Problem 1: Record Keeping

Farmer John has been keeping detailed records of his cows as they enter the barn for milking. Each hour, a group of 3 cows enters the barn, and Farmer John writes down their names. For example over a 5-hour period, he might write down the following list, where each row corresponds to a group entering the barn:

```
BESSIE ELSIE MATILDA
FRAN BESSIE INGRID
BESSIE ELSIE MATILDA
MATILDA INGRID FRAN
ELSIE BESSIE MATILDA
```

Farmer John notes that the same group of cows may appear several times on his list; in the example above, the group of BESSIE, ELSIE, and MATILDA appears three times (even though Farmer John didn't necessarily write their names in the same order every time they entered the barn).

Please help Farmer John count the number of occurrences of the group entering the barn the most.

INPUT FORMAT:

* Line 1: The number of hours, N, for which Farmer John keeps records (1 <= N <= 1000).

* Lines 2..1+N: Each line contains a list of three space-separated cow names. Each name is between 1 and 10 characters and uses only the letters A-Z.

SAMPLE INPUT:

```
5
BESSIE ELSIE MATILDA
FRAN BESSIE INGRID
BESSIE ELSIE MATILDA
MATILDA INGRID FRAN
ELSIE BESSIE MATILDA
```

OUTPUT FORMAT:

* Line 1: The number of occurrences of the group entering the barn the
 most often.

SAMPLE OUTPUT:

3

OUTPUT DETAILS:

The group {BESSIE, ELSIE, MATILDA} enters the barn on three separate occasions.

Problem 2: Cow Baseball

Farmer John's N cows (3 <= N <= 1000) are standing in a row, each located at a distinct position on the number line. They are practicing throwing a baseball around, getting ready for an important game against the cows on the neighboring farm.

As Farmer John watches, he observes a group of three cows (X,Y,Z) completing two successful throws. Cow X throws the ball to cow Y on her right, and then cow Y throws the ball to cow Z on her right. Farmer John notes that the second throw travels at least as far and no more than twice as far as the first throw. Please count the number of possible triples of cows (X,Y,Z) that Farmer John could have been watching.

INPUT FORMAT:

* Line 1: The number of cows, N.

* Lines 2..1+N: Each line contains the integer location of a single
       cow (an integer in the range 0..100,000,000).

SAMPLE INPUT:

```
5
3
1
10
7
4
```

INPUT DETAILS:

There are 5 cows, at positions 3, 1, 10, 7, and 4.

OUTPUT FORMAT:

* Line 1: The number of triples of cows (X,Y,Z), where Y is right of
       X, Z is right of Y, and the distance from Y to Z is between XY
       and 2XY (inclusive), where XY represents the distance from X
       to Y.

SAMPLE OUTPUT:

4

OUTPUT DETAILS:

The four possible triples are the cows as positions 1-3-7, 1-4-7, 4-7-10, and 1-4-10.

Problem 3: Milk Scheduling

Farmer John has N cows that need to be milked (1 <= N <= 10,000), each of which takes only one unit of time to milk.

Being impatient animals, some cows will refuse to be milked if Farmer John waits too long to milk them. More specifically, cow i produces g_i gallons of milk (1 <= g_i <= 1000), but only if she is milked before a deadline at time d_i (1 <= d_i <= 10,000). Time starts at t=0, so at most x total cows can be milked prior to a deadline at time t=x.

Please help Farmer John determine the maximum amount of milk that he can obtain if he milks the cows optimally.

INPUT FORMAT:

* Line 1: The value of N.

* Lines 2..1+N: Line i+1 contains the integers g_i and d_i.

SAMPLE INPUT:

```
4
10 3
7 5
8 1
2 1
```

INPUT DETAILS:

There are 4 cows. The first produces 10 gallons of milk if milked by time 3, and so on.

OUTPUT FORMAT:

* Line 1: The maximum number of gallons of milk Farmer John can
       obtain.

SAMPLE OUTPUT:

```
25
```

OUTPUT DETAILS:

Farmer John milks cow 3 first, giving up on cow 4 since she cannot be milked by her deadline due to the conflict with cow 3. Farmer John then milks cows 1 and 2.

Problem 4: Vacation Planning

Air Bovinia operates flights connecting the N farms that the cows live on (1 <= N <= 20,000). As with any airline, K of these farms have been designated as hubs (1 <= K <= 200, K <= N).

Currently, Air Bovinia offers M one-way flights (1 <= M <= 20,000), where flight i travels from farm u_i to farm v_i and costs d_i (1 <= d_i <= 10,000) dollars. As with any other sensible airline, for each of these flights, at least one of u_i and v_i is a hub. There is at most one direct flight between two farms in any given direction, and no flight starts and ends at the same farm.

Bessie is in charge of running the ticketing services for Air Bovinia. Unfortunately, while she was away chewing on delicious hay for a few hours, Q one-way travel requests for the cows' holiday vacations were received (1 <= Q <= 50,000), where the ith request is from farm a_i to farm b_i.

As Bessie is overwhelmed with the task of processing these tickets, please help her compute whether each ticket request can be fullfilled, and its minimum cost if it can be done.

To reduce the output size, you should only output the total number of ticket requests that are possible, and the minimum total cost for them. Note that this number might not fit into a 32-bit integer.

INPUT FORMAT:

* Line 1: The integers N, M, K, and Q.

* Lines 2..M + 1: Line i+1 contains u_i, v_i, and d_i. (1 <= u_i, v_i <= N, u_i != v_i)

* Lines M + 2..M + K + 1: Each of these lines contains the ID of a single hub (in the range 1..N).

* Lines M + K + 2..M + K + Q + 1: Two numbers per line, indicating a request for a ticket from farm a_i to b_i. (1 <= a_i, b_i <= N, a_i != b_i)

SAMPLE INPUT:

```
3 3 1 2
1 2 10
2 3 10
2 1 5
2
1 3
3 1
```

OUTPUT FORMAT:

* Line 1: The number of ticket requests that can be fullfilled.

* Line 2: The minimum total cost of fulling the possible ticket
       requests

SAMPLE OUTPUT:

```
1
20
```

OUTPUT DETAILS:

For the first flight, the only feasible route is 1->2->3, costing 20.
There are no flights leaving farm 3, so the poor cows are stranded there.

Problem 5: The Bessie Shuffle

Bessie is practicing her card tricks. She has already mastered the Bessie-shuffle -- a shuffle on M (2 <= M <= 100,000) cards that reorganizes the cards so the i-th card from the top is now the P[i]-th card from the top.

Now Bessie is practicing shuffles on larger decks. She has a deck of N cards (M <= N <= 1,000,000,000) conveniently labeled 1 to N. She shuffles this deck by taking the first M cards and performing the Bessie-shuffle on them, placing the shuffled cards back on top of the deck. She then removes the top card from the deck and places it face down. She repeats this process, placing the top cards successively on top of each other, until she is out of cards. When Bessie has less than M cards left, she no longer performs the Bessie-shuffle, but continues to place the top card on top of the others.

Bessie knows that the deck initially started in sorted order, with 1 on top, 2 next, and N on the bottom. Given the description of the Bessie-shuffle, help Bessie compute which cards end up located at Q different specified positions (1 <= Q <= N, Q <= 5,000) in the deck.

50% of test cases will have N <= 100,000.

INPUT FORMAT:

* Line 1: A single line containing N, M and Q separated by a space.

* Lines 2..1+M: Line i+1 indicates the position from the top, P[i], of
  the i-th card in the Bessie-shuffle (1 <= P[i] <= M).

* Lines 2+M..1+M+Q: Line i+1+M contains a single integer q_i
  describing the i-th query. You are to compute the label on
  the card located in position q_i from the top (1 <= q_i <= N).

SAMPLE INPUT:

```
5 3 5
3
1
2
1
```

2
3
4
5

INPUT DETAILS:

Bessie has a deck of 5 cards initially ordered as [1, 2, 3, 4, 5]. Her shuffle is on 3 cards and has the effect of moving the top card to the bottom. There are 5 queries querying each position in the deck.

OUTPUT FORMAT:

* Lines 1..Q: On the i-th line, print a single integer indicating the card at position q_i from the top.

SAMPLE OUTPUT:

4
5
3
1
2

OUTPUT DETAILS:

The shuffle proceeds as:

[1, 2, 3, 4, 5] -> [2, 3, 1, 4, 5] (put 2 face down)
[3, 1, 4, 5] -> [1, 4, 3, 5] (put 1 face down)
[4, 3, 5] -> [3, 5, 4] (put 3 face down)
[5, 4] (put 5 face down)
[4] (put 4 face down)

This produces the final order of [4, 5, 3, 1, 2]