AdaByron 2020

Estadísticas y Soluciones

accenture
Alto rendimiento. Hecho realidad.

ThoughtWorks®

URJC

Clasificación de los problemas

Problema	Categoría
A - Enero: Dickie está en aprietos	Cadenas de markov, grafos, exponenciación rápida.
B - Febrero: En el último minuto	TSP, Backtracking, Programación dinámica, Máscara de bits.
C - Marzo: La moda secundaria	Mapas, ordenamiento
D - Abril: Declaración de la renta	Trie, árboles, vectores, bucles.
E - Mayo: Dickie y las hamburguesas	Búsqueda binaria sobre resultados, matemáticas.
F - Junio: Rebajas de Verano	AdHoc, precisión de doubles.
G - Julio: Torneos de Ajedrez	Pensar, condicionales, strings, AdHoc.
H - Agosto: HH y el bosón	Teorema del resto chino-
I - Septiembre: Vuelta a la rutina	Estructuras de datos.
J - Octubre: Recogiendo Caramelos	Programación dinámica, grafos.
K - Noviembre: La casa encantada	AdHoc.
L - Diciembre: Dora la aisladora	Grafos, puntos de articulación, BFS.

Estadísticas

Problema	# casos de prueba	Espacio en disco
A - Enero: Dickie está en aprietos	63	678KB
B - Febrero: En el último minuto	29	15KB
C - Marzo: La moda secundaria	9	4.3MB
D - Abril: Declaración de la renta	8	2.4MB
E - Mayo: Dickie y las hamburguesas	11	544KB
F - Junio: Rebajas de Verano	50	73MB
G - Julio: Torneos de Ajedrez	16	16KB
H - Agosto: HH y el bosón	102	61KB
I - Septiembre: Vuelta a la rutina	7	58MB
J - Octubre: Recogiendo Caramelos	71	54KB
K - Noviembre: La casa encantada	11	3.9MB
L - Diciembre: Dora la aisladora	13	1.3MB
- **Total**	390	144.27MB

Estadísticas*

Problema	Primer equipo en resolverlo	Tiempo
A - Enero: Dickie está en aprietos	Segment Tree de máximos	177
B - Febrero: En el último minuto	(·_·) (·_·) (·_·)	65
C - Marzo: La moda secundaria	Segment Tree de mínimos	12
D - Abril: Declaración de la renta	Echo	109
E - Mayo: Dickie y las hamburguesas	-	-
F - Junio: Rebajas de Verano	Segment Tree de mínimos	43
G - Julio: Torneos de Ajedrez	UwUntu	9
H - Agosto: HH y el bosón	-	-
I - Septiembre: Vuelta a la rutina	Fish Team	9
J - Octubre: Recogiendo Caramelos	-	-
K - Noviembre: La casa encantada	Segment Tree de mínimos	82
L - Diciembre: Dora la aisladora	Segment Tree de mínimos	173

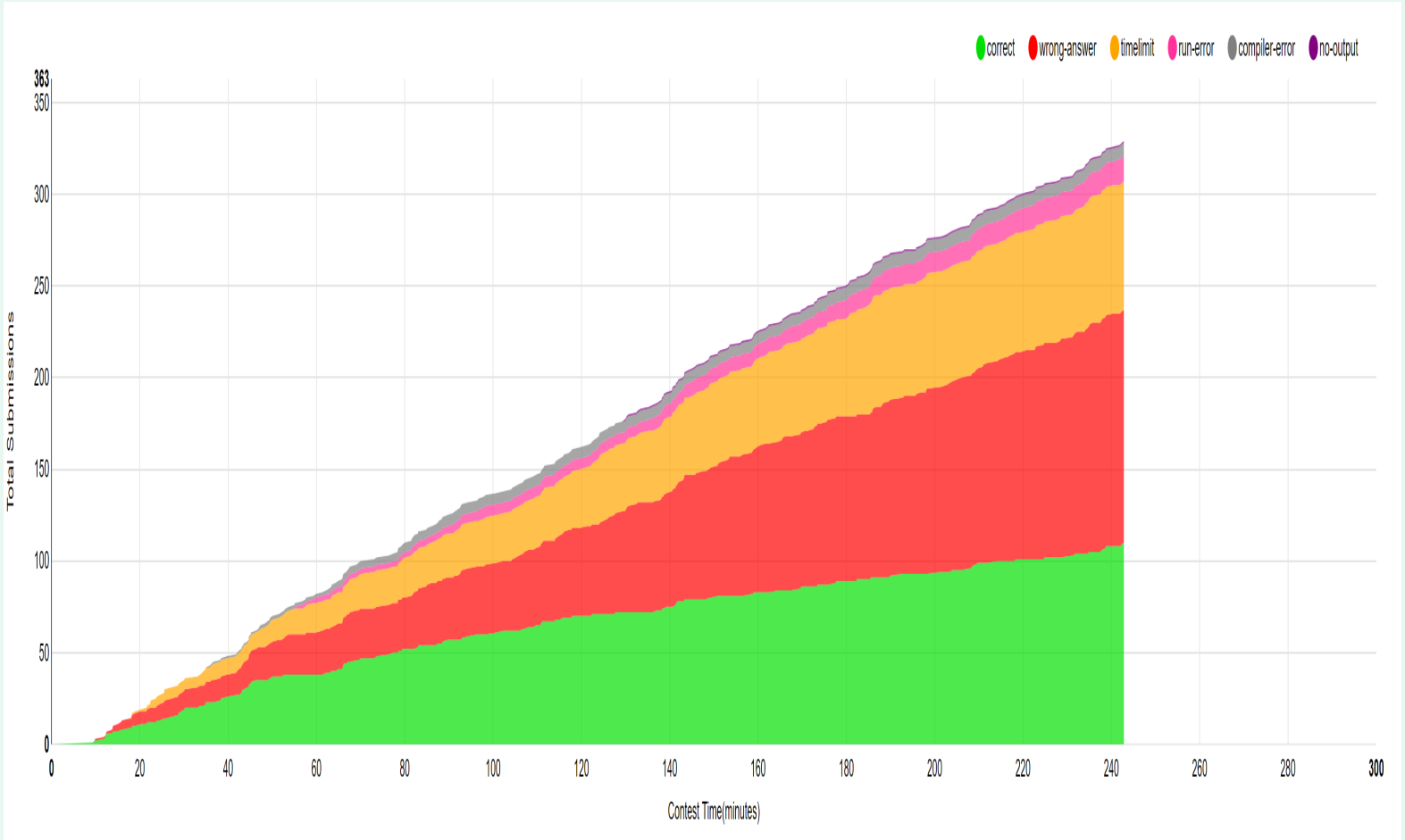
*Antes de congelar el marcador.

Estadísticas*

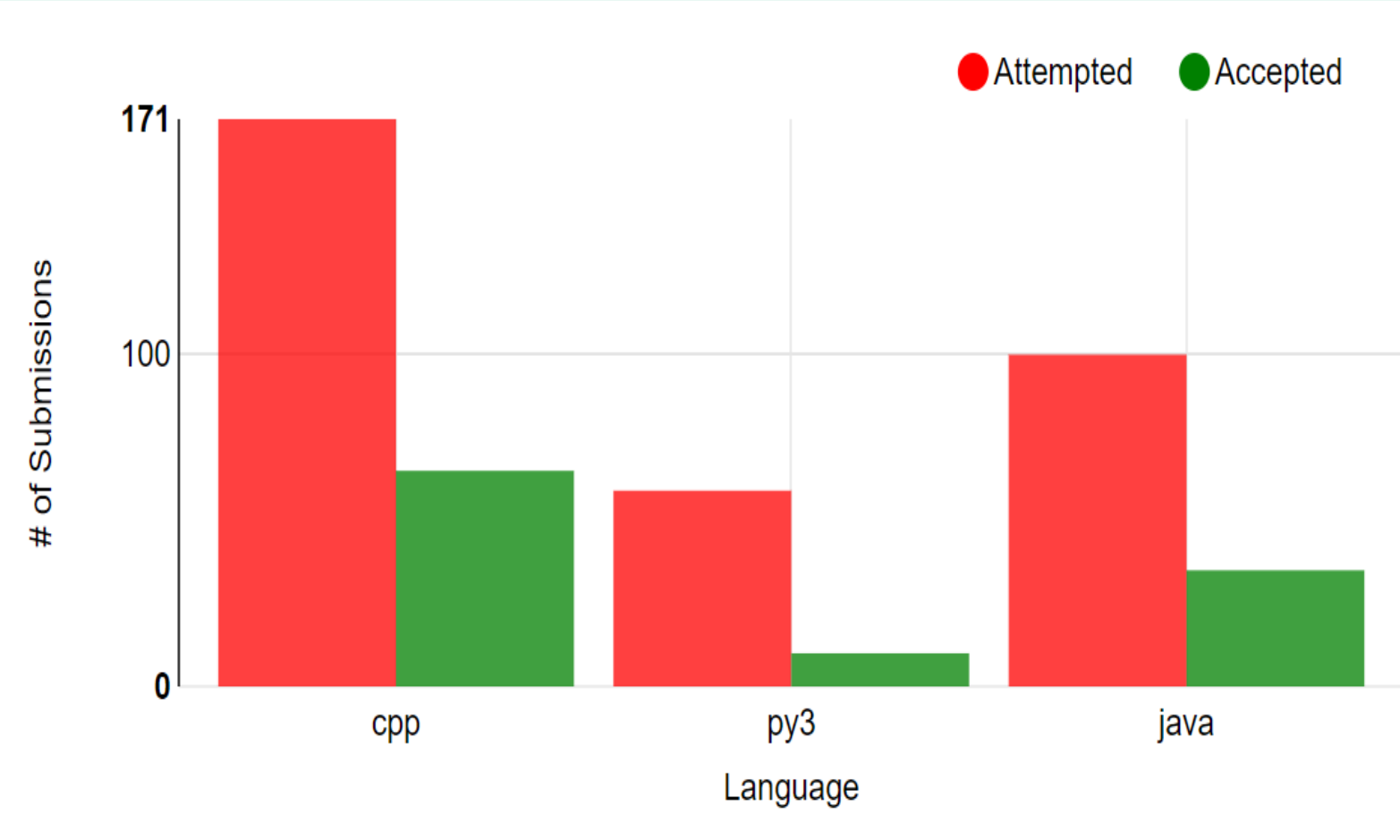
Problema	Envíos	Válidos	% éxito
A - Enero: Dickie está en aprietos	3	1	33 %
B - Febrero: En el último minuto	10	3	30 %
C - Marzo: La moda secundaria	44	15	34 %
D - Abril: Declaración de la renta	30	6	20 %
E - Mayo: Dickie y las hamburguesas	20	0	0 %
F - Junio: Rebajas de Verano	30	10	33 %
G - Julio: Torneos de Ajedrez	30	24	80 %
H - Agosto: HH y el bosón	5	0	0 %
I - Septiembre: Vuelta a la rutina	74	22	30 %
J - Octubre: Recogiendo Caramelos	0	0	0 %
K - Noviembre: La casa encantada	19	10	52 %
L - Diciembre: Dora la aisladora	12	1	8 %

*Antes de congelar el marcador.

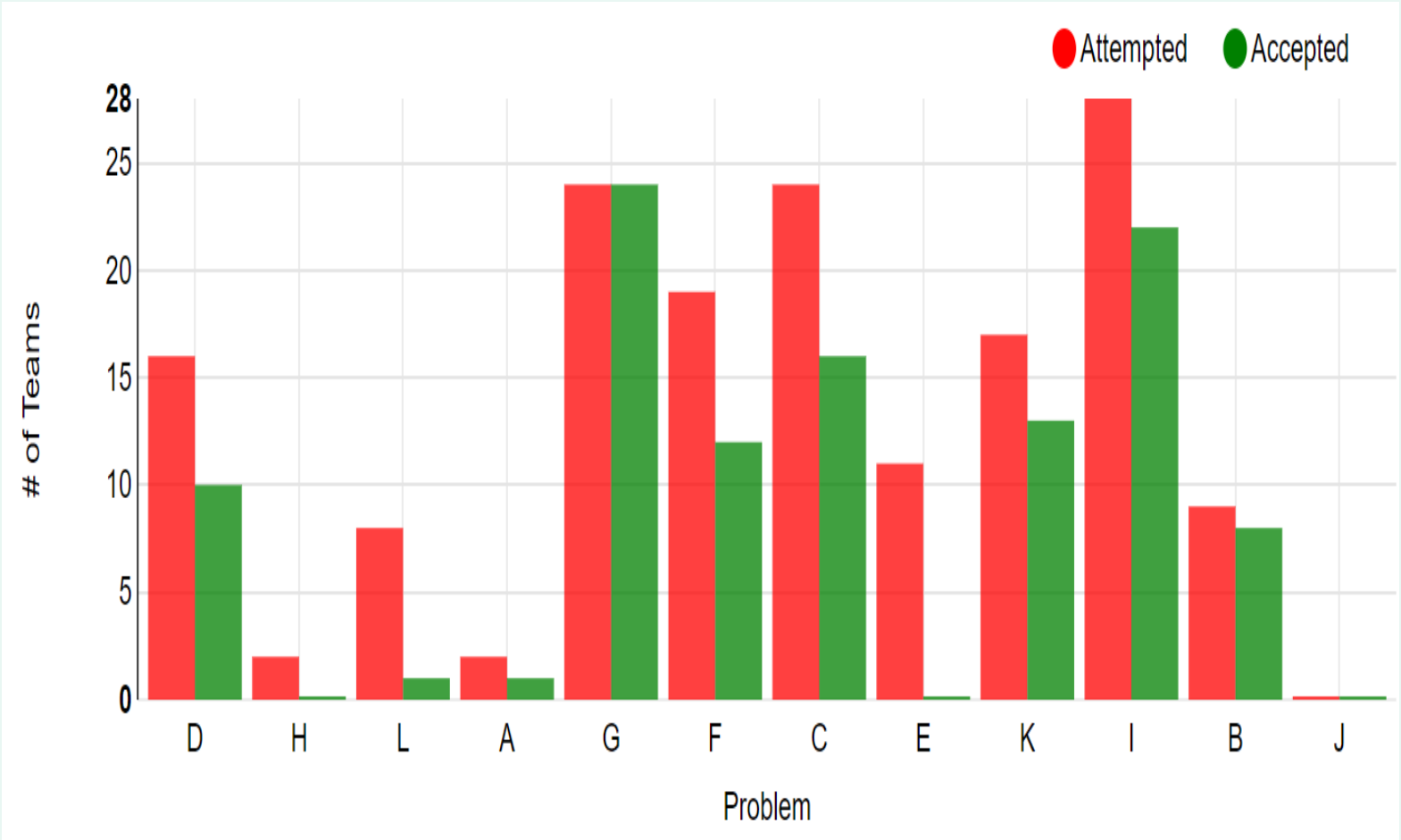
Estadísticas varias



Estadísticas varias



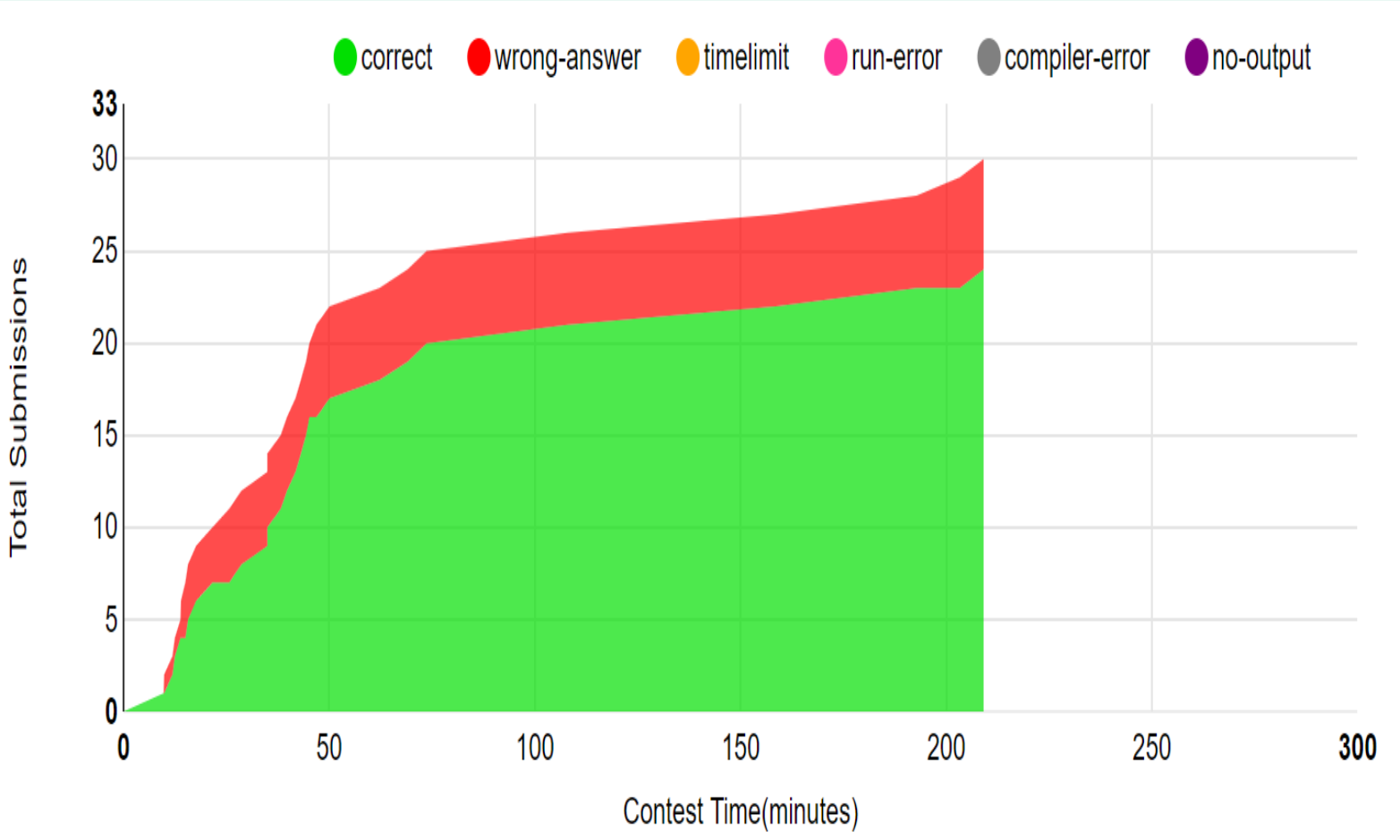
Estadísticas varias



G. Julio: Torneos de Ajedrez

Envíos	Válidos	% éxito
30	24	80 %

G. Julio: Torneos de Ajedrez



G. Julio: Torneos de Ajedrez

¿Problema de implementación completa?

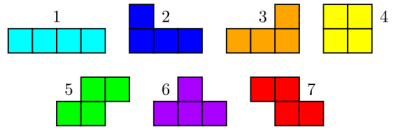
Altura final en Tetris

Tiempo máximo: 2,000 s Memoria máxima: 4096 KiB

El Tetris es un videojuego que se popularizó en los años 80. Consiste en colocar una serie de piezas con distintas formas que van cayendo en un tablero, de tal modo que queden encajadas de la forma más compacta posible.

En este problema vamos a suponer una secuencia de piezas que caen, cada una en una determinada posición y con una determinada orientación que no se pueden cambiar. Las piezas se van amontonando según caen y no se eliminan las filas completas (como ocurre en el juego original). El objetivo es determinar la altura final de cada columna del tablero después de que caigan todas las piezas.

Hay un total de 7 piezas diferentes, mostradas en la figura:



G. Julio: Torneos de Ajedrez

Problema basado en pregunta de entrevista de Google. Dado los puntos de dos tenistas, determinar el ganador del partido. ¿Hace falta implementar?.

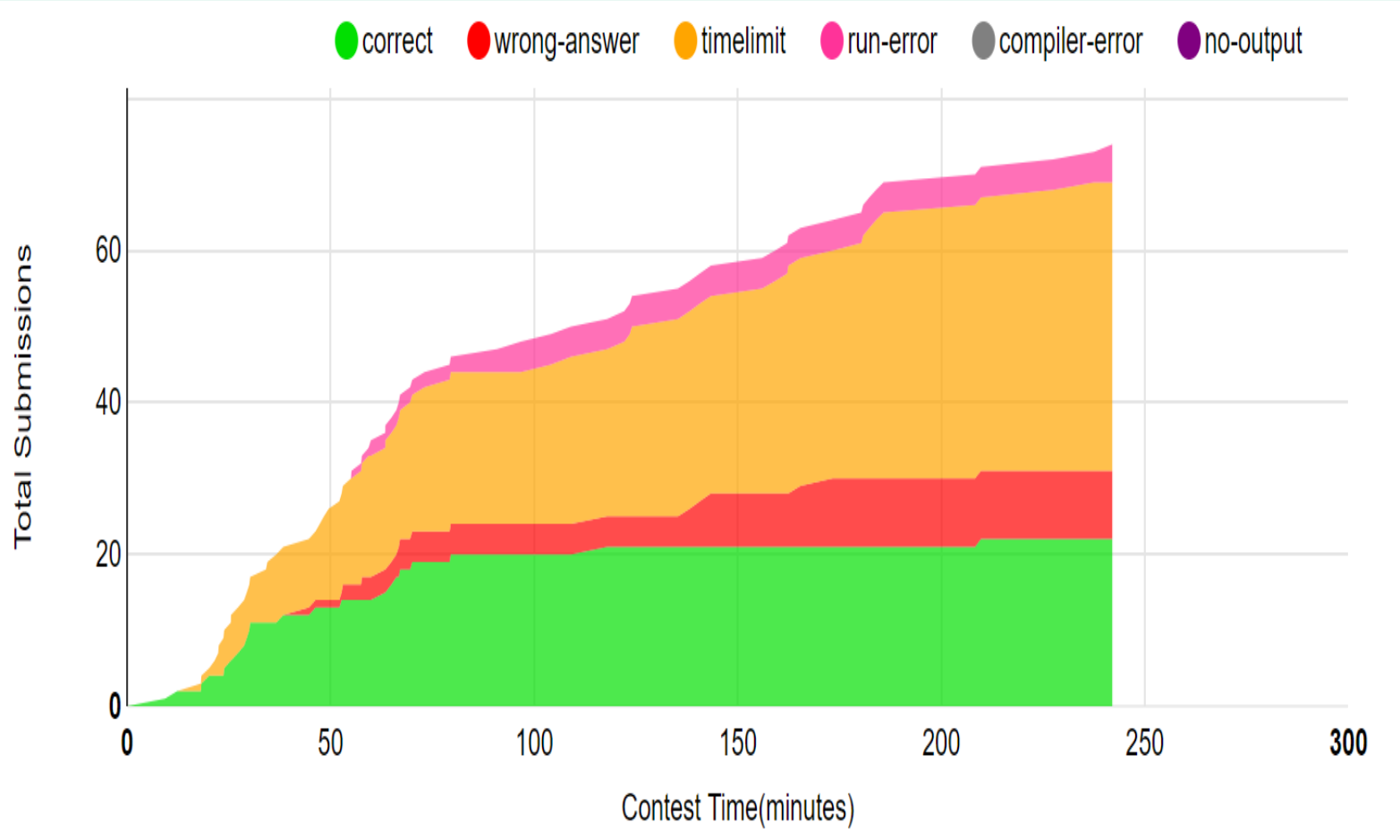
G. Julio: Torneos de Ajedrez

En la partida dada e4 e5 Dh5 Re7 D*e5# simplemente nos interesa el último movimiento con este si aparece ++ o # significa que se ganó una partida. En otro caso tablas. Con el número de jugadas si son pares significa que el último movimiento fue de negras, si son impares de blancas. No hace falta implementarlo completo.

I. Septiembre: Vuelta a la rutina

Envíos	Válidos	% éxito
74	22	30 %

I. Septiembre: Vuelta a la rutina



I. Septiembre: Vuelta a la rutina

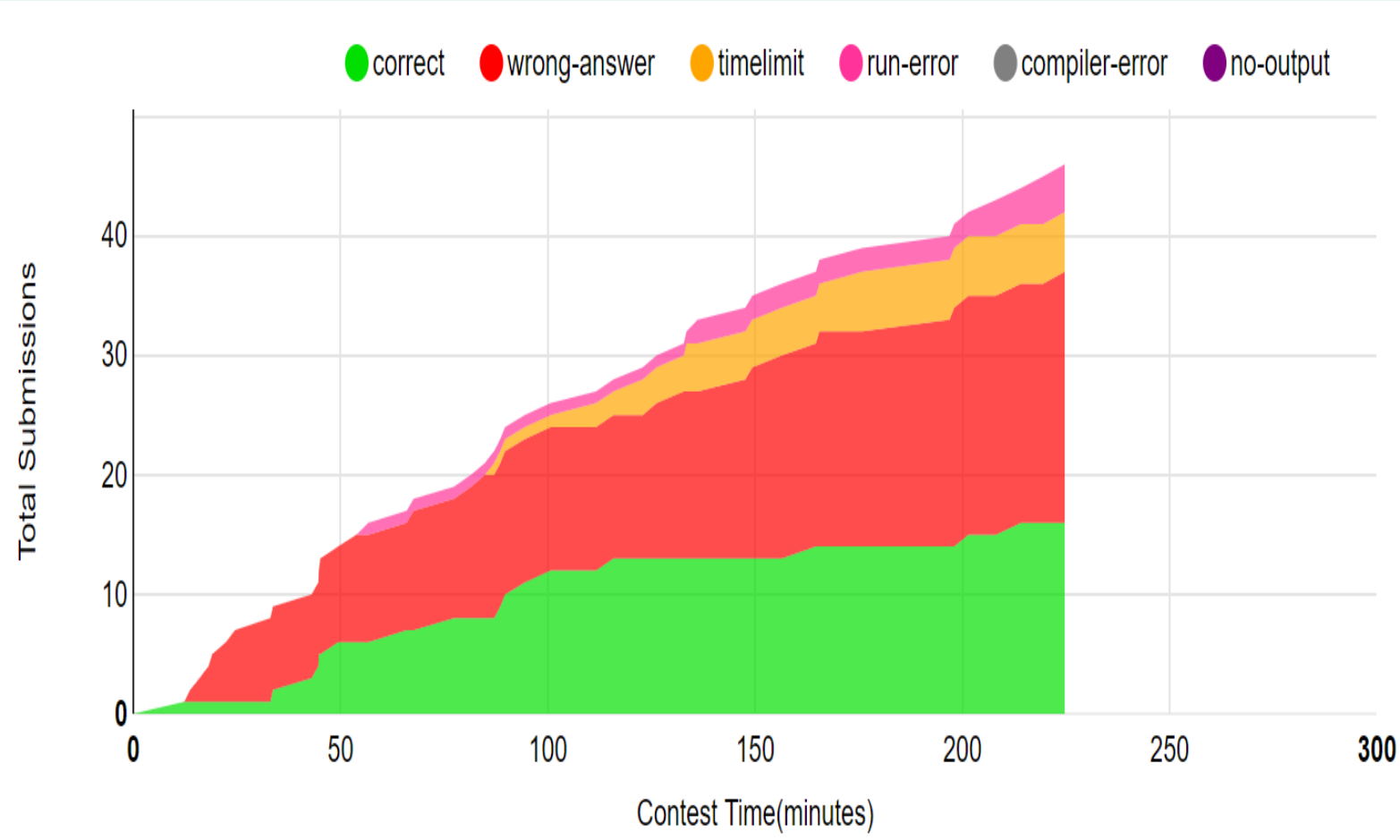
Se pide llevar la cuenta de las tareas pendientes por realizar. Complejidad esperada: O(N)

- Java: HashMap, HashSet
- C/C++: unorderedmap, unorderedset
- Python: Set, Diccionario

C. Marzo: La moda secundaria

Envíos	Válidos	% éxito
44	15	34 %

C. Marzo: La moda secundaria



C. Marzo: La moda secundaria

En este reto se nos pedía que dada una lista de elementos, escribiéramos cual es la moda de todo el conjunto de objetos.
Con una vuelta! No necesariamente era la primera moda la que nos piden!

C. Marzo: La moda secundaria

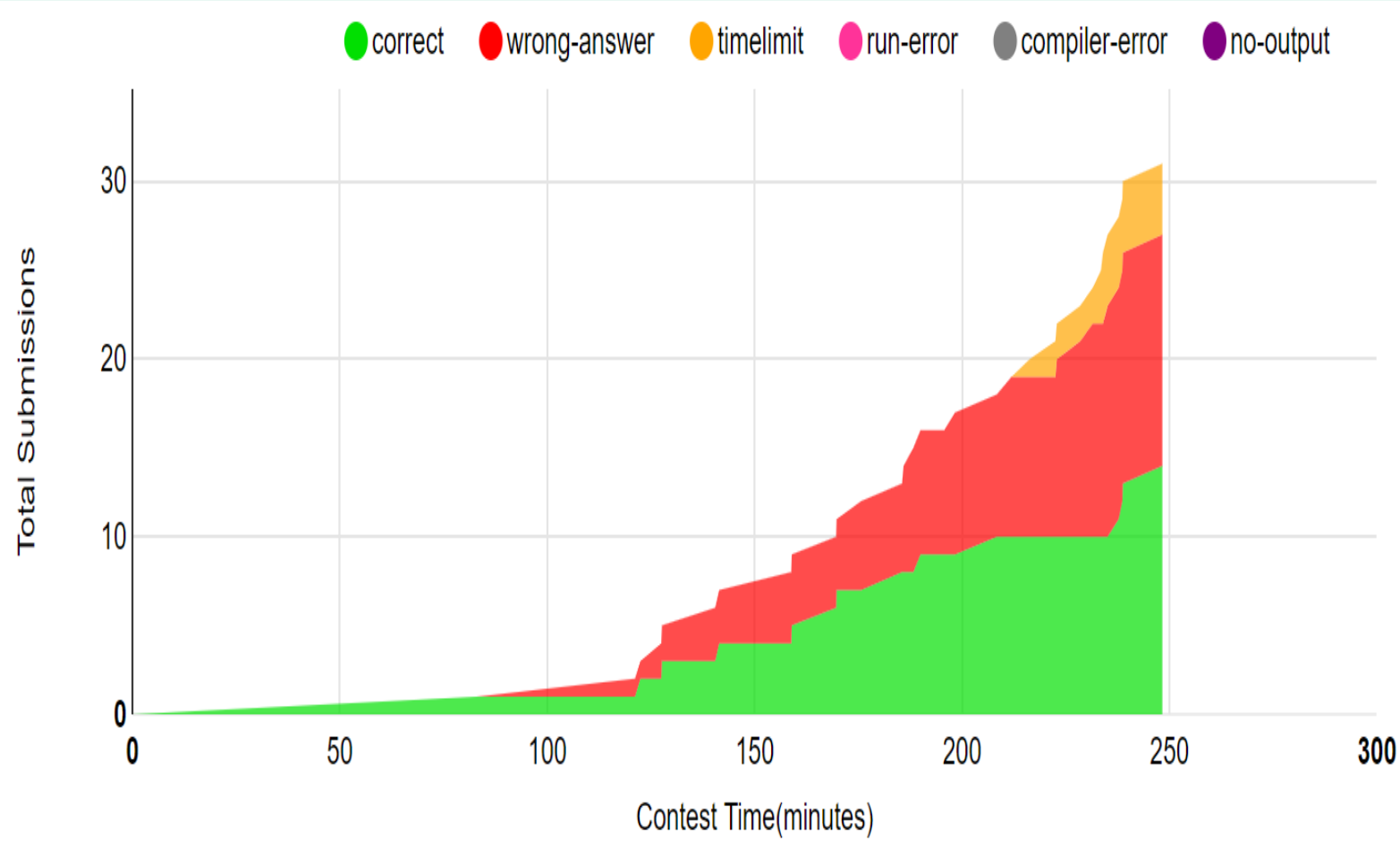
Basta con llevar en un mapa las apariciones de strings y, después de leer la entrada, en otro mapa, ver cuales elementos existen con un número de elementos X
Finalmente, pueden extraer el K-ésimo elemento de la moda y devolver la respuesta, por lo que la complejidad total del problema sería

$$O(NLgN)$$

K. Noviembre: La casa encantada

Envíos	Válidos	% éxito
19	10	52 %

K. Noviembre: La casa encantada



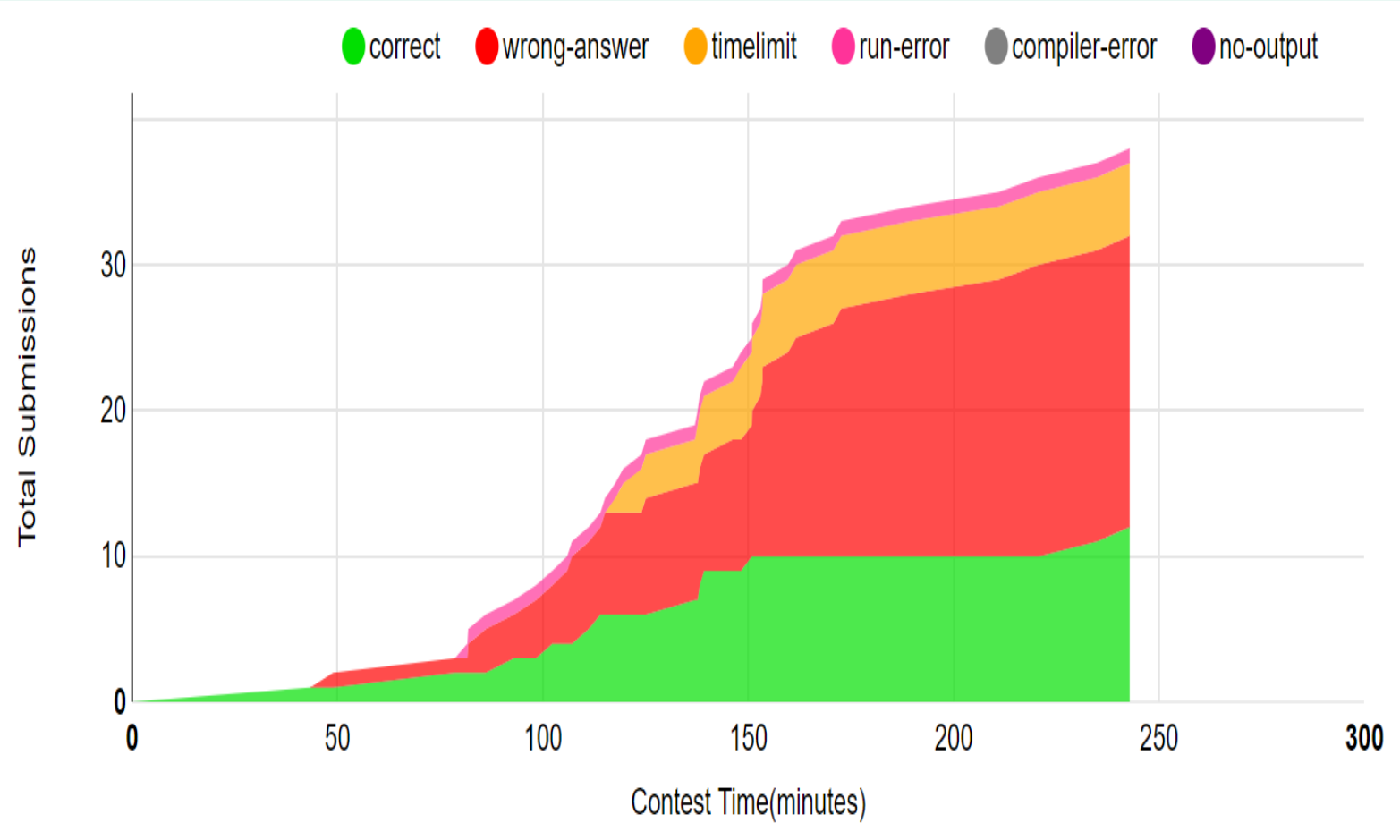
K. Noviembre: La casa encantada

Se puede implementar con un array de boolean para guardar si las habitaciones están cerradas y un array de enteros para guardar la habitación que se desde cada cuarto, ya que se asegura que las habitaciones estarán cerradas solamente una vez.
Es un problema de implementación. Alice y bob empiezan en extremos opuestos de un array. Se debe avanzar dos índices, empezando en los extremos, hacia el centro.
Es importante tener en cuenta que Alice ase moverá primero.

F. Junio: Rebajas de Verano

Envíos	Válidos	% éxito
30	10	33 %

F. Junio: Rebajas de Verano



F. Junio: Rebajas de Verano

El problema estaba puesto para engañar un poco a primera vista con las potencias de dos y hacer pensar en divide y vencerás u otras soluciones en O(N log(N)).
Pero si se piensa detenidamente, Gaben puede comprar todos los juegos menos uno con el mismo descuento.
Al darse cuenta de esto es suficiente con dos cosas:

1. Sumar todos los valores
2. Encontrar el valor mínimo (el juego que debe comprarse sin el descuento).

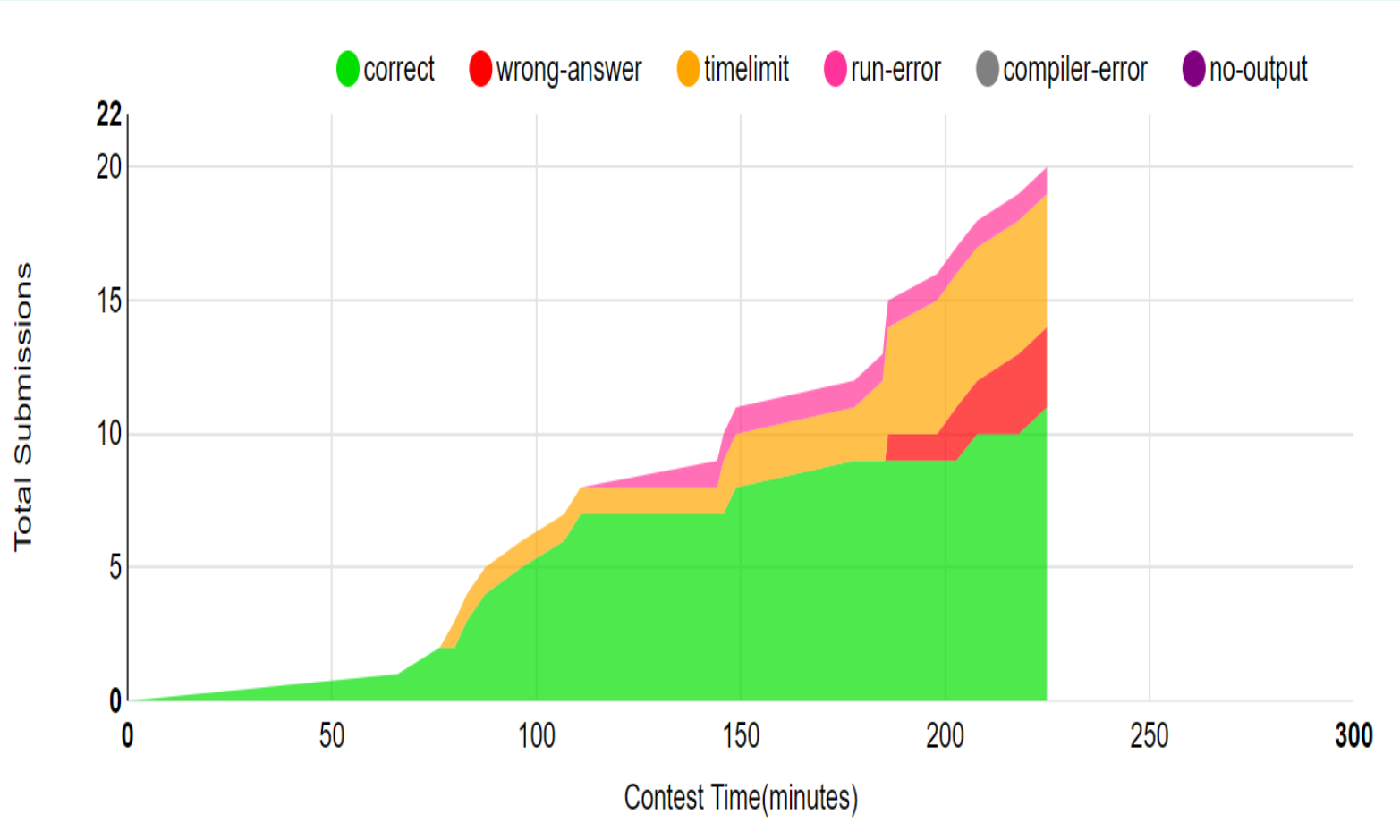
F. Junio: Rebajas de Verano

Solamente queda restarle el menor a la suma, aplicar el descuento y volver a sumar el mínimo.
Hay que tener cuidado con la precisión, float puede dar problemas. Operar con double o pasar a enteros da el resultado correcto.

B. Febrero: En el último minuto

Envíos	Válidos	% éxito
13	3	30 %

B. Febrero: En el último minuto



B. Febrero: En el último minuto

La solución a este problema se corresponde con un problema bien conocido: el Traveling Salesman Problem (TSP). Este problema consiste en que, partiendo de un punto (en este caso, la casa de David) visite todos los demás (las tiendas) y regrese al punto de partida.

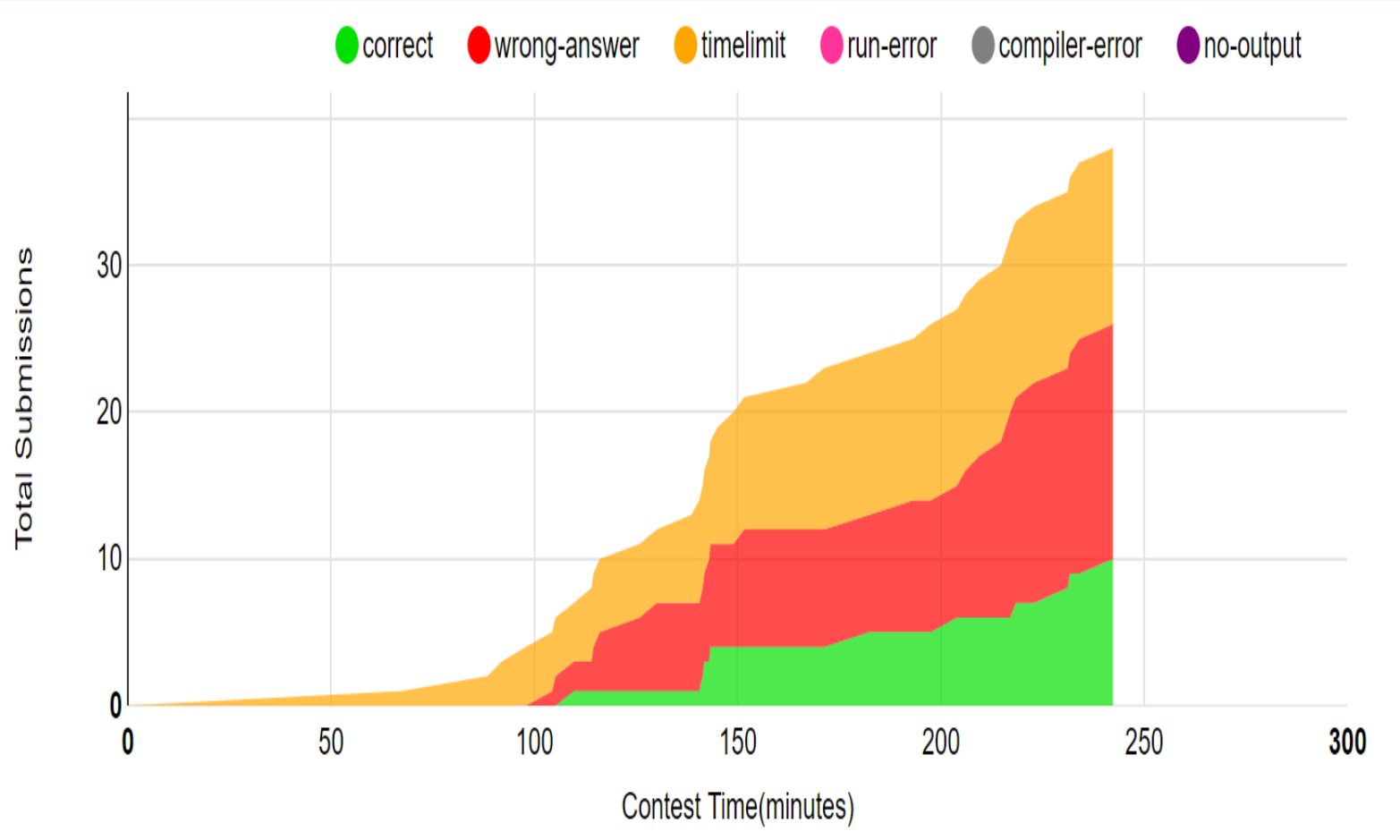
B. Febrero: En el último minuto

Un detalle importante a tener en cuenta es que la casa de David (el origen, N=0) **no es una tienda** y siempre debe partirse desde ella.
Al ser un ciclo, un enfoque de backtracking resuelve el problema. También se puede utilizar una aproximación de DP + Máscaras de bits.

D. Abril: Declaración de la renta

Envíos	Válidos	% éxito
30	6	20 %

D. Abril: Declaración de la renta



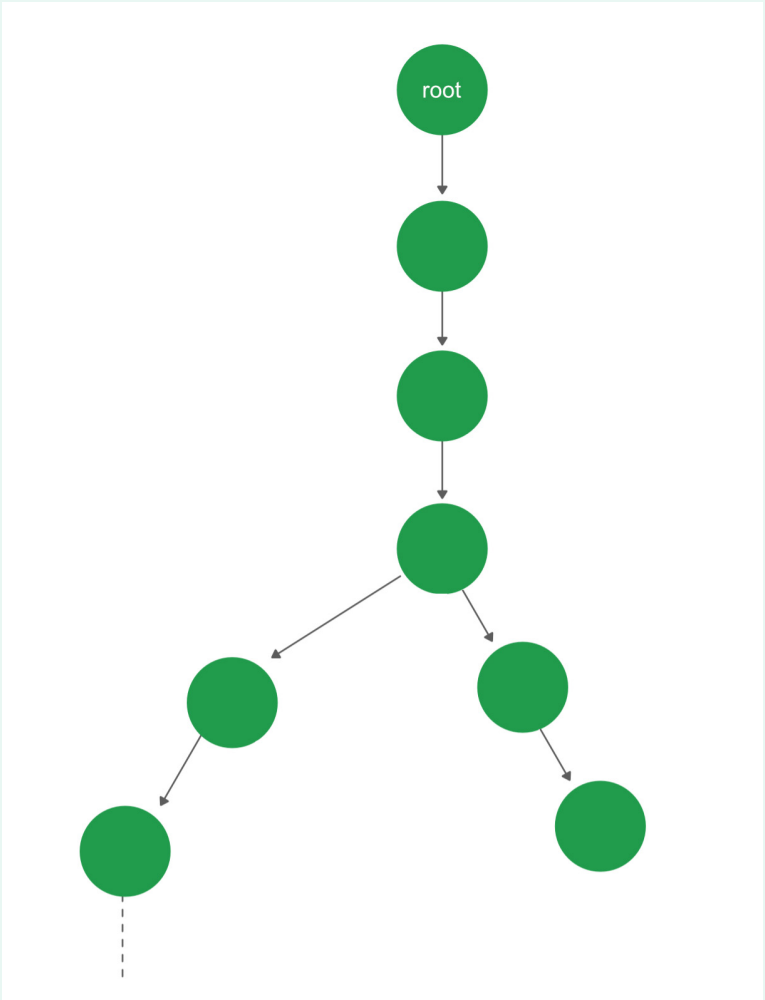
D. Abril: Declaración de la renta

Determinar según la apareciencia el número de prefijos existentes. Si

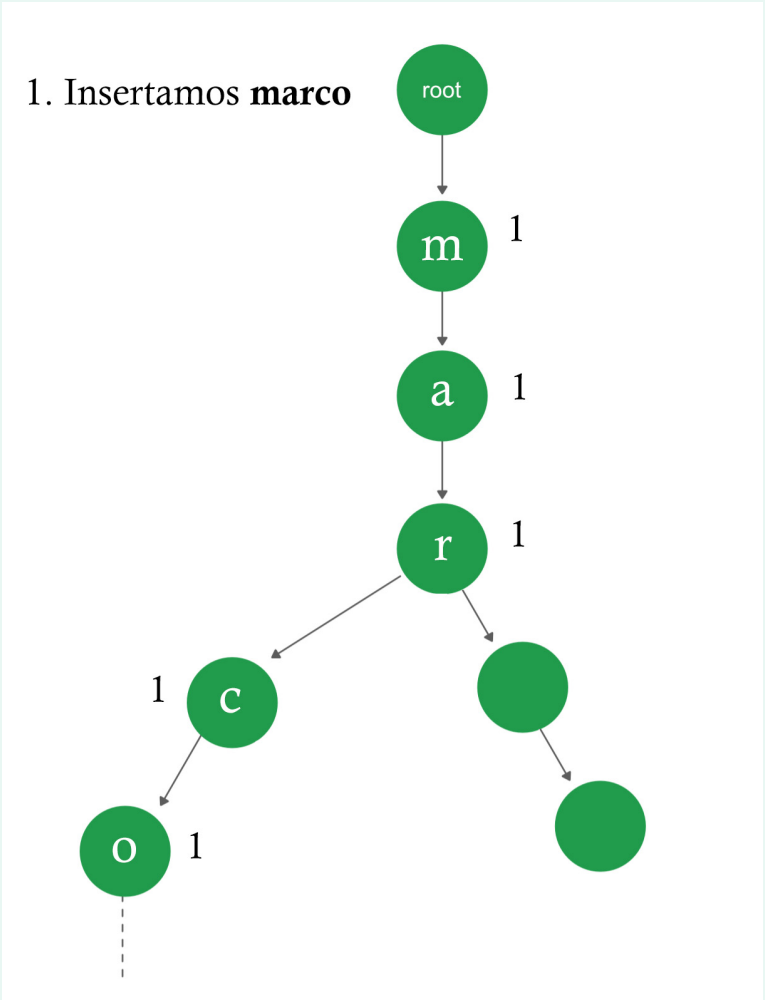
queremos comprobar si existe un prefijo en $O(N * Nombre)$ = TLE. Comprobar en el Trie si existe un prefijo es $O(Nombre)$ = AC ¿Qué es un Trie?

D. Abril: Declaración de la renta

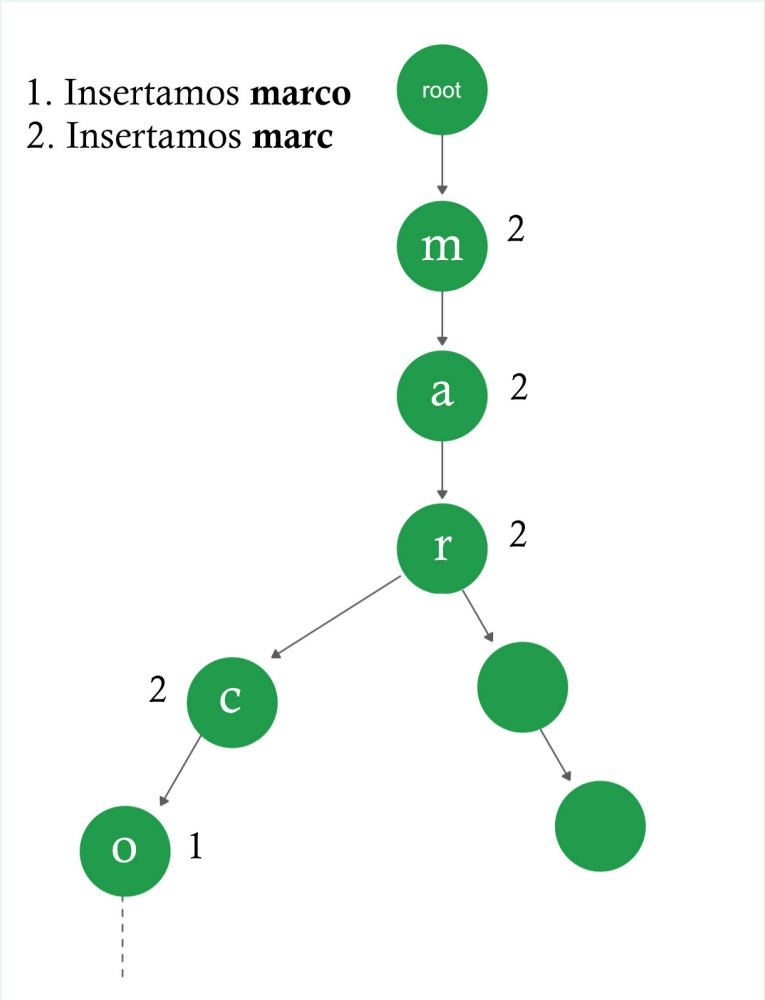
Problema base -> PHONELST en SPOJ



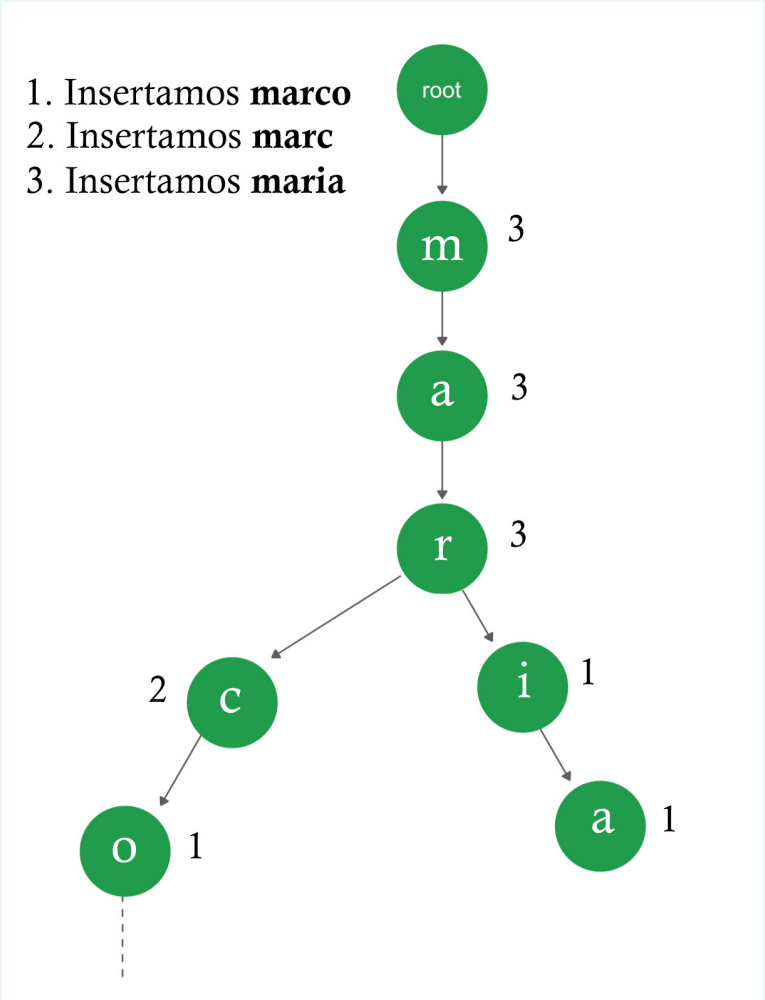
D. Abril: Declaración de la renta



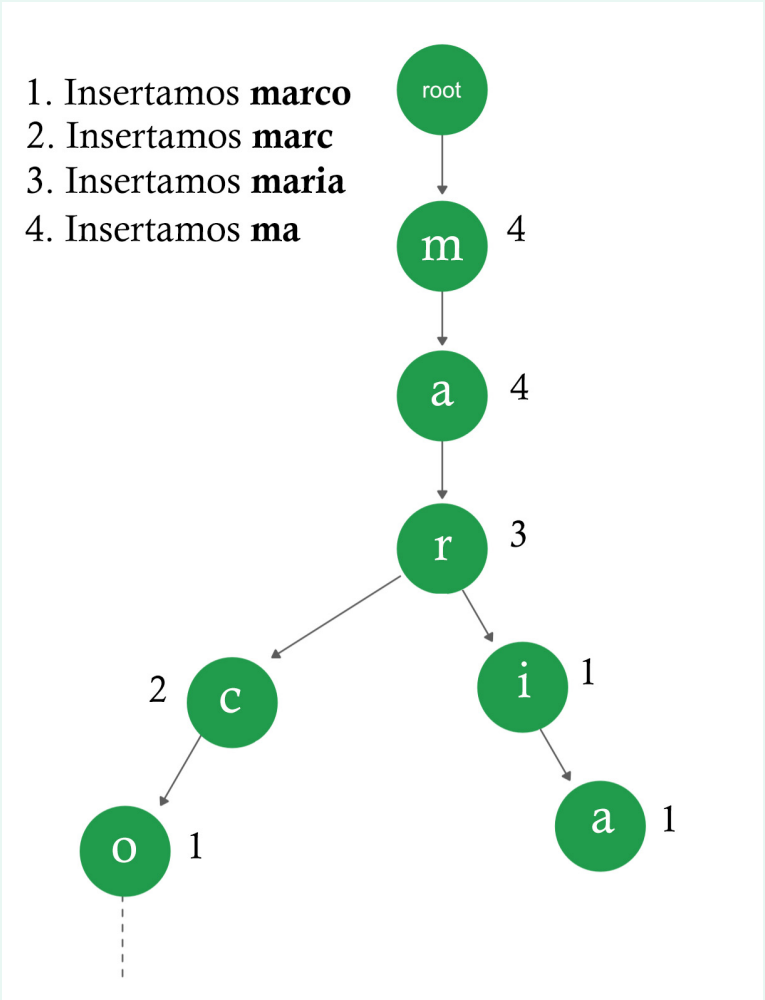
D. Abril: Declaración de la renta



D. Abril: Declaración de la renta



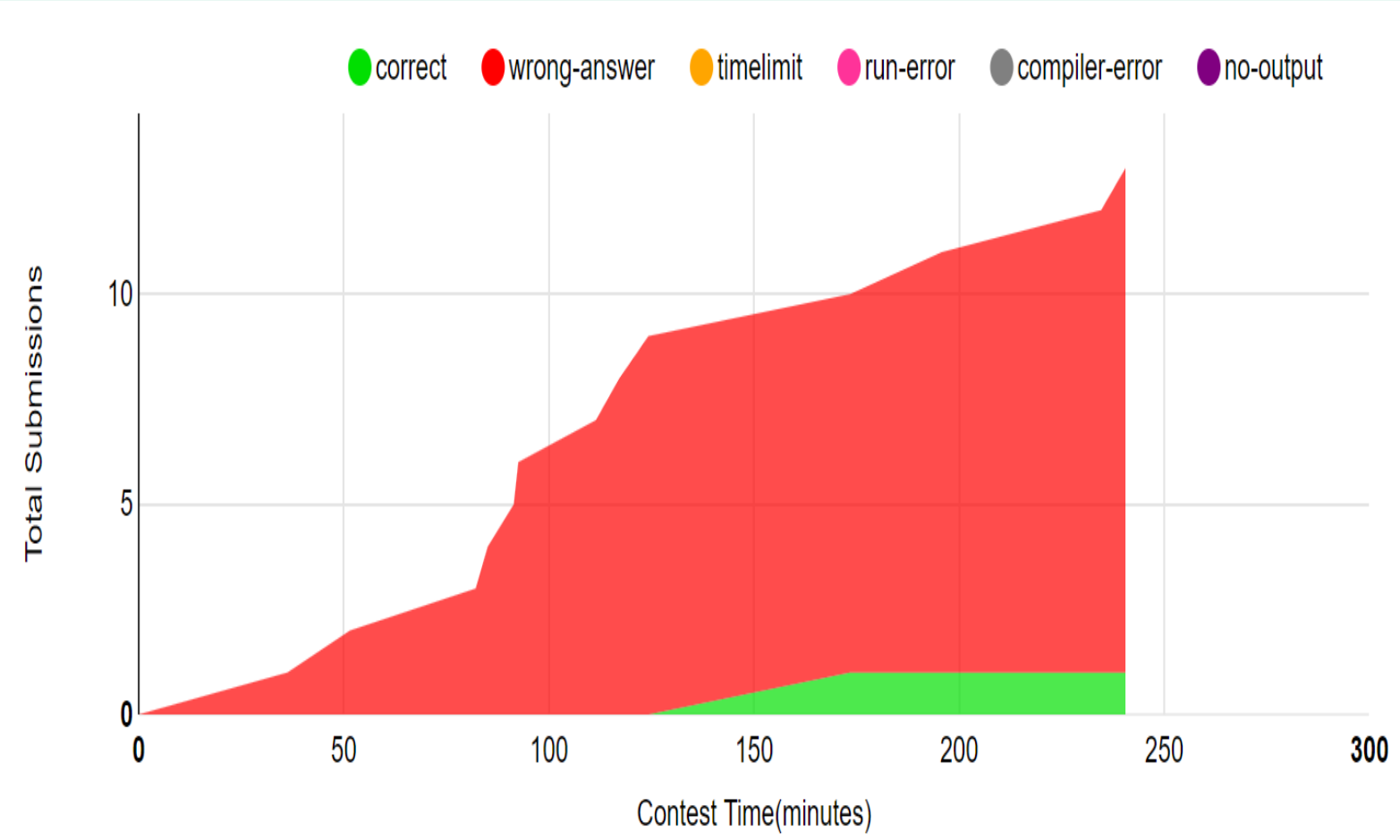
D. Abril: Declaración de la renta



L. Diciembre: Dora la Aisladora

Envíos	Válidos	% éxito
12	1	8 %

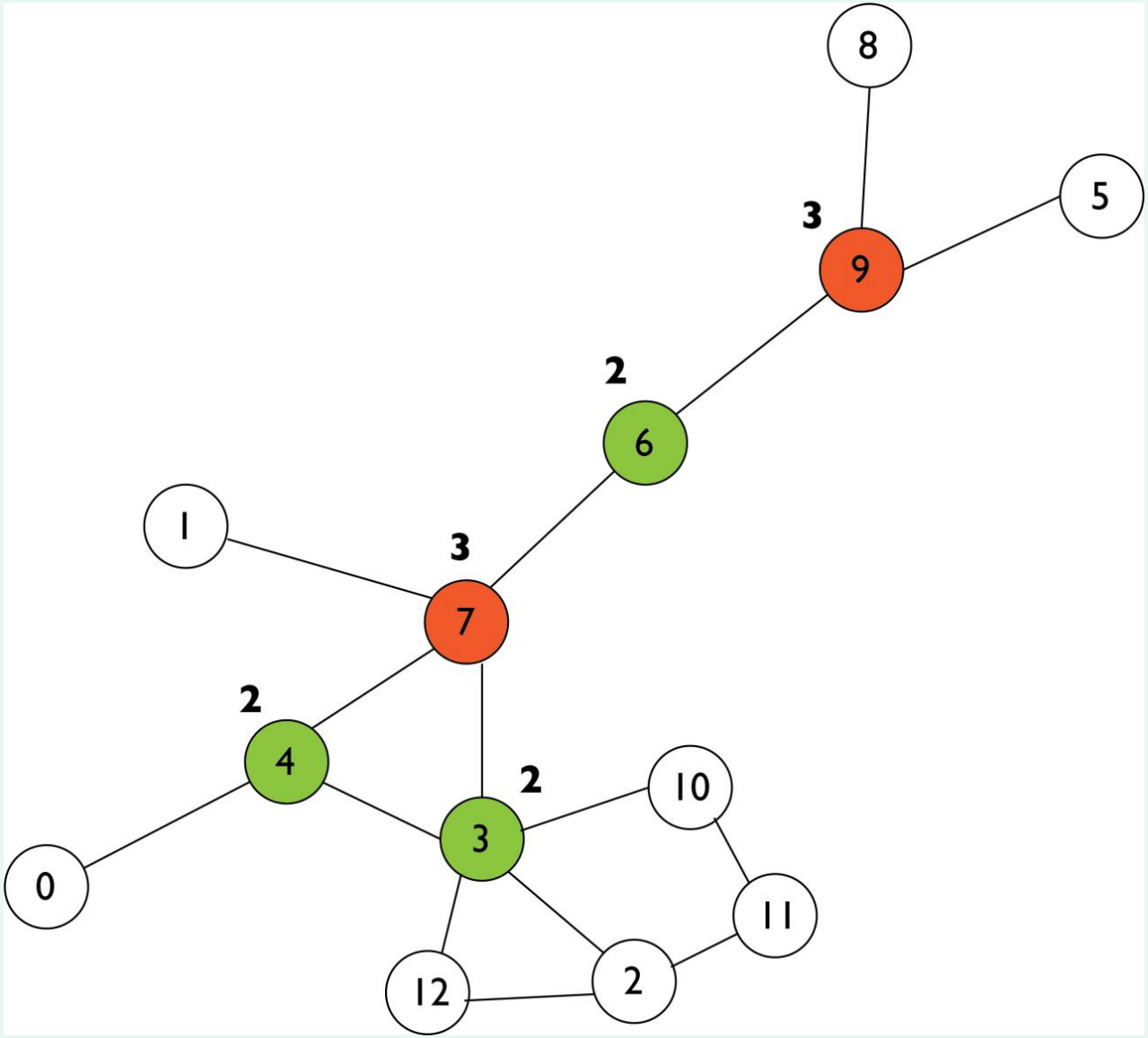
L. Diciembre: Dora la Aisladora



L. Diciembre: Dora la Aisladora

La solución a este problema se corresponde con dos de los problemas típicos dentro de los problemas de grafos: la identificación de puntos de articulación y la extracción de componentes conextas.
La idea principal consiste en identificar los puntos de articulación de un grafo y calcular cuántas componentes conexa tendría el grafo resultante cuando eliminamos cada punto.
Sin embargo, no todas las soluciones entran en tiempo.

L. Diciembre: Dora la Aisladora



L. Diciembre: Dora la Aisladora

Una posible solución que se nos puede ocurrir es solucionar este problema en dos pasos:

1. Calculamos todos los puntos de articulación de un grafo.
2. Por cada punto de articulación:
   1. Extraer del grafo el punto de articulación.
   2. Calcular las componentes conexas resultantes.
   3. Almacenar por cada punto de articulación, el número de componentes conexas generadas.
3. Devolver aquella solución que maximiza el número de componentes conexas, o en caso de que hubiera más de dos puntos de articulación devolver el punto con menor id.

Sin embargo esta solución no entra en tiempo ya que tiene complejidad $\mathcal{O}(V \cdot (V + E))$.

L. Diciembre: Dora la Aisladora

Otra posible opción es identificar puntos de articulación haciendo DFS. u será un punto de articulación si:

- Es la raíz del árbol DFS y tiene, al menos, dos hijos.
- No es la raíz y tiene un hijo, v, tal que no hay ningún vértice en el subárbol con raíz en v que tiene una arista de vuelta a uno de los ancestros de u.

Después se eliminan cada punto de articulación y se extraen componentes conexas... pero tampoco entra en tiempo.

L. Diciembre: Dora la Aisladora

La solución correcta consiste en aplicar el algoritmo de puntos de articulación anterior, y sobre la marcha calcular cuántas componentes conexas dejaría. Cada vez que un nodo se detecta como punto de articulación, deja una componente conexa más.

A. Enero: Dickie está en aprietos

Envíos	Válidos	% éxito
3	1	33 %

A. Enero: Dickie está en aprietos

Este problema se resuelve con cadenas de Markov. Las cadenas de Markov son tipo especial de proceso estocástico discreto en el que la probabilidad de que ocurra un evento depende solamente del evento inmediatamente anterior. En el problema, cada estado se traduce en una ubicación o intersección entre calles. De esta manera, se define una matriz de transición entre estados A, i.e., la probabilidad que Dickie tome una calle que conecta dos ubicaciones; y M vectores de condiciones iniciales, i.e., las ubicaciones de las casas de los amigos de Dickie, donde el vector v_i se representa como un vector con ceros en todas sus posiciones excepto en la posición del amigo i donde se asigna un 1.

A. Enero: Dickie está en aprietos

Para cada amigo i, las probabilidades, representadas en un vector p, de estar en cualquier intersección luego de K minutos y comenzando en la casa del este amigo i se calcula mediante:

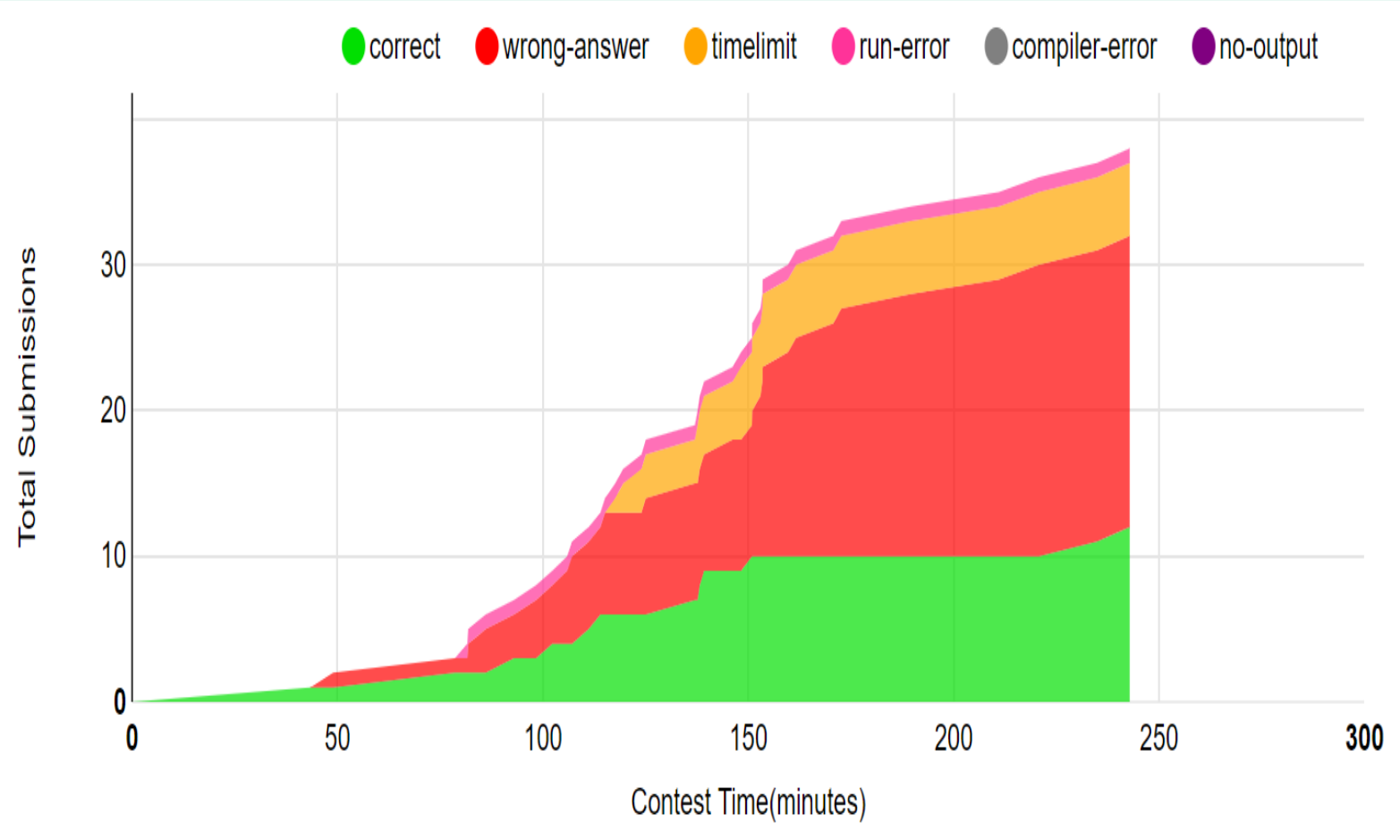
$$p = A^K \, v_i$$

Por lo tanto, la probilidad de llegar a la ubicación N desde la ubicación i es p_N. Por último, dado que la multiplicación de matrices es $O(N^3)$ y hay que exponenciar K veces, se usa exponenciación rápida para bajar la complejidad total del algoritmo de $O(N^3 \, T)$ a $O(log_2 \, (T)N^3)$.

- E. Mayo: Dickie y las hamburguesas.

Envíos	Válidos	% éxito
20	0	0 %

E. Mayo: Dickie y las hamburguesas.



E. Mayo: Dickie y las hamburguesas.

Sea r_i el ratio v_i/w_i, es posible afirmar que la felicicidad α siempre estará comprendida entre el mínimo y máximo ratio de entre todas las hamburguesas. Tomando en cuenta esto, se puede hacer búsqueda binara sobre el valor de la felicidad α y comprobar si es posible alcanzar esa felicidad comiendo K hamburguesas. Sin embargo, la felicidad está expresada en términos de todo el conjunto de hamburguesas y no en términos de cada hamburguesa. De esta forma, es necesario buscar cuánto aporta cada hamburguesa tomando en cuenta que conocemos α:

$$\alpha = \frac{\sum_{i=1}^{k} v_i}{\sum_{i=1}^{k} w_i}$$

$$0 = \frac{\sum_{i=1}^{k} v_i}{\sum_{i=1}^{k} w_i} - \alpha$$

$$0 = \sum_{i=1}^{k} v_i - \sum_{i=1}^{k} w_i \cdot \alpha$$

$$0 = \sum^{k} v_i - w_i \cdot \alpha$$

E. Mayo: Dickie y las hamburguesas.

$$0 \leqslant \sum_{i=1}^{k} v_i - w_i \cdot \alpha$$

Listing 1: Boceto de la solución

```
def isHappinessPossible(alpha, burgers):
    happiness = [ burguer.v - burger.w*alpha for burge
    happiness.sort(reverse=True)
    return sum(happiness[:k]) >= 0

def max_happiness(burgers, k, min_ration, max_ration):
    l = min_ratio, r = max_ratio
    for i in range(50):
        mid = (l + r) / 2.0
        if happiness(mid, burgers, k):
            l = mid
        else:
            r = mid
    return l
```

H. Agosto: HH y el Bosón

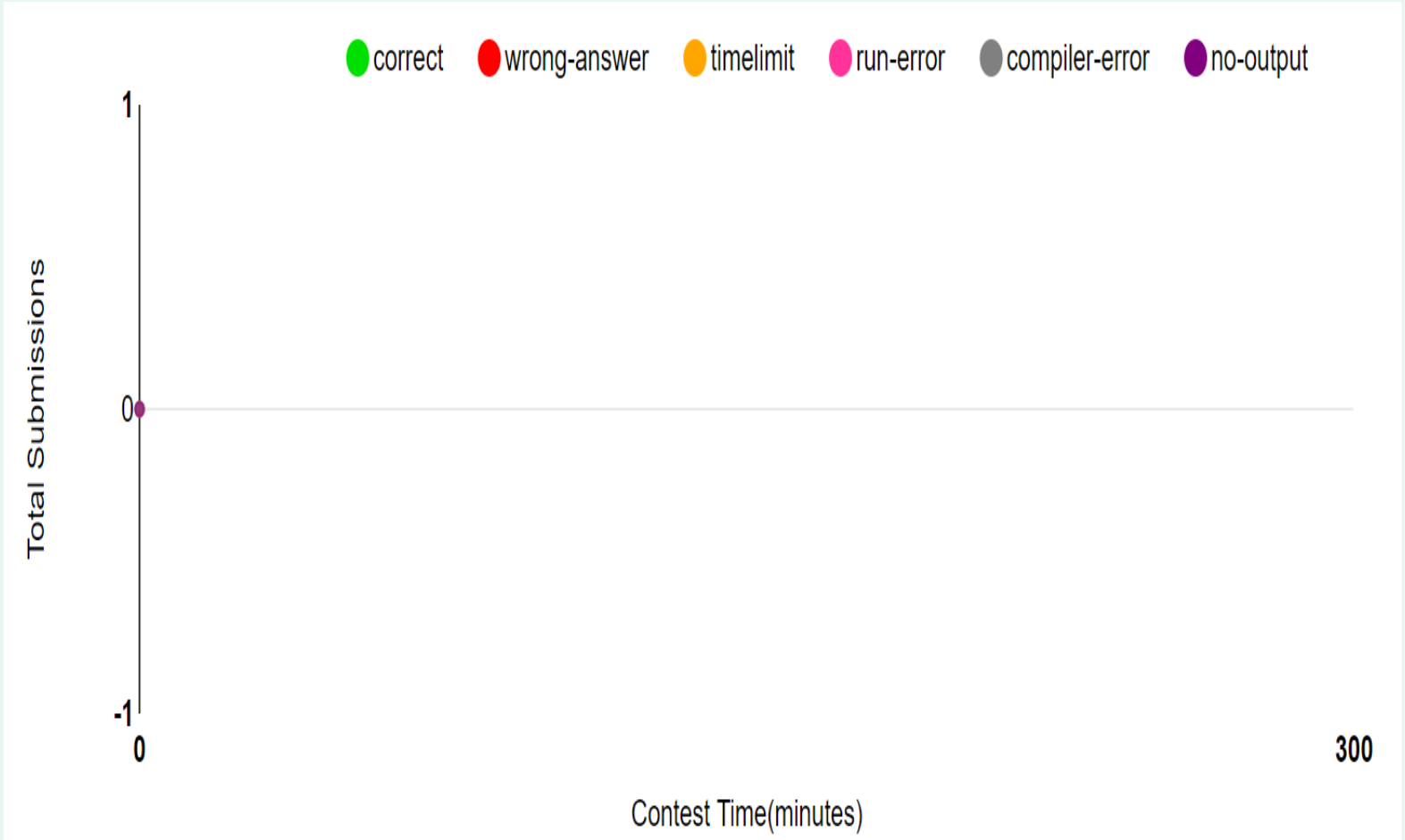
Envíos	Válidos	% éxito
5	0	0 %

El recorrido de cada bosón i se puede traducir en una ecuación, donde el tiempo que toma llegar al ciclo es b_i, la longitud del ciclo es c_i y la cantidad de veces que hace un ciclo es a_i. Si colocamos juntas todas estas ecuaciones, se puede formar un sistema de congruencias simultáneas. Dado que las longitudes de los tubos son números coprimos entre sí, se puede aplicar el teorema del resto chino para resolver este sistema de congruencias.

J. Octubre: Recogiendo Caramelos

Envíos	Válidos	% éxito
0	0	0 %

J. Octubre: Recogiendo Caramelos



J. Octubre: Recogiendo Caramelos

Se nos pedía el número máximo de caramelos que podían extraer Maxx y sus amigos de las casas en noche de Halloween.
Podían ir solo a casas crecientes a pedir caramelos, podían "devolverse" a casas de sus amigos e ir a otro sitio.

J. Octubre: Recogiendo Caramelos

Cosas a tomar en cuenta: Los bolsillos tenían "capacidad infinita", por lo que el rebasamiento de caramelos en una bolsa se podía llevar en los bolsillos, sin embargo, al vaciar la bolsa, todo lo que tenían en los bolsillo se movía a la bolsa. Esto podría representarse como aritmética modular de $(A + B) \mod C$. Podemos darnos cuenta de que este problema es un problema recurrente y por ende, se puede resolver utilizando backtracking.
TLE

J. Octubre: Recogiendo Caramelos

Con los límites, podemos memoizar el índice de las casas, la capacidad de la bolsa y el tiempo que le queda a Maxx. Luego, por la naturaleza del problema, este problema se puede reducir a un problema de programación dinámica.

$$O(N^3T)$$

¿AÑO QUE VIENE?