mBIT Advanced Division Problems

June 7, 2020

These problems are roughly ordered by difficulty. However, we suggest you look through and think about as many problems as you can in the time provided. Good luck and happy coding!

Contents

1 Zoo Tour 3

2 Leaping Lizards 4

3 Raging Rhinos 5

4 Raccoon Mischief 6

5 Turtle Tribulation 7

6 Gorilla Grouping 8

7 Hen Hackers 9

8 Platypus Puddles 11

9 Playlist Shuffle 13

10 Penguin Mayhem 14

11 Sharing Seals 15

12 Zookeepers' Gathering 16

Thanks to Evan Chen for letting us use his style file

The memory limit for all problems is 256 MB.

Time Limits (seconds)			
Problem	**C++**	**Java**	**Python**
Zoo Tour	1	2	2
Leaping Lizards	1	1	1
Raging Rhinos	1	2	2
Raccoon Mischief	1	2	2
Turtle Tribulation	1.5	1.5	1.5
Gorilla Grouping	1	2	2
Hen Hackers	1	1	1
Platypus Puddles	2	3	3
Playlist Shuffle	1	1	1
Penguin Mayhem	1	2	3
Sharing Seals	1	1	1.5
Zookeepers' Gathering	2	2	4

Advice:

Go for partial credit! Even if you don't know how to solve a problem with the full bounds, you can still earn some of the points by coding a solution that works for smaller test cases. **On each problem, we guarantee that 50% of our test cases have reduced input sizes.** This means that suboptimal solutions can earn partial credit.

Watch out for integer overflow. When a problem uses large values, make sure you use `long` (in Java) or `long long` (in C++).

Consider using fast I/O. For problems with large input sizes, you may want to use faster I/O methods. We have increased our time limits so that fast I/O should not be necessary, but it doesn't hurt to be safe. Here is how to use fast I/O in each language:

- In Python, write `from sys import stdin, stdout` at the top of your program. When reading input, use `stdin.readline()`. To write output, use `stdout.write()`.
- In Java, use a custom Scanner class as shown here.
- In C++, write `ios_base::sync_with_stdio(0); cin.tie(NULL); cout.tie(NULL);` at the top of your `main` method. Then you can use `cin` and `cout` as usual.

Special considerations for Python. In Python, avoid multidimensional lists. Nesting lists may slow your program down. You can use a trick known as array flattening instead. Additionally, if you are coding in Python, consider submitting your solution in PyPy. It behaves in the same way as Python, but is often faster (the time limits for PyPy are the same as Python). Putting your code in a `main` method can speed up run time as well. Finally, **do not** use `exit()`.

Ask for clarifications! If you are confused about a problem statement, do not hesitate to message us.

§1 Zoo Tour

An estimated 600 million people visit zoos each year.

Lev is a new tour guide at the zoo, and he busy showing Dr. Roman around. The zoo habitats are arranged in a loop and are labeled from 1 to N. For any i from 1 to $N-1$, the there is a path of length A_i connecting habitats i and $i+1$. There is also a path of length A_N connecting habitats 1 and N. Since Dr. Roman is a veterinarian who may need to get from one habitat to another at a moment's notice, he is interested in the shortest distance between certain pairs of habitats. Dr. Roman will ask Lev Q questions, where each question consists of two habitat numbers. For each question, help Lev out by determining the shortest distance between the two habitats. All paths can be traversed in both directions. Dr. Roman must travel along the paths.

Input Format:

The first line contains N ($1 \leq N \leq 10^5$) and Q ($1 \leq Q \leq 10^5$).

The second line contains N integers: $A_1, A_2, \ldots, A_N$ ($1 \leq A_i \leq 10^9$).

The next Q lines each contain a pair of integers, u and v ($u \neq v$; $1 \leq u, v \leq N$).

Output Format:

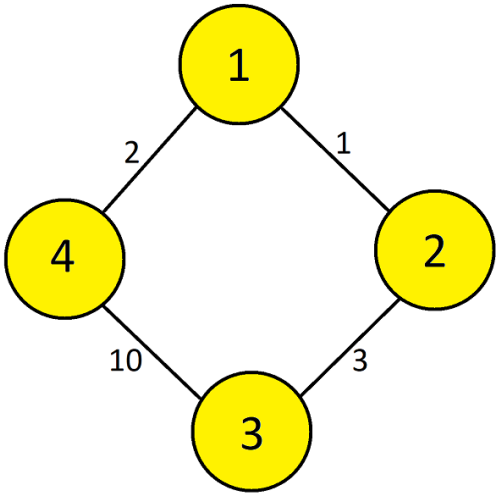
Print the shortest distance between each pair of habitats.

Sample Input:

```
4 3
1 3 10 2
1 3
4 2
3 4
```

Sample Output:

```
4
3
6
```



§2 Leaping Lizards

Some lizard species can detach their tails if caught by predators.

There is an obstacle course with N poles which have distinct heights, h_i. The poles are evenly spaced in a line and are labelled $1, 2, \ldots N$ from left to right. A researcher is analyzing the agility of lizards and wants to see how fast they can jump from the first pole to the last. Current scientific theory states that if the lizard is at pole i, it may jump to pole j if and only if $j > i$ and there does *not* exist pole k between i and j where the slope from i to k is **strictly** greater than the slope from i to j. Determine the least number of poles the lizard must stand on to get to pole N if it starts at pole 1. Include poles 1 and N in this total. Note that it is always possible for the lizard to reach pole N.

Input Format:

The first line contains the number of poles N ($2 \leq N \leq 1000$).

The next line contains the heights of the poles h_i ($1 \leq h_i \leq N$).

Output Format:

One line with the least possible number of visited poles.

Sample Input:

```
9
4 5 1 8 7 2 3 9 6
```

Sample Output:

```
4
```

§3 Raging Rhinos

Fully grown rhinos can weigh over 5000 pounds!

The rhinos are in a fit of rage, and they are ready to duke it out in an all-out frenzy. The rules are as follows: the rhinos all stand in a line facing either to the left or right. All rhinos begin with a positive amount of stamina. At the sound of the bell, all rhinos will begin charging in their respective directions at the same speed, and upon encountering another rhino, they will collide. The rhino with less stamina is knocked out of the line while the rhino with more stamina continues in the same direction, *with its stamina reduced by the stamina of its opponent.* If two rhinos with the same stamina collide, both drop out of the line. Can you determine the state of the rhinos after all collisions? Note that a rhino never changes directions.

Input Format:

The first line contains N, the number of rhinos ($1 \leq N \leq 10^5$).

Among the next N lines, the ith line describes the ith rino from the left. Each line contains two integers s_i and d_i denoting the initial stamina and direction of each rhino ($1 \leq s_i \leq 10^9$). If $d_i = 0$, then the rhino is facing left, otherwise if $d_i = 1$, it is facing right.

Output Format:

Print one line containing M, the number of rhinos still running in on the line.

On the next M lines, print two space-separated integers containing the remaining stamina and direction of each rhino. Describe the rhinos in the order they appear on the line (from left to right).

Sample Input:

```
4
39 1
20 0
21 0
12 0
```

Sample Output:

```
2
2 0
12 0
```

The first and second rhino collide, and the first rhino is victorious with 19 remaining stamina. Then the first and third rhino collide, and the third rhino is victorious with 2 remaining stamina. The third and fourth rhino never touch because they are moving in the same direction.

§4 Raccoon Mischief

The name "raccoon" comes from the Powhatan word "aroughcun" meaning "animal that scratches with its hands."

Raccoons love candy! There are N raccoons numbered from 1 to N, where the ith raccoon starts with A_i pieces of candy. However, the evil zookeeper Alice will adjust their candy amounts Q times. For each adjustment, she will pick positive integers l, r, and x. For each raccoon i in that range $l, l+1, \ldots, r$, she will take away all the candy it has if it has a nonzero amount of candy, or give it x pieces of candy if it has no candy. How many pieces of candy will each raccoon have after all of Alice's antics?

Input Format:

The first line will contain N and Q ($1 \leq N, Q \leq 10^5$).

The next line will contain N integers A_i ($0 \leq A_i \leq 10^9$).

The next Q lines will contain three integers for each adjustment: l, r, and x ($1 \leq l \leq r \leq N$; $1 \leq x \leq 10^9$).

Output Format:

Output one line containing N integers: the amount of candy each raccoon has at the end of all of Alice's adjustments.

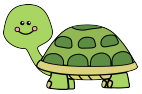
Sample Input:

```
5 2
1 0 4 5 2
1 2 3
1 3 4
```

Sample Output:

```
4 0 0 5 2
```

§5 Turtle Tribulation



Turtles belong to one of the oldest reptile groups in the world!

Turtle-hopping is all the rage these days. There are N turtles on a rectangular lake of width Z which lies on the xy coordinate plane. Each turtle is at a **distinct** y_i value for their y-coordinate and a **not necessarily distinct** x_i value for their x-coordinate. All coordinates are integers between 1 and Z, inclusive. You plan to start on the shell of the turtle with the lowest y-coordinate, then proceed in a series of $N-1$ jumps from shell to shell until you reach the turtle with the highest y coordinate. Your path will visit every turtle in order of increasing y-coordinate. Each jump can be represented as a line segment connecting the starting and ending turtles. To ensure that your route is efficient, you have decided that each one of these line segments must have an absolute slope of at least K (or be vertical).

To satisfy this condition, you can bribe the turtles to move around before you make your jumps. If you give a turtle a carrot, you can convince it to move left or right by one unit. This corresponds to changing an x_i value by 1. Turtles may be bribed multiple times, but they may not leave the lake so x_i must always remain between 1 and Z, inclusive. However, the turtle with the lowest y-coordinate and the turtle with the highest y-coordinate are not hungry, so they cannot be moved. Since carrots are getting expensive these days, you want to know the minimum number of carrots you must use to arrange the turtles in a way that lets you efficiently jump across their shells.

It is guaranteed that in our test cases it will always possible to move the turtles to satisfy the condition.

Input Format:

The first line contains the number of turtles N, Z, and K ($1 \leq N \leq Z \leq 2500$ and $0 < \frac{1}{K} \leq \frac{Z}{3}$). K is a floating point number.

The next N lines contain the x_i and y_i values of the turtle positions ($1 \leq x_i, y_i \leq Z$).

Output Format:

One line with the minimum number of operations necessary.

Sample Input:

```
5 7 0.553537
2 1
6 5
2 4
2 3
6 6
```

Sample Output:

```
4
```

In this example, increase the x_i once for the turtle at $y_i = 3$, increase the x_i twice for the turtle at $y_i = 4$, and decrease the x_i once for the turtle at $y_i = 5$.

§6 Gorilla Grouping

Gorillas are actually generally calm and passive animals. However, the Silverback will become aggressive when its troop is threatened.

Jane is studying the social interactions of a band of gorillas as they meet during the zoo's play time. She has figured out that in gorilla society, each gorilla is assigned a unique positive integer ID which acts as its name. It seems that any pair of gorillas is willing to work together during play time as long as the absolute difference between their IDs is not exactly K. However, when two gorillas have IDs that differ by K, they refuse to collaborate on anything and often end up exhibiting aggressive behavior towards each other.

Jane wants to choose a non-empty subset of these gorillas to live together in the zoo's new experimental primate resort. However, she wants to make sure that all of the gorillas that she chooses are compatible with one another so that no fights break out. How many ways are there for her to choose such a subset? Output your answer modulo $10^9 + 7$.

Input Format:

The first line contains N and K ($1 \leq N \leq 10^5$, $1 \leq K \leq 10^9$).

The next line contains N integers $A_1, A_2, \ldots A_N$ representing the IDs of the gorillas ($1 \leq A_i \leq 10^9$).

Output Format:

Output the number of ways to choose a non-empty subset of gorillas.

Sample Input:

```
7 3
3 4 6 7 9 12 14
```

Sample Output:

```
47
```

§7 Hen Hackers

If chickens listen to classical music, they can lay bigger and heavier eggs.

This problem involves interaction between your program and the grader.

In an attempt to "liberate" their fellow birds, a group of highly intelligent hens are attempting to compromise the zoo's security system. To get access to the zoo's mainframe, the hens first need to guess the administrative password. The hens know that the password is a string built from up to 62 possible characters: a, b, c, ..., z, A, B, C, ..., Z, 0, 1, ... 9. They also know that **no character appears more than once** in the password, which means that it has a maximum length of 62 (and a minimum length of 1). The password is case-sensitive.

We define a guess be *close* if and only if the real password can be obtained by inserting additional characters among the letters and numbers of the original guess. In other words, a guess is close if it is a (not necessarily contiguous) subsequence of the password. Every time the hens guess a password, the zoo's security system will do one of three things:

- If the guess is exactly equal to the password, it will print 'C' and grant the hens access to the mainframe.
- If the guess *close* but not exactly equal to the password, it will print 'Y'.
- If the guess is neither *close* nor exactly equal to the password, it will print 'N'.

The mainframe permits at most 750 guesses before it locks further attempts and alerts the zoo officials of suspicious activity. The hens have hired you to guess the password before you use all of your allotted attempts. Can you help them?

I/O Format:

There will be no initial input. Every time your program makes a guess, the grader will output 'C', 'Y', or 'N', which your program can then read via the usual input channel. The grader will only respond to the first 750 guesses. **Make sure that you end each guess with a new line.**

Sample Interaction:

```
CorectHO7sEBaT3RyS1Apl
                              N
unt2
                              Y
HuNtEr2
                              N
run2the
                              N
hunter2
                              C
```

In this example, the program guessed the password correctly with 5 attempts.

Note: You are responsible for making sure your program terminates after receiving the 'C' response. If you make further queries after guessing the password you may receive a time limit error. Also, make sure that you flush the output stream every time your program makes a guess. For example, in C++ you may use `fflush(stdout)` or `cout << flush`. In Java you can use `System.out.flush()`. In Python you can use `stdout.flush()` after you import `stdout` from the `sys` module. You can learn more about interactive problems here.

§8 Platypus Puddles

Platypi are one of the few species of mammals that lays eggs.

After a night of pouring rain, the platypus exhibit has been flooded. Now there are puddles all over the place. The platypus habitat is an uneven terrain with a square base of $N \times N$ (in meters) when viewed from above. This base can be divided into a grid of N^2 separate 1×1 areas, and each area has a given elevation in meters. This means that the terrain can be represented by a $N \times N$ grid of integers, where each value represents the height of the ground at that location. A drainage system surrounds the four edges of the habitat.

In this problem, water flows throughout the terrain the same way that you would expect in the real world. Whenever the water level is higher in one area than in an adjacent area, the water will flow from the first area to the second area until an equilibrium is reached or the first area runs out of water. Note that tiles are only adjacent in the four cardinal directions (not diagonally). Water on the border of the habitat flows out of the exhibit via the drainage system.

After the rain has stopped and the water levels across the terrain have reached equilibrium, there still may be puddles remaining. The zookeepers want you to determine the total volume of water left in these puddles (in cubic meters). You may assume that a surplus of rain fell during the night, so all puddles hold as much water as they can.

Note: Solutions in Python may not be fast enough to pass the largest test cases on this problem. Consider using Java or C++.

Input Format:

The first line contains a single integer N ($1 \leq N \leq 1000$).

The next N lines will contain N integers each, representing the topology of the terrain. All of these values will be between 0 and 10^6, inclusive.

Output Format:

Output the total volume of the puddles, in cubic meters.

Sample Input:

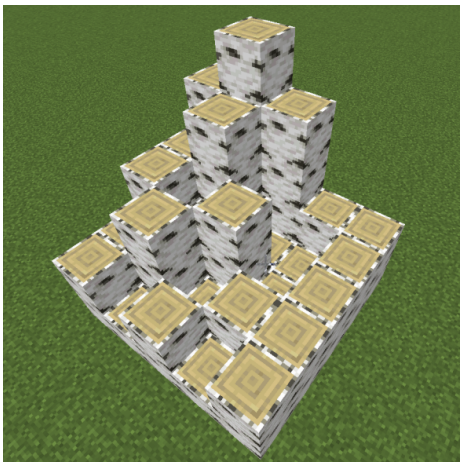
```
5
4 5 4 2 2
3 4 1 1 2
3 2 3 1 2
2 3 1 2 2
2 1 2 1 2
```

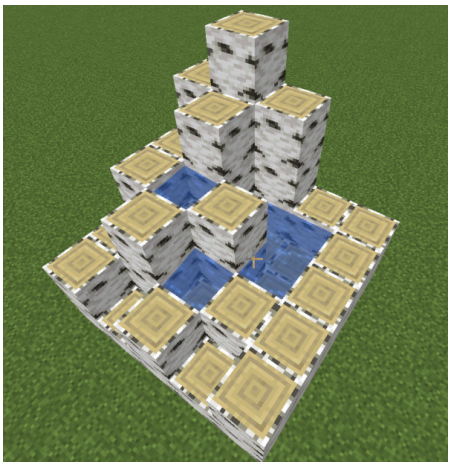
Sample Output:

```
5
```

After the water levels reach equilibrium, the following puddles remain: a 1 by 1 puddle in the third row, second column; a 1 by 1 puddle in the fourth row, third column; and an L-shaped puddle in the second and third rows. All of these puddles have a depth of 1 meter (although in general, other habitats could allow for deeper puddles). Thus, the answer is 5 cubic meters.

Here are Minecraft representations of the habitat before and after the rain, respectively:





§9 Playlist Shuffle

An adult panda eats 12-38 kilograms of bamboo per day!

Rio the red panda is listening to her playlist of music that contains N songs, numbered 1 through N. To navigate through the songs on the playlist, Rio has three buttons labelled *previous*, *next*, and *random*. If she is currently on song i, she can move to song $i-1$ or $i+1$ by pressing the *previous* or *next* buttons, respectively. Note that the playlist is not circular, so she cannot press *previous* when she is currently at song 1, and she cannot press *next* when she is at song N. When Rio presses the *random* button, her music player moves to a song chosen uniformly at random from all N songs. Unfortunately, Rio doesn't have a premium Pandara account so every time she presses *previous* or *next*, she is forced to listen to an X second long advertisement. Every time she presses *random*, Rio must listen to a Y second long ad.

Rio's favorite song, *What Does the Panda Say?*, is located at position B in the playlist. She wants to use these three buttons to navigate to this song while minimizing total length of ads she has to listen to. If Rio has just pressed the *random* button and watched the accompanying ad, what is the expected value of ad length she must now listen to to get to song B? **Do not count the Y second ad she just watched**. Assume Rio acts optimally to minimize expected ad time. Your output will be judged to be correct if its relative error with the correct answer is less than 10^{-6}.

Input Format:

The first line contains four integers N, B, X, Y ($1 \leq N \leq 10^9; 1 \leq B \leq N; 1 \leq X, Y \leq 100$).

Output Format:

Output a single number containing the expected value.

Sample Input:

```
7 4 1 1
```

Sample Output:

```
1.6
```

§10 Penguin Mayhem

Some penguins can swim up to 22 miles per hour!

Ringo has made a magical box of size H by W units, where the top left corner is $(0, 0)$ and the bottom right corner is at (H, W). However, this box is magical so objects will "wrap around" if the hit the edges of the box (this will be formally explained later). There are N penguins in this box numbered $1, \ldots N$, where the ith penguin is located at (X_i, Y_i) at time $t = 0$. Penguin i has a starting velocity of $< P_i, Q_i >$. Formally, this means that the position of penguin i at time t will be $(X_i + P_i \cdot t, Y_i + Q_i \cdot t)$. The wrap around effect means that the first coordinate will be reduced $\pmod W$ and the second coordinate will be reduced $\pmod H$. To reduce a real number $x \pmod M$ for some positive integer M means we add or subtract integer multiples of M to x until x is in the range $[0, M)$. For example, -10.6 and 9.4 both reduce to $4.4 \pmod 5$.

Each penguin has the same mass. In other words, when two or more penguins collide, you can pretend they just go through each other (this is a result of elastic collisions in physics, but you don't need to understand why). Given an integer T, find how many collisions occur from time $t = 0$ to $t = T$, inclusive. A collision is when two penguins occupy the exact same location (x, y) in the box. It is guaranteed that no two penguins will have both the same start position and velocity.

Input Format:

The first line contains one integer N $(1 \leq N \leq 1000)$.

The next line contains two integers H, W $(1 \leq H, W \leq 5000)$.

The next N line contains four integers X_i, Y_i, P_i, Q_i
$(0 \leq X_i < H; 0 \leq Y_i < W; -5000 \leq P_i, Q_i \leq 5000)$.

The final line contains one integer T $(1 \leq T \leq 10^9)$.

Output Format:

Output one integer with the number of collisions that occur before (or at) time T.

Sample Input:

```
2
4 3
1 0 -4 1
1 0 3 0
99
```

Sample Output:

```
9
```

§11 Sharing Seals

A seal's whiskers allow it to detect prey in dark murky waters.

Seal trainer Selkie is training N seals that are sitting in a circle, indexed $0, \ldots, N-1$, sitting in that order. Each seal starts out with A_i pieces of fish. Selkie wants to give the seals fish in Q updates, where each update consists of two positive integers K, T. A single round of this update gives the seal with number i an additional A_{i-K} fish, where the indices are defined modulo N. Formally, each element of the array is set to $A_i = A_i + A_{i-K}$ all at once. This is then repeated T times (note that between each of those T times, we use the recalculated A_i).

After all Q updates are processed, Selkie wants to know how many fish each seal has at the end. Formally, your job is output the final state of the array A_i where each element is taken $\pmod{10^9+7}$.

Input Format:

The first line contains one integer N ($1 \leq N \leq 100$).

The second line contains N integers representing $A_0, \ldots, A_{N-1}$ ($1 \leq A_i \leq 10^9$).

The third line contains one integer Q ($1 \leq Q \leq 1000$).

The next Q lines contain K and T for each query ($1 \leq K \leq N-1$; $1 \leq T \leq 10^5$).

Output Format:

Output the final states of $A_0, \ldots, A_{N-1}$ taken modulo 10^9+7 in one line separated by spaces.

Sample Input:

```
3
1 2 3
2
1 2
2 1
```

Sample Output:

```
16 15 17
```

The state of the A_i after the first update is $9, 7, 8$. The final state after the second update is then $16, 15, 17$.

§12 Zookeepers' Gathering

Havish, Patrick, Danny, and Omkar were the winners of the 2019 November mBIT. We are featuring them in this problem.

Havish, Patrick, Danny, and Omkar are four good friends and fellow zookeepers working at the mBIT zoo. The 10th anniversary of the zoo's opening is approaching soon, so the four zookeepers want to arrange a party to celebrate. However, each of them has work shifts that span certain time intervals. They want to find the longest contiguous interval of time to schedule the party in, such that all four zookeepers are not working during that time.

This would be a trivial task if not for the fickle nature of the zookeepers' schedules. Because the mBIT zoo is always experiencing a wide variety of unforeseen incidents, the four friends' schedules are always changing. A schedule update consists of removing an existing work shift or adding in a new work shift for one of the four zookeepers. After each update, can you find the length of the longest interval of time in which they can schedule their party?

Input Format:

Time will be represented as **discrete integer points** $0, 1, 2, \ldots, N$, and each person's schedule is represented as a set of intervals $[l, r]$ representing their job shifts. The range $[l, r]$ contains the times $l, l+1, \ldots, r$ and has a length of $r - l + 1$. **It is guaranteed that no single zookeeper will have overlapping job shifts.** For example, if Danny has shifts $[a, b]$ and $[c, d]$, then it is guaranteed that either $b < c$ or $d < a$.

For convenience of input, Havish, Patrick, Danny, and Omkar will be the 1st, 2nd, 3rd, and 4th zookeeper, respectively.

The first line contains N, the maximum possible time value ($1 \leq N \leq 10^5$).

Next, there will be four blocks of lines, with the ith block representing the initial schedule of the ith zookeeper ($1 \leq i \leq 4$). The first line of each block contains m_i, the number of initial work shifts the ith zookeeper has. The next m_i lines each contain l_j and r_j, the start and end time of the jth job shift ($1 \leq m_i \leq N + 1; 0 \leq l_j \leq r_j \leq N$).

The next line contains Q, the number of updates made to the schedule ($1 \leq Q \leq 10^5$).

The final Q lines each contain an update consisting of id, t, l, r ($1 \leq id \leq 4; 0 \leq t \leq 1; 0 \leq l \leq r \leq N$).

id denotes which zookeeper's schedule is being updated. t denotes whether an event is being added or removed. If $t = 0$, then add a job shift from time l to r. Otherwise if $t = 1$, remove the job shift that currently lasts from time l to r. It is guaranteed that all updates will be valid.

Output Format:

Print Q integers on separate lines, with the ith integer denoting the longest contiguous block of available party time after the ith update.

Sample Input:

```
10
2
0 1
9 10
1
0 3
1
0 2
1
4 4
4
1 1 9 10
1 0 9 10
2 0 7 9
1 0 2 3
```

Sample Output:

```
6
4
2
2
```

After the 1st update, the longest block of empty time is from time 5 to 10.
After the 2nd update, the longest is from 5 to 8.
After the 3rd update, the longest is from 5 to 6.
After the 4th update, the longest is from 5 to 6.