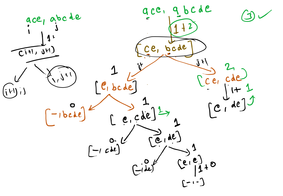
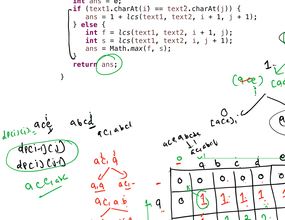
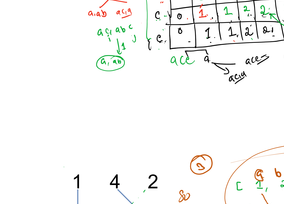
Lec-44
21/04/24 4:53 PM

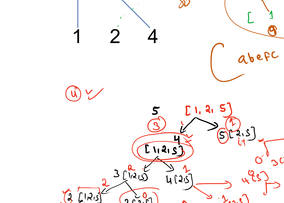


```
public static int lcs(String text1, String text2, int i, int j) {
    if (i == text1.length() || j == text2.length()) {
        return 0;
    }
    int ans = 0;
    if (text1.charAt(i) == text2.charAt(j)) {
        ans = 1 + lcs(text1, text2, i + 1, j + 1);
    } else {
        int f = lcs(text1, text2, i + 1, j);
        int s = lcs(text1, text2, i, j + 1);
        ans = Math.max(f, s);
    }
    return ans;
}
```





```
public static int Coin_ChangeBU(int[] coins, int amount) {
    int[][] dp = new int[amount + 1][coins.length + 1];
    for (int i = 0; i < dp[0].length; i++) {
        dp[0][i]=1;
    }
    for (int a = 1; a< dp.length; a++) {
        for (int i = 1; i < dp[0].length; i++) {
            int inc = 0, exc = 0;
            if (a >= coins[i-1]) {
                inc = dp[a-coins[i-1]][i];
            }
            exc =dp[a][i-1];
            dp[a][i]=inc + exc;
        }
    }
    return dp[dp.length-1][dp[0].length-1];
}
```



You are given two strings, A and B. Answer, what is the smallest number of operations you need to transform A to B?

Operations are:

- Delete one letter from one of strings
- Insert one letter into one of strings
- Replace one of letters from one of strings with another letter

Input

T- number of test cases

For each test case:

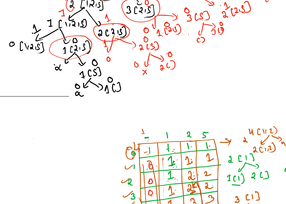
- String A
- String B

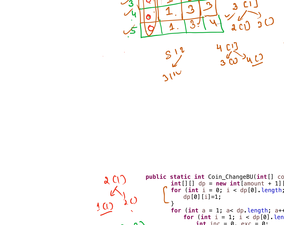
Both strings will contain only uppercase characters and they won't be longer than 2000 characters.

There will be 10 test cases in data set.

Output

For each test case, one line, minimum number of operations.



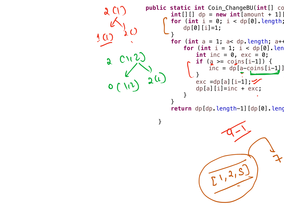


You are packing for a vacation on the sea side and you are going to carry only one bag with capacity S (1 <= S <= 1000).
You also have N (1<= N <= 1000) items that you might want to take with you to the sea side.
Unfortunately you can not fit all of them in the knapsack so you will have to choose.
For each item you are given its size and its value.
You want to maximize the total value of all the items you are going to bring.
What is this maximum total value?

Example 1

Input

```
5 4
1 2 3 2 2
8 4 0 5 3
```

Output

```
13
```