Ex. 1: Car rental

We are going to rent a car and travel by road from the city of Sort to the city of Selection for the Binary Search algorithm birthday party. The road connecting the cities of Sort and Selection is a long straight D kilometers. Between the cities of Sort and Selection, there are intermediate stations (part which may be very close to each other) which are the only places where one can refuel one's car with fuel. In particular, station j is located d_j km from the city of Selection.
Dimitris arrived late at the car rental office and is planning his trip to the city of Selection, in order to catch the start of the party, in exactly T minutes. For your trip, you will rent one of the N cars that are available. Each car i is characterized by the rental cost p_i and its capacity in fuel c_i. That aside, cars are standard. They all have two modes, the economic one, with which they cover a kilometer in T_s minutes and the consumption of C_S liters per km and the sport, with which they cover one kilometer in $T_f < T_s$ minutes and fuel consumption $C_f > C_s$ liters of fuel per kilometer. Fortunately, at least, the switching time from one mode of operation to another is zero and the driver can change as many times as he wants the operation of his vehicle. Dimitris loves sport and is in a hurry to get to his destination, but he wants to be sure that he will not run out of fuel in the middle of the road. Dimitris wants to write a program that calculates the minimum cost of renting a car that can cover the distance from the city of Sort to the city of Selection in no more than T minutes.

Input Data: The program will initially read from the standard input four positive integers, the number N of the available ones, the number K of the intermediate stations, the distance D of the cities of Sort and Selection in kilometers, and the time margin T that Dimitris has at his disposal to complete his journey. In each of the remaining N lines, there will be two positive integers p_i and c_i corresponding to the rental cost and the capacity of car i (the numbering of cars is arbitrary, while there may be cars i and j where $p_i > p_j$ and $c_i < c_j$. In the next line, there will be K the integers $d_1, \ldots, d_k$ that correspond to the distances (in km) of the stations inbetween from the original city of Sort. The numbering of intermediate stations is arbitrary (ie the intermediate stations are not necessarily numbered in ascending or descending distance from the city of Selection), while there may be i ≠ j stations with $d_i = d_j$. In the last line of the input, there will be four positive integers T_s, C_s, T_f, C_f that correspond to the time performance and consumption of the input economic and sporty pace of operation instead.

Output Data: The program must print in the standard output an integer, the minimum cost of renting a car that can cover the distance from the city of Sorting to the city of Selection in time no bigger than T minutes. In case there is no such car, the program must print -1.

Restrictions

$$1 \le N \le 4 \cdot 10^5$$
$$1 \le K \le 2 \cdot 10^5$$
$$1 \le D, p_i, c_i \le 10^9$$
$$1 \le T \le 2 \cdot 10^9$$
$$0 < d_j < D$$
$$1 \le T_f < T_s \le 2 \cdot 10^9$$
$$1 < C_s < C_f < 10^9$$

Input Example

```
3 1 8 10
10 8
5 7
11 9
3
2 1 1 2
```

Output

```
10
```

Ex. 2: Teleports

As you may know, there are infinite parallels to all of them. Morty, Rick's grandson, stole his grandfather's portal gun for to impress Jessica. But, without noticing, he was teleported to another parallel universe. To be precise, we know that M_i, aka Morty of universe I, teleported to universe c_i in {1, ..., N}. After all these unfortunate teleports, we continue to have a single Morty in each of the N parallel universes (i.e. the sequence c = $(c_1, ..., c_N)$ is a transposition of {1, ..., N}). Luckily for them, M portals were left open between some universies which two Mortys can use to exchange worlds. Each open portal j connects two parallel universes α_j , b_j , has width w_j and can be used infinite times. The Mortys must return to the correct universes (ie restore the c sequence to its original form) before the Ricks notice! The Mortys are grumpy in all N universies and don’t want to get crammed into open portals that are narrow.

Write a program that helps the Mortys return comfortably to their (correct) parts, ie. to restore the c sequence to its original form, calculating the maximum width of the narrowest portal they need to use for this purpose.

Input Data: Your program will initially read from the standard input two positive integers, the number N of the parallel universes we are interested in and the number M of portals that have been left opened. In the next line, a transposition c = $(c_1, …, c_N)$ of the set {1, …, N} is given, in which c_i indicates the parallel universe that Morty of the i-th universe ended up in. In the next M lines, M triads of natural numbers are given, which indicate the portals that have been left opened. To be more precise, each open portal j is described by 3 natural numbers; the universes α_j and b_j it connects and the width of it, w_j.

Output Data: Your program must print in the standard output a natural number, that describes the maximum width of the narrowest portal that needs to be used, so the input sequence c = $(c_1, …, c_N)$ can be restored in its initial form. In all given inputs, the sequence c will be able to be restored to its initial form.

Restrictions

$1 \leq N \leq 10^5$
$1 \leq M \leq 10^5$
$1 \leq a_j \neq b_j \leq N$
$1 \leq w_j \leq 10^9$
Όριο χρόνου εκτέλεσης: 1 sec.
Όριο μνήμης: 64 MB.

Input Example

```
4 4
3 2 1 4
1 2 73
1 3 42
2 4 17
2 3 100
```

Output

```
73
```