Knights of the Frozen Throne

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Knights of the Frozen Throne is a famous online game developed and published by ICPC Entertainment, Inc.

A single server called *Zeus Super Computer* (ZSC) handles all requests from all players of the game. Originally, all players' data is stored in ZSC's memory so that the server can respond to a request without querying the database. However, this design hardly scales as the game becomes more and more popular.

As a great engineer in ICPC Entertainment, Tom wants to help the company save costs by using an awesome database with a very large capacity. After some investigation, Tom discovers the behavior patterns of all players, and he can accurately predict when a player will send a request. Now it's time to deploy his optimization design: time-based caching.

To introduce the time-based caching policy, instead of holding all players' data in memory all the time, a player's data is retained in memory only for a while, and ZSC will drop this player's data if he doesn't send any request for some time. Specifically, when the request from one player comes,

- if the player's data is not in memory, ZSC loads his data into memory and records the *last request time* of the player;
- if the player's data is already in memory, ZSC responds to this request immediately, then updates the *last request time* of the player.

Also, ZSC would drop a player's data from memory if X seconds (X is a positive integer parameter to be determined) have elapsed since his last request. In particular, if a request is sent exactly X seconds after the player's *last request time*, ZSC does have to load the player's data into memory. Initially, ZSC's memory is empty.

Tom wants you to help him determine the best positive integer X such that the total cost is minimized.

As for the cost of CPU and memory, it costs a dollars for ZSC to hold one player's data in memory for one second. ZSC's CPU is pretty powerful, and it costs $i \cdot b_i$ if i players' data is loaded in one second.

Input

The input starts with a line of a single integer m ($1 \le m \le 10^5$), indicating the number of game players.

Then follow m lines, specifying the predicted requests of the players. Each of them begins with a single integer k ($1 \le k \le 5 \times 10^5$), denoting the number of requests player i will send. The remaining k integers $p_1, p_2, \cdots, p_k$ ($1 \le p_1 \le p_2 \le \cdots \le p_k \le 10^9$) are the times(in second) predicted by Tom that the player will send request to ZSC. It is guaranteed that the total number of requests sent by all players does not exceed 5×10^5.

The next line contains an integer a ($1 \le a \le 10^4$), the cost for ZSC to hold one player's data in memory for one second.

The last line contains m integers $b_1, b_2, \cdots, b_m$ ($1 \le b_i \le 10^9$), the cost of loading one player's data if i players' data is loaded in one second.

Output

Print two integers in the first line, denoting the minimum total cost and the number of different positive integers X that can achieve this minimum cost. Then list all these values of X in increasing order in the second line.

It can be proved that there are finitely many different positive integers X, such that the total cost is minimized.

Examples

standard input	standard output
4 3 1 4 5 2 2 4 4 2 4 7 8 6 1 3 5 7 8 10 1 2 4 6 5	53 2 3 4
3 3 1 1 4 5 1 4 4 5 7 3 3 4 5 2 6 4 3	55 1 1

Note

The total cost of different values of X is shown in Figure 1.

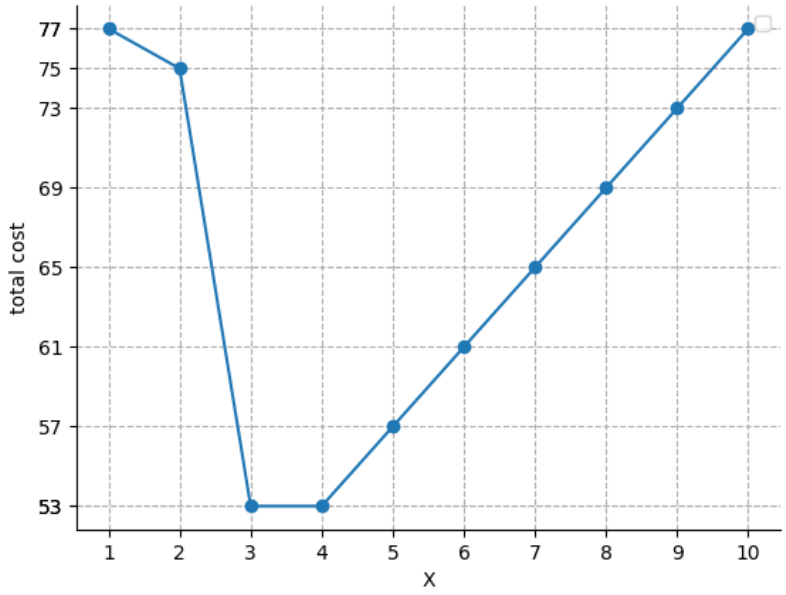

Рис. 1: The first sample data.