Task Overview Sheet

	Garage	Mecho	Regions	Salesman
Type	Batch	Batch	Reactive	Batch
Detailed Feedback	**Full**	Partial	None	Partial
Time Limit (per test case)	1 second	1 second	8 seconds	3 seconds
Memory Limit (per test case)	32 MB	64 MB	128 MB	128 MB
Points	100	100	100	100
	400			

C++ programmers should be aware that using C++ streams (cin / cout) may lead to I/O bottlenecks and substantially slower performance. Please see the technical info sheet for ways to avoid this.

TECHNICAL INFO SHEET

These pages contain helpful information on how to avoid slow input/output performance with C++ streams (cin / cout), how to use 64-bit data types (variables) and how to properly communicate with the grader on interactive tasks. They also include reference for what options are given to the compilers and what stack limitations are in place.

Slow Input / Output with C++ Streams

When solving tasks with very large amounts of input / output data, you may notice that C++ programs using the cin and cout streams are much slower than equivalent programs that use the scanf and printf functions for input and output processing. Thus, if you are using the cin / cout streams we strongly recommend that you switch to using scanf / printf instead. However, if you still want to use cin / cout, we recommend adding the following line at the beginning of your program:

```
ios::sync_with_stdio(false);
```

and also making sure that you never use `endl`, but use `"\n"` instead.

Please note, however, that including `ios::sync_with_stdio(false)` breaks the synchrony between cin / cout and scanf / printf, so if you are using this, you should never mix usage of cin and scanf, nor mix cout and printf.

64-bit Data Types

For some tasks you may need to deal with numbers too large to fit in 32 bits. In these cases, you would have to use a 64-bit integer data type, such as `long long` in C/C++ or `int64` in Pascal. Here is some sample code that illustrates the usage of these data types:

C/C++

```
int main(void) {
    long long varname;
    scanf("%lld", &varname);
    // Do something with the varname variable
    printf("%lld\n", varname);
    return 0;
}
```

Pascal

```
var
    varname: Int64;
begin
    read(varname);
```

```
    { Do something with the varname variable }
    writeln(varname);
end.
```

Communication with Grader on Interactive Tasks

Whenever you solve an interactive task, you always need to flush the buffer of your output after every new line printed on the output. Here is some code to illustrate how to do this under C, C++ and Pascal:

C or C++ with scanf / printf

```
fflush(stdout);
```

In addition, when using scanf, you must avoid reading input in a way that blocks the execution of your program while waiting for white space on standard input. Such blocking might happen if you use `scanf` with a first argument that ends with a space or a new line. In particular, you can safely use "`%d`" as a `scanf` argument, but you should **NOT** use "`%d `" (with a trailing space) or "`%d\n`" (with a trailing new line).

C++ with cin / cout

```
cout << flush;
```

Pascal

```
flush(output);
```

Compiler Options

The following commands will be used to compile solutions of batch and interactive tasks (say the task name is abc):

C

```
gcc –o abc abc.c –std=gnu99 –O2 –s –static –lm –x c
```

C++

```
g++ –o abc abc.cpp –O2 –s –static –lm –x c++
```

Pascal

```
fpc –O2 –XS –Sg abc.pas
```

Stack Limitations

Whenever your program is executed through the contest system, the stack size will only be limited by the memory limit for the corresponding task.

GARAGE

Full Feedback Problem

A parking garage has ***N*** parking spaces, numbered from 1 to ***N*** inclusive. The garage opens empty each morning and operates in the following way throughout the day. Whenever a car arrives at the garage, the attendants check whether there are any parking spaces available. If there are none, then the car waits at the entrance until a parking space is released. If a parking space is available, or as soon as one becomes available, the car is parked in the available parking space. If there is more than one available parking space, the car will be parked at the space with the smallest number. If more cars arrive while some car is waiting, they all line up in a queue at the entrance, in the order in which they arrived. Then, when a parking space becomes available, the first car in the queue (i.e., the one that arrived the earliest) is parked there.

The cost of parking in dollars is the weight of the car in kilograms multiplied by the specific rate of its parking space. The cost does not depend on how long a car stays in the garage.

The garage operator knows that today there will be ***M*** cars coming and he knows the order of their arrivals and departures. Help him calculate how many dollars his revenue is going to be today.

TASK

Write a program that, given the specific rates of the parking spaces, the weights of the cars and the order in which the cars arrive and depart, determines the total revenue of the garage in dollars.

CONSTRAINTS

$1 \le N \le 100$ The number of parking spaces
$1 \le M \le 2{,}000$ The number of cars
$1 \le R_s \le 100$ The rate of parking space s in dollars per kilogram
$1 \le W_k \le 10{,}000$ The weight of car k in kilograms

INPUT

Your program must read from standard input the following data:

- The first line contains the integers ***N*** and ***M***, separated by a space.
- The next ***N*** lines describe the rates of the parking spaces. The s^{th} of these lines contains a single integer R_s, the rate of parking space number ***s*** in dollars per kilogram.
- The next ***M*** lines describe the weights of the cars. The cars are numbered from 1 to ***M*** inclusive in no particular order. The k^{th} of these ***M*** lines contains a single integer W_k, the weight of car ***k*** in kilograms.
- The next ***2*M*** lines describe the arrivals and departures of all cars in chronological order. A positive integer ***i*** indicates that car number ***i*** arrives at the garage. A negative integer ***-i*** indicates that car number ***i*** departs from the garage. No car will

depart from the garage before it has arrived, and all cars from 1 to ***M*** inclusive will appear exactly twice in this sequence, once arriving and once departing. Moreover, no car will depart from the garage before it has parked (i.e., no car will leave while waiting on the queue).

OUTPUT
Your program must write to standard output a single line containing a single integer: the total number of dollars that will be earned by the garage operator today.

GRADING
For a number of tests worth 40 points there will always be at least one available parking space for every arriving car. In these cases no car will ever have to wait for a space.

EXAMPLES

Sample Input	Sample Output
3 4 2 3 5 200 100 300 800 3 2 -3 1 4 -4 -2 -1	5300

Car number 3 goes to space number 1 and pays 300 * 2 = 600 dollars.
Car number 2 goes to space number 2 and pays 100 * 3 = 300 dollars.
Car number 1 goes to space number 1 (which was released by car number 3) and pays 200 * 2 = 400 dollars.
Car number 4 goes to space number 3 (the last remaining) and pays 800 * 5 = 4,000 dollars.

Sample Input	Sample Output
2 4 5 2 100 500 1000 2000 3	16200

1 2 4 -1 -3 -2 -4	

Car number 3 goes to space number 1 and pays 1,000 * 5 = 5,000 dollars.
Car number 1 goes to space number 2 and pays 100 * 2 = 200 dollars.
Car number 2 arrives and has to wait at the entrance.
Car number 4 arrives and has to wait at the entrance behind car number 2.
When car number 1 releases its parking space, car number 2 parks there and pays 500 * 2 = 1,000 dollars.
When car number 3 releases its parking space, car number 4 parks there and pays 2,000 * 5 = 10,000 dollars.

MECHO

Mecho the bear has found a little treasure – the bees' secret honeypot, which is full of honey! He was happily eating his newfound treasure until suddenly one bee saw him and sounded the bee alarm. He knows that at this very moment hordes of bees will emerge from their hives and start spreading around trying to catch him. He knows he has to leave the honeypot and go home quickly, but the honey is so sweet that Mecho doesn't want to leave too soon. Help Mecho determine the latest possible moment when he can leave.

Mecho's forest is represented by a square grid of ***N*** by ***N*** unit cells, whose sides are parallel to the north-south and east-west directions. Each cell is occupied by a tree, by a patch of grass, by a hive or by Mecho's home. Two cells are considered adjacent if one of them is immediately to the north, south, east or west of the other (but not on a diagonal). Mecho is a clumsy bear, so every time he makes a step, it has to be to an adjacent cell. Mecho can only walk on grass and cannot go through trees or hives, and he can make at most ***S*** steps per minute.

At the moment when the bee alarm is sounded, Mecho is in the grassy cell containing the honeypot, and the bees are in every cell containing a hive (there may be more than one hive in the forest). During each minute from this time onwards, the following events happen in the following order:

- If Mecho is still eating honey, he decides whether to keep eating or to leave. If he continues eating, he does not move for the whole minute. Otherwise, he leaves immediately and takes up to ***S*** steps through the forest as described above. Mecho cannot take any of the honey with him, so once he has moved he cannot eat honey again.

- After Mecho is done eating or moving for the whole minute, the bees spread one unit further across the grid, moving only into the grassy cells. Specifically, the swarm of bees spreads into every grassy cell that is adjacent to any cell already containing bees. Furthermore, once a cell contains bees it will always contain bees (that is, the swarm does not move, but it grows).

In other words, the bees spread as follows: When the bee alarm is sounded, the bees only occupy the cells where the hives are located. At the end of the first minute, they occupy all grassy cells adjacent to hives (and still the hives themselves). At the end of the second minute, they additionally occupy all grassy cells adjacent to grassy cells adjacent to hives, and so on. Given enough time, the bees will end up simultaneously occupying all grassy cells in the forest that are within their reach.

Neither Mecho nor the bees can go outside the forest. Also, note that according to the rules above, Mecho will always eat honey for an integer number of minutes.

The bees catch Mecho if at any point in time Mecho finds himself in a cell occupied by bees.

TASK

Write a program that, given a map of the forest, determines the largest number of minutes that Mecho can continue eating honey at his initial location, while still being able to get to his home before any of the bees catch him.

CONSTRAINTS

$1 \leq N \leq 800$	The size (side length) of the map
$1 \leq S \leq 1{,}000$	The maximum number of steps Mecho can take in each minute

INPUT

Your program must read from standard input the following data:

- The first line contains the integers ***N*** and ***S***, separated by a space.
- The next ***N*** lines represent the map of the forest. Each of these lines contains ***N*** characters with each character representing one unit cell of the grid. The possible characters and their associated meanings are as follows:

T denotes a tree
G denotes a grassy cell
M denotes the initial location of Mecho and the honeypot, which is also a grassy cell
D denotes the location of Mecho's home, which Mecho can enter, but the bees cannot.
H denotes the location of a hive

NOTE: It is guaranteed that the map will contain exactly one letter **M**, exactly one letter **D** and at least one letter **H**. It is also guaranteed that there is a sequence of adjacent letters **G** that connects Mecho to his home, as well as a sequence of adjacent letters **G** that connects at least one hive to the honeypot (i.e., to Mecho's initial location). These sequences might be as short as length zero, in case Mecho's home or a hive is adjacent to Mecho's initial location. Also, note that the bees cannot pass through or fly over Mecho's home. To them, it is just like a tree.

OUTPUT

Your program must write to standard output a single line containing a single integer: the maximum possible number of minutes that Mecho can continue eating honey at his initial location, while still being able to get home safely.

If Mecho cannot possibly reach his home before the bees catch him, the number your program writes to standard output must be **-1** instead.

GRADING

For a number of tests, worth a total of 40 points, ***N*** will not exceed 60.

EXAMPLES

Sample Input	Sample Output
7 3 TTTTTTT TGGGGGT TGGGGGT MGGGGGD TGGGGGT TGGGGGT THHHHHT	1

After eating honey for one minute, Mecho can take the shortest path directly to the right and he will be home in another two minutes, safe from the bees.

Sample Input	Sample Output
7 3 TTTTTTT TGGGGGT TGGGGGT MGGGGGD TGGGGGT TGGGGGT TGHHGGT	2

After eating honey for two minutes, Mecho can take steps $\rightarrow\uparrow\rightarrow$ during the third minute, then steps $\rightarrow\rightarrow\rightarrow$ during the fourth minute and steps $\downarrow\rightarrow$ during the fifth minute.

REGIONS

The United Nations Regional Development Agency (UNRDA) has a very well defined organizational structure. It employs a total of N people, each of them coming from one of R geographically distinct regions of the world. The employees are numbered from 1 to N inclusive in order of seniority, with employee number 1, the Chair, being the most senior. The regions are numbered from 1 to R inclusive in no particular order. Every employee except for the Chair has a single supervisor. A supervisor is always more senior than the employees he or she supervises.

We say that an employee A is a manager of employee B if and only if A is B's supervisor or A is a manager of B's supervisor. Thus, for example, the Chair is a manager of every other employee. Also, clearly no two employees can be each other's managers.

Unfortunately, the United Nations Bureau of Investigations (UNBI) recently received a number of complaints that the UNRDA has an imbalanced organizational structure that favors some regions of the world more than others. In order to investigate the accusations, the UNBI would like to build a computer system that would be given the supervision structure of the UNRDA and would then be able to answer queries of the form: given two different regions r_1 and r_2, how many pairs of employees e_1 and e_2 exist in the agency, such that employee e_1 comes from region r_1, employee e_2 comes from region r_2, and e_1 is a manager of e_2. Every query has two parameters: the regions r_1 and r_2; and its result is a single integer: the number of different pairs e_1 and e_2 that satisfy the above-mentioned conditions.

TASK

Write a program that, given the home regions of all of the agency's employees, as well as data on who is supervised by whom, interactively answers queries as described above.

CONSTRAINTS

$1 \le N \le 200{,}000$	The number of employees
$1 \le R \le 25{,}000$	The number of regions
$1 \le Q \le 200{,}000$	The number of queries your program will have to answer
$1 \le H_k \le R$	The home region of employee k (for $1 \le k \le N$)
$1 \le S_k < k$	The supervisor of employee k (for $2 \le k \le N$)
$1 \le r_1, r_2 \le R$	The regions inquired about in a given query

INPUT

Your program must read from standard input the following data:

- The first line contains the integers N, R and Q, in order, separated by single spaces.
- The next N lines describe the N employees of the agency in order of seniority. The k^{th} of these N lines describes employee number k. The first of these lines (i.e., the one describing the Chair) contains a single integer: the home region H_1 of the Chair.

Each of the other ***N-1*** lines contains two integers separated by a single space: employee ***k***'s supervisor S_k, and employee ***k***'s home region H_k.

INTERACTION
After reading the input data, your program must start alternately reading queries from standard input and writing query results to standard output. The ***Q*** queries must be answered one at a time; your program must send the response to the query it has already received before it can receive the next query.

Each query is presented on a single line of standard input and consists of two different integers separated by a single space: the two regions r_1 and r_2.

The response to each query must be a single line on standard output containing a single integer: the number of pairs of UNRDA employees e_1 and e_2, such that e_1's home region is r_1, e_2's home region is r_2 and e_1 is a manager of e_2.

NOTE: The test data will be such that the correct answer to any query given on standard input will always be less than 1,000,000,000.

IMPORTANT NOTE: In order to interact properly with the grader, your program needs to flush standard output after every query response. It also needs to avoid accidentally blocking when reading standard input, as might happen for instance when using `scanf("%d\n")`. Please see the technical info sheet for instructions on how to do this properly.

GRADING
For a number of tests, worth a total of 30 points, ***R*** will not exceed 500.
For a number of tests, worth a total of 55 points, no region will have more than 500 employees.
The tests where both of the above conditions hold are worth 15 points.
The tests where at least one of the two conditions holds are worth 70 points.

EXAMPLE

Sample Input	Sample Output
6 3 4 1 1 2 1 3 2 3 2 3 5 1 1 2 1 3 2 3 3 1	 1 **[flush standard output]** 3 **[flush standard output]** 2 **[flush standard output]** 1 **[flush standard output]**

TESTING

If you would like to test your solution through the contest system's test interface, the input file you provide should include both the input data and all queries, as illustrated in the sample input above.

SALESMAN

The traveling salesman has decided that optimally scheduling his trips on land is an intractable computational problem, so he is moving his business to the linear world of the Danube River. He has a very fast boat that can get him from anywhere to anywhere along the river in no time, but unfortunately the boat has terrible fuel consumption. It costs the salesman ***U*** dollars for every meter traveled upstream (towards the source of the river) and ***D*** dollars for every meter traveled downstream (away from the source of the river).

There are ***N*** trade fairs that the salesman would like to visit along the river. Each trade fair is held for one day only. For each trade fair ***X***, the traveling salesman knows its date T_X, measured in the number of days since he purchased his boat. He also knows the fair's location L_X, measured as the distance in meters from the source of the river downstream to the fair, as well as the number of dollars M_X that the salesman is going to gain if he attends this trade fair. He has to start and end his journey at his waterfront home on the river, which is at location ***S***, measured also in meters downstream from the source of the river.

Help the traveling salesman choose which trade fairs to attend (if any) and in what order, so that he may maximize his profit at the end of his travels. The traveling salesman's total profit is defined as the sum of the dollars he gained at the fairs he attended, minus the total sum of dollars he spent traveling up and down the river.

Keep in mind that if trade fair ***A*** is held earlier than trade fair ***B***, the salesman can visit them only in this order (i.e., he cannot visit ***B*** and then visit ***A***). However, if two fairs are held on the same date, the salesman can visit them both in any order. There is no limit to how many fairs the salesman can visit in a day, but naturally he can't visit the same fair twice and reap the gains twice. He can pass through fairs he has already visited without gaining anything.

TASK

Write a program that, given the date, location and profitability of all fairs, as well as the location of the traveling salesman's home and his costs of traveling, determines the maximum possible profit he can make by the end of his journey.

CONSTRAINTS

$1 \le N \le 500{,}000$	The number of fairs
$1 \le D \le U \le 10$	The cost of traveling one meter upstream (***U***) or downstream (***D***)
$1 \le S \le 500{,}001$	The location of the salesman's home
$1 \le T_k \le 500{,}000$	The day on which fair ***k*** is held

$1 \le L_k \le 500{,}001$ The location of fair k
$1 \le M_k \le 4{,}000$ The number of dollars the salesman would earn if he attends fair k

INPUT

Your program must read from standard input the following data:

- The first line contains the integers N, U, D and S, in this order, separated by single spaces.
- The next N lines describe the N fairs in no particular order. The k^{th} of these N lines describes the k^{th} fair and contains three integers separated by single spaces: the day of the fair T_k, its location L_k, and its profitability for the salesman M_k.

NOTE: All locations given in the input will be different. That is to say, no two fairs will happen at the same location and no fair will happen at the salesman's home.

OUTPUT

Your program must write to standard output a single line containing a single integer: the maximum profit the salesman can possibly make by the end of his journey.

GRADING

For a number of tests, worth a total of 60 points, no two fairs will be held on the same day.
For a number of tests, worth a total of 40 points, none of the numbers in the input will exceed 5,000.
The tests where both of the above conditions hold are worth 15 points.
The tests where at least one of the two conditions holds are worth 85 points.

EXAMPLE

Sample Input	Sample Output
4 5 3 100 2 80 100 20 125 130 10 75 150 5 120 110	50

An optimal schedule would visit fairs 1 and 3 (the ones at locations 80 and 75). The sequence of events and their associated profits and costs would be as follows:

- The salesman travels 20 meters upstream at a cost of 100 dollars. Profit so far: -100
- He attends fair number 1 and earns 100. Profit so far: 0

- He travels 5 meters upstream at a cost of 25. Profit so far: -25
- He attends fair number 3 where he earns 150. Profit so far: 125
- He travels 25 meters downstream to return home at a cost of 75. Profit at the end: 50