the diagonal element.



Scanned by CamScanner

Example

str1 = "adceg" ⟶ to str2 = "abcfg".

	NULL	a	b	c	f	g
NULL	0	1	2	3	4	5
a	1	0	1	* $\boxed{2}$	3	4
d	2	1	1	2	3	4
c	3	2	2	1	2	3
e	4	# $\boxed{3}$	3	2	2	3
g	5	4	4	3	3	$\boxed{2}$

if two of them are same then you can consider anyone.

↓
2 operations

* That cell represents that to convert a
a —to⟶ abc we need 2 insert operations.

a ⟶ a$\underbrace{bc}$ (2 insert).

# That cell represents that to convert adce —to⟶ a we need 3 delete operations.

Code 1 :

```
public static void minedit(String str1, String str2)
{
      int n1 = str1.length();
      int n2 = str2.length();
      int t[][] = new int[n1+1][n2+1];

     for(int i=0; i<=n1; i++){
        for(int j=0; j<=n2; j++){

              if(i==0 && j==0)
                    t[i][j] = 0;

              else if(i==0)
                    t[i][j] = t[i][j-1]+1;
              else if(j==0)
                    t[i][j] = t[i-1][j]+1;
              else
                {
                   if(str1.charAt(i-1)== str2.
                                  charAt(j-1))
                   {
                      t[i][j] = t[i-1][j-1];
                   }
                  else{  int x=0;
                      if(t[i-1][j-1] > t[i-1][j])
                           x = t[i-1][j];
                      else
                           x = t[i-1][j-1];
                      if(x > t[i][j-1])
                                        x = t[i][j-1];
sop(t[n1][n2]);} } } } t[i][j] = x+1;
```