ASSIGNMENT (DP)

1. **MCP Compute**

```
 package dsa.dp;
public class EditDistance {
public int editDistance(String s, String t) {

       int m = s.length();
       int n = t.length();
       int dp [][] = new int[m+1][n+1];
       for(int i=0;i<=m;i++){
           dp[i][0] = i;
       }
for(int j=0;j<=n;j++){
           dp[0][j] = j;
       }
       for(int i=1;i<=m;i++){
           for(int j=1;j<=n;j++){
               if(s.charAt(i-1) == t.charAt(j-1)){
               dp[i][j] = dp[i-1][j-1];
               }
      else{
```

```
 dp[i][j]=1+Math.min(dp[i-

1][j],Math.min(dp[i][j-1],dp[i-1][j-1]));

            }

         }

      }

      return dp[m][n];

}

}
```

2. **Edit Distance**

```
class Solution
{
   //Function to return the minimum cost to react at bottom
   //right cell from top left cell.
   public int minimumCostPath(int[][] grid)
   {
      Queue<int[]> queue = new PriorityQueue<>((a,b)->a[0]-b[0]);
      int [][] costs = new int[grid.length][grid[0].length];
      for(int i = 0; i<costs.length; i++){
         Arrays.fill(costs[i], Integer.MAX_VALUE);
      }
      queue.add(new int[]{grid[0][0], 0, 0});

      int[][] ds = new int[][]{{0,1},{0, -1},{1,0},{-1,0}};

      while(!queue.isEmpty()){
         int[] temp = queue.poll();
         int cost = temp[0];
         int x = temp[1];
         int y = temp[2];


         if(x==grid.length-1 && y == grid[0].length-1){
               return cost;
           }


           for(int k = 0; k<ds.length; k++){
               int newX = x+ ds[k][0];
               int newY = y+ ds[k][1];

               if(isSafe(newX, newY, grid.length, grid[0].length) &&
costs[newX][newY]>cost + grid[newX][newY] ){
                  int newCost = cost + grid[newX][newY];
                  queue.add(new int[]{newCost, newX, newY});
                  costs[newX][newY] = newCost;
               }
           }


       }
          return -1;
    }

    public boolean isSafe(int row, int col, int N, int M){
        if(row < 0 || col < 0 || row >= N || col >= M){
            return false;
        }
        return true;
    }
}
```