Algorithm HW#3

divide_and_conquer

Task. Karatsuba algorithm for fast multiplication

Using Divide and Conquer to multiply two integers X and Y in less time complexity.

Sample Input:

```
X    5432
Y    1678
```

Output:

```
9114896
```

Greedy Algorithm

Task. Fractional Knapsack Problem

Given weights and values of n items, we need to put these items in a knapsack of capacity W to get the maximum total value in the knapsack

Sample Input:

```
Items as (value, weight) pairs    --->  {{60, 10}, {100, 20}, {120, 30}}
Knapsack Capacity  --->   W = 50;
```

Output:

```
Maximum possible value = 240
```

by taking items of weight 10 and 20 kg and 2/3 fraction of 30 kg. Hence total price will be 60+100+(2/3)(120) = 240

Algorithm HW#3

Dynamic Programming

1. Task. Money Change Problem

Given an array of integers that represent coins called coins and an integer amount amount, return the minimum amount of coins it requires to make complete change for amount amount.

If it is not possible to make change return -1.

Constraints:
coins[i] >= 1
Each coin will be unique
You may reuse coins
The sum of the coins used must equal amount exactly

Sample 1. Input:

```
Coins :    [1, 2, 3]
amount:    10
```

Output:

```
4
```

Explanation: We can use two 3 coins & two 2 coins to fully make change for 10. 3 + 3 + 2 + 2 = 10

Sample 2. Input:

```
Coins :    [1, 3, 5, 6, 9]
amount:    90
```

Output:

```
10
```

Explanation: 9 x 10 uses = 90

Sample 3. Input:

```
Coins :    [2]
amount:    5
```

Output:

```
-1
```

Explanation: We cannot make change for 5 with only a 2 coin.

2 Edit Distance

Problem Introduction

The edit distance between two strings is the minimum number of operations (insertions, deletions, and substitutions of symbols) to transform one string into another. It is a measure of similarity of two strings. Edit distance has applications, for example, in computational biology, natural language processing, and spell checking. Your goal in this problem is to compute the edit distance between two strings.

Problem Description

Task. The goal of this problem is to implement the algorithm for computing the edit distance between two strings.

Input Format. Each of the two lines of the input contains a string consisting of lower case latin letters.

Constraints. The length of both strings is at least 1 and at most 100.

Output Format. Output the edit distance between the given two strings.

Sample 1.

Input:

```
ab
ab
```

Output:

```
0
```

Sample 2.

Input:

```
short
ports
```

Output:

```
3
```

An alignment of total cost 3:

s	h	o	r	t	–
–	p	o	r	t	s

Sample 3.

Input:

```
editing
distance
```

Output:

```
5
```

An alignment of total cost 5:

e	d	i	–	t	i	n	g	–
–	d	i	s	t	a	n	c	e

3 Longest Common Subsequence of Two Sequences

Problem Introduction

Compute the length of a longest common subsequence of two sequences.

Problem Description

Task. Given two sequences $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_m)$, find the length of their longest common subsequence

Input Format. First line: $a_1, a_2, \ldots, a_n$. Second line: $b_1, b_2, \ldots, b_m$.

Output Format. Output p.

Sample 1.

Input:

```
X      <A B C B D A B>

Y      <B D CAB A>
```

Output:

```
<B C B A>    |    <BDAB>
```

valid to be sub-sequence <B C A> but not longest one.

4 0-1 Knapsack Problem

Sample 1.

Input:

```
val = [60, 100, 120]
wt = [10, 20, 30]
W = 50
```

Output:

```
220
```


Good Luck