Curso Programación Competitiva 2021

Estadísticas y Soluciones

Grafos II

Clasificación de los problemas

Problema	**Categoría**
A - Biodiversidad	Mapas y Bucles
B - El sueño de Elon Musk	Grafos, Dijkstra
C - Enamorados a distancia	Grafos, BFS y DFS
D - Torneos de Ajedrez	Pensar, condicionales
E - Ataque de los Titanes	Grafos, Kruskal
F - Evitando lag	Grafos, Floyd Warshall

Estadísticas

Problema	**# casos de prueba**	**Espacio en disco**
A - Biodiversidad	13	12MB
B - El sueño de Elon Musk	3	2MB
C - Enamorados a distancia	14	15MB
D - Torneos de Ajedrez	16	3MB
E - Ataque de los Titanes	9	1MB
F - Evitando lag	4	1MB
- **Total**	59	33MB (+-)

Estadísticas*

Problema	Primer equipo en resolverlo	Tiempo
A - Biodiversidad	x.liu.2020	13’
B - El sueño de Elon Musk	¿?	¿?
C - Enamorados a distancia	x.liu.2020	36’
D - Torneos de Ajedrez	s.salazarc.2018	7’
E - Ataque de los Titanes	s.salazarc.2018	30’
F - Evitando lag	s.salazarc.2018	15’

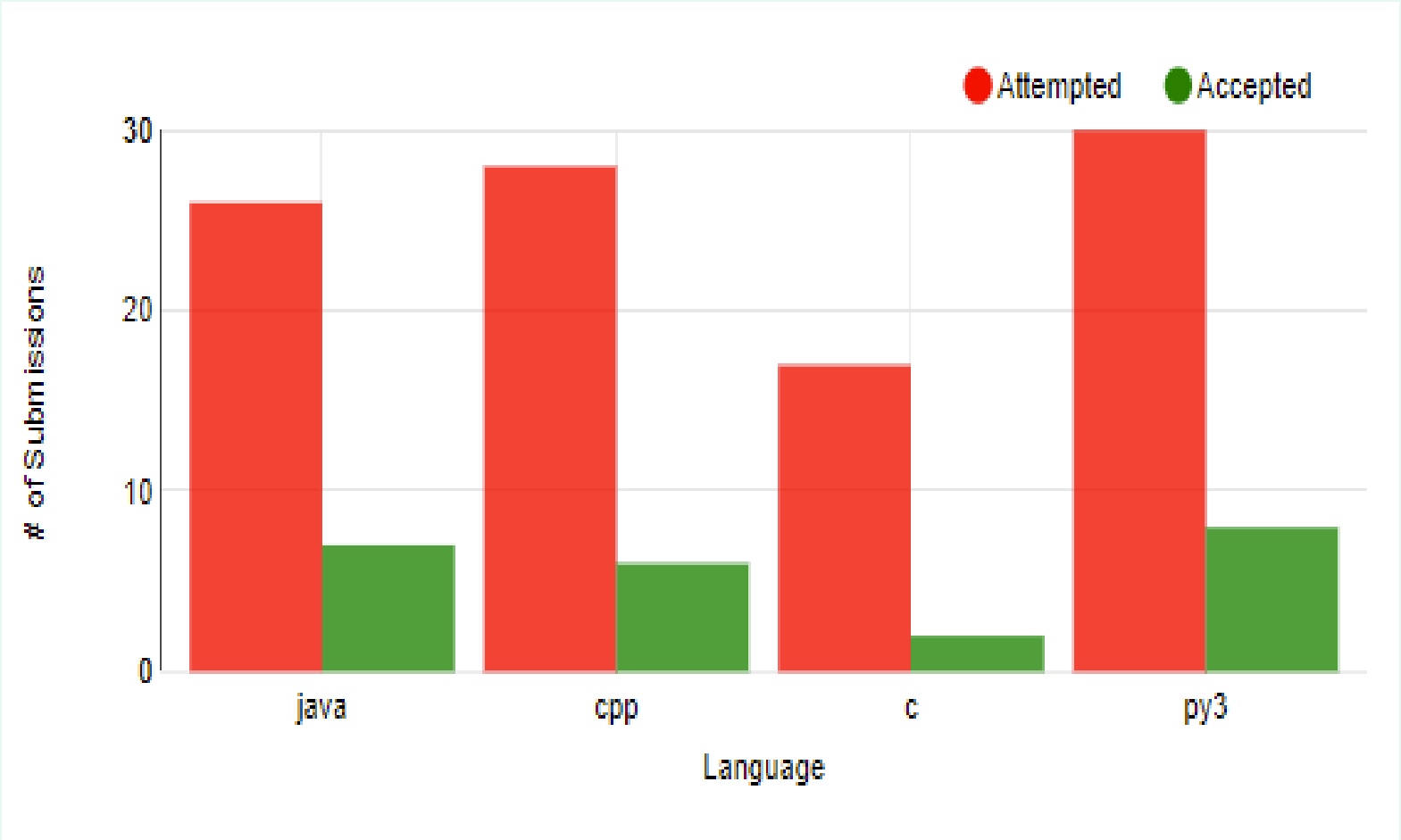
*Antes de congelar el marcador.

Estadísticas*

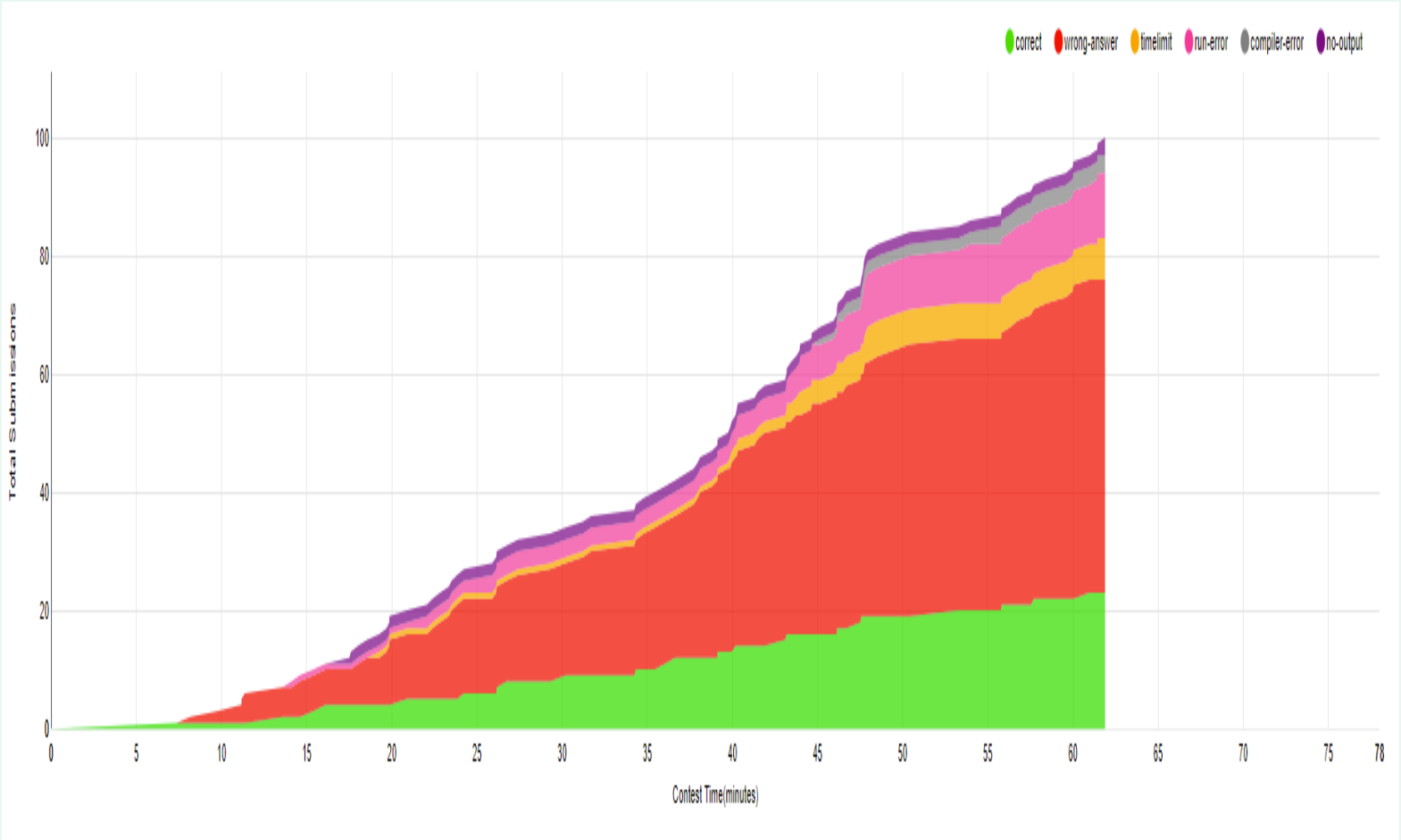
Problema	Envíos	Válidos	% éxito
A - Biodiversidad	38	7	18 %
B - El sueño de Elon Musk	11	0	0 %
C - Enamorados a distancia	2	2	100 %
D - Torneos de Ajedrez	12	6	50 %
E - Ataque de los Titanes	2	2	100 %
F - Evitando lag	7	2	28 %

*Antes de congelar el marcador.

Estadísticas varias



Estadísticas varias



A. Biodiversidad

Envíos	Válidos	% éxito
38	7	18 %

A. Biodiversidad

El problema nos pide que midamos la biodiversidad de un hábitat y demos como resultado la especie que más predomina.

- La especie que más predomina está dada por el número de apariciones en la entrada
- Por cada especie podríamos ver cuántas veces aparece en el input. Pero si cada especie resulta ser única y tenemos que iterar en todo el array, ¡Esto daría TLE!

A. Biodiversidad

Mejor idea:

- Guardamos en un mapa utilizando como clave el string de la especie y como valor el número de apariciones que tiene (si no existe en el mapa, tendría 0 apariciones)
- A medida que leemos la entrada, insertamos en el mapa o utilizamos el elemento que ya existía y sumamos 1
- Finalmente, iteramos sobre el mapa y si el valor del elemento actual es mayor al mejor elemento que hemos encontrado, actualizamos
- ¡Cuidado! Añadimos una restricción adicional para los casos donde haya más de un elemento con el mismo número de elementos
- ¡AC!

D: Torneos de Ajedrez

Envíos	Válidos	% éxito
12	6	50 %

D: Torneos de Ajedrez

¿Problema de implementación completa?

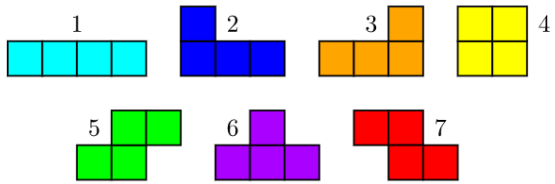
Altura final en Tetris

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:



D: 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?.

D: 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.

E. Ataque de los Titanes

Envíos	Válidos	% éxito
2	2	100 %

E. Ataque de los Titanes

En la serie de Attack On Titan los protagonistas se mueven utilizando un Equipo de Maniobras Tridimensionales, y para ello necesitan utilizar bombonas de gas que impulsen el equipo.

Para poder desplazarse sin utilizar gas, nos piden utilizar líneas de teleférico entre cada puesto de avanzadilla, pero no podemos tirar cable entre cualquier par de puestos, y queremos que la red de teleféricos sea lo más corta posible.

E. Ataque de los Titanes

- Una primera idea podría ser utilizar Dijkstra para saber cuál es la distancia más corta entre cada par de puestos y tirar cable en esas rutas.
- Sin embargo, esto no garantiza que la red de carreteras final sea la de menor coste.

E. Ataque de los Titanes

- Enfoque correcto: Kruskal / Prim para generar un MST de manera que garanticemos minimizar el coste total, que se corresponde con los metros de cable que necesitaremos.
- Finalmente necesitaremos calcular el coste sabiendo que cada 5m de cable cuesta 1€ y que el cable se compra en packs de 5m.

F. Arreglando el lag

Envíos	Válidos	% éxito
7	2	28 %

F. Arreglando el lag

Tenemos un grafo en el que cada nodo es un ordenador y cada arista la latencia entre los nodos que conecta. Objetivo: Contar cuantas rutas entre cualquier par de nodos superan la latencia máxima.

F. Arreglando el lag

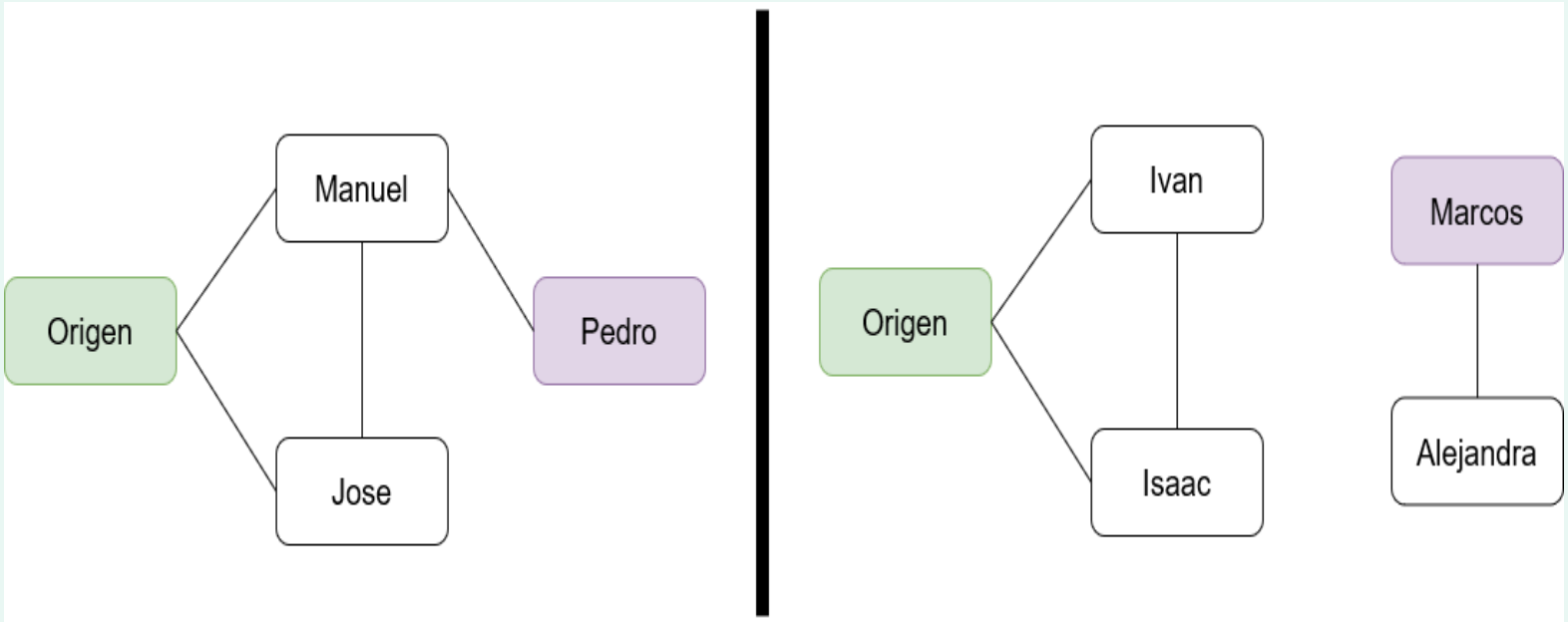
Complejidad máxima esperada: $O(n^3)$ (por el tamaño de la entrada) Idea: 1. Calcular distancias entre cada par de nodos usando: - N Dijkstras - Floyd-Warshall 2. Crear un segundo grafo, mismos nodos, arista entre dos nodos si la ruta más corta entre 3. Solución: Aristas máximas - Aristas del grafo Donde aristas máximas: $N * (N-1)/2$

C. Enamorados a distancia

Envíos	Válidos	% éxito
2	2	100 %

C. Enamorados a distancia

Conociendo todas las amistades de la red social, debemos comprobar si somos capaces de localizar al chico de nuestros sueños o no.
Podemos verlo fácilmente a partir de los dos ejemplos del enunciado:



C. Enamorados a distancia

Lo que debemos hacer es comprobar si existe en el grafo un camino desde nosotros mismos hasta el chico de nuestros sueños. Esto se puede hacer a través de una búsqueda en anchura (BFS) o una búsqueda en profundidad (DFS).
Es importante tener en cuenta cómo viene la entrada:

1. Los nodos del grafo vienen como Strings, debemos convertirlas a números a través de un mapa.
2. La entrada indica cuáles son nuestros amigos: Estos son los puntos desde los que debemos lanzar el BFS / DFS.

B. El sueño de Elon Musk

Envíos	Válidos	% éxito
11	0	0 %

B. El sueño de Elon Musk

Elon Musk quiere quiere cumplir su sueño y demostrar al mundo que Star Wars es real. Los ingenieros de SpaceX han creado una nave capaz de llegar hasta los lugares más lejanos de la galaxia. Por otro lado, han calculado el combustible necesario para viajar desde un plantea hasta otro. ¿Será capaz de lograrlo?

B. El sueño de Elon Musk

La entrada:

- Número de casos a resolver
- Un grafo ponderado y no dirigido:
  - Vértices (planetas)
  - Aristas (ruta de un planeta a otro)
  - Peso de la arista (combustible)
- Plantea origen y planeta destino.

La salida:

- Combustible necesario para ir desde el planeta origen, hasta el planeta destino.

B. El sueño de Elon Musk

¿Cómo resolverlo?

Bae: Come over
Dijkstra: But there are so many routes to take and I don't know which one's the fastest
Bae: My parents aren't home
Dijkstra:

Dijkstra's algorithm

Graph search algorithm

Not to be confused with Dykstra's projection algorithm.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.[1][2]

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes,[2] but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

Dijkstra's algorithm