

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN
FACULTAD DE INGENIERIA DE PRODUCCION Y SERVICIOS
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
LABORATORIO - ANÁLISIS Y DISEÑO DE ALGORITMOS

GABRIELA MALENA QUISPE QUISPE

PROFESOR: ALEX FLOREZ FARFAN

AULA -9

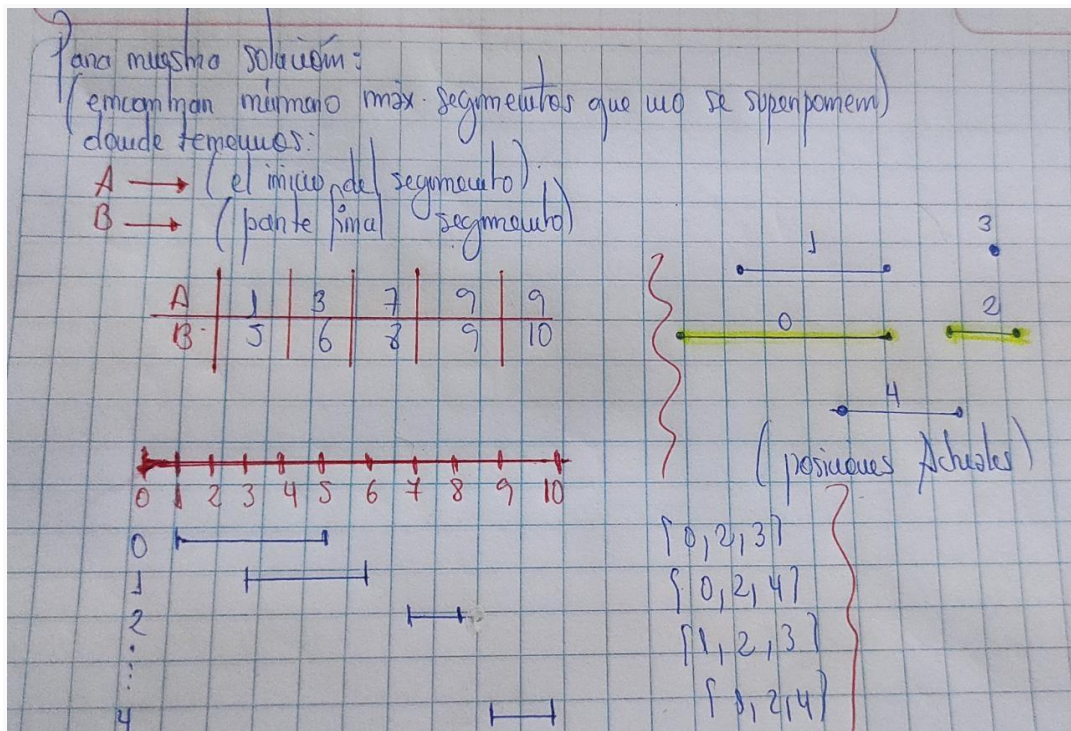


UNSA
UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

1. MaxNonoverlappingSegmentsen

En este ejercicio nuestro objetivo es encontrar el número máximo de segmentos máximos (no se superponen) en un conjunto de segmentos indicados en dos matrices, para lo cual los segmentos se clasifican previamente por posición final.

Para este ejercicio básicamente revisamos la posición actual, que vendría a ser el final de cada segmento, luego escaneo las matrices para encontrar líneas que comienzan después de la posición actual y cuento los segmentos(total) cuando encuentro un segmento que no se superpone.

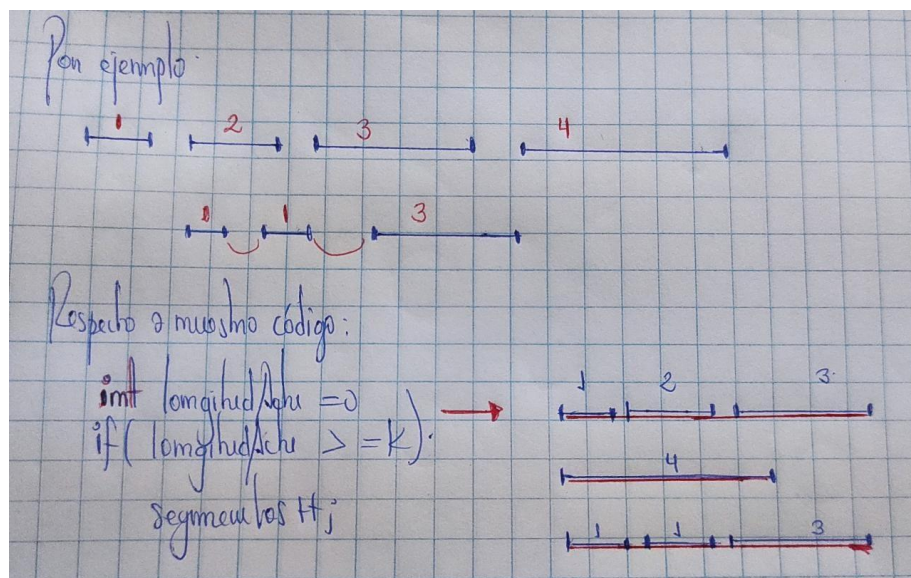


Prueba del código:


```
Test Output ✓  
  
| Compilation successful.  
  
| Example test: ([1, 3, 7, 9, 9], [5, 6, 8, 9, 10])  
| OK  
  
Your code is syntactically correct and works properly on the  
Note that the example tests are not part of your score. On sub  
will assess your solution.
```

2. TieRopes

La idea sería por ejemplo tener N de longitud mínima en el cual vamos a intentar quitar la última primero y luego quitar la primera. Sin importar si estas están atadas o no, no importa, y vamos desplazando estos segmentos hacia la izquierda lo más que podamos. Usando este algoritmo, tratamos de imprimir de izquierda a derecha.



Prueba del código:

```
Test Output 

| Compilation successful.



| Example test: (4, [1, 2, 3, 4, 1, 1, 3])
| OK

Your code is syntactically correct and works properly on the example test.
Note that the example tests are not part of your score. On submission at least 8 test cas
will assess your solution.
```

3. Bank Queue

Primero declaramos la cadena vector, dentro de la estructura indicamos nuestras variables dinero y el tiempo. posteriormente en un método aparte hacemos las comparaciones de cada persona respecto al su dinero, ahora colocamos a la persona en la cola, si este lugar que ocupa esta persona se encuentra ocupado y los demás lugares también , entonces en nuestro código paramos el bucle (se detiene cuando no haya más personas en nuestra cola). Dentro de nuestra clase principal, vamos a declarar la cantidad total de personas(cola), el tiempo que se cierra el banco y las dos otras variables que nos indica el problema, declaramos la cola. También usamos el Sort para ir ordenando de primero a último en orden ascendente. imprimiendo finalmente la cantidad máxima de dinero que puede obtener la persona en la cola.

Prueba del código:

Sumisión			
IDENTIFICACIÓN	FECHA	PROBLEMA	ESTADO
CASOS DE PRUEBA			
8152943	20:01:18	Cola bancaria	 Aceptado
			

```
Compiler output

./BankQueueee.cpp: In function 'int main()':
./BankQueueee.cpp:37:10: warning: ignoring return value of 'int scanf(const char*, ...)',
 37 |     scanf("%d %d", &total, &minutos);
    |     ~~~~~^~~~~~
./BankQueueee.cpp:43:14: warning: ignoring return value of 'int scanf(const char*, ...)',
 43 |     scanf("%d %d", &efectivo, &timeTotal);
    |     ~~~~~^~~~~~
```

4. A Vicious Pikeman

$$t_I = ((At_{yo-1} + B) \bmod C) + 1, yo \in [1, N - 1]$$

$$1 \leq N \leq 10^4$$

Para este problema: tenemos nuestro módulo = 1000000007.

Prueba del código:

Sumisión			
IDENTIFICACIÓN	FECHA	PROBLEMA	ESTADO
8153082	CASOS DE PRUEBA		
	21:40:36	Un piquero vicioso (fácil)	✓ Aceptado
✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓			

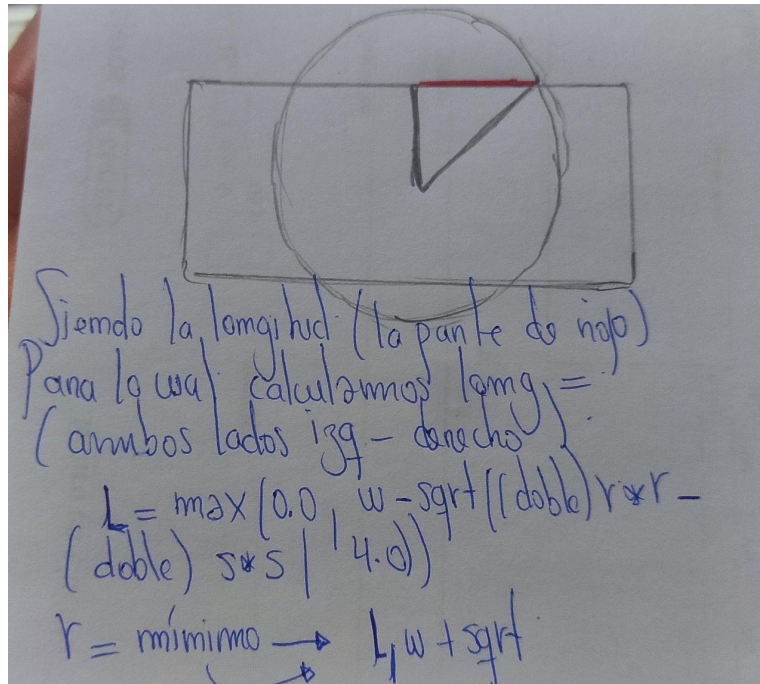
```
1 3
2 2 2 1
1 1

...Program finished with exit code 0
Press ENTER to exit console.
```

5. Watering Grass

Básicamente el ejercicio es dar la longitud y el ancho de un césped rectangular, y luego dar la posición horizontal de una serie de tuberías de agua y el radio del área que pueden cubrir las tuberías de agua; el lugar donde se puede colocar cada tubería de agua. instalado es el centro longitudinal del césped, y lo mínimo que se debe pedir varias tuberías de agua pueden hacer que todo el césped esté cubierto por agua y producir la menor cantidad; si no se puede cubrir, nuestra salida será -1.

Para resolver este problema necesitamos conocer los puntos cortos izquierdo y derecho de la unión de cada cobertura de tubería de agua y el césped, y luego almacenarlo en la estructura, aunque el radio de la tubería de agua r en la pregunta es de tipo int, si el dato es muy grande, si lo elevamos al cuadrado resultará muy grande, entonces necesitamos el radio del tipo double. Primero comenzamos por el extremo izquierdo, lo que necesitamos encontrar es el punto que se puede cubrir y el límite derecho más lejano que se encuentra con el césped, como un rango de tubería de agua, realizado esto buscamos la siguiente tubería de agua hasta que la tubería pueda cubrir el borde derecho del césped.



Inicialmente estaba buscando hallar los radios, de A y B, pero no me acepto todos los casos de prueba, por lo cual tome un radio de posición, y también con ayuda del sort (ordene de manera creciente aumentando) similar al ejercicio anterior donde uso Sort.

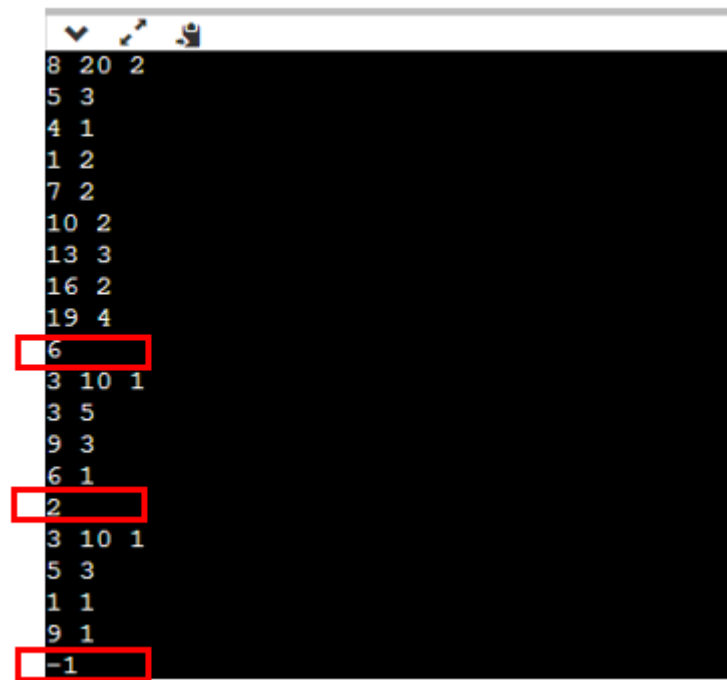
Prueba del código:

Inicialmente cometí un error.

Sumisión			
IDENTIFICACIÓN	FECHA	PROBLEMA	ESTADO
CASOS DE PRUEBA			
8153306	2021-12-10 23:42:22	Regar la hierba	✗ Respuesta incorrecta
<input checked="" type="checkbox"/> <input type="checkbox"/>			

Posteriormente me acepto

Sumisión			
IDENTIFICACIÓN	FECHA	PROBLEMA	ESTADO
CASOS DE PRUEBA			
8153350	00:21:29	Regar la hierba	✓ Aceptado
<input checked="" type="checkbox"/> <input checked="" type="checkbox"/>			



A terminal window with a black background and white text. The text consists of a list of numbers, some of which are grouped by spaces. Four specific numbers are highlighted with red rectangular boxes: 6, 2, -1, and -1. The numbers are arranged in a vertical list, with some having multiple spaces between them, suggesting a list of numbers with varying lengths or a specific format.

```
8 20 2
5 3
4 1
1 2
7 2
10 2
13 3
16 2
19 4
6
3 10 1
3 5
9 3
6 1
2
3 10 1
5 3
1 1
9 1
-1
```