Soluciones Curso 27/3

Grafos: Representación, recorridos, componentes conexas…

¡Esperamos que os hayan gustado los problemas!

Por: Isaac Lozano, Raúl Martín, Iván Martín, Jakub Jan Luzcyn

Estadísticas de los problemas

Problema	Primer AC General	Primer AC (Solo alumnos del curso)
A: Enamorados a distancia	David Morán (6’)	Jorge Sendarrubias (10’)
B: Poniendo Baldosas	David Morán (44’)	-
C: Pagando la hipoteca	Alberto Maurel (8’)	Laura Medina (16’)
D: Esquivando al MamáRadar	Alberto Maurel (66’)	-
E: Benjamín Botones	Alberto Maurel (17’)	Jorge Sendarrubias (48’)

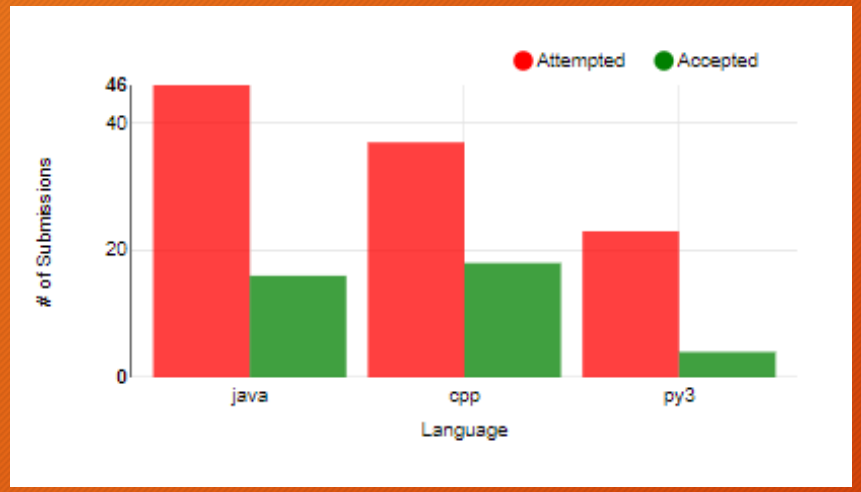
Estadísticas antes del Freeze

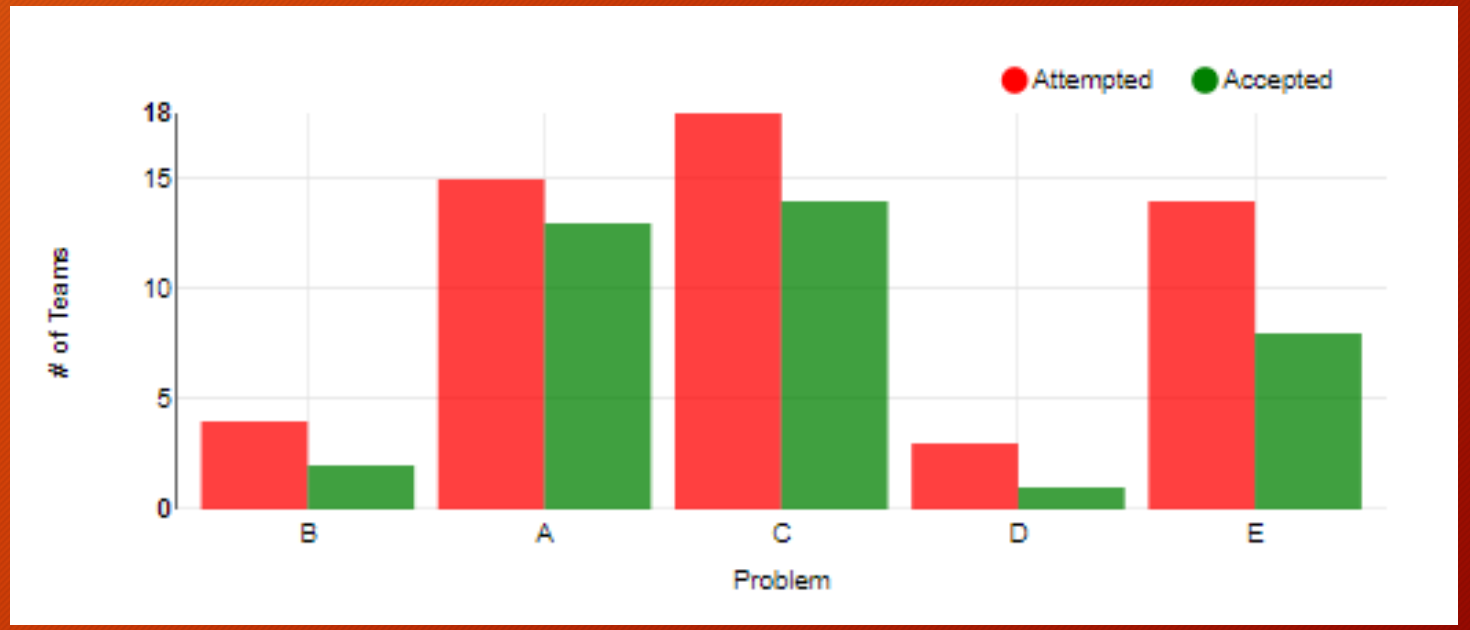
Estadísticas de los problemas

Problema	Casos de prueba	Ratio de ACs
A: Enamorados a distancia	14	13/27
B: Poniendo Baldosas	12	2/5
C: Pagando la hipoteca	22	14/44
D: Esquivando al MamáRadar	7	1/5
E: Benjamín Botones	3	8/22

Estadísticas antes del Freeze

Estadísticas de los problemas





Estadísticas antes del Freeze

Problema C

Pagando la hipoteca

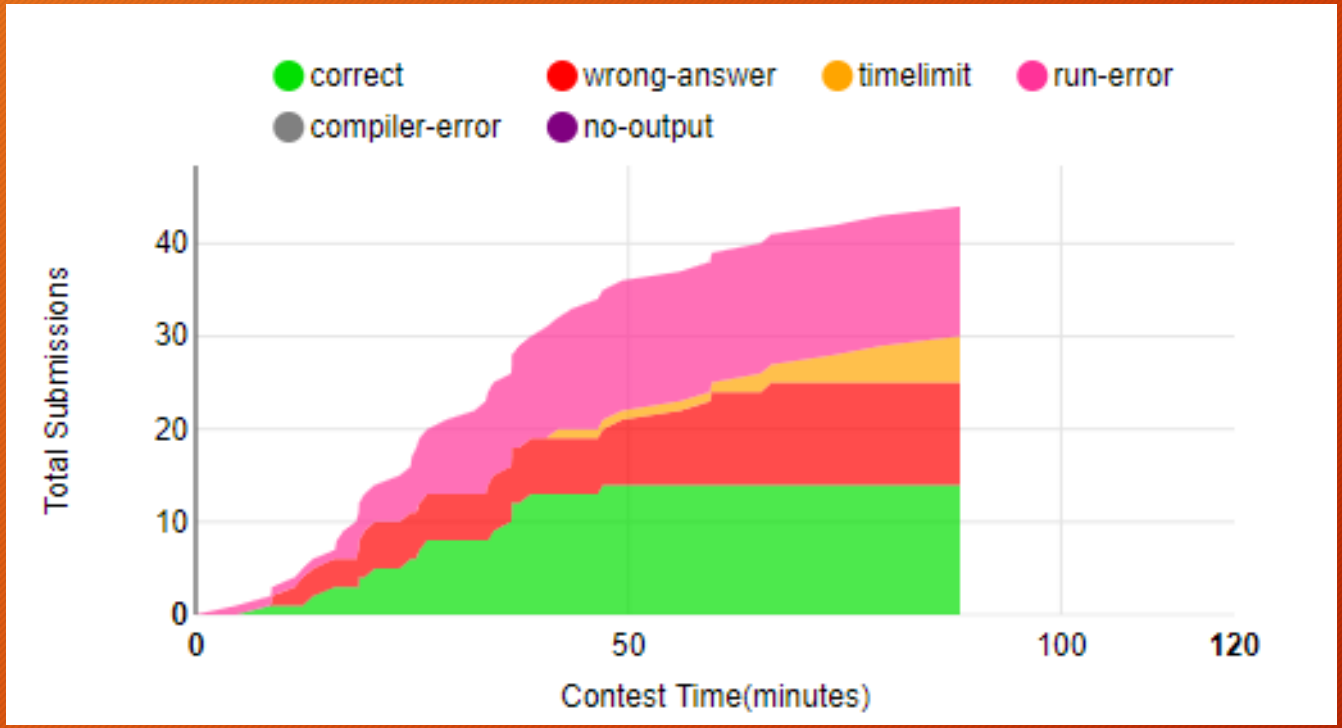
Autor: Iván Martín de San Lázaro

Primer AC general: Alberto Maurel (8’)

Primer AC alumno curso: Laura Medina (16’)

Pagando la hipoteca

C



Total (Antes del Freeze):

44 envíos
14 Accepted
11 Wrong Answer
14 RunTime Error
5 Timelimit Exceeded

Pagando la hipoteca

C

Uno de los problemas fáciles del concurso.

Dada una lista de productos y su precio, sumar cuanto es el total de los productos que llevamos en los bolsillos.

Para solucionarlo se utiliza un Mapa que relacione el nombre de cada producto con su precio. Tras sumar el total de los productos, sumamos los ahorros que ya teníamos, y ese es el dinero que tenemos.

La solución es la resta Hipoteca – Dinero.

Pagando la hipoteca

C

¡Ojo! Dos trucos del problema en los que habéis caído (Por eso los WA y RTE):

- (Hipoteca – Dinero) puede ser negativo si tenemos más dinero del que nos piden. En ese caso imprimimos 0, porque nos quedarían 0 bayas por pagar de la hipoteca, y tendríamos dinero sobrante
- Límites. Era necesario usar un tipo de dato que soportara 2^{63}, long en java y long long int en c/c++. En Python no hace falta nada especial.

¡¡Lo ponía en el ejemplo!! --------→

```
Crown 500000
LongBook 2000
Fossil 100000
Cherry 500
```

Pagando la hipoteca

C

Solución en Python (8 líneas):

```
mapa = dict()
for i in range(int(input())):
    linea = input().split()
    mapa[linea[0]] = int(linea[1])
objetos = int(input())
dinero = sum([mapa[input()] for i in range(objetos)]) + int(input())
hipoteca = int(input())
print(0 if hipoteca-dinero < 0 else hipoteca-dinero)
```

Yay! Hemos pagado la hipoteca de Munchi

Problema E

Benjamín Botones

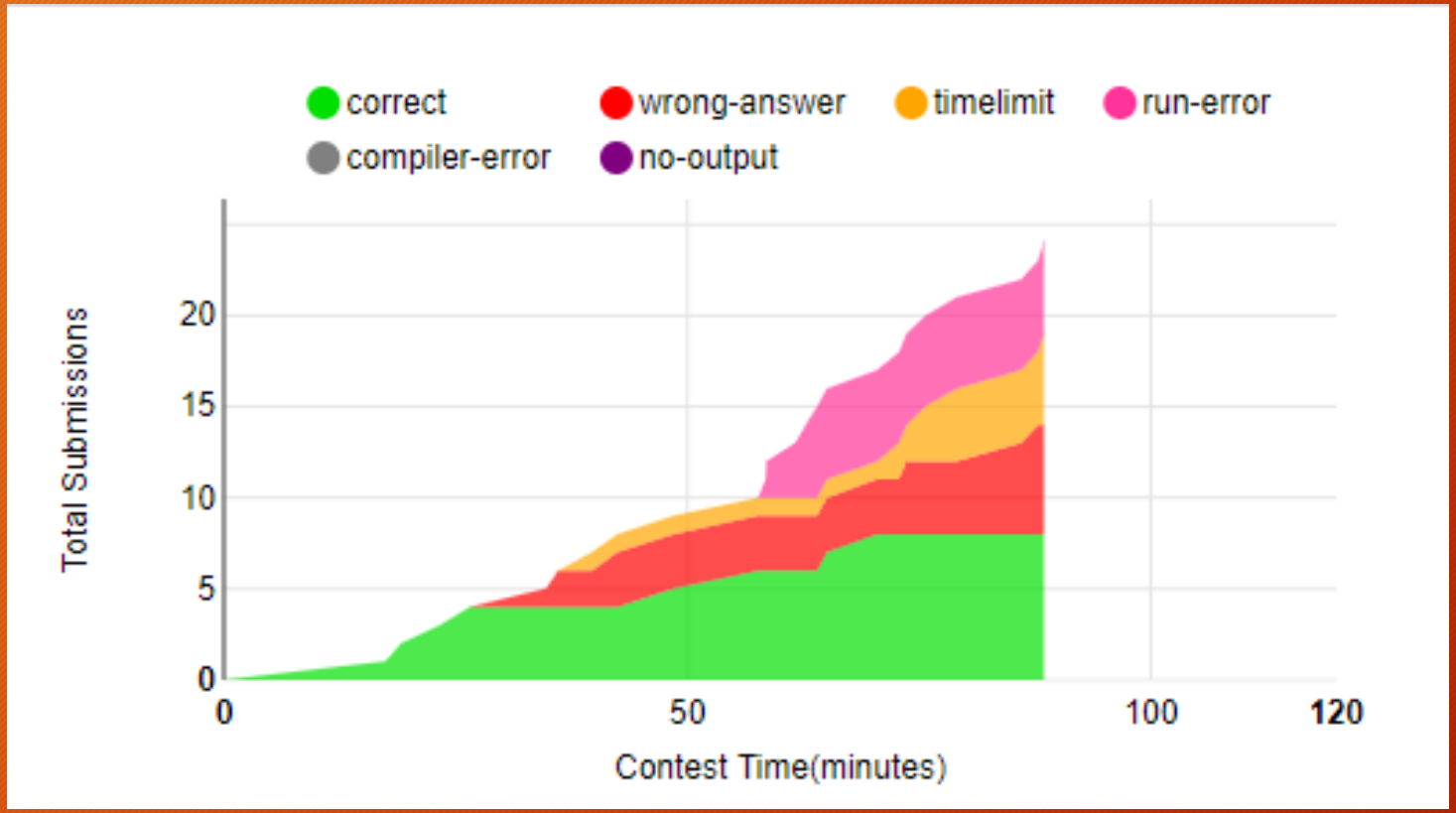
Autor: Raúl Martín Santamaría

Primer AC general: Alberto Maurel (17’)

Primer AC alumno curso: Jorge Sendarrubias (48’)

Benjamín Botones

E



Total (Antes del Freeze):

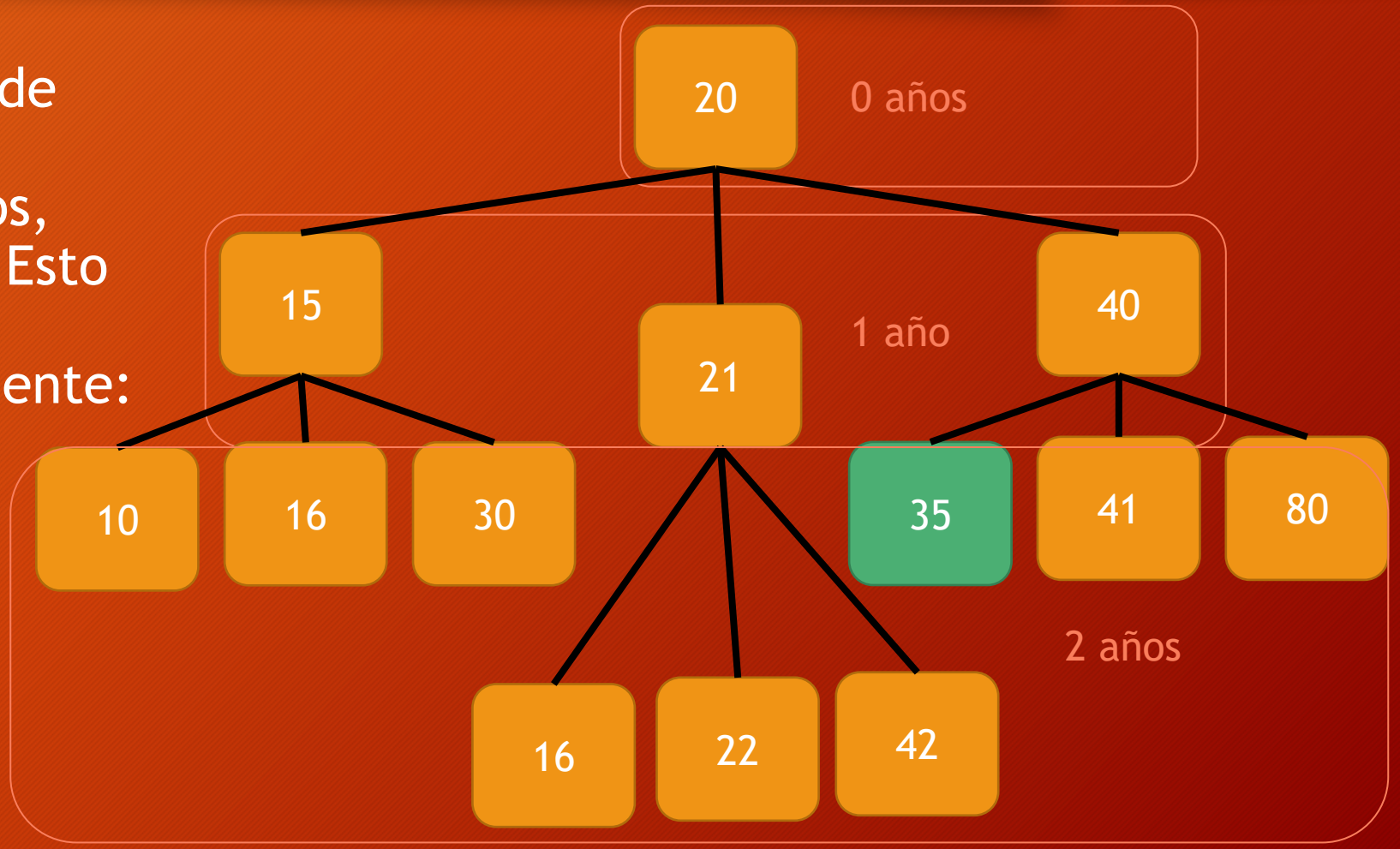
24 envíos
8 Accepted
6 Wrong Answer
6 RunTime Error
4 Timelimit Exceeded

Benjamín Botones

E

Benjamín Botones cambia de edad cada año, con tres opciones: Perder cinco años, cumplir uno o duplicarlos. Esto nos genera un árbol de posibilidades como el siguiente:

(Segundo caso de prueba del ejemplo)



Benjamín Botones

E

Para saber el número mínimo de años que tardaría Benjamín en llegar a la edad deseada, tenemos que utilizar un BFS (Búsqueda en anchura).

Un BFS consiste en, por cada nodo, meter sus nodos hijos en una cola, e ir sacando nodos de la cola hasta encontrar el destino (O vaciar la cola sin haberlo encontrado).

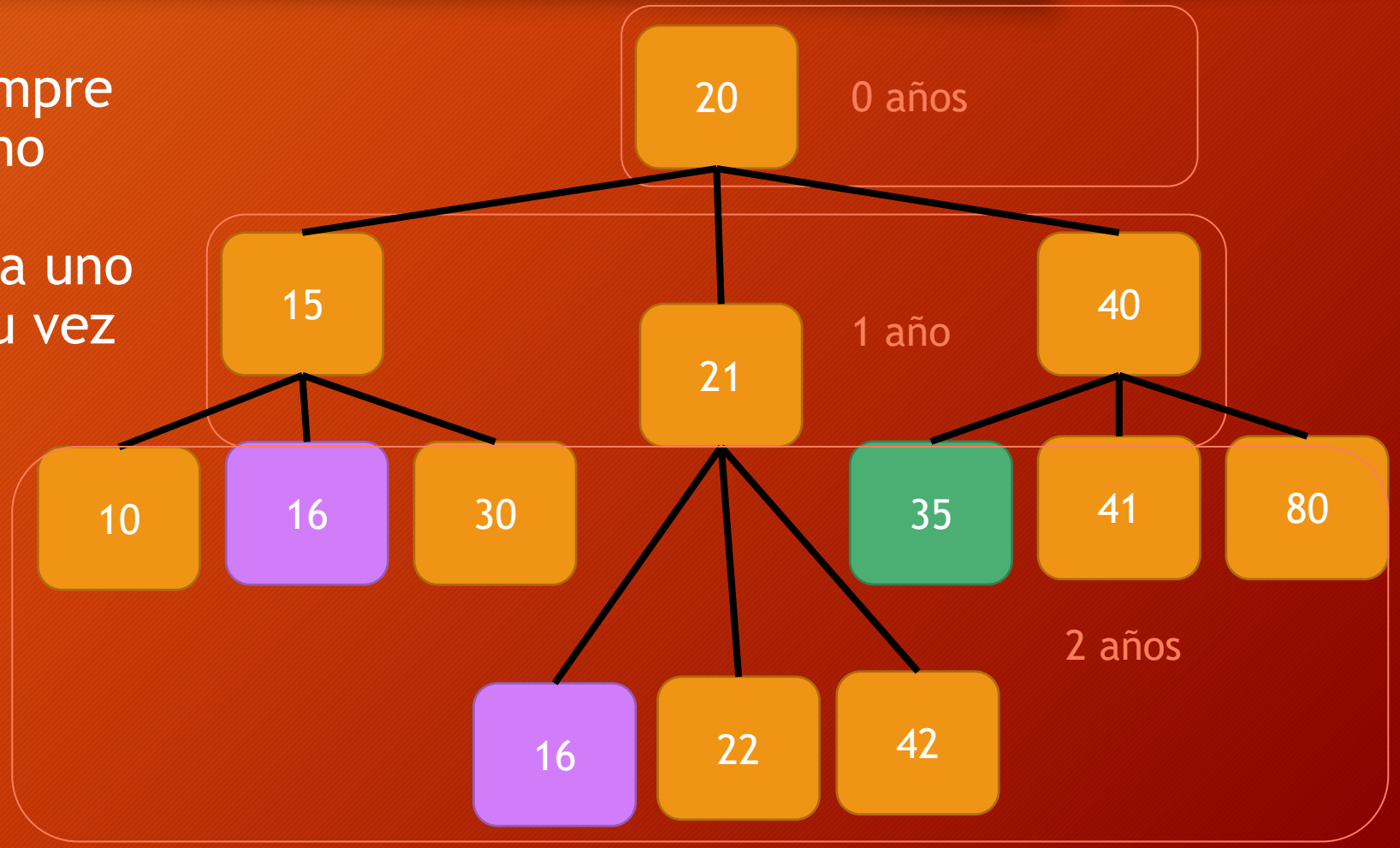
Importante no introducir elementos repetidos en la cola. Cada nodo siempre va a producir los mismos hijos, así que sería repetir una rama ya recorrida (Ejemplo en la siguiente diapositiva)

Benjamín Botones

E

Nodo “16” se repite, y siempre producirá 11, 17 y 32. De no eliminar 16 de la cola, añadiríamos dos veces cada uno de estos números, que a su vez seguirían repitiendo… Tardaríamos mucho más en llegar al objetivo.

TimeLimit Exceeded.



Benjamín Botones

E

Para saber en qué nivel del árbol estamos (Cuantos cumpleaños necesitaría tener Benjamín para llegar a la edad deseada), tenemos dos opciones:

1) Crear una nueva cola para cada nivel. Vamos vaciando una cola con todos los padres y metiendo los hijos a otra distinta. Cada vez que vaciemos nuestra cola de padres, hemos avanzado un nivel.

Benjamín Botones

E

2) Utilizar un elemento “marcador” para contar niveles: En la cola introducimos al principio el origen y el elemento marcador. A continuación, recorremos la cola como normalmente, y cada vez que encontremos el marcador, lo introducimos de vuelta. Si al sacarlo, la cola está vacía, el caso es imposible. Este elemento puede ser un número fuera de límites (Como -100 por ejemplo)

Cola: 20 | -100 | 15 | 21 | 40 | -100 | 10 | 16 | 30

Continuación: 22 | 42 | 35 | 41 | 80 | -100

16 no está repetido!

Cada vez que saquemos de la cola el elemento marcador, es que avanzamos un nivel.

Problema A

Enamorados a distancia

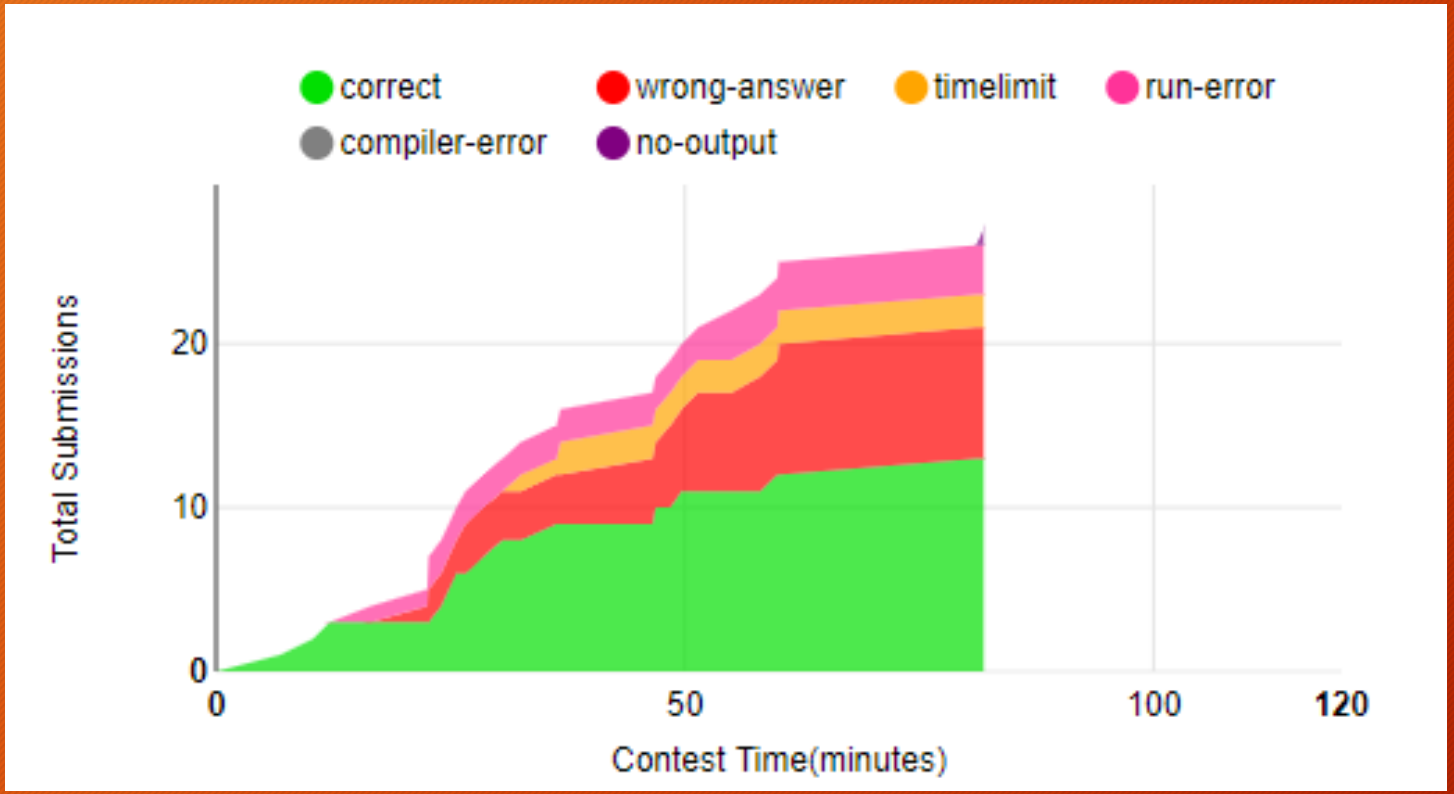
Autor: Iván Martín de San Lázaro

Primer AC general: David Morán (6’)

Primer AC alumno curso: Jorge Sendarrubias (10’)

Enamorados a distancia

A



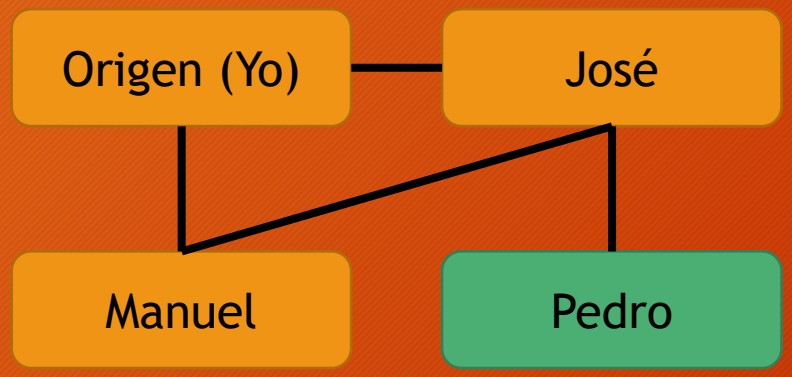
Total (Antes del Freeze):

29 envíos
13 Accepted
8 Wrong Answer
4 RunTime Error
3 Timelimit Exceeded
1 No Output

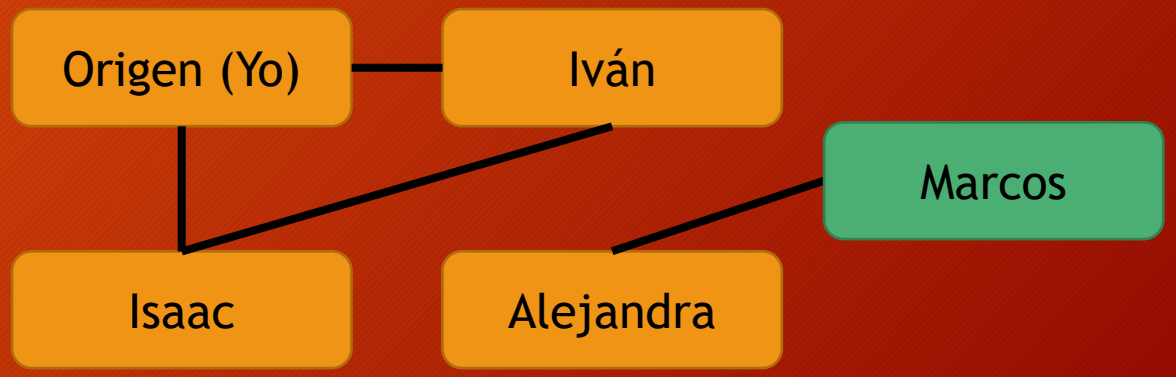
Enamorados a distancia

A

Modelando el problema como un grafo donde cada persona es un nodo y cada amistad una arista, la solución es tan sencilla como ver si desde nuestro nodo podemos alcanzar al nodo de la persona de quien nos hemos enamorado, aunque no sea nuestro amigo directo. Dicho de otra forma, ver si ambos nodos están en la misma componente conexa del grafo.



Encontré a mi príncipe azul



Helado y galletas

Enamorados a distancia

A

Para saber si podemos alcanzar o no el nodo que buscamos, podemos utilizar cualquiera de los recorridos sobre grafos, tanto BFS como DFS son válidos, utilizando como origen nuestro propio nodo.

Cuidado con introducir elementos repetidos en la cola/pila al recorrer el grafo: Puede costarnos TLE como hemos avisado antes.

Usando DFS, los nodos pueden llegar a 10000 => RTE (Stack Overflow). Se recomienda no hacerlo recursivo. Es mejor utilizar una pila.

Para poder trabajar fácilmente con el grafo, ya que las entradas son String, hay que utilizar un Mapa para convertir las String a números.

Problema B

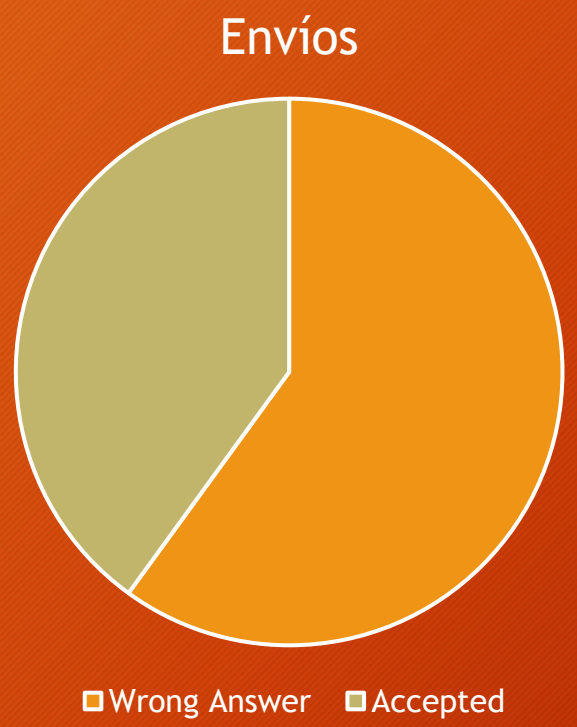
Poniendo baldosas

Autor: Jakub Jan Luczyn

Primer AC: David Morán (44’)

Poniendo Baldosas

B



Total (Antes del Freeze):

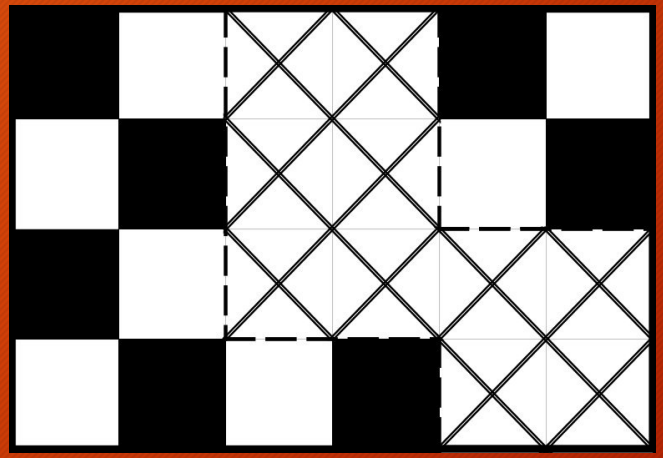
5 envíos
2 Accepted
3 Wrong Answer

Poniendo Baldosas

B

El problema de poner baldosas consiste en colorear todas las baldosas del suelo de color negro o blanco, sin que una baldosa negra toque a otra baldosa negra, ni una blanca toque a otra blanca.

Es un grafo bipartito.



Poniendo Baldosas

B

Una de las dificultades del problema está en que en la entrada vienen distintos grupos de baldosas, pero luego estos grupos pueden juntarse, teniendo que cumplir la condición de bipartito.

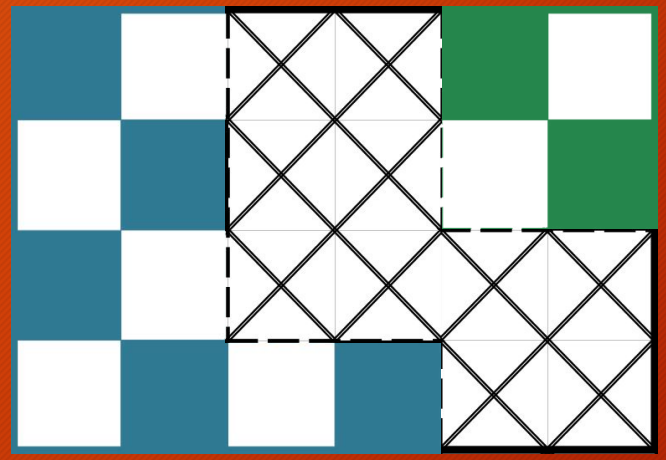
Lo que se debe hacer es leer la entrada completa antes de comenzar a colorear el grafo. Podemos hacerlo en una matriz, marcando como “true” las baldosas que hay que colorear y “false” las que no.

¡Ah! Y no, dividir el total de baldosas por la mitad para saber cuantas blancas y cuantas negras hay, no vale.

Poniendo Baldosas

B

A continuación, recorremos el suelo entero, buscando de forma individual cada componente conexa. En el ejemplo, pintaremos primero el grupo azul y luego el grupo verde. Da igual como lo coloreemos, al final de cada componente conexa, sumaremos el máximo de los dos colores a las casillas blancas y el mínimo a las negras. De esta forma, maximizamos las baldosas blancas.



Problema D

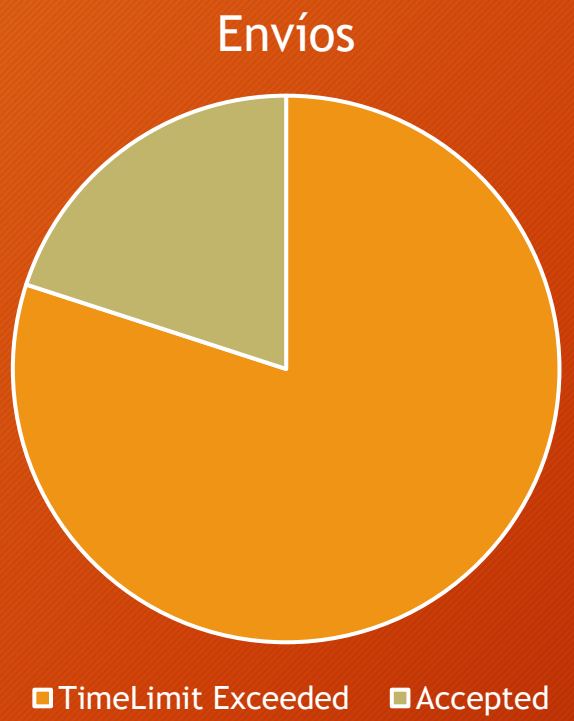
Esquivando al MamáRadar

Autor: Iván Martín de San Lázaro

Primer AC: Alberto Maurel (66')

Esquivando al MamáRadar

D



Total (Antes del Freeze):

5 envíos
4 TimeLimit Exceeded
1 Accepted

Esquivando al MamáRadar

D

Para los que no lo hayáis leído… aquí va un resumen:

Queremos comprar el Call Of Duty sin que nos pillen las vigilantes de la ciudad: Las madres. Se han repartido por toda la ciudad para vigilar.

Si cualquiera de las madres puede alcanzar el punto de origen del chico que quiere comprar el juego; le pillarán en el acto.

En caso contrario, si desde el origen se puede alcanzar la tienda, conseguiremos el juego. Si no podemos alcanzar la tienda, ¡lo hemos planeado todo mal!

La entrada consistía de la descripción gráfica de la ciudad, la localización de las madres (Que es fija) y N casos con el origen del chico y el punto de donde se ubica la tienda.

Esquivando al MamáRadar

D

Problema bastante difícil. ¡No os preocupéis los que ni lo habéis intentado!

Veamos como resolver el problema en 3 pasos:

- Leer la entrada
- Trabajar con un grafo en 2 dimensiones
- Saber si desde casa se puede llegar a la tienda sin TLE
- Saber si ninguna madre nos puede alcanzar sin TLE

Esquivando al MamáRadar

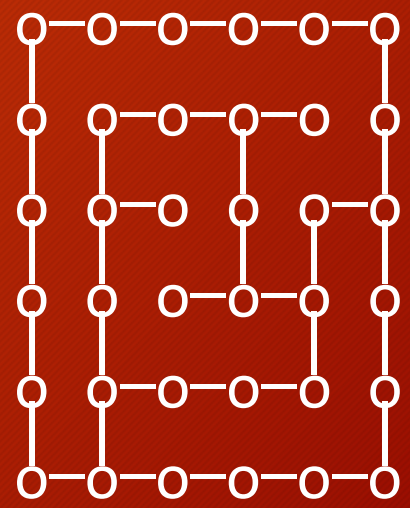
D

Paso 1: Leer la entrada

Nos garantizan que la ciudad es cuadrada, que está llena de edificios y que todas las intersecciones son transitables.

Por tanto, no tenemos que leer las “o”, ni los espacios en las posiciones de intersección.

Solo tenemos que distinguir entre “|”, “-” y “ “ para las posiciones donde haya una calle.



Esquivando al MamáRadar

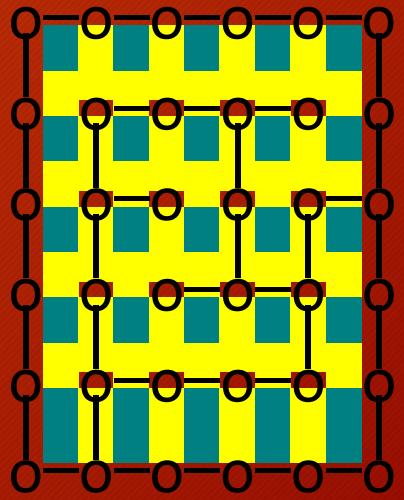
D

Paso 1.2: Leer la entrada y construir un grafo

En este caso, N=5. La entrada es 11*11. Tenemos que acceder solo a las posiciones impares, 1, 3, 5, 7 y 9, en ancho y alto. Como además nos ayudan y nos dan las paredes externas…

Para cada intersección (x, y), podemos acceder a las posiciones (x+1, y), (x-1, y), (x, y+1), (x, y-1) sin miedo a tener excepción por salirnos del array.

Para cada intersección o nodo (x, y), puede tener de 0 a 4 vecinos, dependiendo de si esas 4 posiciones anteriores son espacios o líneas cortadas.

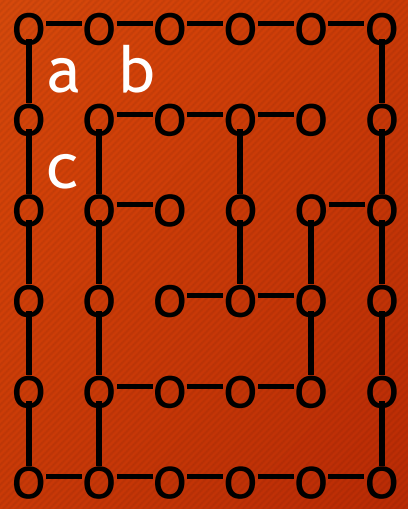


Esquivando al MamáRadar

D

Paso 1.3: Construir un grafo

De esta forma construimos un grafo:



Intersección a

Posición en el array de caracteres = (1, 1)

(0,1) = (1,0) => Cortado

(2,1) = (1,2) => Paso libre

Vecinos de a => b y c

Esquivando al MamáRadar

D

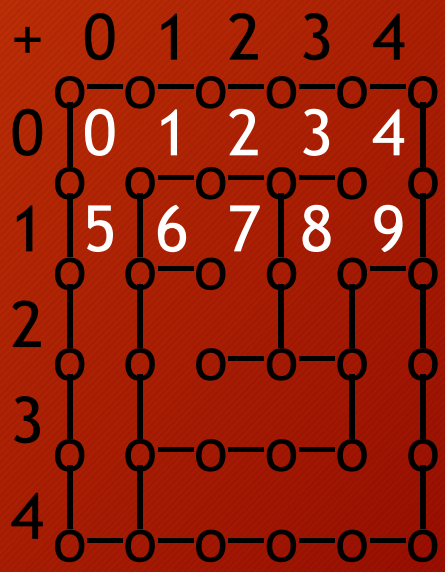
Paso 2: Trabajar con un grafo 2D

Trabajar con un grafo en dos dimensiones es posible, pero podemos simplificar los nodos para hacer más sencillo el problema.

Intentaremos asignar a cada nodo un identificador único. Para un nodo (x, y) podemos asignarle un id único con la fórmula [id = x*ancho + y]. Así, cada nodo es un único número y es más fácil y rápido implementar recorridos en él.

Vecinos de 0 = 1, 5. Vecinos de 1 = 0, 2.

Vecinos de 6 = 7.



Esquivando al MamáRadar

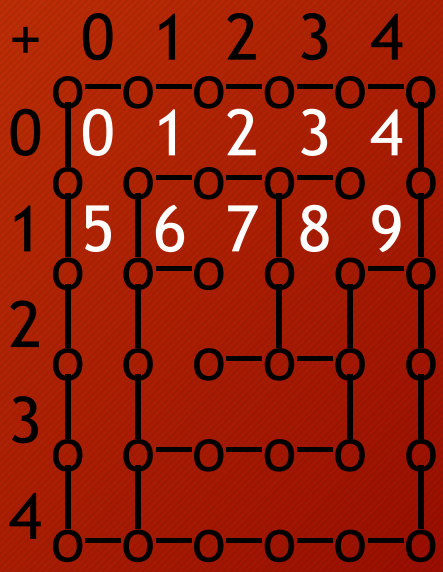
D

Paso 2: Trabajar con un grafo 2D

Con esta identificación, resulta mucho más fácil trabajar con el grafo. Al leer las entradas de las madres, las tiendas y los chicos, podemos convertirlas a este nuevo sistema.

Alternativamente, podemos crear un objeto nodo que almacene los dos índices:

```
class nodo {
int x;
int y;
}
```



Esquivando al MamáRadar

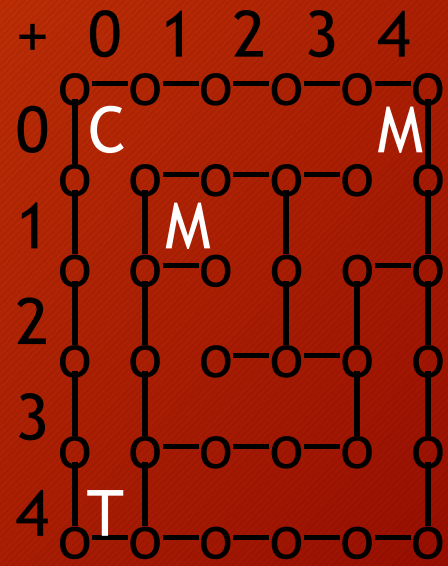
D

Paso 3: Como solucionar el problema.

Debemos comprobar dos cosas:

1.- Si alguna de las madres M nos alcanza.

2.- Si desde el punto inicial C podemos llegar a la tienda T



1º caso del ejemplo

Esquivando al MamáRadar

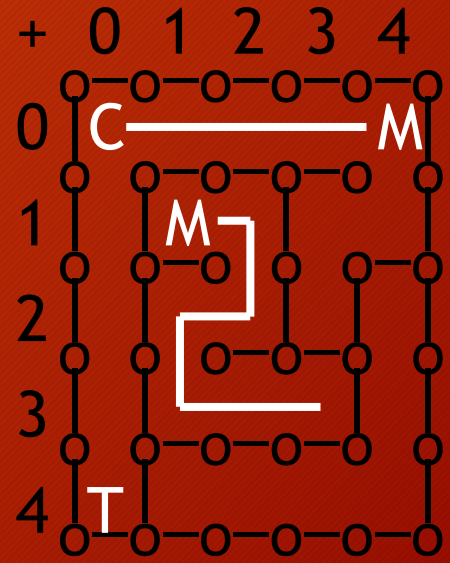
D

Paso 3: Como solucionar el problema.

1º acercamiento: Lanzar M recorridos (BFS, DFS), y si en alguno alcanzamos C, pillamos al chico.

TimeLimit Exceeded

El número de madres es alto, y el grafo puede ser enorme.



1º caso del ejemplo

Esquivando al MamáRadar

D

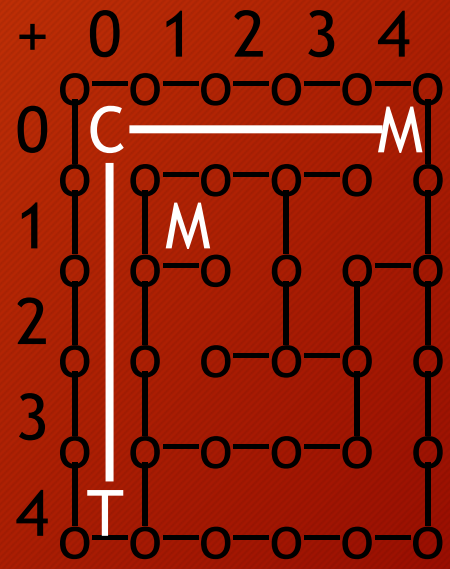
Paso 3: Como solucionar el problema.

2º acercamiento: Lanzar un recorrido desde C (BFS, DFS), y si en alguno alcanzamos cualquier M, pillamos al chico.

Si no se puede, lanzar otro recorrido buscando la tienda.

TimeLimit Exceeded

El grafo es enorme, y lanzamos 2 recorridos por cada caso.



1º caso del ejemplo

Esquivando al MamáRadar

D

Paso 3: Como solucionar el problema.

3º acercamiento: Lanzar un recorrido desde C (BFS, DFS), y si en alguno alcanzamos cualquier M, pillamos al chico. En el mismo recorrido buscamos la tienda T.

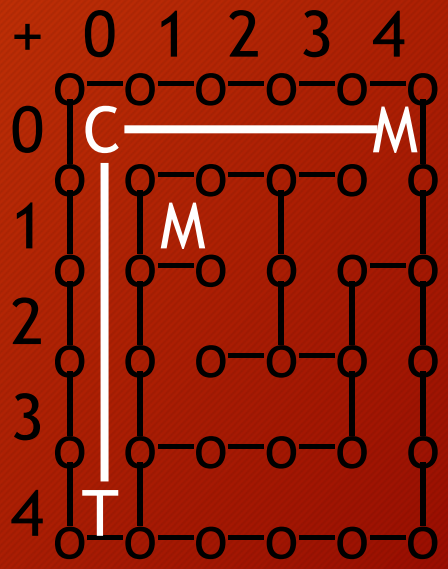
TimeLimit Exceeded

El grafo es enorme, y tendríamos que lanzarlo para cada caso individual.

Si los límites son tamaño = 1000, casos = 60000

Eso nos daría… 1000*1000 nodos * 60000 casos =

60.000 millones de operaciones!



1º caso del ejemplo

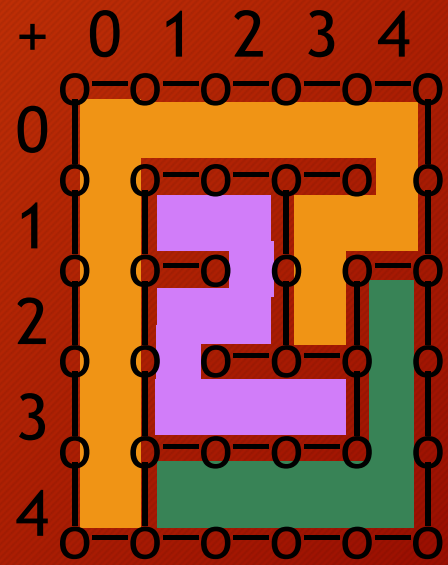
Esquivando al MamáRadar

D

Paso 3: Como solucionar el problema.

Solución: Calcular al principio las componentes conexas.

Antes de leer las madres, lanzamos recorridos por todo el grafo para mapear las componentes conexas. En este caso tenemos 3, la naranja, la morada y la verde.



Ejemplo

Esquivando al MamáRadar

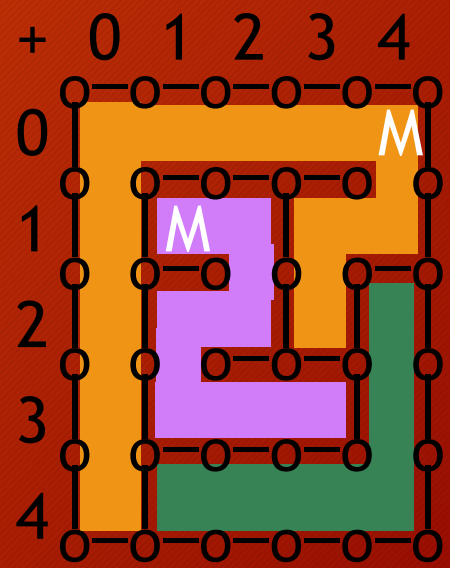
D

Paso 3: Como solucionar el problema.

Solución: Calcular al principio las componentes conexas.

A continuación, marcamos todas las componentes conexas en las que haya alguna madre. En este caso, la naranja y la morada.

Así, sabemos que toda posición en naranja o morado será pillada por alguna madre.



Ejemplo

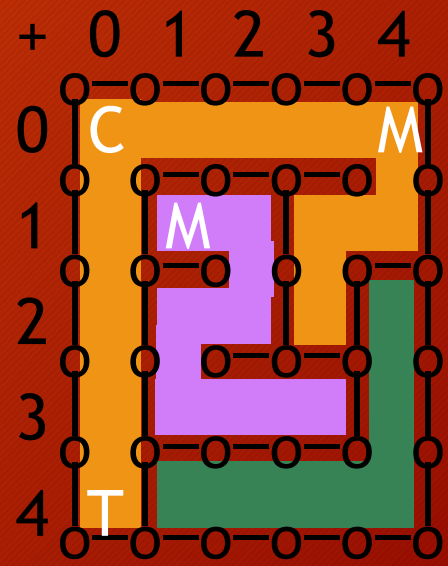
Esquivando al MamáRadar

D

Paso 3: Como solucionar el problema.

Solución: Calcular al principio las componentes conexas.
Ahora, cada caso se resuelve en O(1).
- Si C está en una componente conexa con M => PILLADO

En caso contrario:
- Si C está en la misma componente conexa que T => JUEGO
- Si C y T están en distintas componentes conexas => MALO



1º caso del ejemplo

Esquivando al MamáRadar

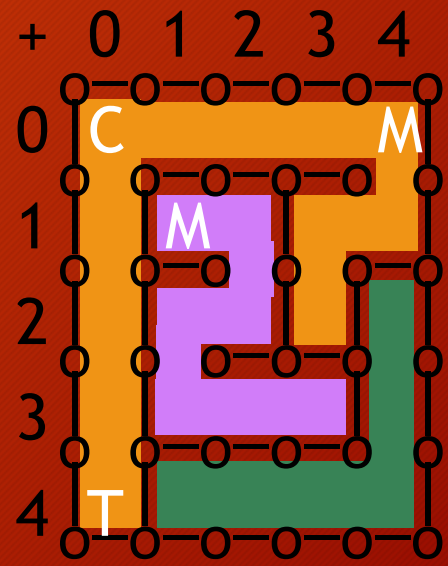
D

Paso 3: Como solucionar el problema.

1º caso de ejemplo:

C está en una componente conexa donde hay M (La naranja)

PILLADO



1º caso del ejemplo

Esquivando al MamáRadar

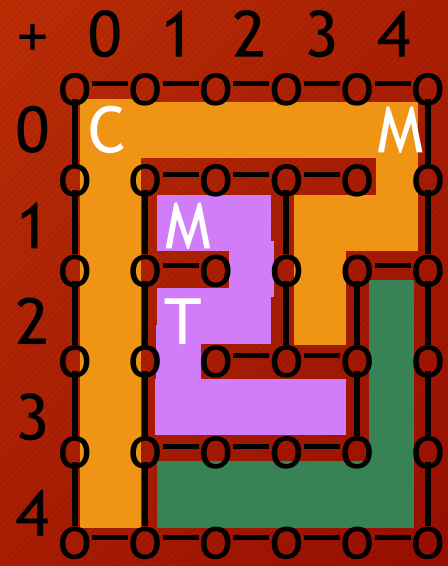
D

Paso 3: Como solucionar el problema.

2º caso de ejemplo:

C está en una componente conexa donde hay M (La naranja)

PILLADO



2º caso del ejemplo

Esquivando al MamáRadar

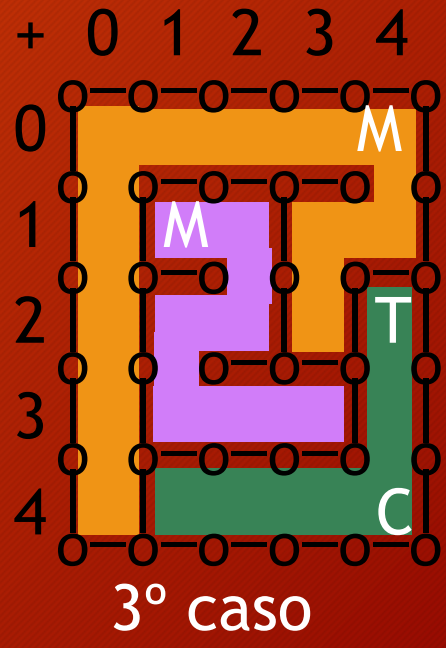
D

Paso 3: Como solucionar el problema.

3º caso de ejemplo:

C está en una componente conexa donde no hay M (La verde)

C y T están en la misma componente conexa = JUEGO



3º caso del ejemplo

Esquivando al MamáRadar

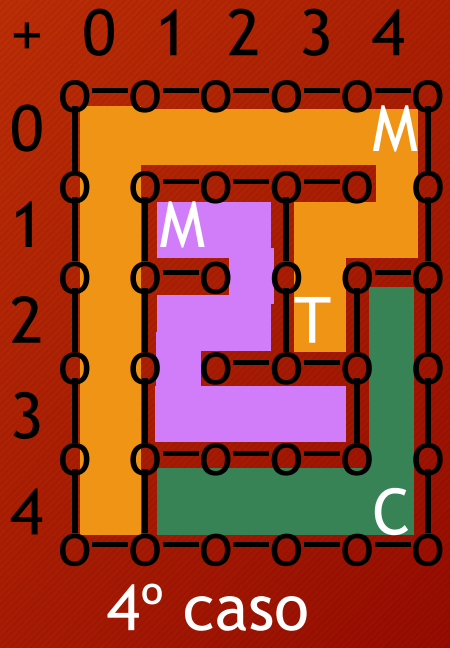
D

Paso 3: Como solucionar el problema.

4º caso de ejemplo:

C está en una componente conexa donde no hay M (La verde)

C y T están en distintas componentes conexas = MAL PLANEADO



4º caso del ejemplo