Task 1: `CollectingMushrooms`

Lim Li the Crab is running a mushroom plantation in her backyard. Her mushroom plantation can be modelled as a grid of R rows and C columns, and each grid square of her mushroom plantation can either be empty, contain a mushroom, or contain a sprinkler. For example, her mushroom plantation could look like this:

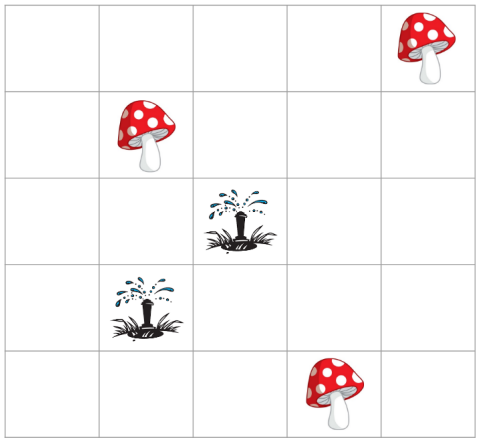

Figure 1: A mushroom farm with $R = 5$ and $C = 5$.

The distance between a sprinkler and a mushroom is defined as the maximum of their separation in the two axes. In other words, if the mushroom is located at row X_m and column Y_m while the sprinkler is located at row X_s and column Y_s, their distance will be $\max(|X_s - X_m|, |Y_s - Y_m|)$. Sprinklers only have a limited range, so a sprinkler can only water a mushroom if the distance between them is at most D. For example, if $D = 1$, the areas reachable by the two sprinklers will be:

Mushrooms can only grow and be harvested if enough sprinklers are watering it. Specifically, a mushroom will be *harvestable* if at least K sprinklers are watering it. Count the number of *harvestable* mushrooms Lim Li can collect in her plantation.

Input

The first line of input will contain four integers: R, the number of rows, C, the number of columns, D, the maximum distance between a sprinkler and a watered mushroom, and K, the

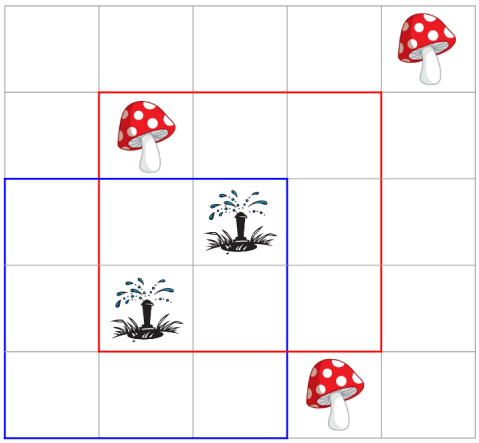

Figure 2: Diagram showing the range of the sprinklers.

minimum number of sprinklers required for a mushroom to be harvestable.

The next R lines of input will contain C characters each, containing a grid representing the mushroom plantation. Each character will represent the contents of a particular grid square, in the following way:

- '`.`' represents an empty grid square,
- '`M`' represents a grid square containing a mushroom,
- '`S`' represents a grid square containing a sprinkler.

Output

The output should contain one line with one integer, the maximum number of mushrooms Lim Li can harvest.

Subtasks

The maximum execution time on each instance is 1.0s. Your program will be tested on sets of input instances that satisfy the following restrictions:

- $2 \leq RC \leq 500000$,
- $1 \leq D \leq \max(R, C)$,

- $1 \leq K \leq RC$,
- there will be at least one mushroom,
- there will be at least one sprinkler.

Subtask	**Marks**	Additional Constraints
1	9	$1 \leq R, C \leq 100$, $D = \max(R, C)$, $K = 1$
2	10	$1 \leq R, C \leq 100$, $D = \max(R, C)$
3	18	$1 \leq R, C \leq 100$, $D = 1$, $K = 1$
4	23	$1 \leq R, C \leq 500$, no. of mushrooms ≤ 500, no. of sprinklers ≤ 500
5	19	$R = 1$
6	21	-

Sample Testcase 1

This testcase is valid for subtasks 3, 4 and 6.

Input	Output
5 5 1 1 ....M .M... ..S.. .S... ...M.	1

Sample Testcase 1 Explanation

Since the range of each sprinkler is only 1, meaning sprinklers can only reach adjacent squares, only the mushroom at (2, 2) is watered.

Sample Testcase 2

This testcase is valid for subtasks 1, 2, 4 and 6.

Input	Output
4 4 4 1 .... .M.. ..MM ...S	3

Sample Testcase 2 Explanation

Since the range of each sprinkler is 4, the lone sprinkler on the plantation can water all the mushrooms.

Sample Testcase 3

This testcase is valid for subtasks 4, 5 and 6.

Input	Output
1 8 5 2 SM..MM.S	2

Sample Testcase 3 Explanation

Each mushroom requires both sprinklers to be within range, since $K = 2$. Only two mushrooms satisfy this condition, the second and third mushrooms from the left.

Sample Testcase 4

This testcase is valid for subtasks 4 and 6.

Input	Output
5 5 2 2 ....M .M... ..S.. .S... ...M.	2

Sample Testcase 4 Explanation

Since the range of each sprinkler is 2, the mushroom at (2, 2) and the mushroom at (5, 4) can be watered by both sprinklers.

Task 2: `Journey`

Kuno Kunibertson is moving house from Amman to Brasília. His new employer pays the trip and allows him to make as many stop-overs as he wants; the only two conditions are: (1) the i-th stop-over city of the trip should be nearer to Brasília than the $(i-1)$-th stop-over city and (2) the trip should not take more than m nights. Furthermore, as Brasília is a transport hub, there is a direct flight from each city to Brasília. Kuno Kunibertson has to supply the following information to the trip evaluation:

1. Three integers n, m, h where n is the number of cities which might be visited on the trip, m is an upper bound on the number of nights the trip can take and h is the number of follow-on flights per stop-over city (for the ease of notation, the cities are numbers $0, 1, \ldots, n-1$ where 0 is Amman and $n-1$ is Brasília).
2. For each stop-over city, a list of exactly h possible follow-on flights and each such flight together is specified by the next city together with a minimum number of nights for a hotel stay at the next city Only those flights which go nearer to Brasília are relevant, the others should be ignored by the program.

The trip evaluation agency will then plan out a suitable trip for him with some artificial intelligence program, which optimises the trips according to a data base of tourist attractions. However, in order not to overload its computers, the trip evaluation agency has the following requirement:

> $(*)$ The number of itineraries which use up k nights $(k < m)$ should not be more than 500000000 (that is, 5×10^8).

Note that for counting, two itineraries are considered to be different, if one of the following conditions applies:

- one itinerary contains the stopover in city i and the other one does not;
- the departure day from city i is different in both itineraries;
- both itineraries go from city i to city j, but using different flights (this can be the case if Kuno Kunibertson lists two or more flights from city i to city j in his data).

For example, if the itinerary is Amman - Cairo - Dakar - Brasília, then an itinerary which departs from Cairo on day 2 differs from another itinerary which departs from Cairo on day 3. Furthermore, if there are two flights from Dakar to Brasília (even if they depart and arrive on the same days), the itineraries are different since they use these different flights.

As there is a penalty when the condition $(*)$ is violated, Kuno Kunibertson asks his friend Herbert Hercules to write for him a computer program which computes the number of itineraries which use up to k days.

Input format

Your program must read from standard input.

The first row of the input consists of three integers n, m and h, where n is the number of cities, m is the strict upper bound of the number of nights used and h is the number of follow-on flights per city. We assume city 0 is Amman and city $n-1$ is Brasília. We also assume city i is closer to Brasília than city j if $i > j$.

Then, the input contains $n-1$ rows, where the ith row, $i = 0, 1, \ldots, n-2$, consists of h pairs of integers (j, k), which means that Kuno Kunibertson can fly from city i to city j and stays at least k nights. If there are several flights to the same city, they are considered different. All $2h$ integers are listed in one row separated by spaces. Below is an example.

```
4 4 3
1 2 2 2 3 0
2 2 2 3 3 0
3 1 3 3 3 0
```

Here city 0 stands for Amman, city 1 might stand for Cairo, city 2 for Dakar and city 3 for Brasília. So the data stands for the following information.

City	Flight a To City	Min. Stay	Flight b To City	Min. Stay	Flight c To City	Min. Stay
Amman	Cairo	2	Dakar	2	Brasília	0
Cairo	Dakar	2	Dakar	3	Brasília	0
Dakar	Brasília	1	Brasília	3	Brasília	0

Output format

Your program must output to standard output only.

The output is a list of m integers between 0 and 500000001 which are, respectively, the number of itineraries from Amman to Brasília taking 0, 1, ..., $m-1$ nights, respectively. If the number of itineraries is 500000001 or more, the output should be 500000001.

The correct answer for the above input example is as follows:

```
1 1 3 6
```

Explanation

The itineraries are the following, where flight a is the first, flight b the second and flight c the third flight out of a city; the overall number of itineraries is 11.

Total Nights	Itinerary	Day	From	To	Flight	Nights at Next City
0	0	0	0	3	c	0
1	0	0	0	3	c	1
2	0	0	0	1	a	2
		2	1	3	c	0
	1	0	0	2	b	2
		2	2	3	c	0
	2	0	0	3	c	2
3	0	0	0	1	a	2
		2	1	3	c	1
	1	0	0	1	a	3
		3	1	3	c	0
	2	0	0	2	b	2
		2	2	3	a	1
	3	0	0	2	b	3
		3	2	3	c	0
	4	0	0	2	b	2
		2	2	3	c	1
	5	0	0	3	c	3

Subtasks

There are four subtasks; each subtask will be tested on several instances and the maximum execution time on each instance is 1.0 seconds. The subtasks and their marks are as follows.

Subtask	**Marks**	**Criteria**
1	20	at most one stop-over, $n, m, h \leq 10$
2	23	no minimum-stay, $n, m, h \leq 20$
3	26	$n, m, h \leq 100$
4	31	$n \leq 10000$, $m \leq 400$, $h \leq 100$

At most one stop-over means that all outgoing flights from a stop-over location go directly to Brasília. No minimum stay means that the stay in each stop-over can be any number (including 0) and that Kuno Kunibert can report to work on arrival in Brasília.

Sample Testcase 1

This testcase consists of test data with at most one stop over. It is valid for subtasks 1, 3 and 4.

Input	Output
4 8 3 1 0 2 1 3 0 3 1 3 2 3 0 3 1 3 1 3 0	2 5 11 17 23 29 35 41

Sample Testcase 2

This testcase consists of test data with no minimum stay. It is valid for subtasks 2, 3 and 4.

Input	Output
4 11 3 1 0 2 0 3 0 0 0 2 0 3 0 0 0 0 0 3 0	4 8 13 19 26 34 43 53 64 76 89

Sample Testcase 3

This testcase consists of test data which generate large output numbers so that the rule comes into place which limits the maximum of an output number to 500000001. The test case is valid for subtasks 2, 3 and 4.

Input	Output
8 8 8 1 0 1 0 1 0 1 0 1 0 1 0 1 0 7 0 2 0 2 0 2 0 2 0 2 0 2 0 2 0 7 0 3 0 3 0 3 0 3 0 3 0 3 0 3 0 7 0 4 0 4 0 4 0 4 0 4 0 4 0 4 0 7 0 5 0 5 0 5 0 5 0 5 0 5 0 5 0 7 0 6 0 6 0 6 0 6 0 6 0 6 0 6 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0 7 0	960800 6702725 26746084 80092734 199948848 439388874 500000001 500000001

Note that artificial linebreaks have been inserted into the output for this example to avoid an overlong line. These linebreaks are for display-purposes only and not required in the output.

Task 3: `LightningRod`

Singapore has anywhere between 171 and 186 lightning days on average a year. Each square kilometer of land in Singapore can be struck up to 16 times annually. This makes Singapore one of the lightning capitals of the world.

Gug the architect surveys N buildings from left to right, and notices that the top of building i, from left to right, has coordinates (X_i, Y_i). Gug wants to protect all the buildings by planting lightning rods on top of some buildings. A lightning rod protects the building it is planted on, and all buildings that lie on or under the 45° line of depression leftwards and rightwards. In other words, a lightning rod on building i protects building j if and only if $|X_i - X_j| \leq Y_i - Y_j$.

Help Gug find out the minimum number of lightning rods required to protect all buildings.

Input format

Your program must read from standard input.

The input starts with a single integer, N, in a single line. N denotes the total number of buildings.

N lines will then follow with 2 integers each, the i^{th} line will contain X_i and Y_i. This indicates that the peak of the i^{th} building is at (X_i, Y_i). You can assume $X_i \leq X_{i+1}$, in other words, X_i is increasing.

Note: The input size for subtasks 1, 6 and 7 is extremely large, so it is only possible to obtain full credit using C++ fast input. The attachment consists of a template that uses C++ fast input to read from standard input.

Output format

Your program must print to standard output.
Output a single integer, denoting the minimum number of lightning rods required to protect all buildings.

Subtasks

The maximum execution time on each instance is 1.0s. Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	N	X_i, Y_i
1	4	$2 \leq N \leq 10\,000\,000$	$0 \leq X_i \leq 10^9, Y_i = 1$
2	7	$N = 2$	$0 \leq X_i, Y_i \leq 10^9$
3	12	$2 \leq N \leq 20$	$0 \leq X_i, Y_i \leq 10^9$
4	21	$2 \leq N \leq 2\,000$	$0 \leq X_i, Y_i \leq 10^9$
5	26	$2 \leq N \leq 200\,000$	$0 \leq X_i, Y_i \leq 10^9$
6	10	$2 \leq N \leq 10\,000\,000$	$X_i = i, 0 \leq Y_i \leq 1$
7	20	$2 \leq N \leq 10\,000\,000$	$0 \leq X_i, Y_i \leq 10^9$

Sample Testcase 1

This testcase is valid for all subtasks.

Input	Output
2 1 1 2 1	2

Sample Testcase 1 Explanation

Both buildings must have lightning rods.

Sample Testcase 2

This testcase is only valid for subtasks 2 to 7.

Input	Output
2 1 0 2 1	1

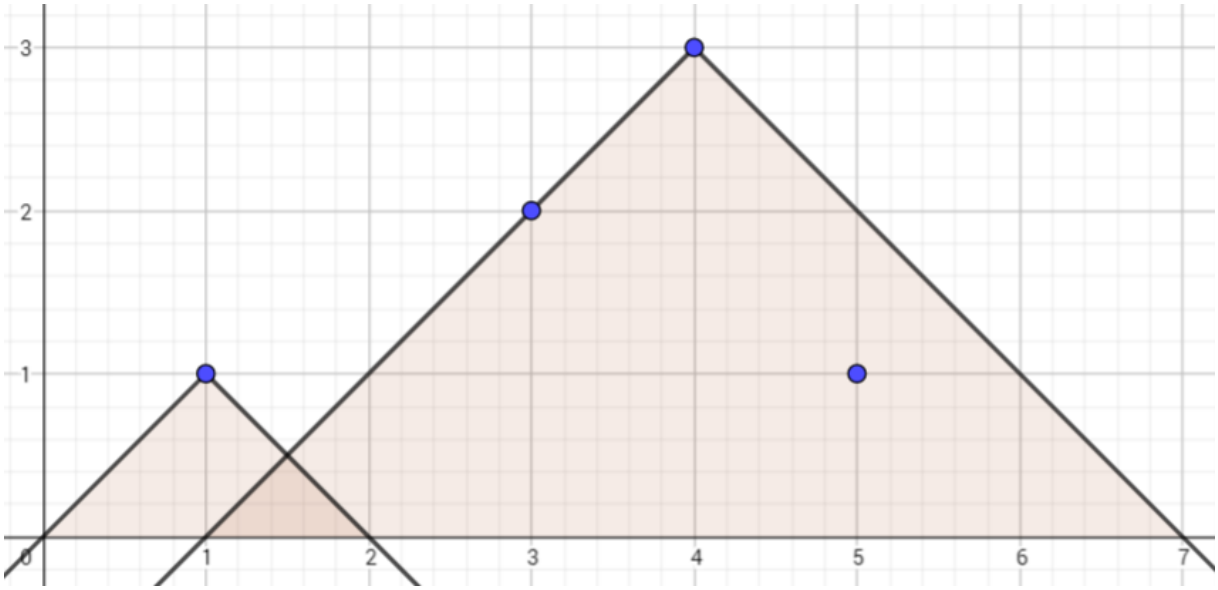

Figure 3: Sample 3, where Gug sees 4 buildings.

Sample Testcase 2 Explanation

A lightning rod can be planted on building 2.

Sample Testcase 3

This testcase is only valid for subtasks 3, 4, 5, 7.

Input	Output
4 1 1 3 2 4 3 5 1	2

Sample Testcase 3 Explanation

Lightning rods can be planted on buildings 1 and 3 (see Figure 3).

Task 4: `CityMapping`

Peanut is going for a holiday to the city Silvermill, and therefore he needs a map of Silvermill. Unfortunately, his computer is broken, and so he cannot Google for the map.

Silvermill is a city consisting of N road junctions, numbered 1 to N, and $N-1$ roads, with each road i connecting two road junctions A_i and B_i in both directions. This road also takes W_i minutes to traverse in either direction. It is guranteed that the city is fully connected, meaning it is possible to travel from any road junction to any other road junction using the roads in the city. To avoid congestion, the governor of Silvermill has decided that there will be **no more than three roads connected to each junction** in the city.

Peanut needs to obtain a map of Silvermill. In other words, he needs to obtain (A_i, B_i, W_i) for $i = 1, \ldots, N-1$.

Peanut has hired a cartographer in Silvermill to accomplish this task. As the cartographer is not very skilled, he can only answer a very simple question: what is the minimum amount of time, in minutes, required to travel between road junctions X and Y? Peanut did not pay enough money to this cartographer, so he only answers Q such questions before he quits his job.

Peanut needs you to help him to design a list of questions so that he can map out Silvermill and enjoy his holiday. Can you help him?

Implementation Details

In this task, you are asked to implement the following function: (It is different from the normal task, which requires the program to read from standard input and output to standard output.)

```
void find_roads(int N, int Q, int A[], int B[], int W[])
```

This function will be called exactly once with two input parameters and three output parameters. The two input parameters are N and Q. N is the number of road junctions while Q is the maximum number of queries allowed. The three output parameters are A, B and W. You are required to determine the $N-1$ roads in the city and return them in the arrays A, B and W. The endpoints of the roads and the roads themselves can be in any order.

You are allowed to call the following grader function to complete the task:

```
long long get_distance(int X, int Y)
```

This function will return a single integer, the time taken to travel between road junctions X and Y, in minutes. If you call this function more than Q times or provide invalid road junction numbers as arguments, the program will terminate immediately and you will be given a *Wrong Answer* verdict.

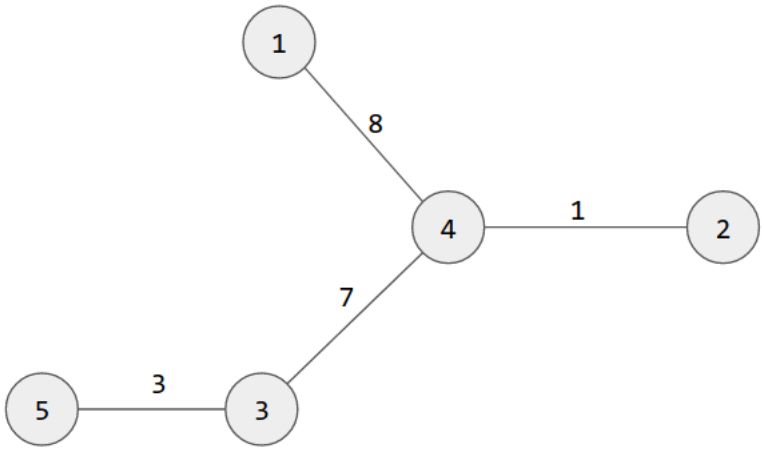

Figure 4: A city with 5 road junctions and 4 roads.

Consider the city map found in Figure 4. Suppose $Q = 500000$. Then, you are asked to implement the function:

```
find_roads(5, 500000, A[], B[], W[])
```

A possible interaction between your program *find_roads* and the grader functions *get_distance* could be as follows:

- `get_distance(5, 4) = 10`

  The *get_distance* function is asked to find the distance between road junctions 4 and 5. The answer is 10, as it takes 3 minutes to get from road junction 5 to 3, then another 7 minutes to get from road junction 3 to 4.

- `get_distance(2, 4) = 1`

  The *get_distance* function is asked to find the distance between road junctions 2 and 4. The answer is 1, as it takes 1 minute to directly travel from road junction 2 to 4.

- `get_distance(1, 3) = 15`

  The *get_distance* function is asked to find the distance between road junctions 1 and 3. The answer is 15, as it takes 8 minutes to get from road junction 1 to 4, then another 7 minutes to get from road junction 4 to 3.

- `get_distance(1, 2) = 9`

  The *get_distance* function is asked to find the distance between road junctions 1 and 2. The answer is 9, as it takes 8 minutes to get from road junction 1 to 4, then another 1 minutes to get from road junction 4 to 2.

At this point, the contestant's *find_roads* function decides it has enough information to deduce the map of the city. It sets $A = [3, 4, 4, 5]$, $B = [4, 1, 2, 3]$ and $W = [7, 8, 1, 3]$ and then terminates. This is a possible correct answer. The endpoints of the roads and the roads themselves can be provided in any order.

Subtasks

The maximum execution time on each instance is 1.0s. Your program will be tested on sets of input instances that satisfy the following restrictions:

- $2 \leq N \leq 1000$,
- $1 \leq A_i, B_i \leq N$,
- $1 \leq W_i \leq 10^9$.

Subtask	**Marks**	Q	Additional Constraints
1	9	$Q = 500000$	$W_i = 1$.
2	16	$Q = 500000$	No additional constraint.
3	13	$Q = 12000$	Each junction has at most **two roads** connected to it, $W_i = 1$.
4	19	$Q = 12000$	Each junction has at most **two roads** connected to it.
5	43	$Q = 25000$	No additional constraint. Please read *Scoring* section for further details.

Scoring

Subtask 5 is a special scoring subtask. Your score depends on the maximum number of queries q you make in any testcase.

- If $q > 25000$, you will score 0 points.
- If $12000 < q \leq 25000$, you will score $10 - 10 \times \frac{q-12000}{13000}$ points.

- If $6500 < q \leq 12000$, you will score $40 - 30 \times \frac{q-6500}{5500}$ points.
- If $q \leq 6500$, you will score 43 points.

Testing

You may download the grader file, *grader.cpp*, the header file, *citymapping.h*, a solution template, *citymapping.cpp*, and the compilation command, *compile.sh* under *Attachments*. Edit the solution template *citymapping.cpp* directly when implementing your solution, and run *compile.sh* to compile the executable. A small and large testcase for each subtask are also provided for your reference. The grader input format is provided below.

Grader Input Format

- one line with three integers: N, the number of road junctions, Q, the maximum number of queries allowed, and S, the subtask number.
- $N - 1$ lines, with the i-th line containing three integers: A_i, B_i, W_i.

Task 5: `Safety`

Squeaky the Mouse has recently gained an appreciation for the visual arts, and is now attempting his own work of art to be put on display in the most prestigious visual arts festival in town.

His artwork consists of many stacks of similar-sized illuminated cubes arranged to form a line. More precisely, there are N stacks, numbered from 1 to N from left to right, and stack i contains $S[i]$ cubes. The following is one possibility of Squeaky's artwork:

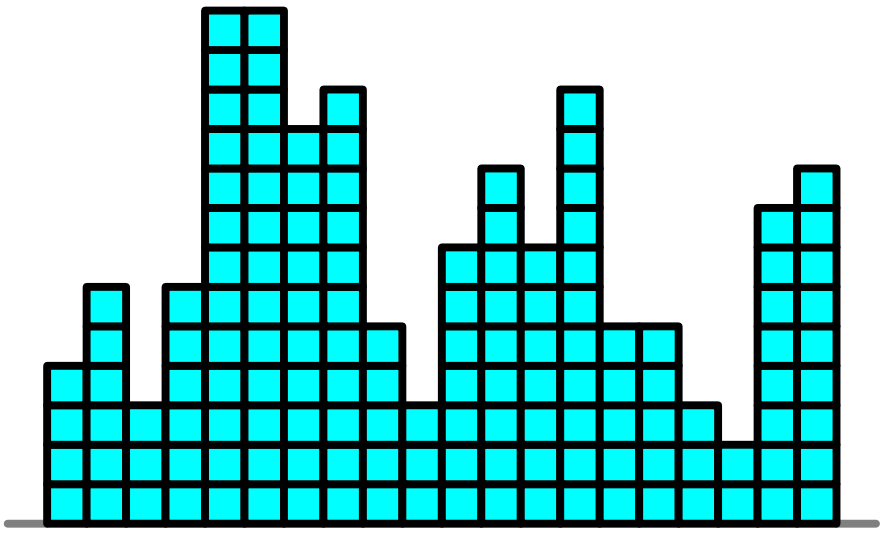

Figure 5: Possible artwork configuration, with $N = 20$

As the cubes are massive, assembling the cubes to form the artwork is a very exhausting task, and Squeaky was only able to complete its assembly a few days before the start of the festival.

Just as he thought he was able to finally take a break, the safety committee came around to assess his artwork. The safety committee at this festival has been very particular and uncompromising ever since a disastrous mishap occurred during the festival last year.

When Squeaky saw the committee huddling together and speaking to one another in hushed voices, his heart sank. He knew that they found an issue with his work. Eventually, some members of the committee approached him and explained their concern: Some of the stacks might topple over when visitors bump into the artwork. Specifically, the artwork is only safe if the heights of adjacent stacks differ by no more than H cubes; equivalently, $|S[i] - S[i+1]| \leq H$ for all $1 \leq i \leq N-1$.

They then gave him two choices – either alter the artwork to make it safe, or remove the artwork completely.

Of course, having spent so much effort on this piece, Squeaky didn't consider removing the artwork to be an option at all, so he opted to alter the artwork by adding and removing some cubes. As carrying cubes around is tiring, he wants to minimize the amount of work he needs to do.

Formally, he wants to minimise the number of steps needed to make his artwork safe, where each step is one of the following:

- Add one cube to the top of stack k
- Remove one cube from the top of stack k

Help Squeaky determine the minimum number of steps he needs to make his artwork safe.

Input format

Your program must read from standard input.
The first line of input contains two positive integers, N and H.
The second line of input contains N non-negative integers. The i^{th} integer on this line is $S[i]$.

Output format

Your program must output a single integer — the minimum number of operations required to make the artwork safe.

Subtasks

The maximum execution time on each instance is 1.0s.

For all instances, the input will satisfy the following constraints:

- $1 \leq N \leq 200\,000$
- $0 \leq H \leq 1\,000\,000\,000$
- $0 \leq S[i] \leq 1\,000\,000\,000$ for all $1 \leq i \leq N$

Your program will be tested on sets of input instances, as follows:

Subtask	Marks	Additional Constraints
1	3	$N \leq 10$, $S[i] \leq 4$
2	4	$N \leq 14$, $H \leq 1$, $S[i] \leq 4$
3	9	$N \leq 10$, $H \leq 2$
4	5	$H = 0$
5	6	$N \leq 500$, $S[i] \leq 400$
6	11	$N \leq 500$, $S[i] \leq 5\,000$
7	11	$N \leq 5\,000$, $S[i] \leq 5\,000$
8	22	$N \leq 5\,000$
9	29	No additional constraints

Sample Testcase 1

This testcase is valid for Subtasks 3, 5-9.

Input	Output
`6 1` `2 10 0 2 4 3`	`10`

Explanation

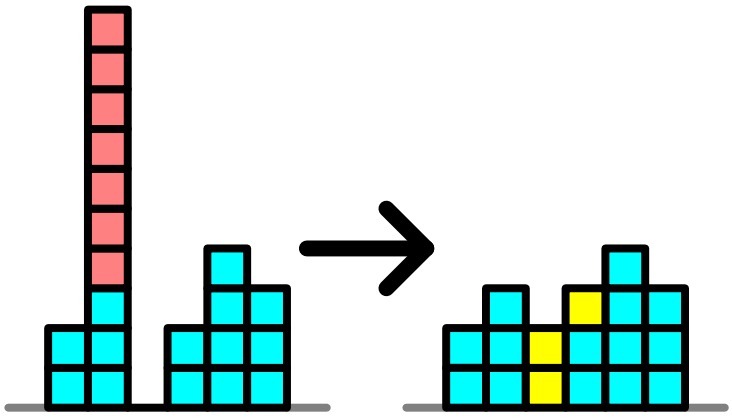

Figure 6: Possible solution for sample testcase 1. Original artwork is on the left; altered artwork is on the right. Cubes highlighted in red represent those that need to be removed from the original artwork; cubes highlighted in yellow represent those that need to be added to the altered artwork.

The above figure demonstrates a way to make the artwork safe. The artwork is safe because the height of each pair of adjacent stacks differ by no more than one.

It requires ten steps (total number of cubes highlighted red and yellow). As there is no way to make the artwork safe that requires less than ten steps, the correct output is `10`. Note that there may be other ways to make the artwork safe that also require exactly ten steps.

Sample Testcase 2

This testcase is valid for Subtasks 5-9.

Input	Output
`6 3` `2 10 2 6 4 3`	`6`

Explanation

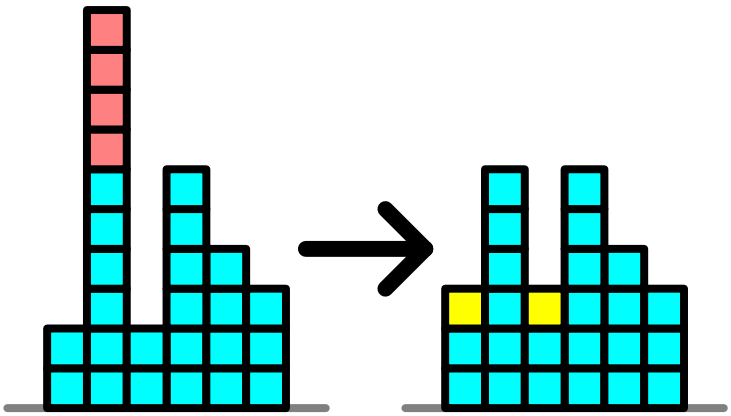

Figure 7: Possible solution for sample testcase 2.

At least six steps are needed to make the artwork safe. Hence, the correct output is `6`.

Sample Testcase 3

This testcase is valid for Subtasks 1-3, 5-9.

Input	Output
`4 1` `1 4 1 4`	`4`

Explanation

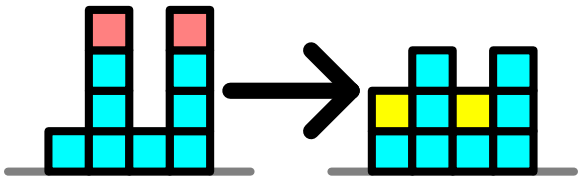

Figure 8: Possible solution for sample testcase 3.

At least four steps are needed to make the artwork safe. Hence, the correct output is `4`.

Sample Testcase 4

This testcase is valid for Subtasks 3, 5-9.

Input	Output
`10 1` `10 9 8 7 6 5 4 3 2 1`	`0`

Explanation

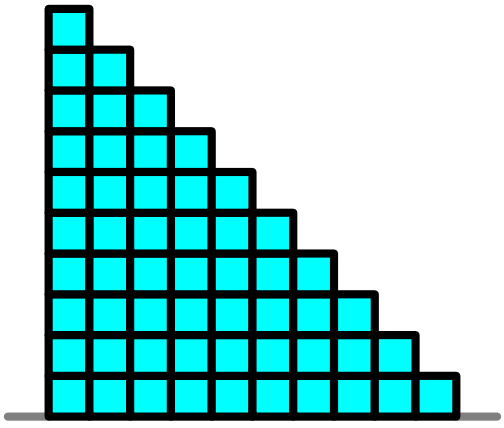

Figure 9: Artwork in sample testcase 4.

The artwork is already safe, so no alterations are needed. Hence, the answer is `0`.

Sample Testcase 5

This testcase is valid for all subtasks.

Input	Output
3 0 1 1 3	2

Explanation

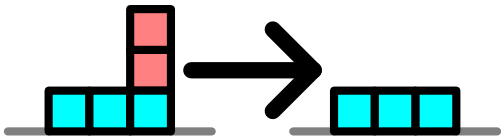

Figure 10: Possible solution for sample testcase 5.

At least two steps are needed to make the artwork safe. Hence, the correct output is `2`.