

## Picarats Collection (picarats)

Professor Layton<sup>1</sup> is a series of puzzle games where the Professor and his small assistant Luke are challenged with a mystery to solve.

During the investigation the duo walks around the city and encounters various puzzles of different levels of difficulty: some of them are very easy, others are very challenging. Each of the  $N$  puzzles has a known positive level of difficulty  $P_i$ , expressed in *Picarats*, that denotes how hard it is to solve without any help.

Fortunately, during their journey they can also find *hint coins*. A coin is a magic aid that, when used, reduces the difficulty of a puzzle.

After the Professor and his assistant solve a puzzle, they can proceed to the next one, collecting all the *hint coins* they find in the journey. The decision on where to go next is entirely up to them, with the only requirement that after solving a puzzle one cannot come back to it again! The Professor is also very methodic: when he encounters a puzzle he has to solve it, he is unable to just ignore it and proceed.

Luke is trying to imitate the Professor, but he is scared not to be smart enough. He wants to solve the mystery, going from the puzzle 0 to the puzzle  $N - 1$  using *the least amount of smartness needed*. Solving a puzzle requires you to be as smart as its difficulty.

You can lower the difficulty of a puzzle by spending a *hint coin*, halving and rounding down its difficulty. For example a puzzle worth 15 *Picarats* with a *hint coin* is worth 7 *Picarats*, with 2 coins 3 *Picarats* and with 3 just 1 *Picarats*. Spending any more coin reduces the difficulty to zero, essentially revealing the solution!

The *smartness* one needs to solve the mystery is therefore the difficulty of the hardest puzzle solved. What is the value of this difficulty, assuming you start with  $C_0$  coins?



*Calm down, Luke.*

Among the attachments of this task you may find a template file `picarats.*` with a sample incomplete implementation.

### Input

The first line of the input contains 3 integers:  $N$ ,  $M$  and  $C_0$ , respectively the number of puzzles in the game, the number of connections between puzzles and the number of coins Luke has before the start of the game.

The next line contains  $N$  integers  $P_i$ , the difficulties of the puzzles in *Picarats*.

The next  $M$  lines contain 3 integers each:  $a$ ,  $b$ ,  $c$ . They indicate that after solving the puzzle  $a$  one can go to puzzle  $b$  collecting  $c$  hint coins in the journey. The order matters: it does not mean that one can also go from  $b$  to  $a$ .

### Output

You need to write a single line with an integer: the minimum smartness Luke needs to solve the mystery.








<sup>1</sup>[https://en.wikipedia.org/wiki/Professor\\_Layton](https://en.wikipedia.org/wiki/Professor_Layton)

## Constraints

- $2 \leq N \leq 10\,000$ .
- $1 \leq M \leq 50\,000$ .
- $0 \leq C \leq 100$  where  $C$  is the total number of coins you can ever find in the game, including  $C_0$ .
- $0 \leq P_i \leq 10^9$  for each  $i = 0 \dots N - 1$ .
- Luke cannot go arbitrarily from one puzzle to another one unless an explicit connection between them is present.

## Scoring

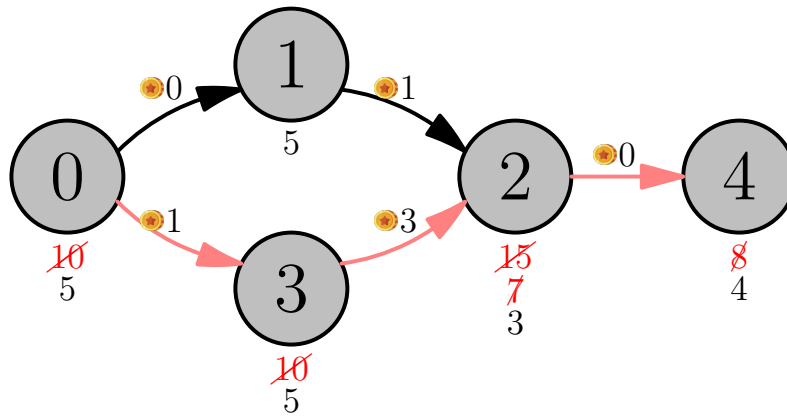
Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** (0 points)      Examples.  

- **Subtask 2** (10 points)      At each puzzle there is at most one next puzzle and  $C = 0$ .  

- **Subtask 3** (13 points)       $N \leq 10$  and  $C \leq 10$ .  

- **Subtask 4** (12 points)       $C = 0$ .  

- **Subtask 5** (21 points)       $C = 1$ .  

- **Subtask 6** (19 points)       $P_i = P_j$  for all  $i, j$ .  

- **Subtask 7** (25 points)      No additional limitations.  


## Examples

input	output
5 5 2 10 5 15 10 8 0 1 0 0 3 1 1 2 1 3 2 3 2 4 0	5
5 4 1 100 51 123 40 6 0 1 0 1 2 0 2 3 0 3 4 0	100

## Explanation



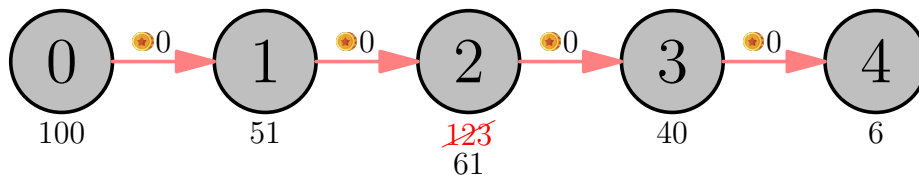
Representation of the first sample case.

In the **first sample case** the optimal solution (highlighted in red) is:

- Start at puzzle 0 with  $C_0 = 2$  coins.
- Spend a coin, halving its difficulty.
- Proceed to puzzle 3 collecting 1 coin. Now Luke has 2 coins in his pocket.
- Spend a coin halving its difficulty.
- Proceed to puzzle 2 collecting 3 coins. Now Luke has 4 coins in his pocket.
- Spend 2 coins halving its difficulty twice.
- Proceed to puzzle 4 collecting 0 coins. Now Luke has 1 coin in his pocket.
- Spend 1 coin halving its difficulty.

Now Luke has solved the final puzzle, effectively solving the mystery! The hardest puzzles he solved were 0 and 3, both with the highest difficulty: 5.

Note that by going through puzzle 1 the solution is suboptimal: he can spend a coin on 0, then 2 coins on 2 but he has no coins left to spend on 4, increasing the solution to 8. If instead he spends only a coin on 2, the solution would be 7 and still suboptimal.



Representation of the second sample case.

In the **second sample case** Luke has a single path to the solution. He can spend his only coin on the puzzle 2, lowering the difficulty from 123 to 61. The hardest puzzle he has to solve is the first.