Birla Institute of Technology and Science, Pilani Second Semester 2017-2018, DSA (CS F211)

Lab Assignment #11

1. Imagine you have a collection of **N** wines placed next to each other on a shelf. For simplicity, let's number the wines from left to right as they are standing on the shelf with integers from **1 to N**, respectively. The price of the ith wine is pi. (prices of different wines can be different).
   Because the wines get better every year, supposing today is the year 1, on year y the price of the ith wine will be y*pi, i.e. y-times the value that current year.
   You want to sell all the wines you have, but you want to sell exactly one wine per year, starting on this year. One more constraint - on each year you are allowed to sell only either the leftmost or the rightmost wine on the shelf and you are not allowed to reorder the wines on the shelf (i.e. they must stay in the same order as they are in the beginning).
   You want to find out, what is the maximum profit you can get, if you sell the wines in optimal order?
   Input:
   N
   N space separated integers such that ith integer denotes the price of ith wine pi
   Output:
   Maximum Profit
   Sample Input:
   4
   1 4 3 2
   Sample Output:
   If the prices of the wines are (in the order as they are placed on the shelf, from left to right): p1=1, p2=4, p3=2, p4=3. The optimal solution would be to sell the wines in the order **p1, p4, p3, p2** for a total profit 1 * 1 + 3 * 2 + 2 * 3 + 4 * 4 = 29.

2. Given a tree *T* of *N* nodes, where each node *i* has C_i coins attached with it. You have to choose a subset of nodes such that no two adjacent nodes (i.e. nodes connected directly by an edge) are chosen and sum of coins attached with nodes in chosen subset is maximum.
   Input:
   N
   N space separated integers denoting coins attached with the ith node
   N-1 lines follow where each line is of the form u,v and denotes an edge between u and v
   Output:
   Maximum Sum of Coins
   Sample Input:
   5
   3 1 2 3 4
   1 2
   1 3
   3 4
   3 5

Sample Output:

```
10
```

Explanation: We select the subset of nodes as 1, 4, 5

Time Complexity: O(N)

3. You are given N stones, labelled from 1 to N. The i-th stone has the weight W[i]. There are M colours, labelled by integers from 1 to M. The i-th stone has the colour C[i] (of course, an integer between 1 to M, inclusive). You want to fill a Knapsack with these stones. The Knapsack can hold a total weight of X. You want to select exactly M stones; one of each colour. The sum of the weights of the stones must not exceed X. Since you paid a premium for a Knapsack with capacity X (as opposed to a Knapsack with a lower capacity), you want to fill the Knapsack as much as possible.

Write a program that takes all the above values as input and calculates the best way to fill the Knapsack – that is, the way that minimizes the unused capacity. Output this unused capacity.

Input:

The first line of input contains the integer T, the number of test cases. Then follows the description of T test cases. The first line of each test case contains three integers, N, M and X, separated by single space. The next line contains N integers, W[1], W[2], W[3] … W[N], separated by single space. The next line contains N integers C[1], C[2], C[3] … C[N], separated by single space.

Output:

An optimal way of filling the Knapsack minimizes unused capacity. There may be several optimal ways of filling the Knapsack. Output the unused capacity of the Knapsack (a single integer on a line by itself) for an optimal way. If there is no way to fill the Knapsack, output -1. Output T lines, one for each test case.

Constraints:

1 ≤ T ≤ 10

1 ≤ M ≤ 100

M ≤ N ≤ 100

1 ≤ W[i] ≤ 100

1 ≤ C[i] ≤ M

1 ≤ X ≤ 10000

Sample Input:

```
4
9 3 10
2 3 4 2 3 4 2 3 4
1 1 1 2 2 2 3 3 3
9 3 10
1 3 5 1 3 5 1 3 5
1 1 1 2 2 2 3 3 3
3 3 10
3 4 4
1 2 3
3 3 10
3 3 3
1 2 1
```

Sample Output:

```
0
1
-1
-1
```

Explanation:

In the first test case you can select stone 2, stone 5 and stone 9. The knapsack will be completely full. Of course, there are several other ways to select stones such that the knapsack is full. The unused capacity in all such ways is 0.

In the second test case you cannot select stones such that the knapsack is completely full. You can select stones {1, 4, 9}, such that the unused capacity is 10-1-1-5 = 3. But there is a better way. Select stones {2, 5, 8}. The unused capacity is 10-3-3-3 = 1. This is the optimal way. There is another way that is optimal. Select stones {1, 5, 9}. The unused capacity is 10-1-3-5 = 1.

In the third test case there is only one option. Select stones {1, 2, 3}. The total weight will be 11. This is more than what the knapsack can hold. In the fourth test case there is no stone of colour 3. Thus, there is no valid selection of stones possible.

4. Donald Trump has n snacks arranged in front of him and all these snacks are arranged in a row. Each snack has a colour which is one among 100 different colours. He wants to mix all these snacks together. At each step, he is going to take two snacks that are next to each other and mix them together and put the resulting snack in their place. When mixing two snacks of colors a and b, the resulting snack will have the color (a+b) mod 100. Also, there will be some wastage in the process. The amount of waste generated when mixing two snacks of colours a and b is a*b. Find out what is the minimum amount of waste that Trump can get when mixing all the snacks together.

Input:

Number of test cases

The first line of each test case will contain n, the number of snacks, 1 <= n <= 100.

The second line will contain n integers between 0 and 99 - the initial colors of the snacks.

Output:

Minimum wastage

Sample Input:

```
2
18 19
3
40 60 20
```

Sample Output:

```
342
2400
```

5. Yash is very unhappy that people are copying in DSA lab test. He plans to develop a plagiarism checker to stop this. It takes code snippets identified by characters A to Z. The goal is to detect how close two programmes are. Each program is a sequence of code snippets and hence would be represented by a string. Given representation of two codes, find the longest common subsequence of the two codes. A subsequence is any sequence of characters that occurs in the string in the same relative order but not necessarily in a contiguous fashion.

Input:

Two strings
Output:
Length of the longest common subsequence and the subsequence
Sample Input:
AGGTAB
GXTXAYB
Sample Output:
4
GTAB
Time Complexity: O(NM) where N and M are lengths of the strings.

6. Refer to problem 1. Now, Yash wants to speed up the process and so he decides to take three strings at a time. Find the longest common subsequence.
Input:
3 strings
Output:
Length of the longest common subsequence and the subsequence
Sample Input:
dabjcld
axbbmczcd
aqbuicdt
Sample Output:
4
abcd
Time Complexity:
O(MNO) – where M N and O are the lengths of strings.

7. Tanmay has gone to a fish market. The market is such that there is a definite price for each type of fish but the price be mentioned. Tanmay has to bargain. Each bargain involves Tanmay quoting the price and the shopkeeper accepting or not. If Tanmay quotes more than the actual price, the shopkeeper accepts and Tanmay cannot bargain further with that shopkeeper. Luckily, there may be multiple shopkeepers and the fish can be bought from any one shop. Find the minimum number of bargains Tanmay should make to buy the fish at its optimal price.
Input:
N and K where N is the price range, i.e., price is between 1 and N both included. And K is the number of shopkeepers.
Output:
Minimum number of bargains.
Sample Input:
36 2
Sample Output:
8
Time Complexity: O(kn^2)

8. Given an array of integers, the task is to find the longest subsequence with a geometric progression in it. The common ratio should be an integer.
Input:
N – size of array
The array
Output:

K – the size of longest subsequence with geometric progression

Sample Input:

6

1 3 7 9 21 27

Sample Output:

4

Time Complexity: O(n^2)

Explanation: 1 3 9 27 is the subsequence.

9. You are given two strings. Using 3 operations, insert, delete and edit, find the number of operations required to convert the first string to the second. Insert – inserting a character in the string. Delete – remove any character from the string. Edit – change any character from the string.

   Input:

   Two strings

   Output:

   K – minimum number of operations

   Sample Input:

   Fday

   Friday

   Sample Output:

   2

   Time Complexity: O(mn) where m and n are lengths of strings

   Explanation: 2 inserts have to be made. ‘r’ and ‘i’.

10. Shikhar is distributing chocolates to his friends on his birthday. He has a lot of chocolates. He has N friends and he has to distribute in a certain fashion. He distributes in rounds. In each round, he can give 1,3 or 5 chocolates to all friends but one who doesn’t get any chocolates in that round. He has to make sure that all of them get equal numbers. And also each one of them initially has some number of chocolates. Find the minimum number of rounds before each person can have same number of chocolates.

    Input:

    N – number of friends

    An array depicting initial number of chocolates each one has

    Output:

    K – minimum number of rounds

    Sample Input:

    4

    2 2 3 7

    Sample Output:

    2

    Explanation: 1 chocolate in first round to 0,1,3 and 5 in second round to 0,1,2. All have 8 in the end.