Curso Programación Competitiva: Soluciones Sesión 1

12/03 2021

¡Gracias por participar!

ETSII – URJC. Grupo de Programación Competitiva

@urjc_cp, @etsii_urjc

Estadísticas de los problemas

*antes del Freeze

Programando el online de GTA	s.perezc.2018_CP [6’]
Ayudame Atica!	j.vega.2019_CP [18’]
Pito Pito Gorgorito	a.pinaz.2020_CP [20’]
Tramitando papeleo	¿?
Elecciones en la URJC	¿?
El Temible Monstruo Come Números	¿?

Estadísticas de los problemas

*antes del Freeze

Problema	Categoría	Casos de prueba
Programando el online de GTA	Conjuntos (Sets), Ad-hoc	6 (10,5 MB)
Ayudame Atica!	Arrays, bucles, sumas	3 (60 KB)
Pito Pito Gorgorito	Arrays, módulos	4 (1.11 MB)
Tramitando papeleo	Colas, Simulación	10 (200 KB)
Elecciones en la URJC	Mapas, Pensar	6 (416 KB)
El Temible Monstruo Come Numeros	Memorización, arrays, pensar	5 (30 MB)

Envíos totales

Submissions: 212

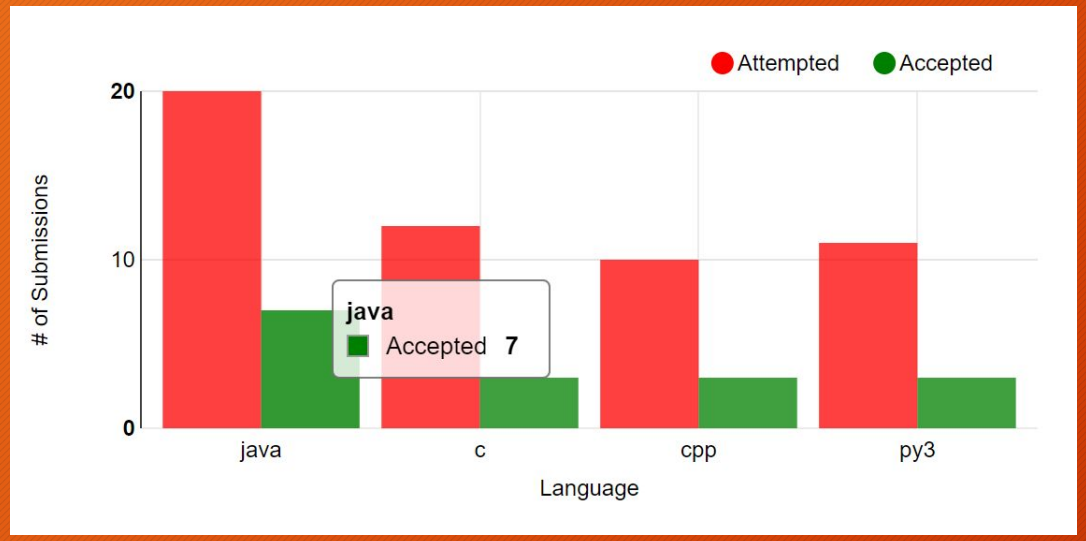
Accepted Submissions: 52 (~25%)

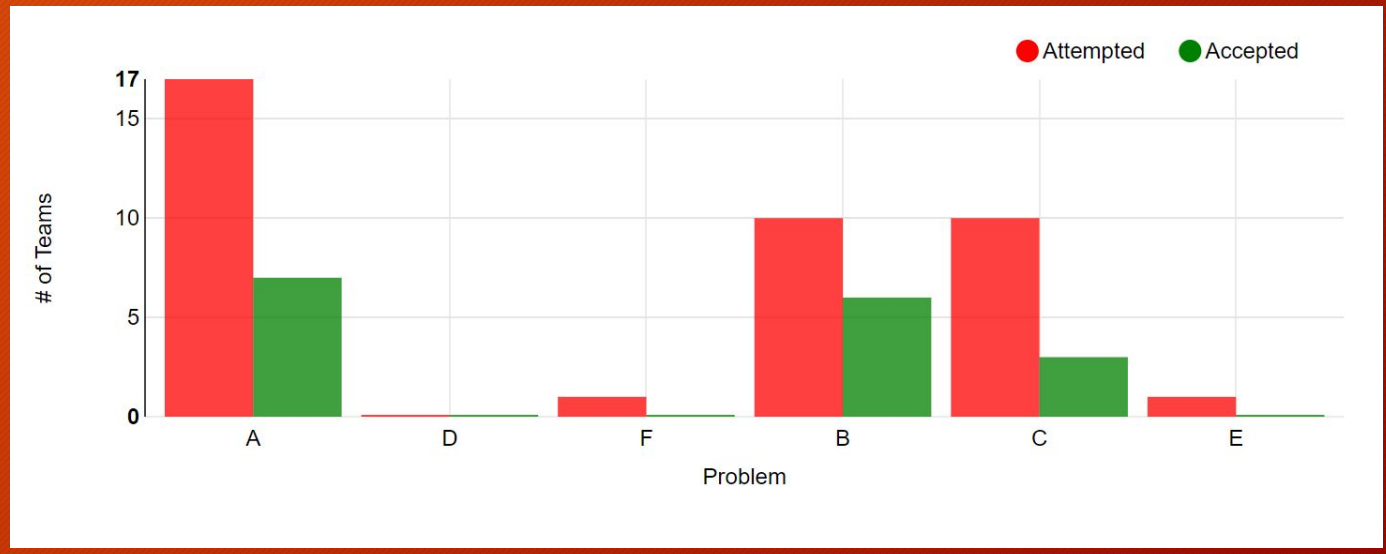
Teams: 82

Clarifications: 13

Estadísticas de los problemas

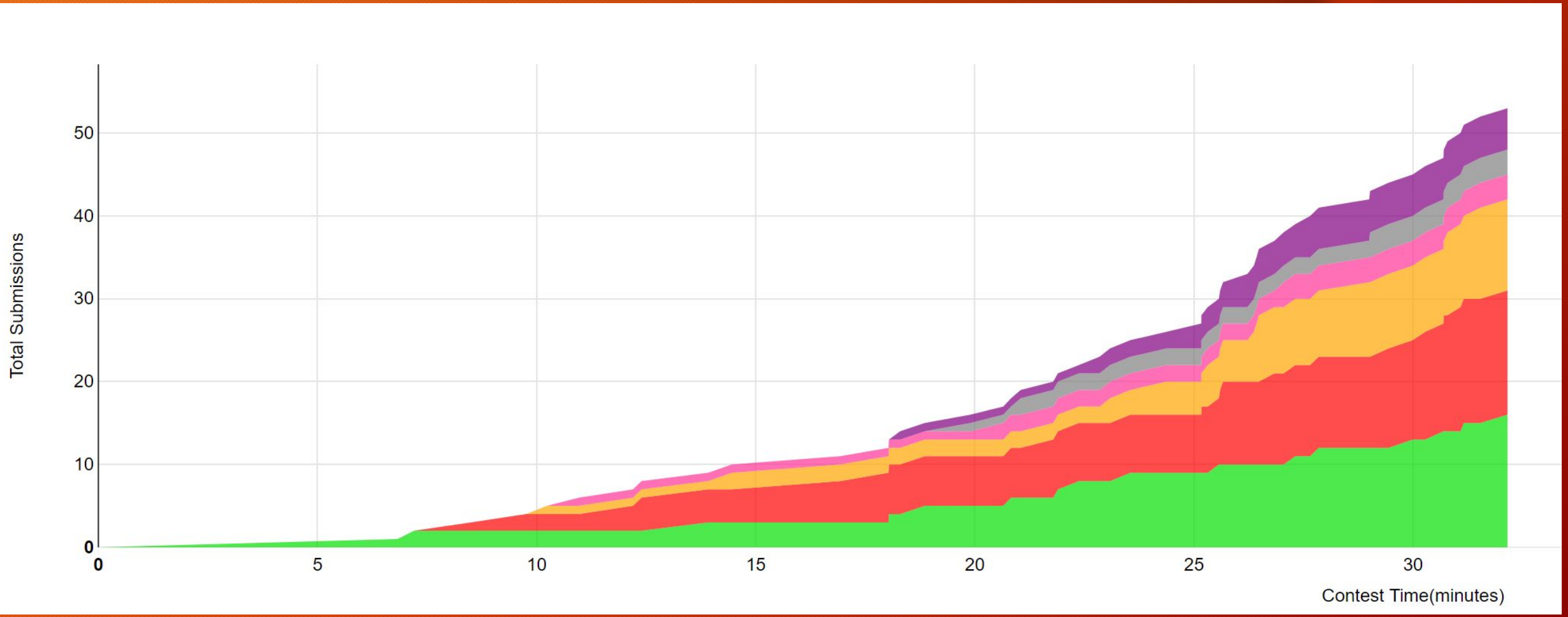
*antes del Freeze





Estadísticas

*antes del Freeze



Problema A

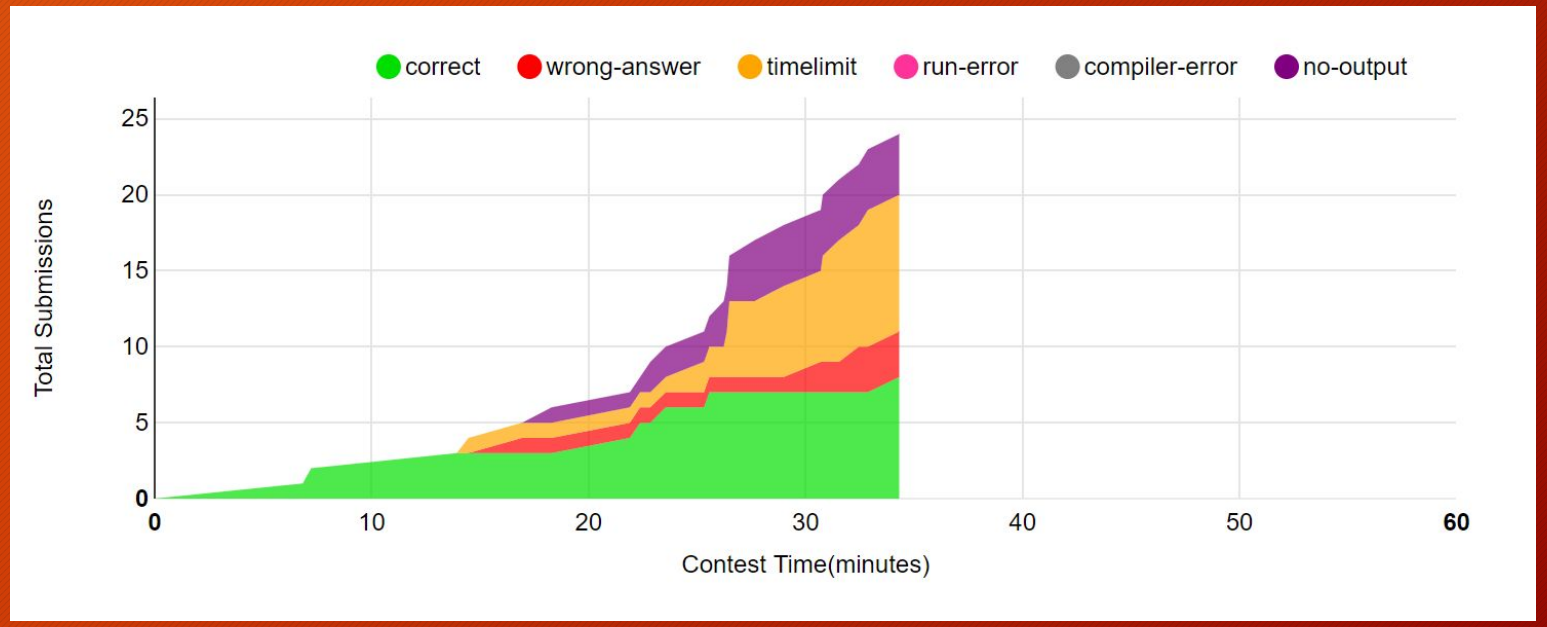
Programando el online de GTA

Autor: Iván Martín

Primer AC: c.perezc.2018_CP [6’]

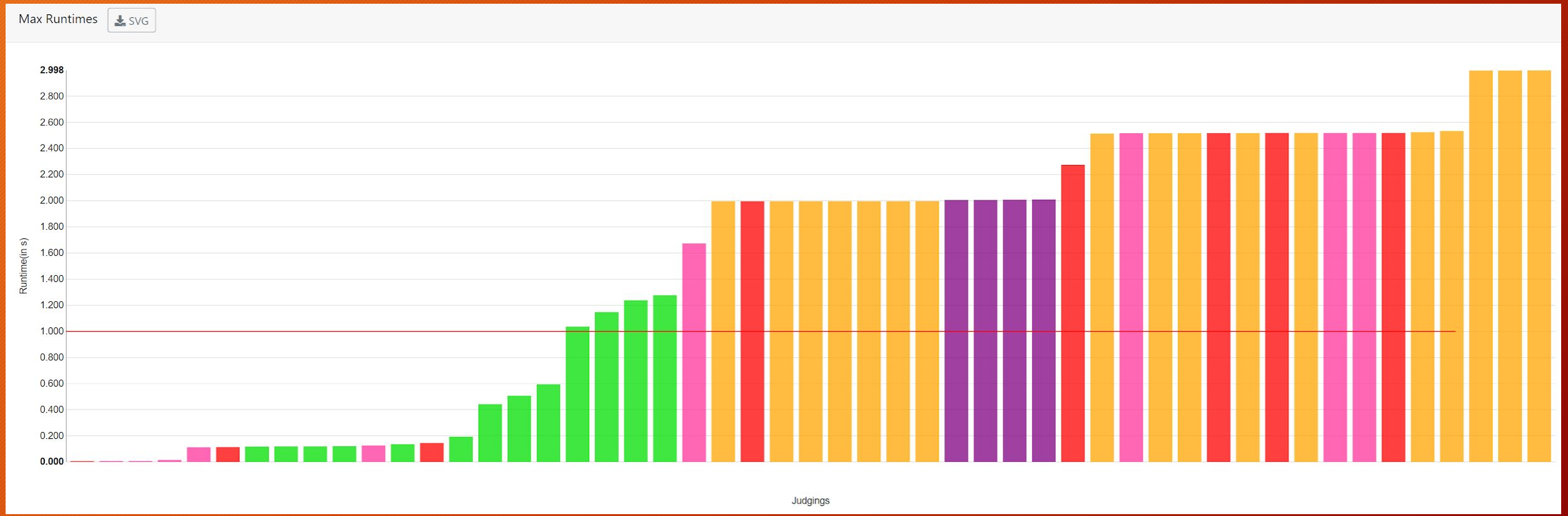
Programando el online de GTA

A



Programando el online de GTA

A



Programando el online de GTA

A

¿Leíste el artículo que pasamos la semana pasada? ¡Si lo leíste, este problema es trivial!

El primer paso importante es leer bien el enunciado y comprobar qué nos piden: Solo necesitamos saber si un objeto existe o no, por lo que no es necesario guardar ni su nombre ni su precio. No hace falta trabajar con los demás datos, ni crear clases adicionales.

Programando el online de GTA

A

Una primera idea podría ser almacenar en un array, vector o lista los objetos del juego. El problema es que el número de objetos es muy elevado (Hasta 10^6) y cada operación de buscar por toda la lista nos obliga a recorrer los 10^6 objetos e ir comprobando uno por uno si es el buscado.

Hay una estructura de datos que permite comprobar si un objeto existe o no sin necesidad de buscarlo, ¡un conjunto!

Añadiendo los identificadores de los objetos a un conjunto, podemos saber en complejidad constante si está o no ¡Accepted!

Problema B

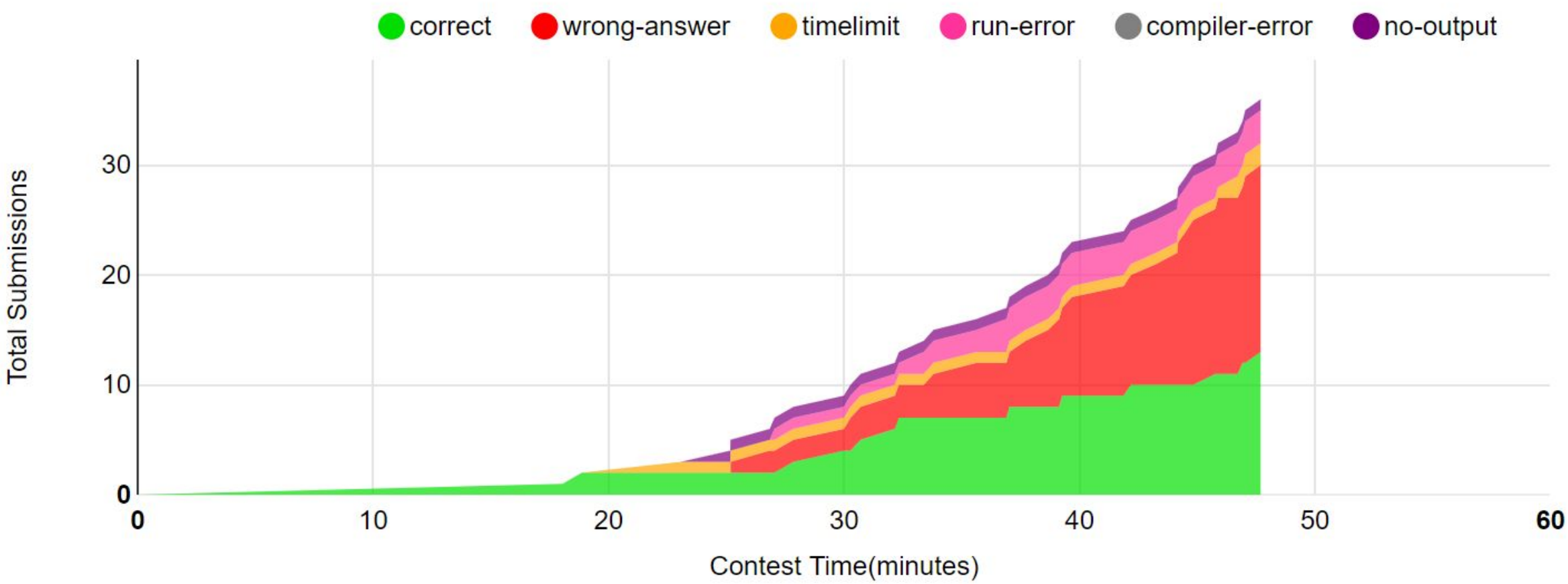
Ayudame Atica!

Autor: Raúl Martín

Primer AC: j.vega.2019_CP

Ayudame Atica!

B



Ayudame Atica!

B

Problema de aritmética fácil.

Solución: Acumular `Max(0, 50 - nota)` ← ¡Cuidado con resta negativa!

`IF acumulado > horasDisponibles → NOS VEMOS EN JUNIO...`

`else → APRUEBA TODO`

Problema C

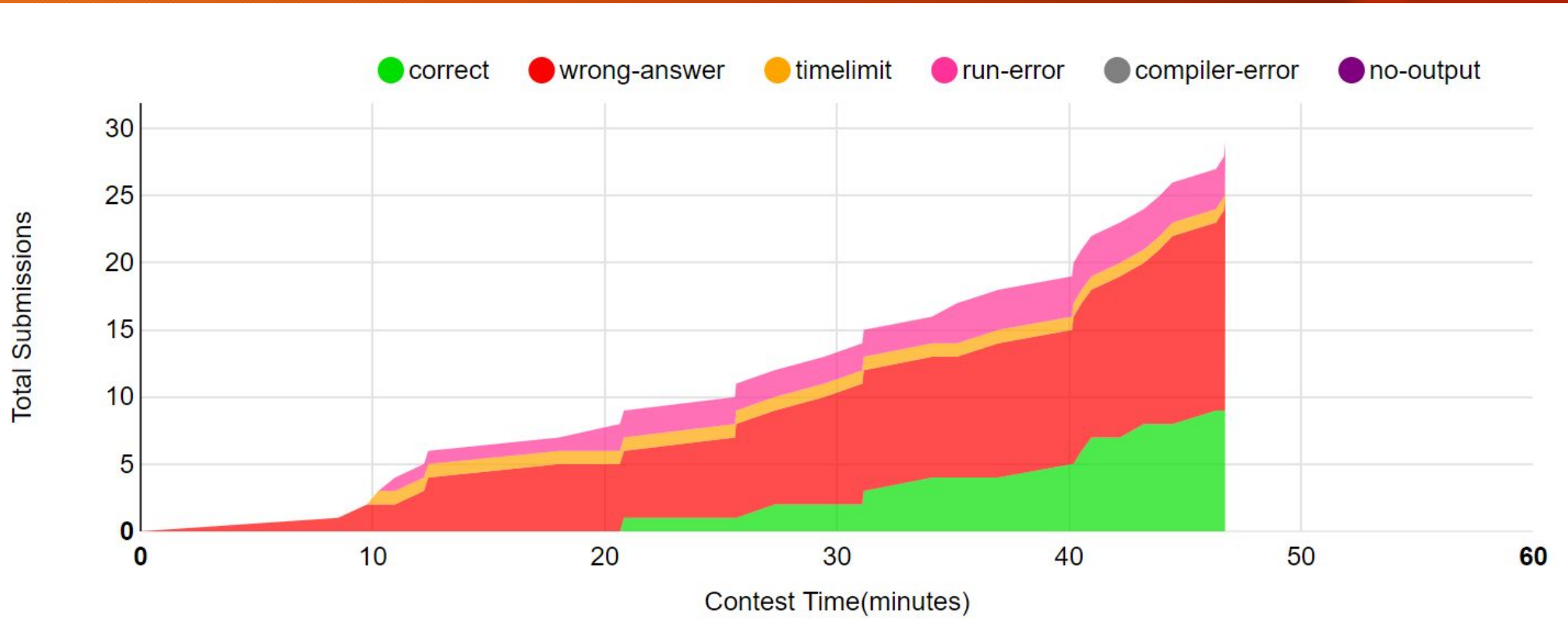
Pito Pito Gorgorito

Autor: Isaac Lozano

Primer AC: a.pinaz.2020_CP

Pito Pito Gorgorito

C



Pito Pito Gorgorito

C

Solución:
Simularlo → TimeLimit

Solución Correcta:

```
seLaLiga = numeroPersonas MOD sílabas
if seLaLiga == 0 → seLaLiga = numeroPersonas
for n in personas:
    if n == seLaLiga: SI else: NO
```

Problema D

Tramitando papeles

Autor: Iván Martín

Primer AC: ¿?

Tramitando papeles

D

La mejor forma de solucionar este problema es simular el proceso completo. Para ello, necesitamos una estructura de datos que nos permita sacar el primer elemento, y añadir uno al final ¡Una cola!

Vamos sacando el primer elemento de la cola, lo ponemos al final si va a tardar más de T/5 (siempre y cuando el tiempo actual no sea mayor o igual a 10) y vamos tramitando los papeles.

También necesitaremos una estructura de datos auxiliar para llevar la cuenta de los papeles que dejaremos para después, eliminandolos tras la primera vez que nos los encontremos. Un arreglo de booleanos o un set resuelven esto perfectamente.

Problema E

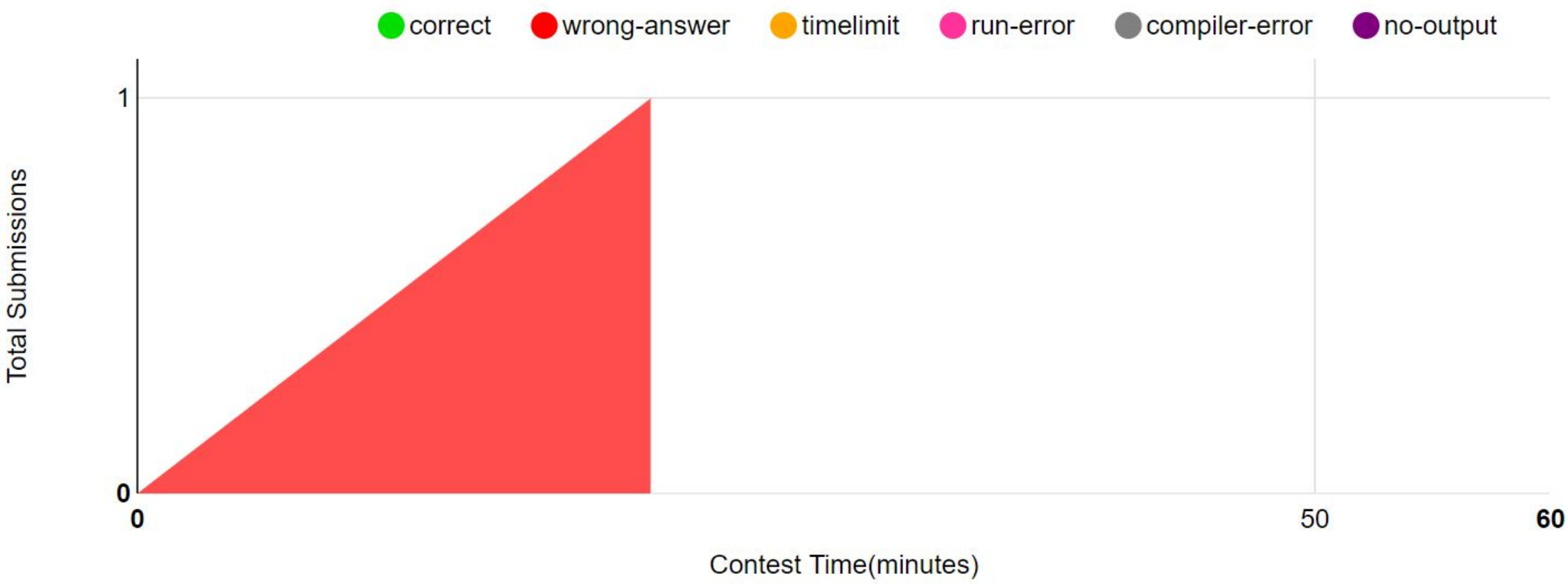
Elecciones en la URJC

Autor: Iván Martín de San Lázaro

Primer AC:

Elecciones en la URJC

E



Elecciones en la URJC

E

Este problema tenía muchas cosas que tener en cuenta. Se resumen en 2:

- Cómo analizar los resultados de cada campus
- Cómo saber quién ha ganado

Elecciones en la URJC

E

Pensemos en los resultados de cada campus. Observemos el ejemplo

- Opción 1: Se conocen los resultados de todos los candidatos

(Por ejemplo: Vicálvaro)

En este caso, apuntamos que cada candidato tiene dichos votos

Candidato	Votos
Héctor	9
Álvaro	5
María	2
Pablo	4

Elecciones en la URJC

E

Pensemos en los resultados de cada campus. Observemos el ejemplo

- Opción 2: Se conocen los resultados de todos los candidatos menos de uno

(Por ejemplo: Argüelles)

El enunciado explica que cada candidato solo sale una vez

Así que concluimos que el que falte se lleva todos los votos

Candidato	Votos
Héctor	34
Álvaro	15
María	12
Pablo	9

Elecciones en la URJC

E

Pensemos en los resultados de cada campus. Observemos el ejemplo

- Opción 3: Se conocen los resultados de todos los candidatos menos de dos o más

(Por ejemplo: Móstoles)

No tenemos certeza de quién se lleva los votos que faltan

Así que los anotamos como votos posibles para Álvaro o para Pablo

Candidato	Votos	Posibles
Héctor	64	0
Álvaro	15	50
María	32	0
Pablo	9	50

Elecciones en la URJC

E

Sabiendo esto, contemos los resultados. Parece obvio que Álvaro ha ganado, tiene 65 votos… ¿Pero los tiene de verdad? ¡No! Es una suposición.

De los 50 votos que faltan, se los podría haber llevado Álvaro o Pablo.

Sumando los 50 votos, o bien Álvaro o bien Héctor Podrían haber ganado

Candidato	Votos	Posibles
Héctor	64	0
Álvaro	15	50
María	32	0
Pablo	9	50

Elecciones en la URJC

E

Tras analizar los resultados de cada campus, cada candidato tiene dos cuentas, los votos certificados y los que podría haber conseguido si todos los votos desconocidos fueran a ese candidato.

Ahora podemos analizar candidato por candidato si ha ganado o no. Cada candidato, en el mejor de los casos tendrá sus votos asegurados + todos los que faltan por votar, y sus rivales ni un voto más. Si con esta suma supera a los votos asegurados del resto de rivales, es candidato a la victoria.

Problema F

El Temible Monstruo Come Números

Autor: David Morán

Primer AC: ¿?

El Temible Monstruo Come Números

F

- El temible monstruo come números trata de un problema que, a pesar que se resuelve usando un array, ¡Requiere saber un poco de mates!
  - Un número, en principio, tiene 2 divisores principales, 1 y él mismo
  - Por este mismo principio, para un divisor válido X que divide a un N, N/X tambien es un divisor válido. E.g. 3 es divisor de 12, pero 12/3 = 4, tambien lo es
  - Teniendo en cuenta este principio, se puede iterar de 1 hasta sqrt(N) y, siempre y cuando N%i == 0 entonces, tanto i como N/i tambien es divisible

El Terrible Monstruo Come Números

F

- ¡Sin embargo!
  - Hay que tener en cuenta que para números cuadrados solo puedes contar una vez, por lo que para un número, digamos, 49, 7 es divisor, pero 7 tambien lo es (no podemos contar dos veces!)
  - Así mismo, para 1 tambien ocurre lo mismo (pero este caso no estaba contemplado en ese problema)

El Terrible Monstruo Come Números

F

- Calcular todo esto en tiempo de ejecución = TLE
  - Necesitas hacer el precálculo de todos los números posibles antes de leer el caso, esto es debido a que:
    - Para cualquier caso de prueba, los números no variarán la cantidad de divisores
    - N=100,000 y T=100. Muchísimas operaciones!
    - Mejor guardarlos en un array y consultar para cualquier número X del 2 al 100,000. Cuántos divisores tiene
  - Finalmente, por cada número, guardamos cuántos divisores tiene, ordenamos todos de mayor a menor (para garantizar el menor número de “bajas”) y restamos hasta vencer al monstruo.
  - Si no podemos vencerlo. ¡Hay que correr!