const double PI = acos(-1.0);

template<class T> T gcd(T a, T b) { return b ? gcd(b, a % b) : a; }

template<class T> T lcm(T a, T b) { return a / gcd(a, b) \* b; }

/// 01 背包

for(int i = 1; i <= n; ++i)

for(int j = W; j >= w[i]; --j)

f[j] = max( f[j],// no fang

f[ j - w[i] ] // fang

+ value[i]); // + value

/// 完全 背包 ( 无限 )

for(int i = 1; i <= n; ++i)

for(int j = w[i]; j <= W; ++j )

dp[j] = max( dp[j],

dp[j - w[i]]

+ value[i]);

/// 多重 背包 (第 i 件 最多 可用 n[i] 件 )

for(int i = 0; i < nkind; ++i)

{

scanf("%d %d %d", &value[i], &weight[i], &bag[i]);

for(int j = 0; j < bag[i]; ++j) // 枚举 个数

{

for(int k = nvalue; k >= value[i]; --k) // 01

{

dp[k] = max( dp[k],

dp[ k - value[i]] + weight[i]);

}

}

} printf("%d\n", dp[nvalue]);

// DP 数塔

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

Ans = 30

for(int i = n; i >= 1; i--)

for(int j = 1; j <= i; j++)

f[i][j] = max(f[i + 1][j], f[i + 1][j + 1]) + a[i][j];

cout << f[1][1] << endl;

// kitten和sitting的编辑距离是3

string s1, s2;

int dp[1005][1005];

int dfs(int pos1, int pos2)

{

if(pos1 == -1 && pos2 == -1) return 0;

if(pos1 == -1) return pos2 + 1;

if(pos2 == -1) return pos1 + 1;

if(dp[pos1][pos2] != -1) return dp[pos1][pos2];// 记忆化搜索

if(s1[pos1] == s2[pos2]) return dp[pos1][pos2] = dfs(pos1 - 1, pos2 - 1);

return dp[pos1][pos2] = min(dfs(pos1, pos2 - 1), min(dfs(pos1 -1, pos2), dfs(pos1 - 1, pos2 - 1))) + 1;

}

int main()

{

while(cin >> s1 >>s2)

{

memset(dp, -1, sizeof dp);

cout << dfs(s1.size() - 1, s2.size() - 1) << endl;

}

}