NOI 2012 TASKS OVERVIEW

Tasks
Task 1: MODSUM
Task 2: PANCAKE
Task 3: FORENSIC
Task 4: WALKING

Notes:

1. Each task is worth 25 marks.
2. Each task will be tested on a few sets of input instances. Each set is worth a few marks, adding up to a total of 25 marks per task. For each set, if your program produces the correct output for all the input instances, then you obtain all the marks; otherwise, you do not obtain any mark for that set.
3. The input instances vary in size and complexity, leading to different levels of difficulty. All input instances in a same set give roughly the same level of difficulty.
4. The maximum execution time on each input instance is 5 seconds. Each instance will be tested in a runtime environment allocated with 32 MB.

HAPPY PROGRAMMING!

Task 1: MODSUM

In this task, you are given the following function f with n parameters:

$$f(x_1, \ldots, x_n) = (((x_1 + x_2 + \ldots + x_n)^4 + 2 \times (x_1 + x_2 + \ldots + x_n)^2) \mod 5) + 1$$

As arguments, f accepts only integer values. Your task is to compute the sum of all values of f, where each input x_i ranges from an integer value v_i to w_i. In other words, you need to compute

$$\sum_{x_1=v_1}^{w_1} \sum_{x_2=v_2}^{w_2} \cdots \sum_{x_n=v_n}^{w_n} f(x_1, \ldots, x_n)$$

For example, if $n = 3$, $v_1 = 2$, $w_1 = 3$, $v_2 = 10$, $w_2 = 12$, $v_3 = 17$ and $w_3 = 17$, then the result should be 19, since $f(2, 10, 17) = 4$, $f(2, 11, 17) = 1$, $f(2, 12, 17) = 4$, $f(3, 10, 17) = 1$, $f(3, 11, 17) = 4$ and $f(3, 12, 17) = 5$.

Important note: You can assume that the result will always be less than 1,000,000.

Input format

Your program must read from the standard input. The input consists of n, where $1 \leq n \leq 1000$, followed by n pairs of numbers, v_i and w_i, each of which ranges from 0 to 100. For each pair v_i and w_i, you can assume that $v_i \leq w_i$. In our example above, the input is:

```
3 2 3 10 12 17 17
```

Output format

Your program must write to the standard output the required sum. In our example above, the output will be:

```
19
```

Input instances

Your program will be tested on 5 sets of input instances as follow:

1. (5 marks) All instances in this set satisfy $n \leq 6$.
2. (5 marks) All instances in this set satisfy $n \leq 20$.
3. (5 marks) All instances in this set satisfy $n \leq 100$.
4. (5 marks) All instances in this set satisfy $n \leq 200$.
5. (5 marks) All instances in this set satisfy $n \leq 1000$.

Task 2: `PANCAKE`

You are given a stack of N pancakes, where $1 \leq N \leq 8$. The pancake at the bottom and at the top of the stack has index 0 and index $N-1$ respectively. The size of a pancake is its diameter which is a positive integer. All pancakes in the stack have different sizes. A stack `A` of $N = 5$ pancakes with size 3, 8, 7, 6, and 10 can be visualized as:

```
4 (top)          10
3                 6
2                 7
1                 8
0 (bottom)        3
------------------------
index i        A[i]
```

We want to sort the stack in *descending order*, that is, the largest pancake is at the bottom and the smallest pancake is at the top. To make the problem more real-life like, sorting a stack of pancakes can only be done by a sequence of pancake 'flips', denoted by *flip*(i). The *flip*(i) operation inserts a spatula into the stack, lifts up pancakes from index i to index $N-1$ and flips them. As a result, the position of the pancakes from index i to $N-1$ are reversed.

For example, stack `A` can be transformed to stack `B` via *flip*(0), i.e. inserting a spatula and flipping the pancakes from index 0 and 4. Stack `B` can be transformed to stack `C` via *flip*(3). Stack `C` can be transformed to stack `D` via *flip*(1), and so on. Our target is to make the stack sorted in descending order, i.e. we want the stack to be like stack `E`, using minimum number of flips.

```
4 (top)       10  <--   3  <--   8  <--   6               3
3              6        8  <--   3        7     ...       6
2              7        7        7        3               7
1              8        6        6  <--   8               8
0 (bottom)     3  <--  10       10       10              10
---------------------------------------------------------------
index i     A[i]     B[i]     C[i]     D[i]     ...     E[i]
```

Bill Gates (Microsoft founder, former CEO, and current chairman) wrote only one research paper so far, and it is about this pancake sorting[1].

In this question, given the starting configuration of N pancakes, your task is to compute the *minimum* number of flips required to sort them.

Input format

The first line of input contains an integer T which is the number of test cases. Each of the next T lines corresponds to a test case. Each test case starts with the number of pancakes N, and then followed by a sequence of N integers indicating the size of each pancake. The bottom pancake (i.e the pancake with index 0) appears first in the sequence. Examples will be given in the next page.

Output format

For each test case, output in one line the minimum number of flips required to sort the test case. Hence, the output contains T lines with one integer per line. Examples will be given in the next page.

Input instances

Your program will be tested on 5 sets of input instances as follow:

1. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 1 flip. The pancakes sizes range from 1 to N and we have $1 \leq T \leq 200$.
2. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 2 flips. The pancakes sizes range from 1 to N and we have $1 \leq T \leq 200$.
3. (4 marks) It is guaranteed that all test cases can be sorted with *at most* 2 flips. The pancakes sizes range from 1 to $1,000,000$ and we have $1 \leq T \leq 200$.
4. (8 marks) Here, the minimum number of flips is not constrained, and thus could be large. The pancakes sizes range from 1 to $1,000,000$, and we have $1 \leq T \leq 200$.
5. (5 marks) Same as the fourth criteria above, but with $1 \leq T \leq 6000$. Since the number of test cases can be large, your program must be (very) efficient.

[1]Gates, W. and Papadimitriou, C. *Bounds for Sorting by Prefix Reversal,* Discrete Mathematics, 27, 47-57, 1979.

More Samples

Here are examples for each of the 5 input sets.

Example for set 1

Input

```
2
4 4 3 2 1
8 8 7 6 5 4 1 2 3
```

Output

```
0
1
```

Remark. The first test case is already sorted in descending order, so no flip is needed. The second test case can be sorted in descending order in 1 flip: *flip*(5).

Example for set 2

Input

```
3
4 4 3 2 1
8 8 7 6 5 4 1 2 3
5 5 1 2 4 3
```

Output

```
0
1
2
```

Remark. The first and the second test cases are exactly the same as the example for set 1. The third test case can be sorted in descending order in 2 flips: *flip*(3) then *flip*(1).

Example for set 3

Input

```
2
5 555555 111111 222222 444444 333333
8 1000000 999999 999998 999997 999996 999995 999994 999993
```

Output

```
2
0
```

Remark. The first input stack can be sorted in descending order by 2 flips: *flip*(3) then *flip*(1). The second input stack is already sorted in descending order, so no flip is needed.

Example for set 4 & 5

Input

```
3
5 555555 111111 222222 444444 333333
8 1000000 999999 999998 999997 999996 999995 999994 999993
5 3 8 7 6 10
```

Output

```
2
0
4
```

Remark. The first and the second test cases are exactly the same as the example for set 3. The third test case (which is the example given in the task description) can be sorted in 4 flips by the following two possible sequences:

- Solution 1: *flip*(0), *flip*(1), *flip*(2), *flip*(1): 4 flips.
- Solution 2: *flip*(1), *flip*(2), *flip*(1), *flip*(0): also 4 flips.

There is no way to sort using less than 4 flips, and thus 4 is the minimum possible.

Task 3: `FORENSIC`

0	1	2	3	4	5	6	7	8	9
2	5	4	4	-1	1	-1	3	0	8

The table above shows the content of an array `A`, with the indices in the top row. The array stores the *pointers* of a linear linked list. In this task, a pointer is simply an integer value. The pointer in the first node is stored in $A[0]$, that is, the value $A[0]$ indicates the location of the second node. The pointer in the second node is stored in $A[A[0]]$, whereas the pointer in the third node is stored in $A[A[A[0]]]$, and so on. If the pointer has value -1, it indicates end of the linked list. In the above example, value of $A[0]$ is 2, and thus the second pointer is stored in $A[2]$. The value of $A[2]$ is 4 and thus the third pointer is stored in $A[4]$. The value of $A[4]$ is -1, indicating that there is no more subsequent node. The sequence of pointers is shown below and the linked list has 3 nodes.

$$A[0] = 2 \quad \rightarrow \quad A[2] = 4 \quad \rightarrow \quad A[4] = -1$$

You have received the entries of an array as described above, and you are also told that one of the entries is being replaced. However, the location of that entry, and its new value are not known. Note that it is possible that the new value is the same as the original value, that is, the array remains unchanged. As a forensic expert, you want to recover the original array. To achieve that, you want to modify a single entry, so that the modified array represents a linked list with the largest number of nodes. Consider the above example:

- Changing the entry $A[4]$ to 6 leads to a linked list of 4 nodes.
- Changing the entry $A[0]$ to 7 leads to a linked list of 4 nodes.
- Changing the entry $A[0]$ to 9 leads to an invalid linked list.
- Changing the entry $A[2]$ to 7 leads to a linked list of 5 nodes.

Among all possible changes to one entry, including not changing any entry, the recovered list with 5 nodes is the largest possible. Thus, it is likely that the original entry of $A[2]$ is 7. Here, your task is to compute the size of the largest possible recovered linked list.

Input format

Your program must read from the standard input. The input consists of two lines. The only integer in the first line is N, the size of the array. The second line gives the N integers in the array, starting from $A[0]$. Each integer in the array is larger or equal to -1, and smaller than N. For the above example, the input is:

```
10
2 5 4 4 -1 1 -1 3 0 8
```

Output format

Your program must write to the standard output the number of nodes in the largest recovered linked list. For the above example, the output will be:

```
5
```

Input instances

Your program will be tested on 4 sets of input instances as follow:

1. (5 marks) All instances in this set satisfy $1 < N \leq 20$.
2. (5 marks) All instances in this set satisfy $20 < N \leq 3000$. Moreover, the number of nodes in the recovered linked list is not more than 100.
3. (5 marks) All instances in this set satisfy $20 < N \leq 3000$. Unlike the instances in the above, the size of the recovered linked list can be more than 100.
4. (10 marks) All instances in this set satisfy $3000 < N \leq 20000$.

More Samples

Be careful, there are a number of special cases. Don't miss any of them. Here are a few more samples.

Input

```
4
0 0 0 0
```

Output

```
1
```

Remark. Changing $A[0]$ to -1 give a linked list of 1 node.

Input

```
6
1 2 3 4 5 -1
```

Output

```
6
```

Remark. The input (without any change) has the largest linked list.

Input

```
10
2 5 4 4 0 1 -1 3 0 8
```

Output

```
4
```

Remark. Changing $A[4]$ to 6 gives a linked list of 4 nodes.

Input

```
10
2 5 4 4 -1 1 -1 6 0 8
```

Output

```
5
```

Remark. Changing $A[4]$ to 7 gives a linked list of 5 nodes.

Task 4: `Walking`

Consider a road of length ℓ. There are n persons. The ith person, $i = 1, \ldots, n$, will start walking from the beginning of the road at time t_i and will move at a constant speed v_i until arrival at the end of the road. We assume no two persons start walking at the same time, and no two persons arrive at the same time.

If the ith and jth person meet each other on the road, they will become friends. Mathematically, for the ith and jth persons where $t_i < t_j$, they will become friends if and only if $\ell/v_i + t_i > \ell/v_j + t_j$.

Your task is to find the size of the maximum set of persons who are friends of each other.

Example 1

Suppose the length of the road is $\ell = 1000$, there are 4 persons who start walking at time $t_1 = 0$, $t_2 = 1$, $t_3 = 2$ and $t_4 = 3$ respectively, and their walking speeds are $v_1 = 2$, $v_2 = 3$, $v_3 = 1$ and $v_4 = 4$ respectively.

Then, the 1st person will meet with the 2nd person since $1000/2+0 > 1000/3+1$. Similarly, we have the following table.

	1st	2nd	3rd	4th
1st		friend	not friend	friend
2nd			not friend	friend
3rd				friend

Hence, the 1st, 2nd and 4th persons are friends of each other. In fact, this is the maximum set. We report 3 to be the size of the maximum set of persons who are friends of each other.

Example 2

Suppose the length of the road is $\ell = 1000$, there are 4 persons who start walking at time $t_1 = 0$, $t_2 = 1$, $t_3 = 2$ and $t_4 = 3$ respectively and their walking speeds are $v_1 = 1$, $v_2 = 1$, $v_3 = 1$ and $v_4 = 1$ respectively.

Then, none of them will meet each other. Hence, we report 1 to be the size of the maximum set of persons who are friends of each other.

Input format

Your program must read from the standard input. The input consists of $n+1$ lines. The first line contains two integers ℓ and n separated by space, where $100 \le \ell \le 10000$ and $1 \le n \le 500$. Each of the next n lines contains two integers. For the $(i+1)$-th line, it contains the two integers t_i and v_i separated by space, where $0 \le t_i \le 1000$ and $1 \le v_i \le 100$. For example 1, the input file looks like:

```
1000 4
1 3
2 1
0 2
3 4
```

For example 2, the input file looks like:

```
1000 4
0 1
2 1
1 1
3 1
```

Output format

Your program must write to the standard output an integer which is the size of the maximum set of persons who are friends of each other. For example 1, the output will be:

```
3
```

For example 2, the output will be:

```
1
```

Input instances

Your program will be tested on 5 sets of input instances as follow:

1. (3 marks) The number of persons is at most 40. The output, that is the size of the maximum set of persons who are friends of each other, will be at most 6.
2. (3 marks) The number of persons is at most 150. The output will be at most 12.
3. (5 marks) The number of persons is at most 250. The output will be at most 16.
4. (7 marks) The number of persons is at most 350. The output will be at most 22.
5. (7 marks) The number of persons is at most 500. The output can be any value ranges from 1 to 500.