# Programación Competitiva

Alex Josué Flórez Farfán

Magister en Ciencias de la Computación

Ciencia de la Computación – UNSA
Segundo semestre 2021

# Primera Nota

- Examen de entrada 15%
- Repositorio Github 25%
- Primer Examen 60%
- Puntos extra en clase

# GitHub

- Identificación
- readme
- License
- .gitignore
- Directorios para cada aula
- Nombres de archivos
- Message on commit

# Difference Array

# Haybale Stacking

- http://www.usaco.org/index.php?page=viewproblem2&cpid=104

Starts with N (1 <= N <= 1,000,000, N odd) empty stacks, Bessie is given a sequence of K instructions (1 <= K <= 25,000), each of the form "A B", meaning that Bessie should add one new haybale to the top of each stack in the range A..B.

Compute the median height afterwards

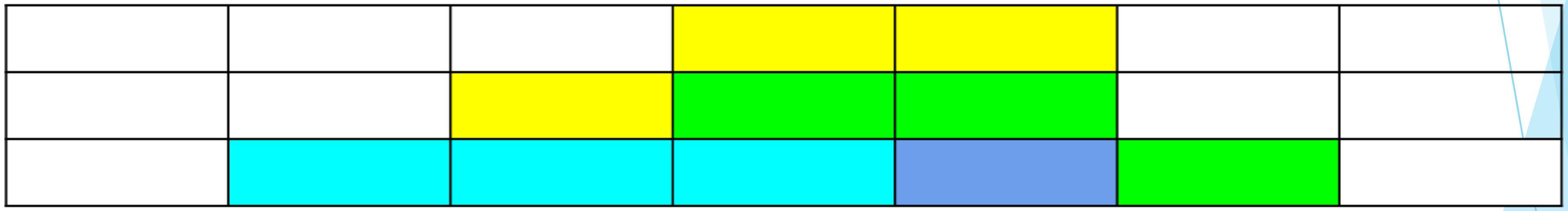
# Haybale Stacking

N stacks, K instructions of form “A B”

```
Input:
7 4
5 5
2 4
4 6
3 5
Output: 1
```

# Haybale Stacking

N=7 K=4 5 5 2 4 4 6 3 5



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Indices: | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Height: | 0 | 1 | 2 | 3 | 3 | 1 | 0 |

Heights sorted 0 0 1 1 2 3 3

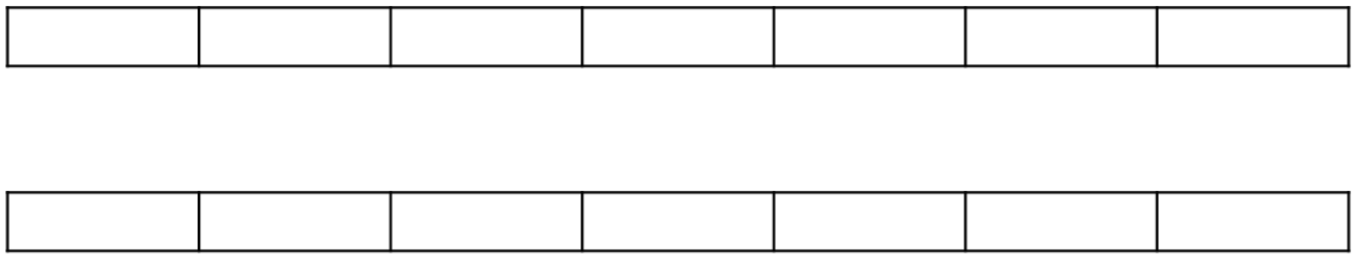
Median (output): 1

# Haybale Stacking. Difference Array

- Updates are ranges.
- Instead of directly maintaining the stack heights, we can maintain the differences of adjacent heights.
- Given an array A[0..n-1], its difference array is defined as
  - `D[0] = A[0]`
  - `D[k] = A[k] - A[k-1]  when 0 < k < n`
- One range update on A (which update all elements in one continuous range by a constant) corresponds to 2 updates on D
- Given the difference array D, reconstructing A takes linear time

# Haybale Stacking. Difference Array

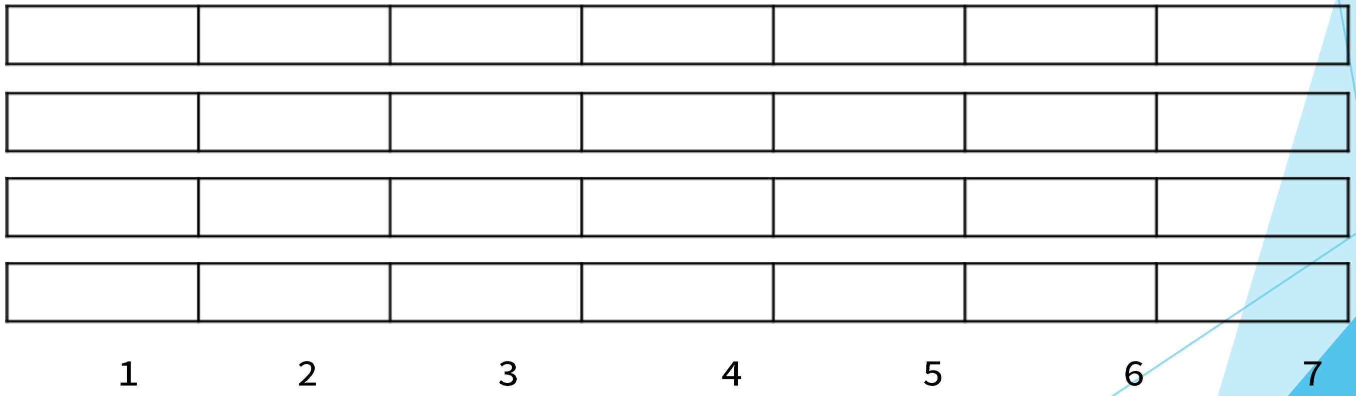
- D[0] = A[0]
- D[k] = A[k] - A[k-1]  when 0 < k < n



# Haybale Stacking. Difference Array

N stacks, K instructions of form “A B”

```
Input:
7 4
5 5
2 4
4 6
3 5
Output: 1
```



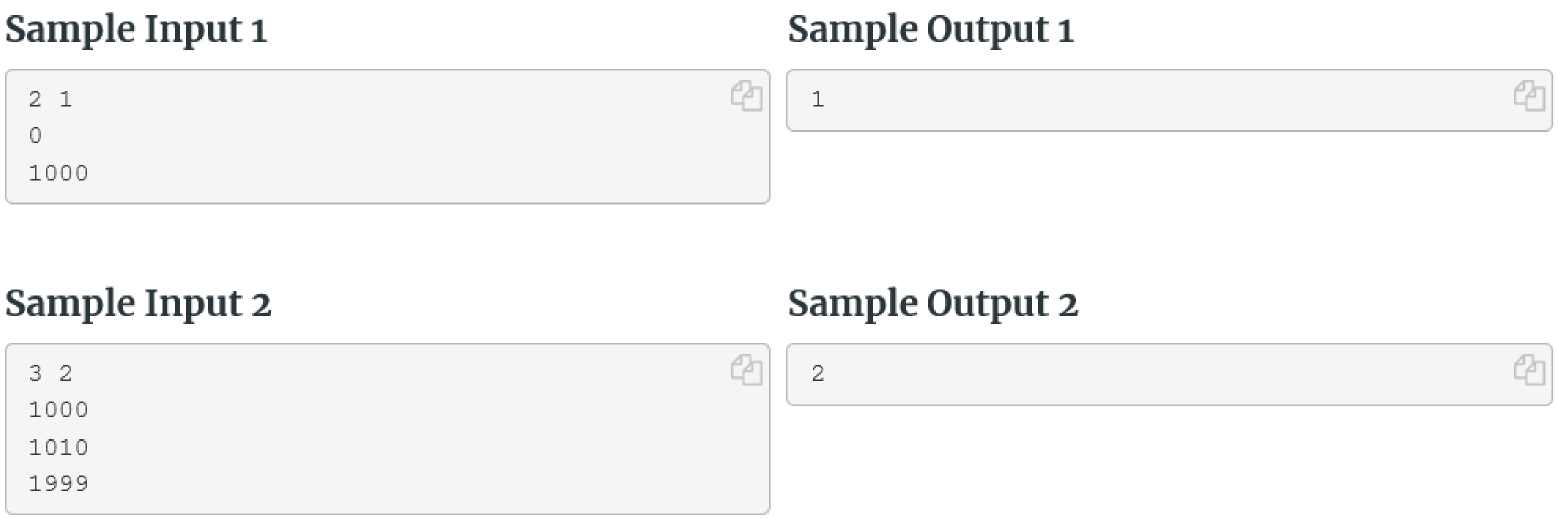
# Haybale Stacking. Difference Array

|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 5 5 |  |  |  |  | +1 | -1 |  |
| 2 4 |  | +1 |  |  | -1 |  |  |
| 4 6 |  |  |  | +1 |  |  | -1 |
| 3 5 |  |  | +1 |  |  | -1 |  |
| Final Difference Array | 0 | 1 | 1 | 1 | 0 | −2 | −1 |
| Final height | 0 | 1 | 2 | 3 | 3 | 1 | 0 |

# Disastrous Downtime

https://open.kattis.com/problems/downtime

Given the starting times of N (1≤N≤100,000) requests, each of which lasts 1000 milliseconds, and K (1≤K≤100,000), the maximum number of requests a server can handle concurrently, compute the minimum number of servers needed to process all requests.



**Sample Input 1**

```
2 1
0
1000
```

**Sample Output 1**

```
1
```

**Sample Input 2**

```
3 2
1000
1010
1999
```

**Sample Output 2**

```
2
```

# Disastrous Downtime

```
Input:    N=7 K=2
```

```
1000 1010 1500 1999 2000 2010 2999
```

```
Output:   2
```

- Explanation: Request 1 starts at 1000 and ends at 2000. At time 1999, there are 4 requests.

# Disastrous Downtime. Difference Array

We just need to keep track of the active number of requests at any time, which can be changed by two kinds of events:

(1) starting of a request (+1 event), and

(2) ending of a request (-1 event).

Sorting events based on time, and we have a difference array.

# Disastrous Downtime. Difference Array

```
Input:    N=7 K=2

1000   1010   1500   1999   2000   2010   2999
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Time | | | | | | | | | | | |
| Diff | | | | | | | | | | | |
| Acc | | | | | | | | | | | |