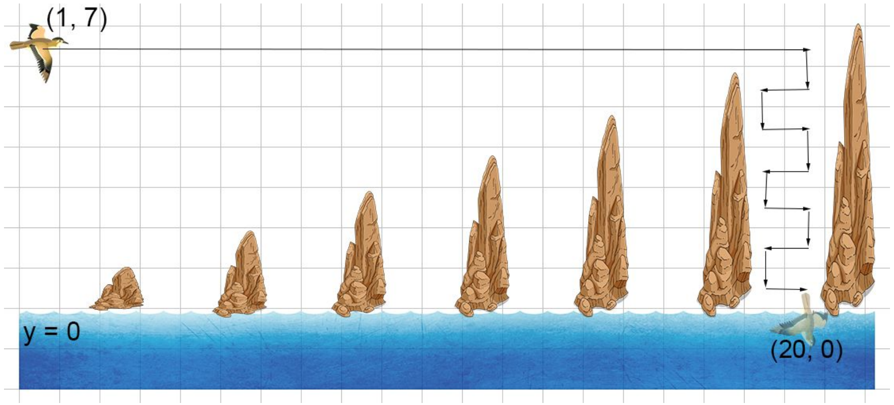
J184 - Mysterious Area

Percy Wong {percywtc}

The Problem



7 1 2 3 4 5 6 7	DROP 20 32

OUTPUT

DROP / ESCAPE
x- / y-coordinate
distance travelled

SUBTASKS

For all cases:
$1 \leq N \leq 2 \times 10^5$.
It is guaranteed that $H_{1...N}$ is a permutation of $1 ... N$.

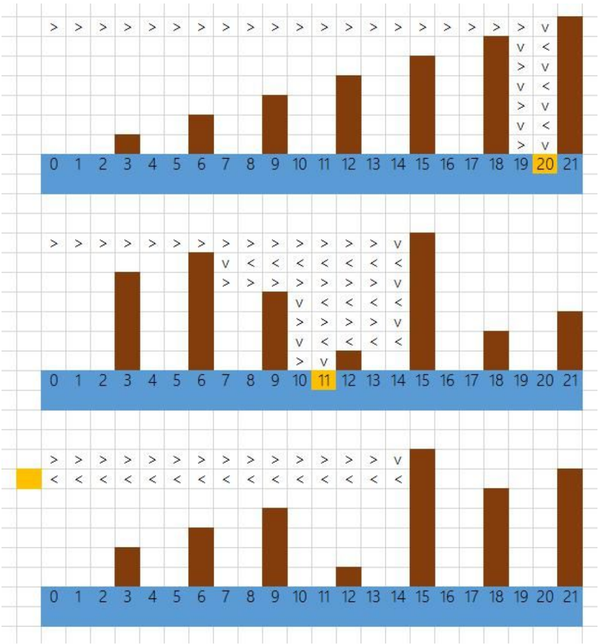
	Points	Constraints
1	10	$N = 2$
2	17	$1 \leq N \leq 50$
3	24	$1 \leq N \leq 2000$
4	19	It is guaranteed that the bird can always escape
5	30	No additional constraints

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Problem Idea By - percywtc

Testdata By - jeremy624; percywtc; microtony



Statistics

0 points	33 + 11 + 1 + 0 = 45
10 points	5 + 2 + 0 + 0 = 7
19 points	0 + 0 + 3 + 0 = 3
27 points	0 + 1 + 0 + 0 = 1
29 points	1 + 3 + 9 + 1 = 14
51 points	0 + 2 + 1 + 0 = 3
70 points	0 + 1 + 1 + 4 = 6
100 points	0 + 0 + 0 + 3 = 3

First solved by **hccheng1** at **1h 59m 18s**

SUBTASKS

For all cases:

$1 \leq N \leq 2 \times 10^5$.

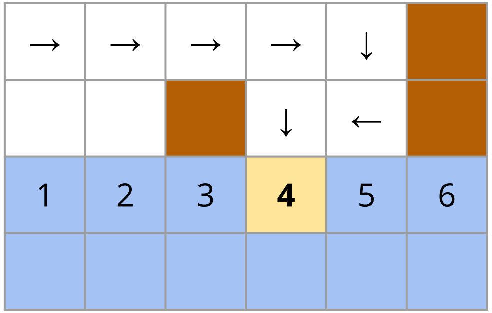
It is guaranteed that $H_{1...N}$ is a permutation of $1...N$.

	Points	Constraints
1	10	$N = 2$
2	17	$1 \leq N \leq 50$
3	24	$1 \leq N \leq 2000$
4	19	It is guaranteed that the bird can always escape
5	30	No additional constraints

Solution 1 - The First Subtask

10 points for just handling two cases: **H = {1, 2}** and **H = {2, 1}**

So we can simply “hardcode” them after solving them on our own

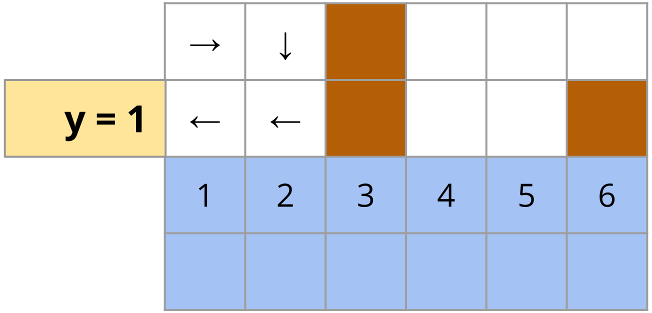


INPUT

2
1 2

OUTPUT

DROP
4
7



INPUT

2
2 1

OUTPUT

ESCAPE
1
4

Solution 1 - The First Subtask

This solution can only solve Subtask 1, nothing else :)

Subtask	Score	Max Score
1	10	10
2	0	17
3	0	24
4	0	19
5	0	30
Total	**10**	**100**

Solution 1 - The First Subtask

PSEUDOCODE

```
ReadLine(N)
ReadLine(a, b)
If (a = 1)
   ...
Else
   ...
```

Solution 2 - Escape

Considering only the cases that the bird can escape

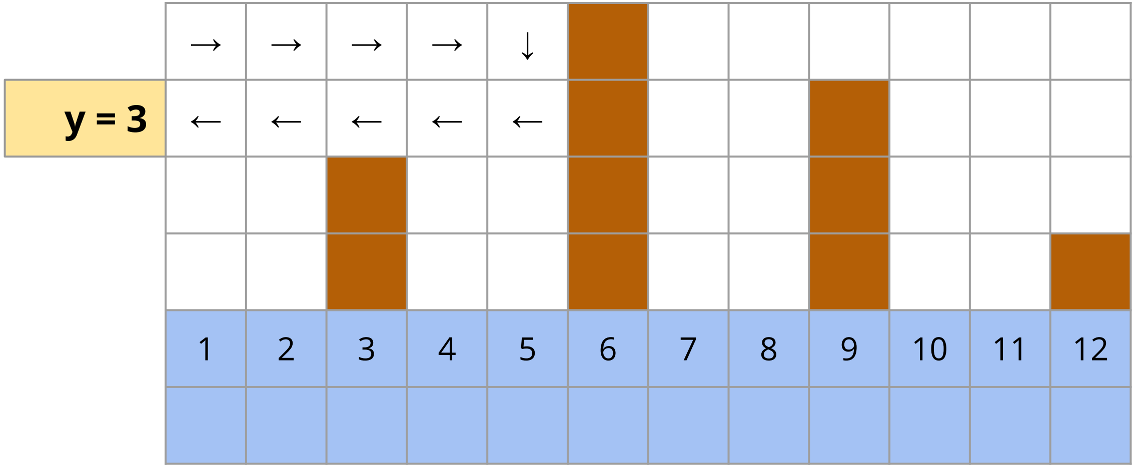
It only happens iff the pillar with **H = N - 1** is on the left of that with **H = N**
Thus, its y-coordinate is **N - 1**, and the distance travelled is **6 × Position(N) - 2**

INPUT

```
4
2 4 3 1
```

OUTPUT

```
ESCAPE
3
10
```



Solution 2 - Escape

This solution can only solve Subtask 4, nothing else :)

Subtask	Score	Max Score
1	0	10
2	0	17
3	0	24
4	19	19
5	0	30
Total	**19**	**100**

Solution 2 - Escape

PSEUDOCODE

```
Read(N)
For i = 1 .. N
  Read(x)
  Pos[x] = i
PrintLine('ESCAPE')
PrintLine(N - 1)
PrintLine(6 * Pos[N] - 2)
```

Solution 3 - Simulation

We can just simply store the entire grid with a 2-d array:
Marking `A[i][j]` as TRUE only if the cell **(i, j)** is occupied by the pillars

With fine implementation, the code should be able to work in $O(N^2)$,
Which can pass Subtask 1, 2 and 3 within time limit

Subtask	Score	Max Score
1	10	10
2	17	17
3	24	24
4	0	19
5	0	30
Total	**51**	**100**

Solution 3 - Simulation

PSEUDOCODE

```
For i = 1 .. N
  For j = 1 .. H[i]
    A[i * 3][j] = True
cur_x = 1
cur_y = N
dist = 0
dir = +1
```

```
While (cur_x > 0 AND cur_y > 0)
  dist++
  cur_x += dir
  If (A[cur_x][cur_y] = True)
    dir *= -1
    cur_x += dir
    cur_y--
If (cur_x = 0)
  PrintLine('ESCAPE')
Else
  PrintLine('DROP')
```

Solutions Summary

Solutions		1 - First Sub	2 - Escape	3 - Sim
Subtask	Max Score	Score		
1	10	10	0	10
2	17	0	0	17
3	24	0	0	24
4	19	0	19	0
5	30	0	0	0
Total	**100**	**10**	**19**	**51**

cumulative

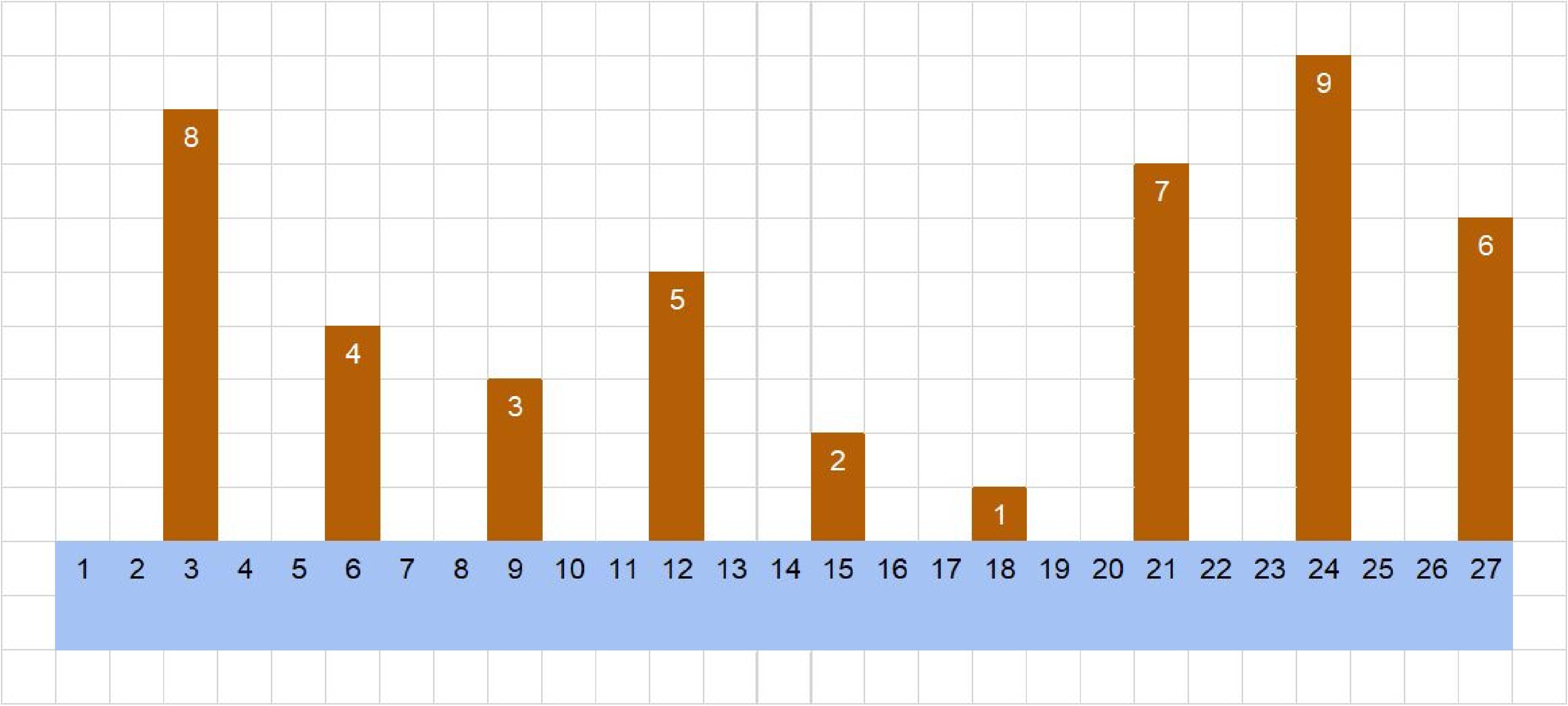
Score
10
17
24
19
0
70

Solution 4 - Optimized Solution

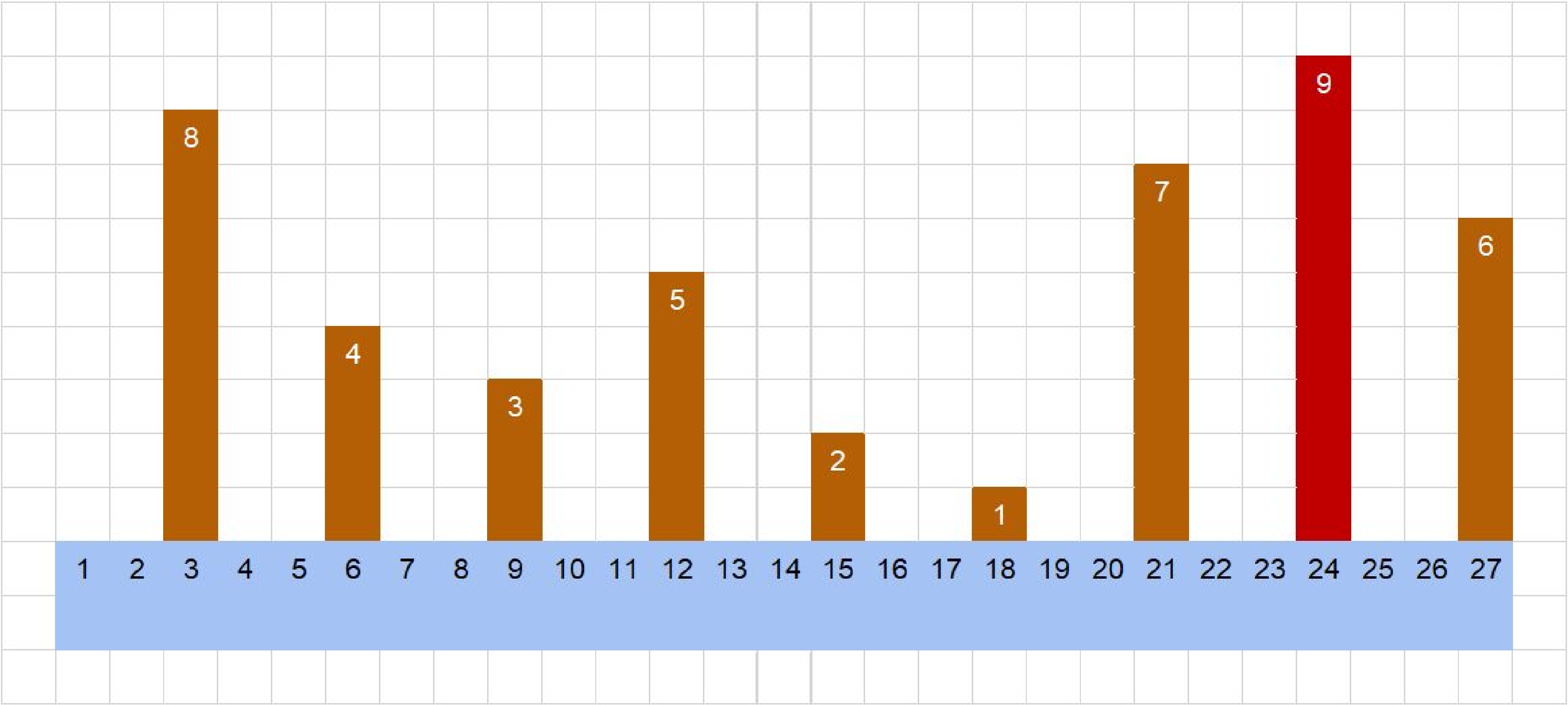
We can see that if the bird cannot escape,
The interval of its x-coordinate keep squeezing

We can maintain the left bound and the right bound of its x-coordinate,
By updating it every time its y-coordinate decreases by one

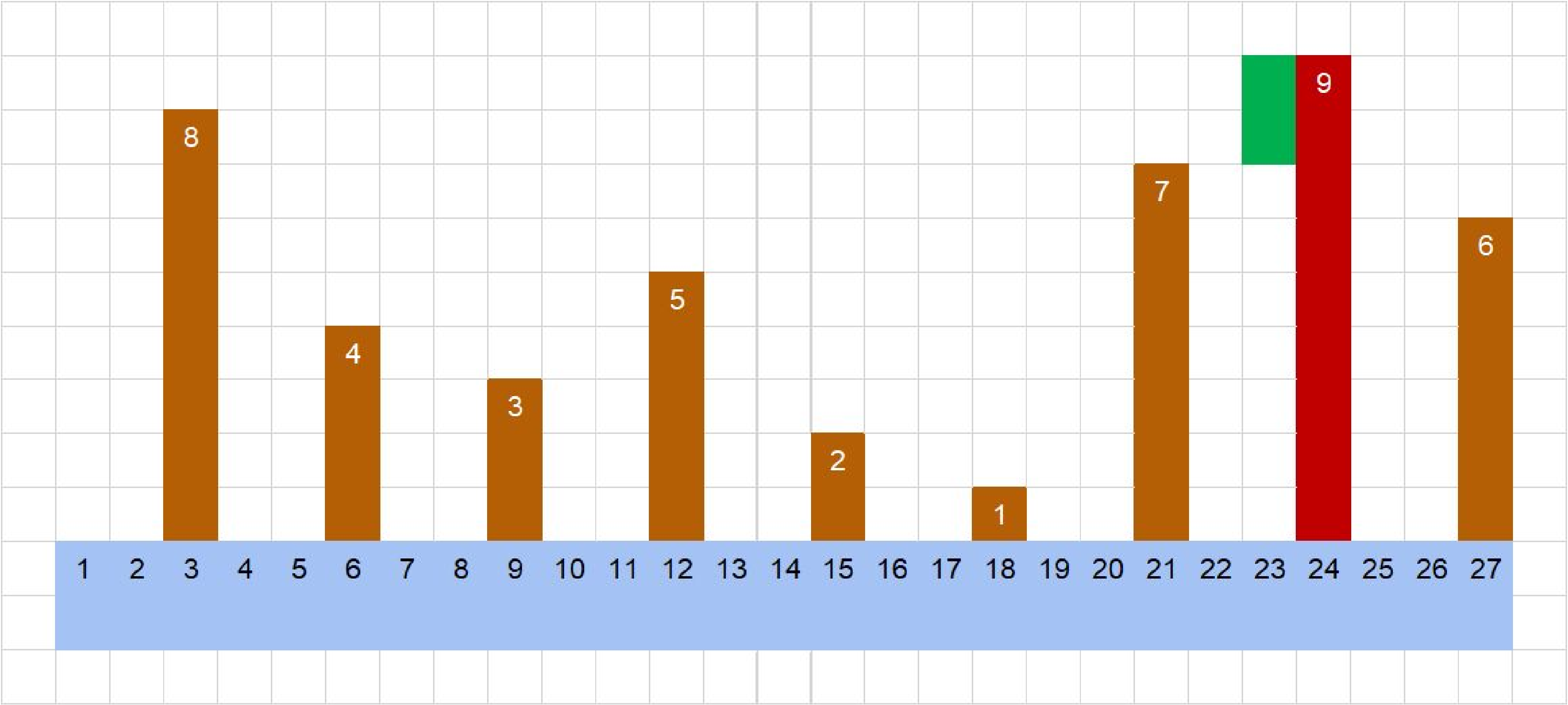
Initially at **y = 9**



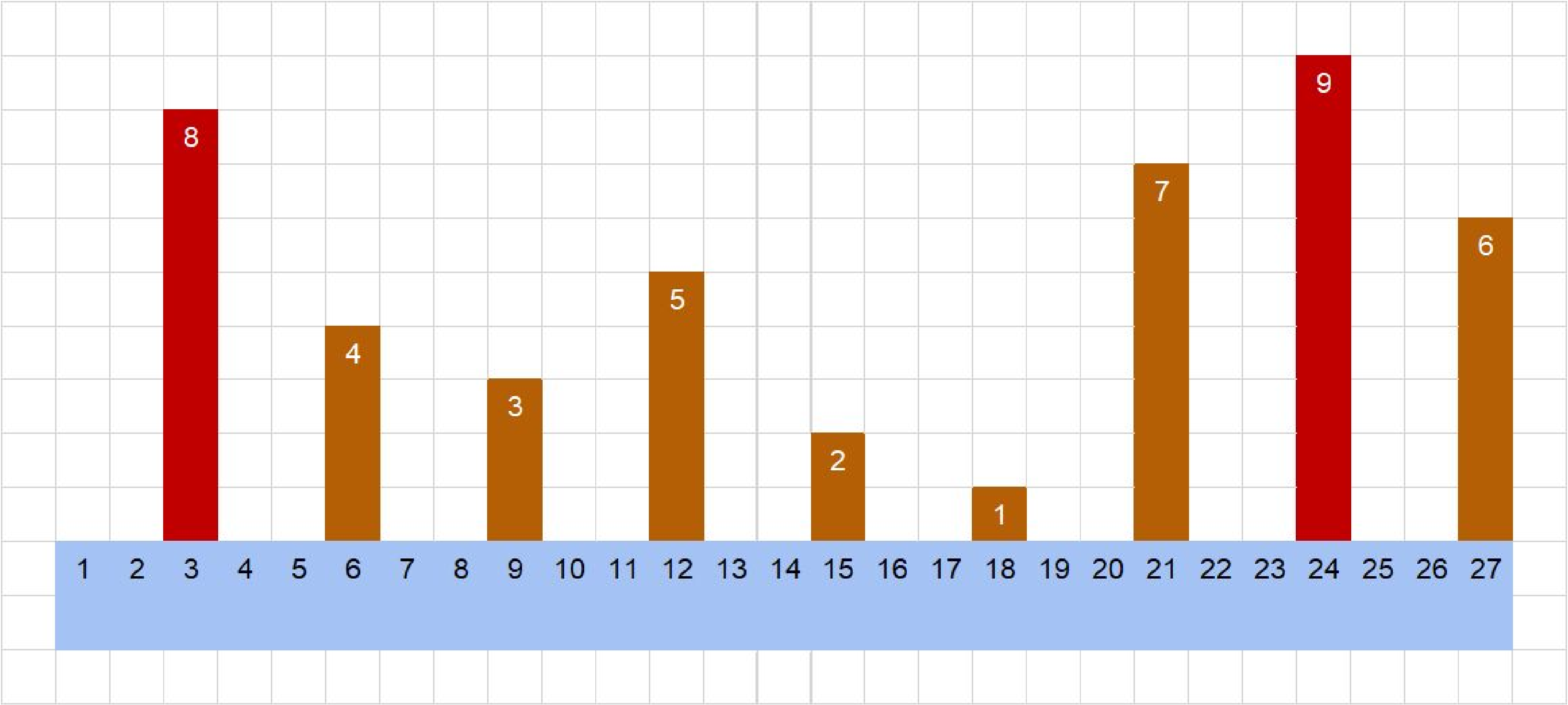
Initially at **y = 9**



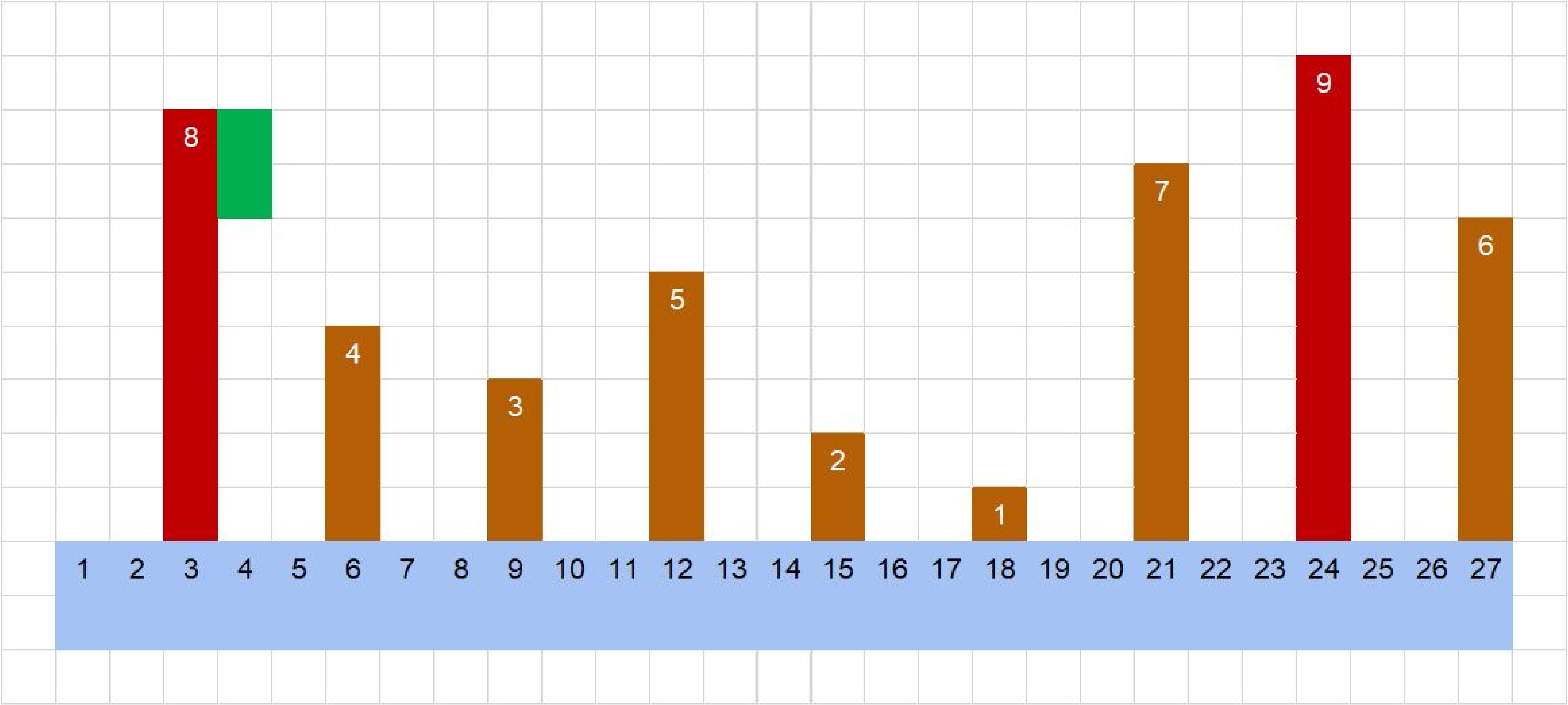
Changing from **y = 9** to **y = 8**



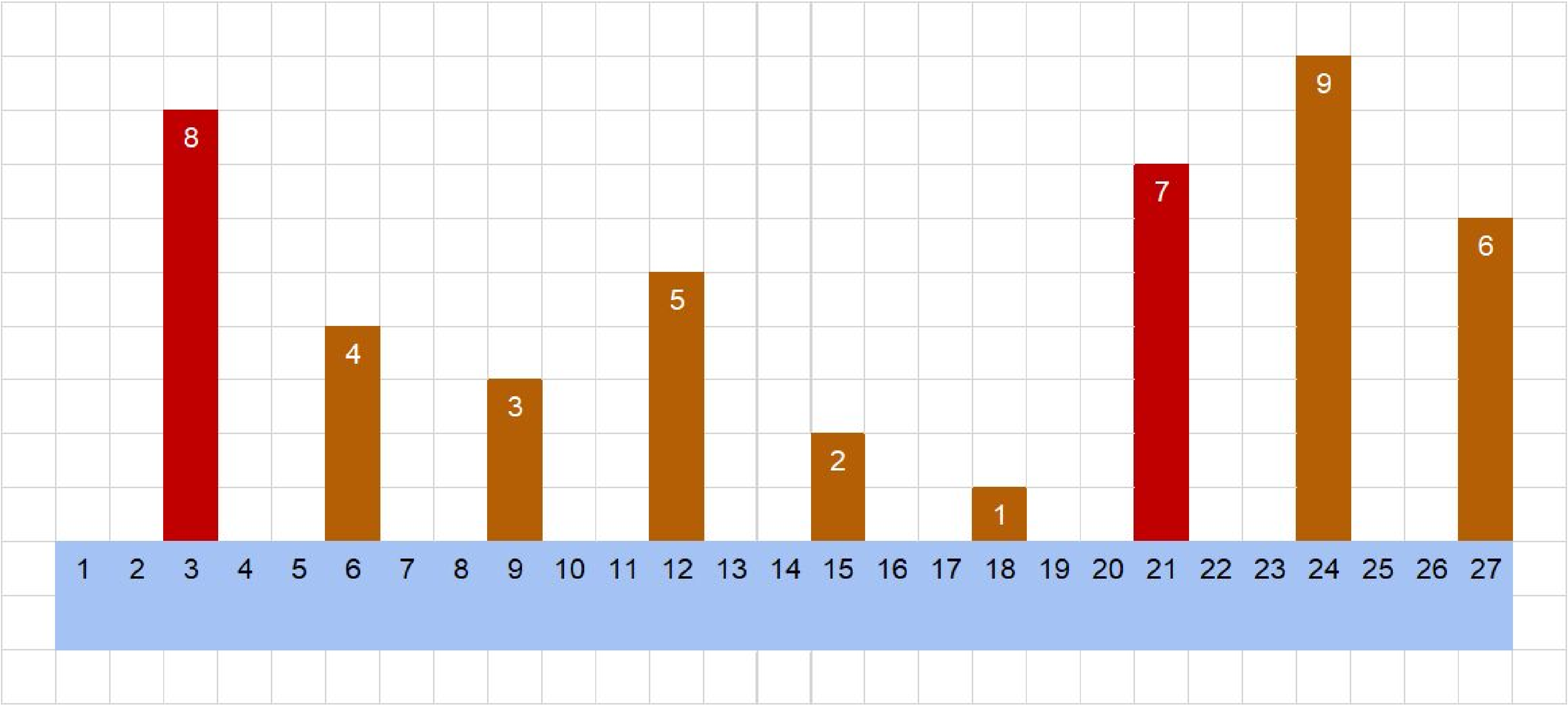
Changing from **y = 9** to **y = 8**



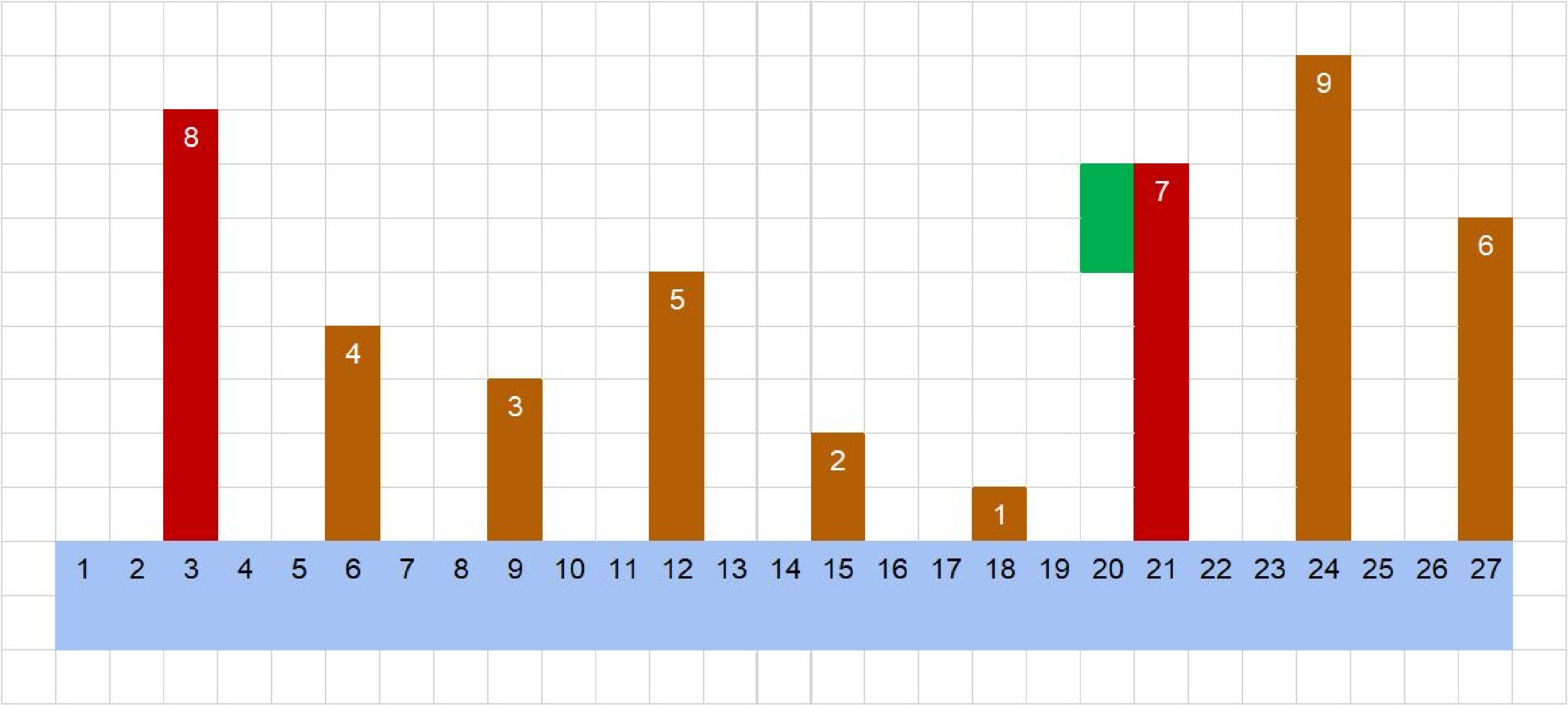
Changing from **y = 8** to **y = 7**



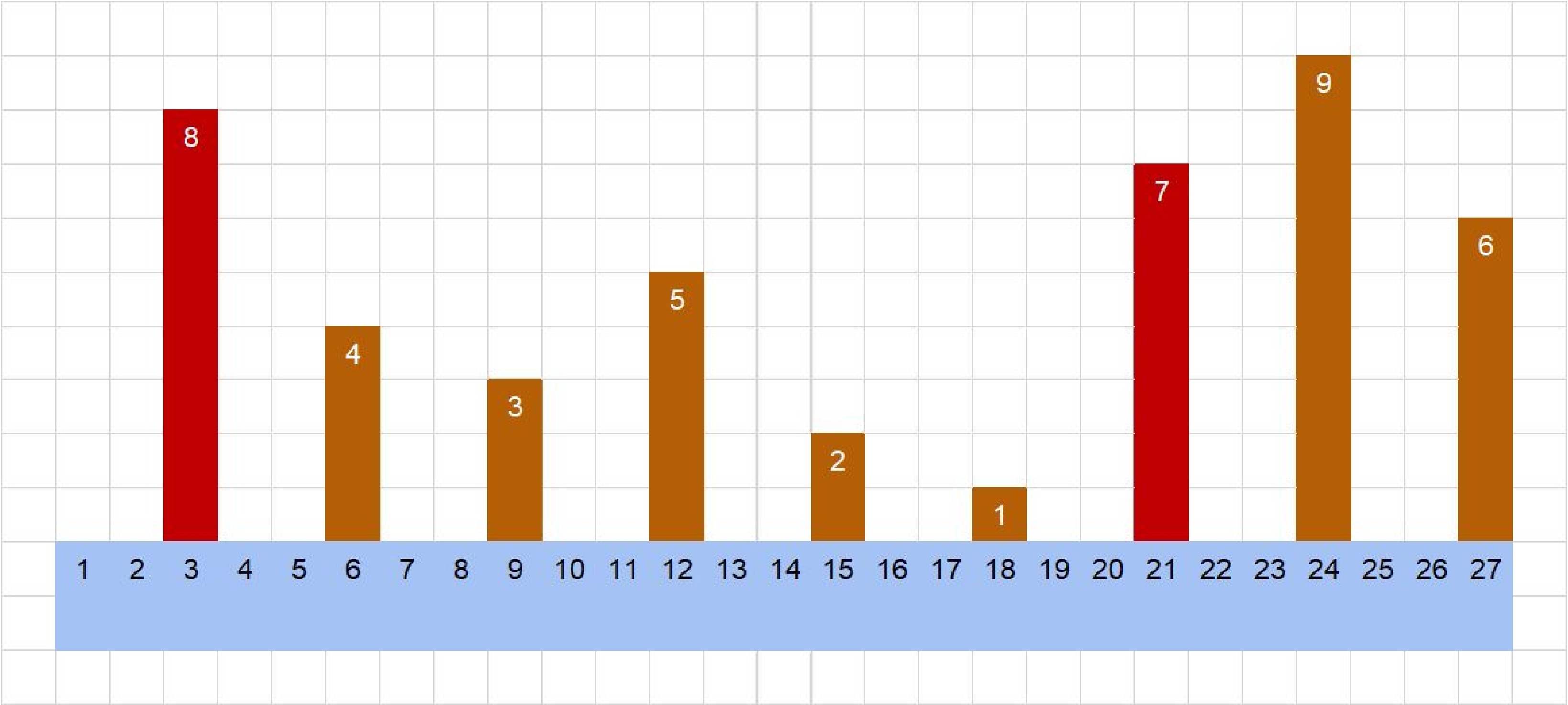
Changing from **y = 8** to **y = 7**



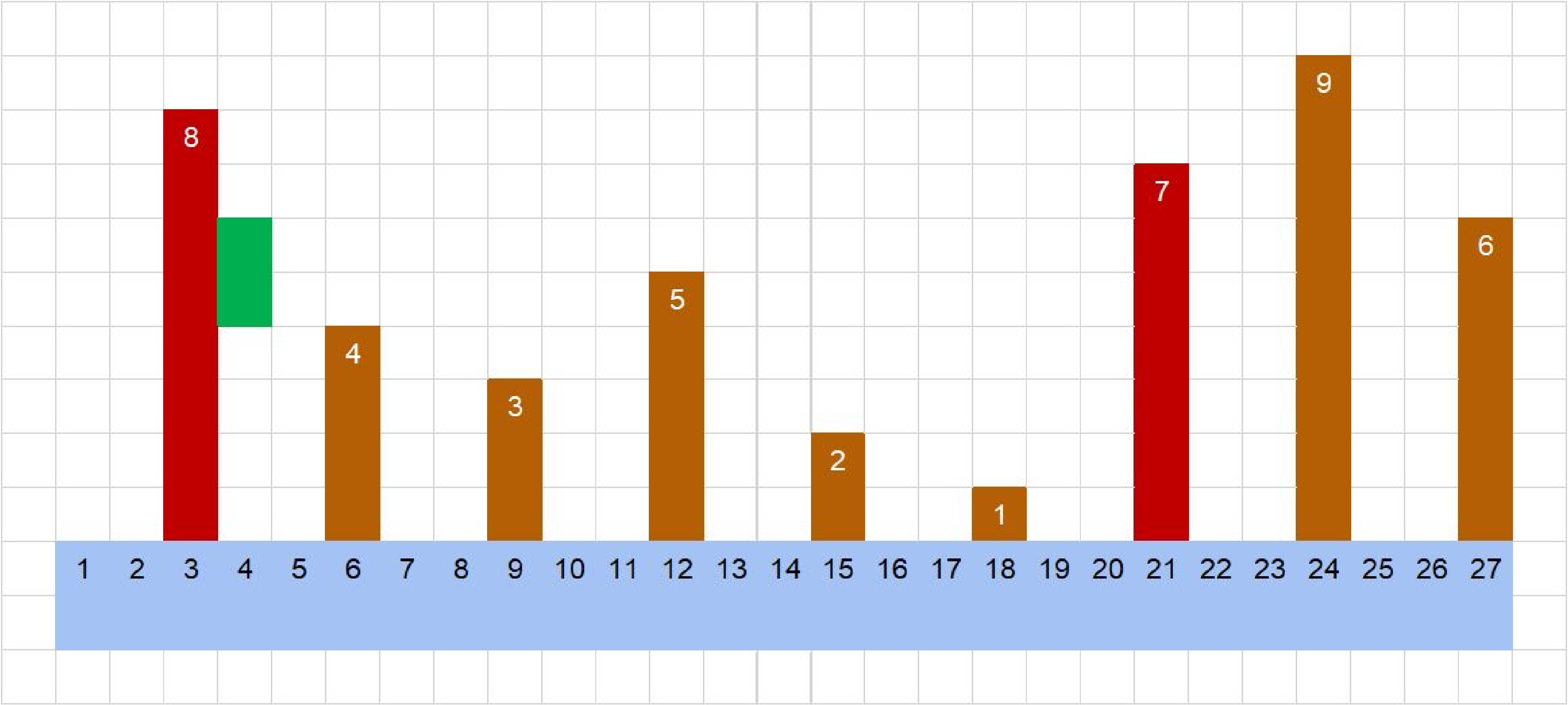
Changing from **y = 7** to **y = 6**



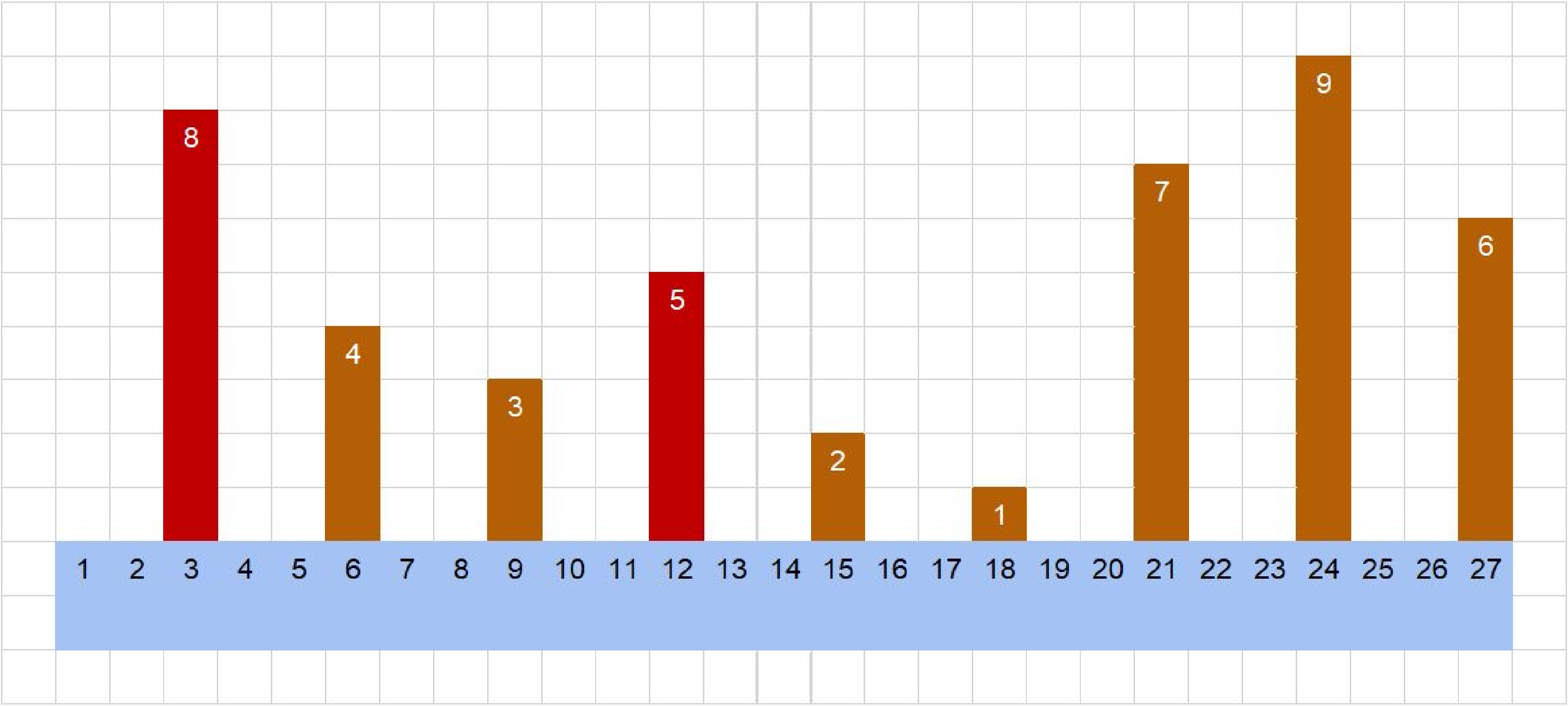
Changing from **y = 7** to **y = 6**



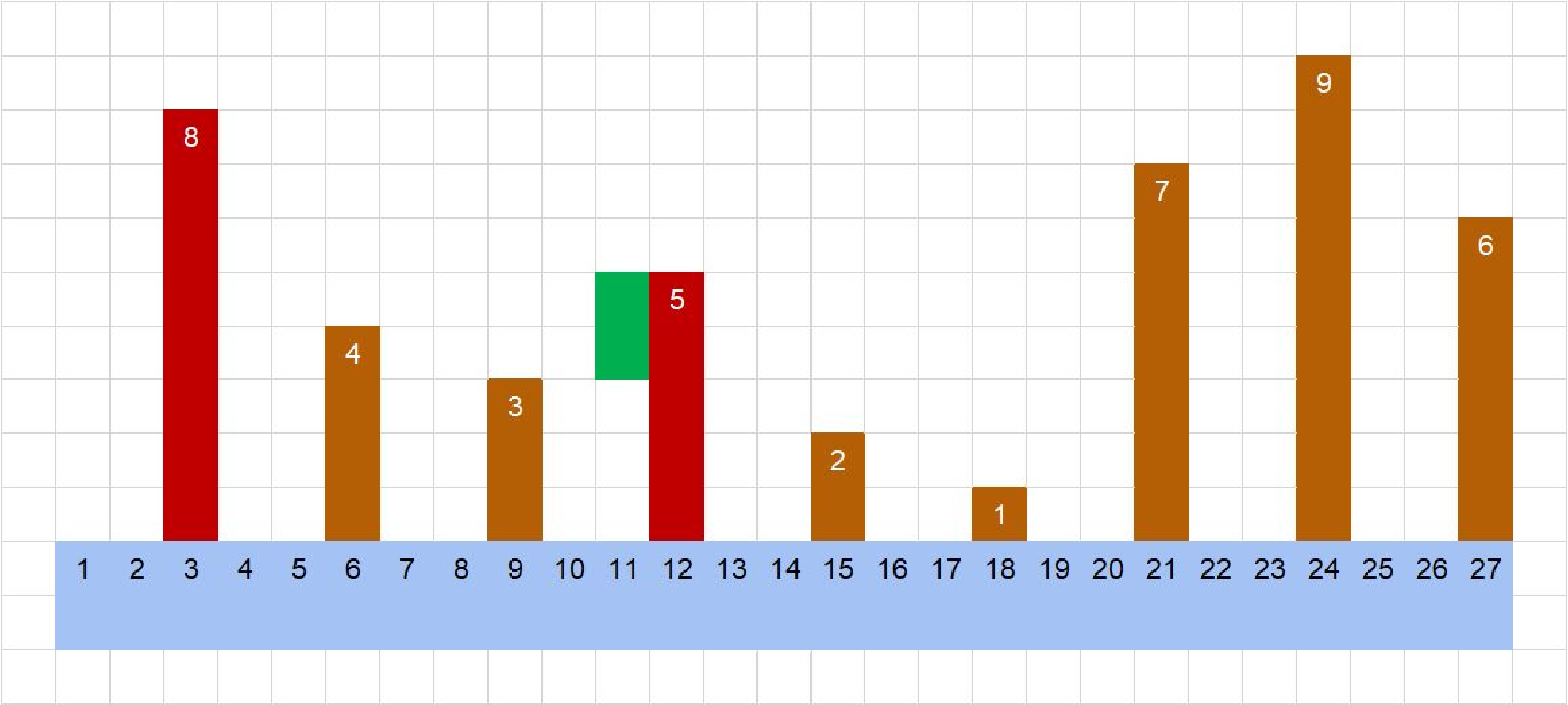
Changing from **y = 6** to **y = 5**



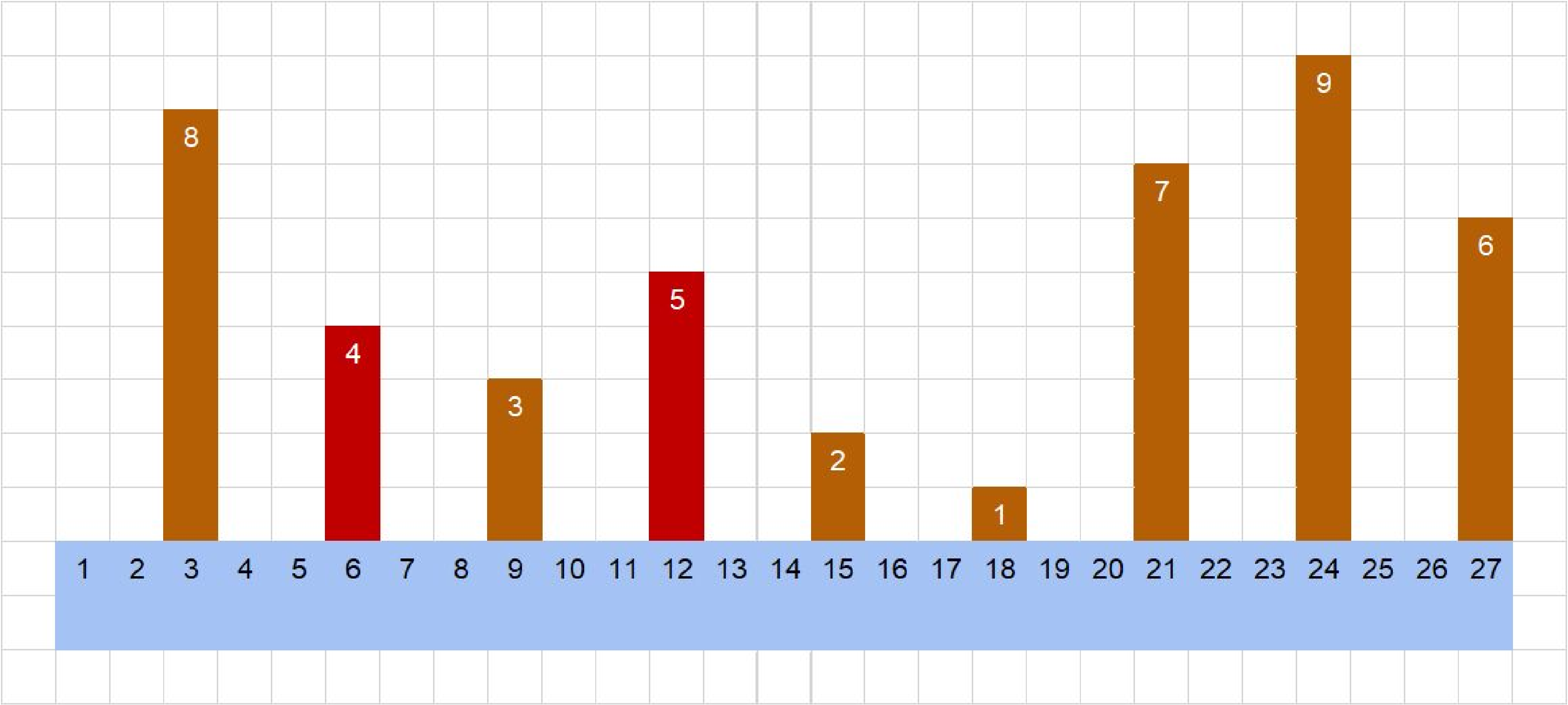
Changing from **y = 6** to **y = 5**



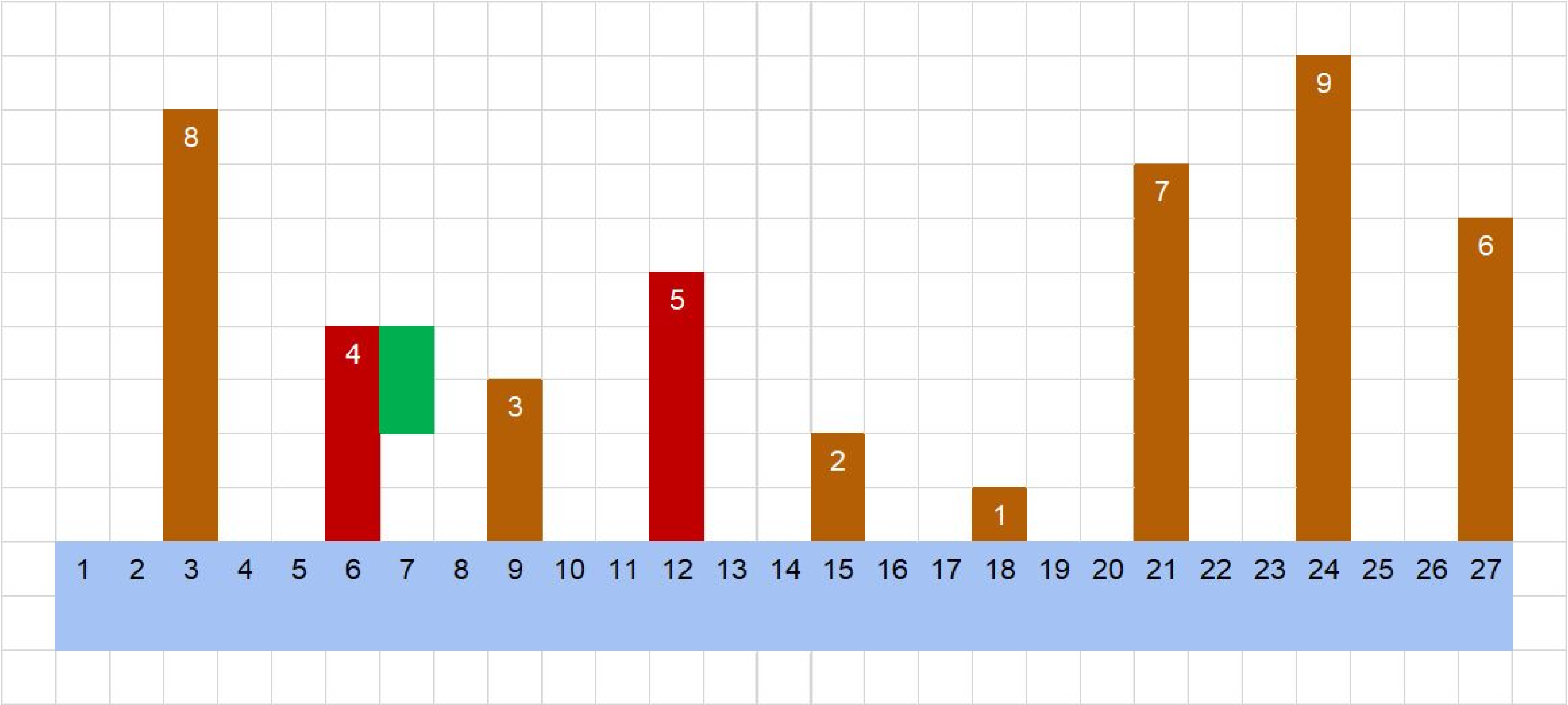
Changing from **y = 5** to **y = 4**



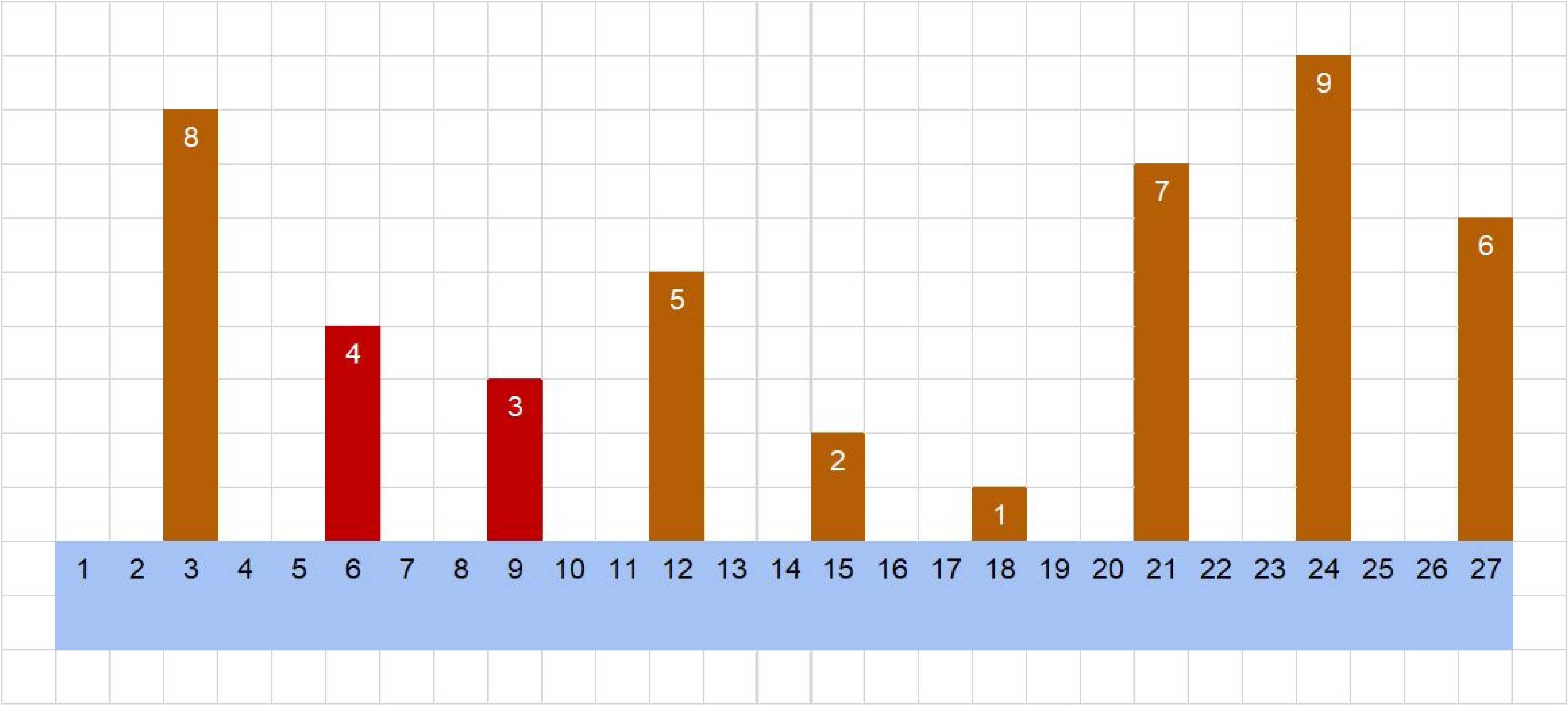
Changing from **y = 5** to **y = 4**



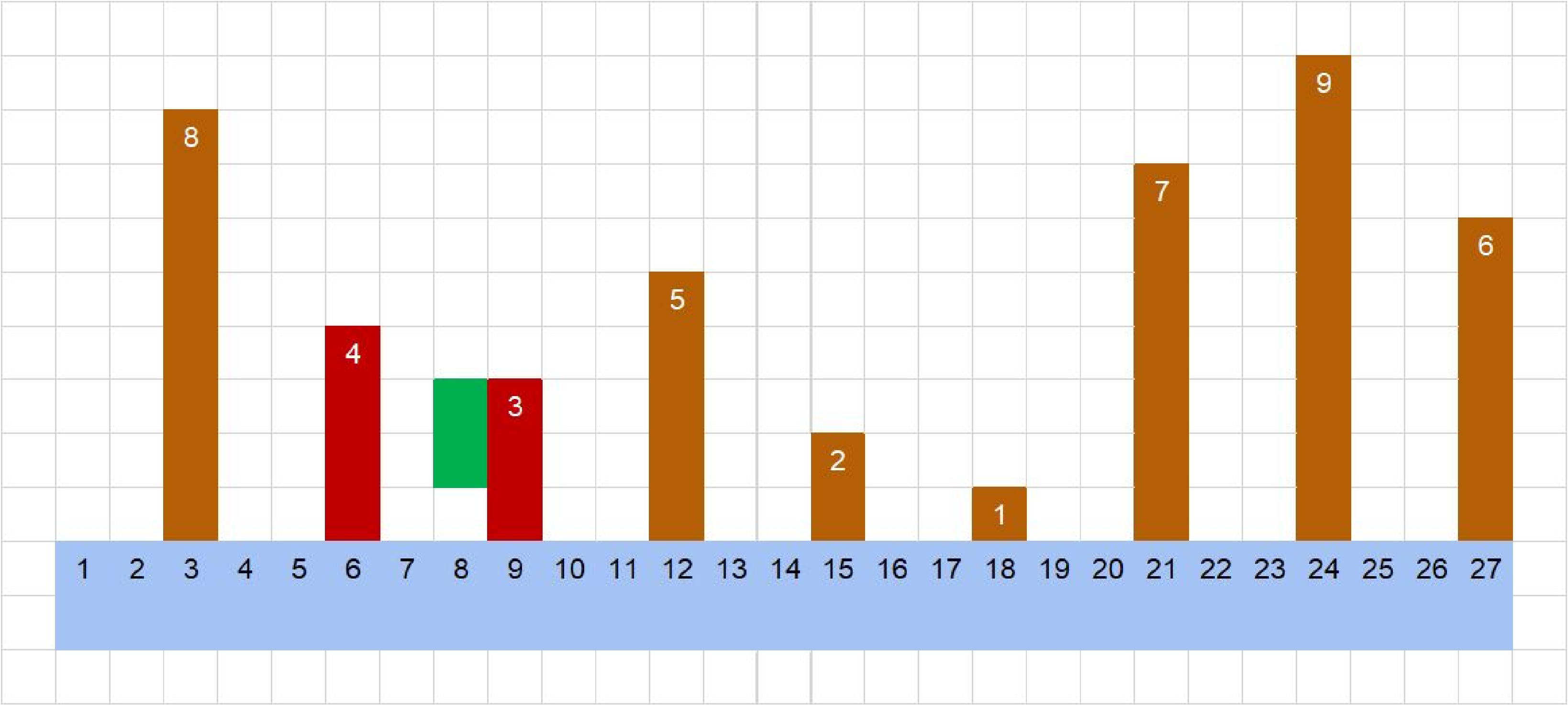
Changing from **y = 4** to **y = 3**



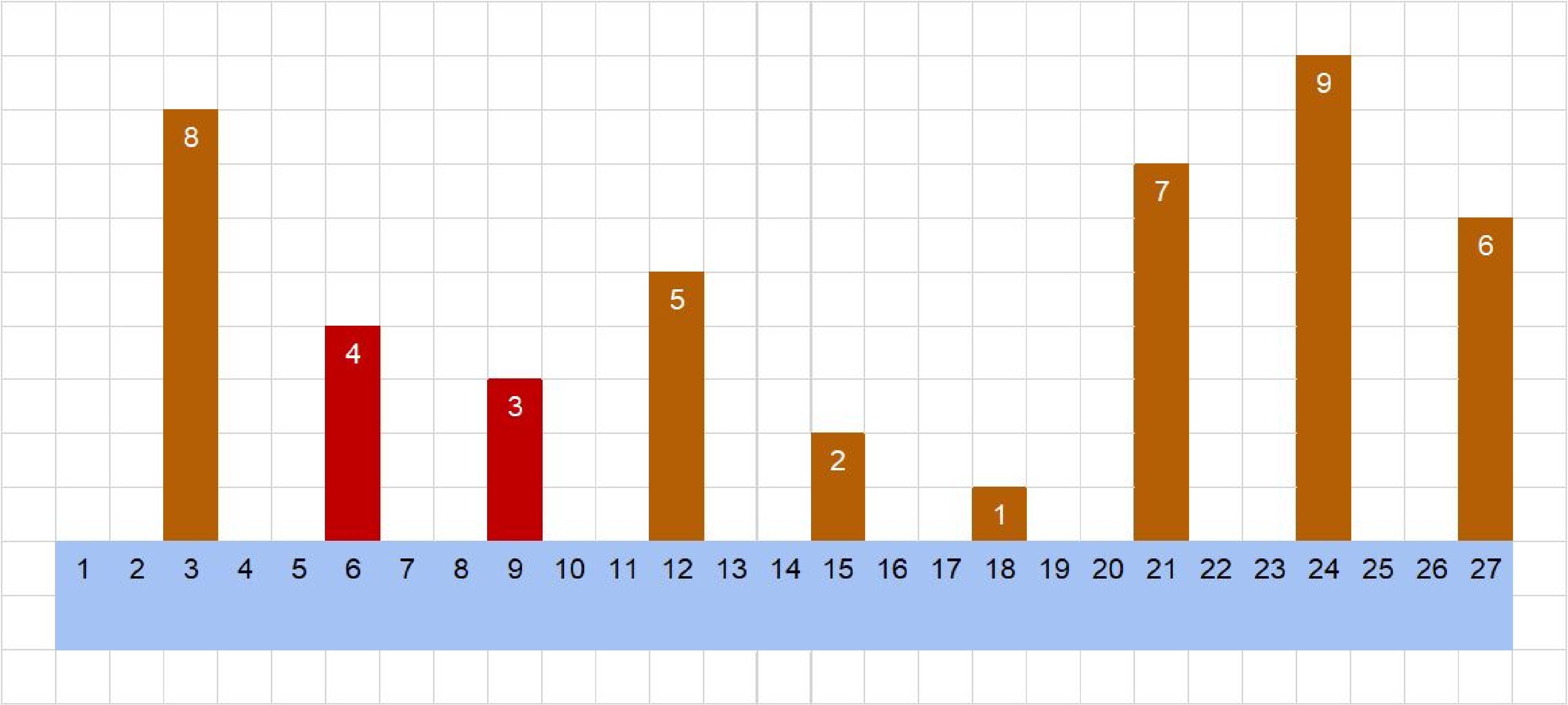
Changing from **y = 4** to **y = 3**



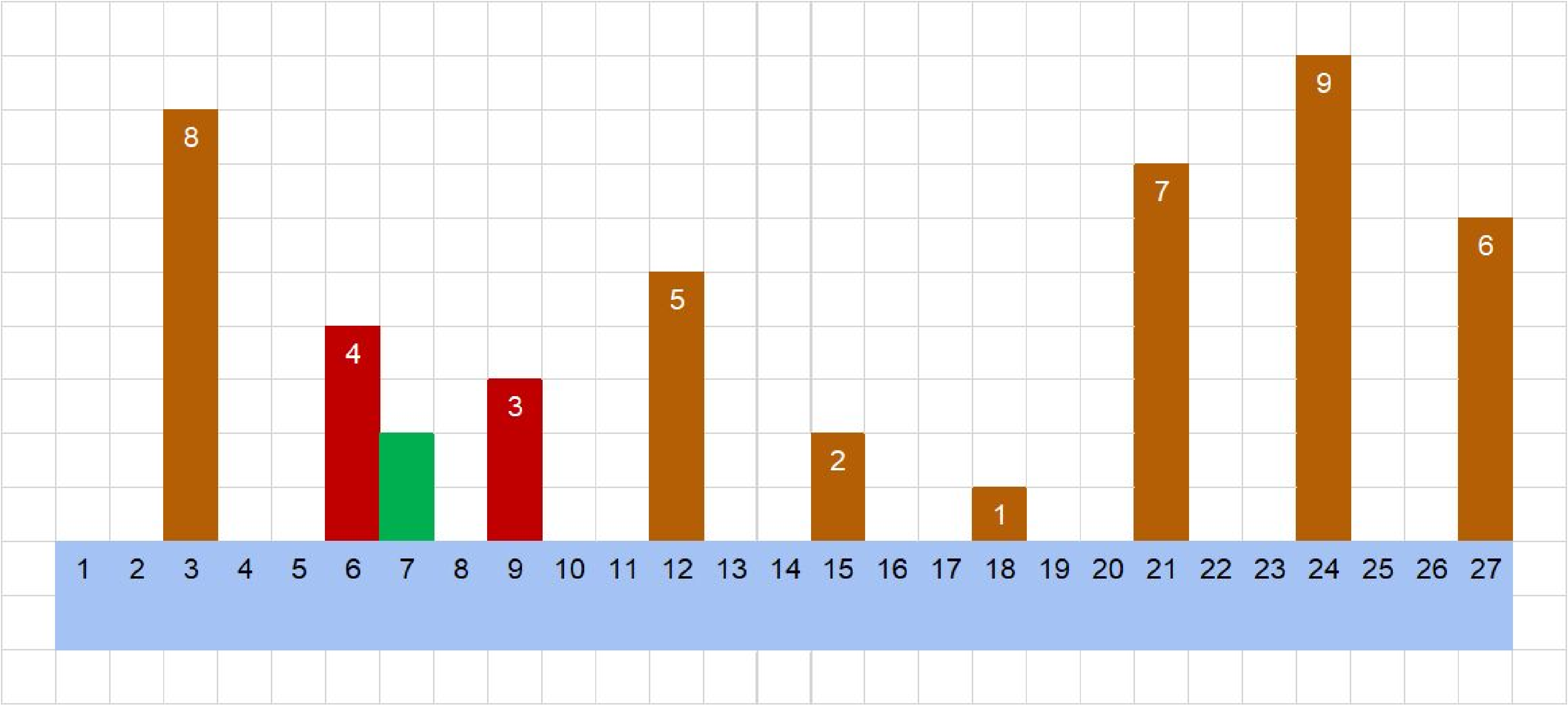
Changing from **y = 3** to **y = 2**



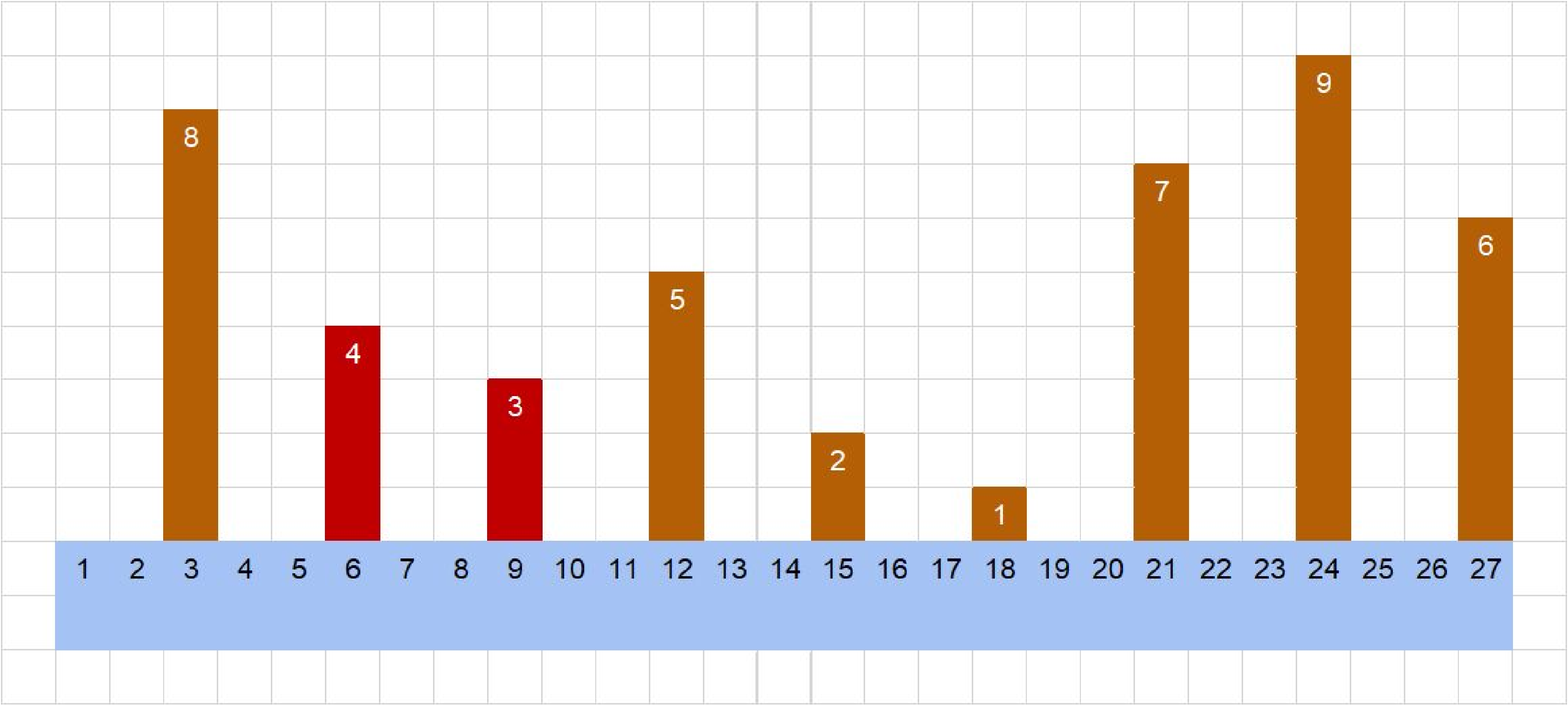
Changing from **y = 3** to **y = 2**



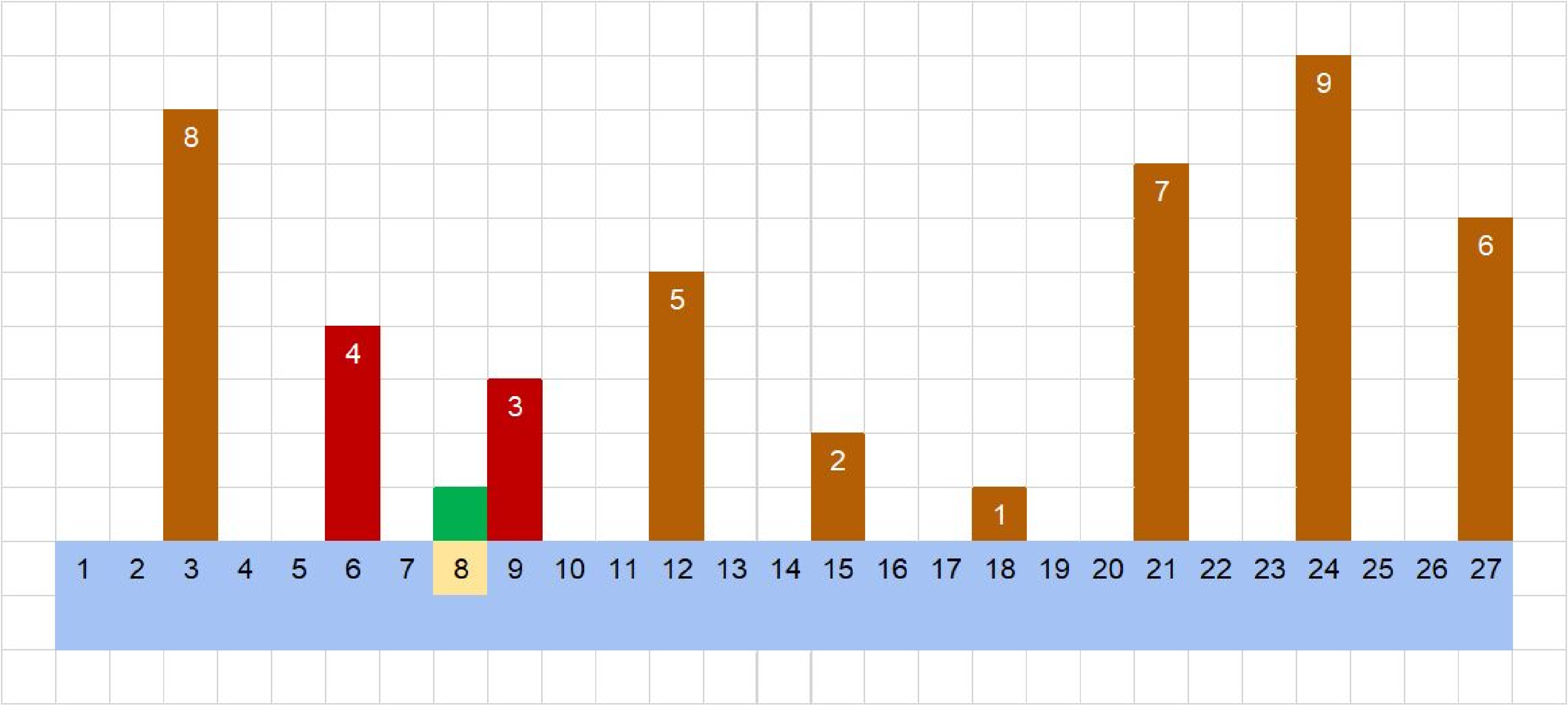
Changing from **y = 2** to **y = 1**



Changing from **y = 2** to **y = 1**



Changing from **y = 1** to **y = 0**



Solution 4 - Optimized Solution

In each decrement of y-coordinate, *O(1)* for updating bounds
Total has **N** decrement of y-coordinate as initial **y = N**

Therefore, overall time complexity is *O(N)*

Can pass the time limit for all test cases :)

Subtask	Score	Max Score
1	10	10
2	17	17
3	24	24
4	19	19
5	30	30
Total	**100**	**100**