G. Adilbek and the Watering System

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Adilbek has to water his garden. He is going to do it with the help of a complex watering system: he only has to deliver water to it, and the mechanisms will do all the remaining job.

The watering system consumes one liter of water per minute (if there is no water, it is not working). It can hold no more than c liters. Adilbek has already poured c_0 liters of water into the system. He is going to start watering the garden right now and water it for m minutes, and the watering system should contain at least one liter of water at the beginning of the i-th minute (for every i from 0 to $m - 1$).

Now Adilbek wonders what he will do if the watering system runs out of water. He called n his friends and asked them if they are going to bring some water. The i-th friend answered that he can bring no more than a_i liters of water; he will arrive at the beginning of the t_i-th minute and pour all the water he has into the system (if the system cannot hold such amount of water, the excess water is poured out); and then he will ask Adilbek to pay b_i dollars for each liter of water he has brought. You may assume that if a friend arrives at the beginning of the t_i-th minute and the system runs out of water at the beginning of the same minute, the friend pours his water fast enough so that the system does not stop working.

Of course, Adilbek does not want to pay his friends, but he has to water the garden. So he has to tell his friends how much water should they bring. Formally, Adilbek wants to choose n integers k_1, k_2, ..., k_n in such a way that:

- if each friend i brings exactly k_i liters of water, then the watering system works during the whole time required to water the garden;
- the sum $\sum\limits_{i=1}^{n} k_i b_i$ is minimum possible.

Help Adilbek to determine the minimum amount he has to pay his friends or determine that Adilbek not able to water the garden for m minutes.

You have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 5 \cdot 10^5$) – the number of queries.

The first line of each query contains four integers n, m, c and c_0 ($0 \le n \le 5 \cdot 10^5, 2 \le m \le 10^9, 1 \le c_0 \le c \le 10^9$) — the number of friends, the number of minutes of watering, the capacity of the watering system and the number of liters poured by Adilbek.

Each of the next n lines contains three integers t_i, a_i, b_i ($0 < t_i < m, 1 \le a_i \le c, 1 \le b_i \le 10^9$) — the i-th friend's arrival time, the maximum amount of water i-th friend can bring and the cost of 1 liter from i-th friend.

It is guaranteed that sum of all n over all queries does not exceed $5 \cdot 10^5$.

Output

For each query print one integer — the minimum amount Adilbek has to pay his friends, or -1 if Adilbek is not able to water the garden for m minutes.

Example

input Copy

```
4
1 5 4 2
2 4 2
0 4 5 4
```

Educational Codeforces Round 74 (Rated for Div. 2)

Finished

Practice

→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Practice

You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++11 5.1.0 ▾

Choose file: 选择文件 未选择任何文件

Submit

→ Problem tags

data structures greedy sortings

No tag edit access

→ Contest materials

- Announcement #1 (en)
- Announcement #2 (ru)
- Tutorial #1 (en)
- Tutorial #2 (ru)

```
2 5 3 1
1 2 4
3 1 3
2 3 5 1
2 1 1
1 4 3
```

output Copy

```
6
0
-1
4
```