CS3230 Design and Analysis of Algorithms

Programming Assignment 1

Released August 31, 2020 8:00 AM

Due October 5, 2020 11:59 PM

Problems

Yggdrasil (80 marks)

Time limit: 2 seconds
Memory limit: 256 MB

Astrid has planted a sapling on hallowed ground. Legend has it that, when this tree grows to certain heights, it will grant its owner the powers of a dragon.

The tree has just been planted, and is currently 0 meters tall. It has a *growth rate* r, which is initially r_{orig}.

Every day, Astrid can choose to do *exactly one* of the following:

- Apply a *growth acceleration* fertilizer, which will *increase* r by 1;
- Apply a *growth deceleration* fertilizer, which will *decrease* r by 1;
- Do nothing, leaving r unchanged.

Note that she cannot apply a growth deceleration fertilizer if r is already 1.

Every night, the tree grows by r meters. The tree will magically disappear once its height becomes strictly greater than h meters.

Certain heights are considered *sacred*. Every day that a tree is exactly at a sacred height, Astrid receives greater power. Therefore, she wants to act in such a way as to maximize the number of days the tree's height is sacred.

This isn't a "treevial" problem, so she needs your help!

Input

The first line of input contains two integers r_{orig} and h, the original growth rate and the greatest height the tree can have before it disappears.

The second line of input contains a string s of h characters. The i^{th} character in this string is `s` if a height of i meters is sacred, and `.` otherwise.

Output

Output a single integer on a line by itself, the maximum number of days the tree's height can be sacred, if Astrid acts optimally.

Scoring

For all groups, $1 \leq r_{\text{orig}} \leq h$.

Group	Marks	h	Additional constraints
1	8	$h \leq 25$	The inputs for this group are public.
2	7	$h \leq 25$	s contains exactly one `s`.
3	10	$h \leq 25$	
4	21	$h \leq 2\,000$	
5	9	$h \leq 10\,000$	
6	25	$h \leq 50\,000$	

Example

Input

```
7 24
......ss......s....s...s
```

Output

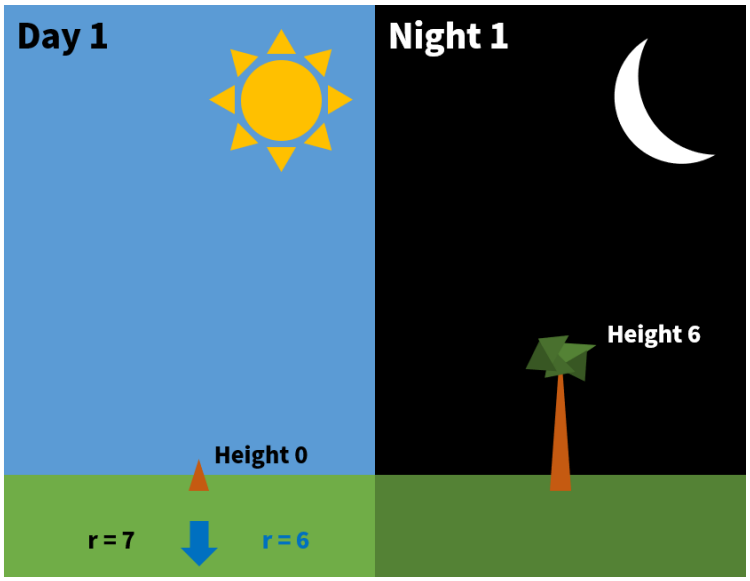
```
3
```

Note

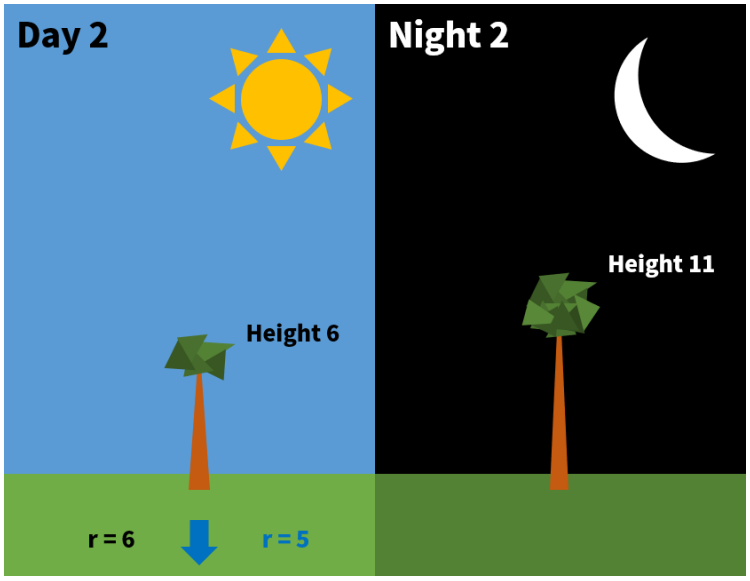
The following images illustrate the example. The sacred heights are 7, 8, 15, 20 and 24 meters, respectively.

One way of achieving 3 days where the height is sacred is as follows.

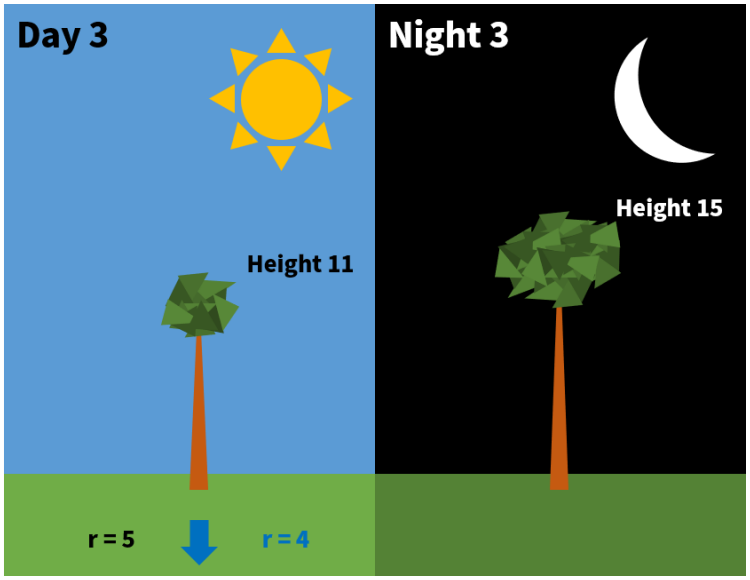
On the first day, Astrid should apply a growth deceleration fertilizer, which will decrease r from 7 to 6. That night, the tree grows to 6 meters:



On the second day, Astrid should again apply a growth deceleration fertilizer, which will decrease r from 6 to 5. That night, the tree grows to 11 meters:

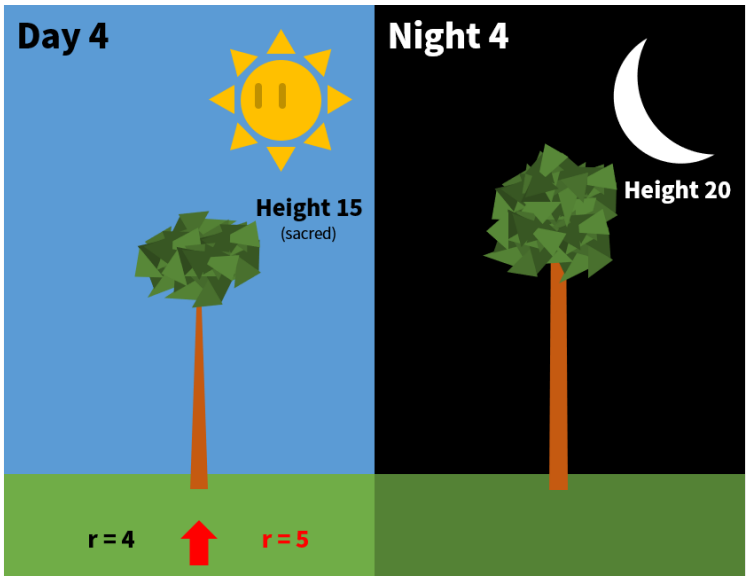


On the third day, Astrid should once again apply a growth deceleration fertilizer, which will decrease r from 5 to 4. That night, the tree grows to 15 meters:



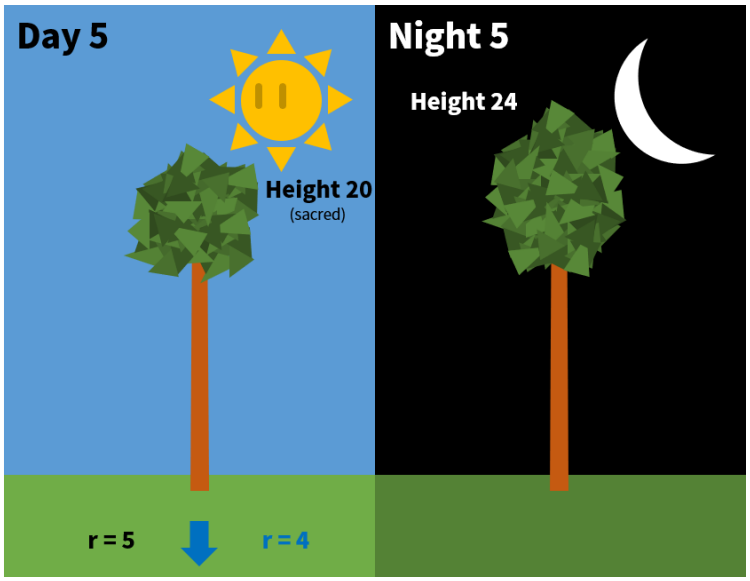
The tree is at a sacred height on the fourth day.

On the fourth day, Astrid should apply a growth acceleration fertilizer, which will increase r from 4 to 5. That night, the tree grows to 20 meters:



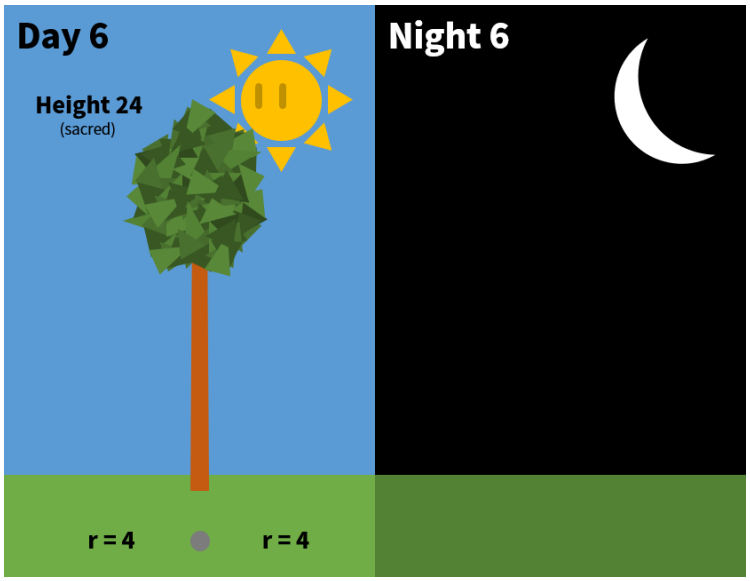
The tree is at a sacred height on the fifth day.

On the fifth day, Astrid should apply a growth deceleration fertilizer, which will decrease r from 5 to 4. That night, the tree grows to 24 meters:



The tree is at a sacred height on the sixth day.

On the sixth day, Astrid should do nothing; r retains its value of 4. That night, the tree grows to 28 meters and disappears:



There are 3 days where the tree is at a sacred height.

It can be proven that 4 or more such days is not possible, therefore 3 is the maximum.

The five inputs for **Group 1** have been made public to assist you with debugging. They are accessible here:

https://drive.google.com/drive/u/0/folders/1fpW7X6rZEOXqA5SApjsdNZ7uk1SVOMg2

Each input is a `.in` file, which is just a text file that can be opened by most text editors. Besides the first test (`1.in` and `1.out`, which corresponds to the example provided here), the corresponding outputs are not provided, but the inputs are small enough to be easily solved by hand.

It is possible to obtain the nominal 8 points by solving these inputs manually and hardcoding the answers into your program, but we suggest using them to construct and debug a more general solution. You are free to discuss the expected outputs for these inputs with your coursemates or teaching assistant if you have uncertainties about your understanding of the problem.

Rock Solid Friendship (20 marks)

Time limit: 2 seconds
Memory limit: 256 MB

Sweetie Belle and Button Mash need some rocks to build their house. Their blueprint calls for exactly n rocks, with integer sizes $s_1, s_2, \ldots, s_n$.

Not wanting to look for too many rocks, they've decided to look for just *one* massive rock of some integer size, and mine it repeatedly to get rocks of smaller sizes; among the remaining rocks, they should have the n rocks they need (along with, possibly, some extra).

Denote by $\lfloor x \rfloor$ the largest integer not greater than x, and $\lceil x \rceil$ the smallest integer not less than x.

When a rock of size $s > 1$ is mined, it breaks and two rocks, with sizes $\lfloor s/2 \rfloor$ and $\lceil s/2 \rceil$, are produced. These two rocks may again be mined to produce smaller rocks, and so on. Note that rocks of size 1 cannot be mined further.

Let's call a rock *useful* if it can produce the n rocks they need by mining it any number of times (possibly zero).

There are two parts to this task.

Part 1

Sweetie Belle has found a rock of size m. Is it useful?

Part 2

Button Mash decides that instead of trying many candidate rocks, they should first find out the size of the smallest useful rock, and *then* look for a rock of that size.

Does a useful rock even exist, and if so, what is the size of the smallest one?

Input

The first line of input contains a single integer p, either 1 or 2, denoting that the task to be solved is **Part 1** or **Part 2**, respectively.

For **Part 1**, the second line of input contains two integers n and m, the number of rocks they need and the size of the rock Sweetie Belle found, respectively.

For **Part 2**, the second line of input contains a single integer n, the number of rocks they need.

For both parts, the third line of input contains n integers $s_1, s_2, \ldots, s_n$, the sizes of the rocks they need.

Output

For **Part 1**, if the rock Sweetie Belle found is useful, output `YES`; otherwise, output `NO`.

For **Part 2**, if a useful rock exists, output a single integer, the size of the smallest such rock; otherwise, output `IMPOSSIBLE`.

Scoring

For all groups,

- $1 \leq n \leq 200\,000$;
- $1 \leq s_i \leq 10^9$.

Group	Marks	p	Additional constraints
1	5	$p = 1$	$1 \leq m \leq 10^{18}$
2	15	$p = 2$	

Examples

Input 1

```
1
3 13
2 2 6
```

Output 1

```
YES
```

Input 2

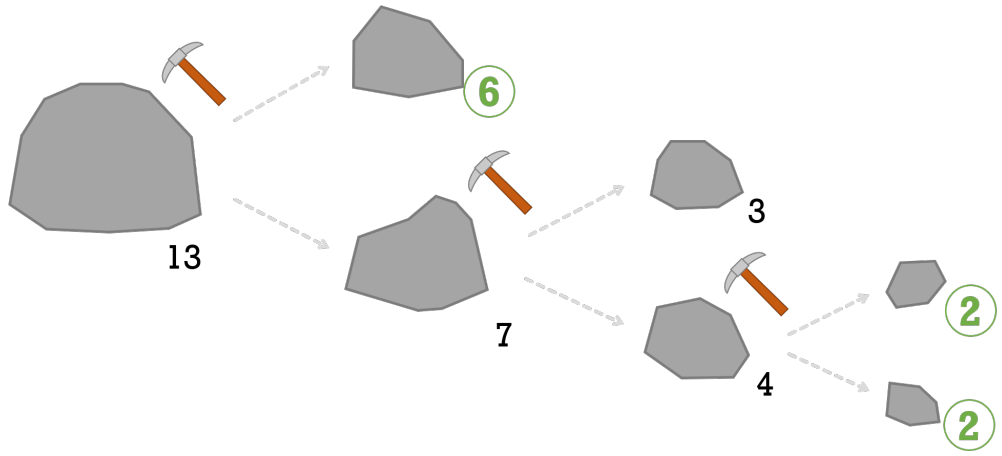
```
2
3
2 2 6
```

Output 2

```
11
```

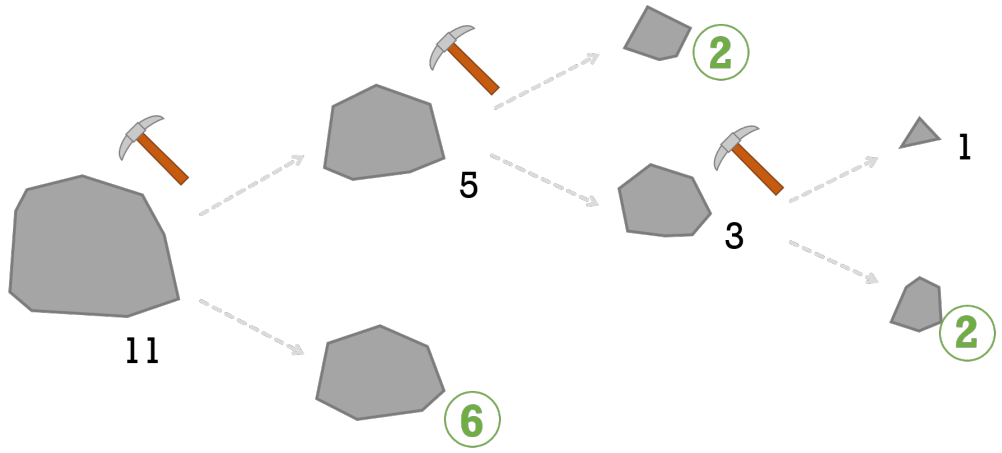
Note

The following image illustrates the first example. One way of producing the required rocks (highlighted in green) is shown:



Note that a rock of size 3 is left over.

The following image illustrates the second example. It shows that a rock of size 11 is useful:



It is easy to show that any useful rock must be of size at least 10.

On the other hand, a rock of size 10 is not useful, because it cannot produce a rock of size 6: mining it for the first time produces two rocks of size 5, and mining those rocks further can only result in smaller rocks.

The next smallest candidate is 11, which the above image shows is useful. Therefore, a rock of size 11 is the smallest possible useful rock.