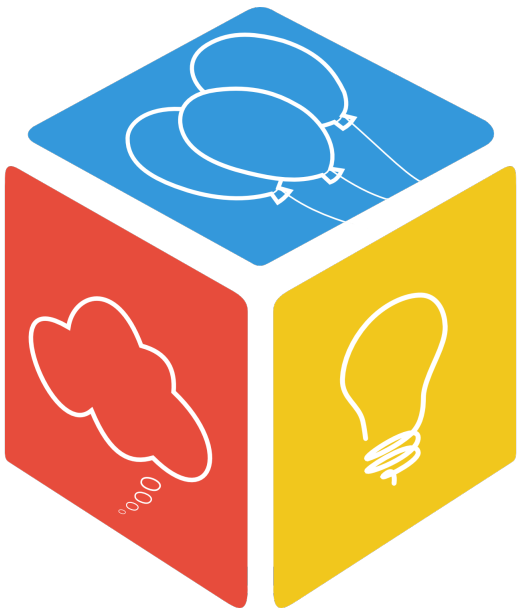
Editorial Maratón de Programación UFPS 2018



Programación Competitiva
UFPS

Universidad Francisco de Paula
Santander

Junio 09 de 2018

Maratón Interna UFPS

La maratón de programación interna UFPS 2018 contó la participación de 20 equipos de la Universidad Francisco de Paula Santander y contó con 13 problemas inéditos:

- A - Attractive Subsequence
- B - Valentine's Day
- C - Circumscribed Recursion
- D - Dreams
- E - Elseif statement does not exist
- F - Free Robot

Maratón Interna UFPS

- G - Acrobat
- H - Harry and the golden egg
- I - Incomplete Team
- J - Shuffle Cards
- K - Universal Language I
- L - Lázaro System
- M - Valyrio muño engos ñuhys issa

Maratón Abierta UFPS

La maratón de programación abierta UFPS 2018, ejecutada a traves de la Red de Programación Competitiva RPC y en la cual participaron 115 equipos de 8 paises de latinoamerica contó con 13 problemas inéditos:

- A - Attractive Subsequence
- B - Beautiful Tree
- C - Circumscribed Recursion
- D - Being a cheater
- E - Elseif statement does not exist
- F - Free Robot

Maratón Abierta UFPS

- G - Acrobat
- H - Harry and the golden egg
- I - Universal Language I
- J - Valyrio muño engos ñuhys issa
- K - Shuffle Cards
- L - Lázaro System
- M - Natural Sort

Maratón de Programación UFPS 2018

En total la maratón de programación UFPS 2018 contó con 16 ejercicio, 3 que aparecieron únicamente en la versión interna, 3 únicamente en la versión abierta, y 10 en ambas versiones.

A continuación se describe la aproximación a las soluciones de los 16 problemas, ordenados del más fácil al más difícil (según el criterio de los problemsetters).

1. Incomplete Team

Manuel solo puede asistir cada M competencias. Carlos cada N competencias.

Manual asistirá a las competencias: M, 2M, 3M, 4M, 5M...

Carlos asistirá a las competencias: N, 2N, 3N, 4N, 5N...

1. Incomplete Team

Manuel solo puede asistir cada M competencias. Carlos cada N competencias.

Manual asistirá a las competencias: M, 2M, 3M, 4M, 5M...

Carlos asistirá a las competencias: N, 2N, 3N, 4N, 5N...

Es decir, Manuel asiste a las competencias que son múltiplos de M, y Carlos a las que son múltiplos de N.

Por tanto la respuesta es el mínimo común múltiplo entre M y N.

2. Dreams

Simulación simple. Almacenar los valores leídos e imprimir aquellos no leídos.

Solución simple: arreglo de booleans.

Importante: Tener cuidado con los espacios. No debe haber un espacio antes del primer número, ni después del último.

3. Circumscribed Recursion

Área del cuadrado $= l^2$

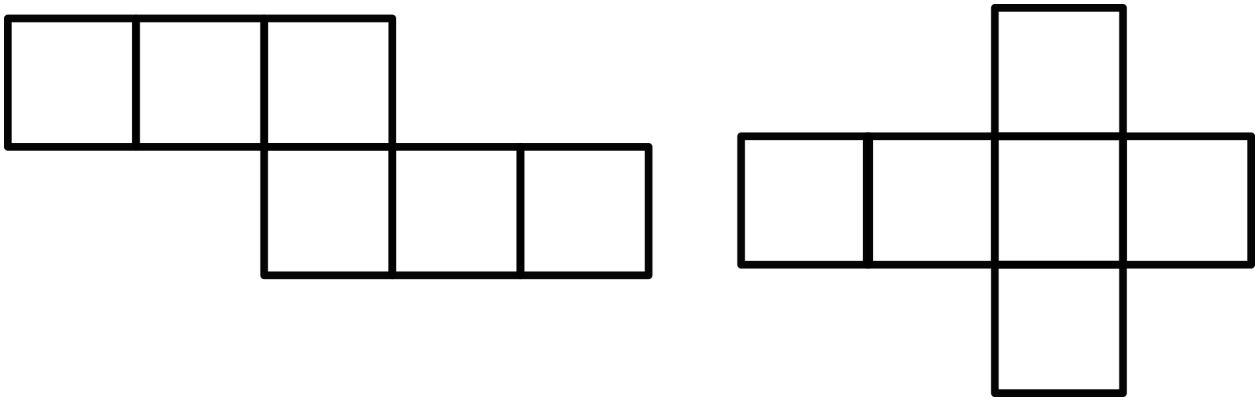
Área del círculo $= \pi r^2$

Solución 1. Con estas dos fórmulas basta para calcular el área de la figura (el radio o lado de la figura puede hallarse recursivamente hasta el primero).

Solución 2. Con las fórmulas previamente dadas, generar una formula general sencilla para todos los casos cuadrado, y otra para los casos círculo.

Nota: No es necesario usar valores flotantes, el problema es entero. Usar tipo de dato long.

4. Valentine's day



Existen varias formas de armar la figura, pero finalmente todas las demás son variaciones de estas dos. Verificar que alguna de las variaciones de (X, Y, Z) encajan en alguna variación de (A, B).

5. Shuffle Cards

Definamos un concepto:

- **Altura agrupada de una pila de cartas**: cantidad de conjuntos de cartas contiguas en la misma dirección.

El objetivo es convertir la altura agrupada en 1 (un solo subconjunto con todos las cartas en la misma dirección, cuya dirección sea 'B').

5. Shuffle Cards

Puede demostrarse que cada movimiento puede como máximo disminuir en 1 la altura agrupada. Por tanto para alcanzar una altura agrupada de 1, la respuesta será la altura agrupada inicial - 1. Sin embargo al finalizar este proceso la dirección de la pila puede ser 'A'. En este caso debe hacerse un giro más.

6. Harry and the golden Eggs

Simplificando el problema, debe hacerse una inversión en ciertos caracteres de corte: (coma, comilla, punto y coma, punto, espacio o salto de linea). Cada vez que se encuentre un punto de corte, deben imprimirse en orden inverso todos los caracteres entre el punto de corte anterior y el actual. Luego, imprimir el mismo punto de corte.

Caso especial: Tener cuidado con las palabras que inician en mayúscula.

6. Harry and the golden Eggs

Simplificando el problema, debe hacerse una inversión en ciertos caracteres de corte: (coma, comilla, punto y coma, punto, espacio o salto de linea). Cada vez que se encuentre un punto de corte, deben imprimirse en orden inverso todos los caracteres entre el punto de corte anterior y el actual. Luego, imprimir el mismo punto de corte.

Caso especial: Tener cuidado con las palabras que inician en mayúscula.

7. Universal Language 01

Problema de conversión de bases. Dado un lenguaje de m letras l_i, podemos dar a cada letra un valor ascendente de 1 hasta m. El valor de una letra específica se considera v_i.

La conversión se realiza de la forma común, como se convertiría de octal o binario a decimal, utilizando m como base.

7. Universal Language 01

Problema de conversión de bases. Dado un lenguaje de m letras l_i, podemos dar a cada letra un valor ascendente de 1 hasta m. El valor de una letra específica se considera v_i.

La conversión se realiza de la forma común, como se convertiría de octal o binario a decimal, utilizando m como base.

8. Valyrio

Consideremos:

- $f(x)$ cantidad de palabras válidas de tamaño x.
- $fv(x)$ cantidad de palabras válidas de tamaño x que inician por una vocal.
- $fc(x)$ cantidad de palabras válidas de tamaño x que inician por una consonante.
- c, cantidad de consonantes.
- v, cantidad de vocales.

8. Valyrio

Si una palabra es hermosa, podemos quitar su primer caracter y seguirá siendo hermosa (y seguir repitiendo este proceso hasta que quede un solo caracter, siendo hermosas todas las palabras surgidas en el proceso).

A la inversa, podemos decir que cualquier consonante, seguida por una palabra hermosa que inicie por vocal, es hermosa. Una vocal, seguida por cualquier palabra hermosa, es hermosa.

Por tanto:

8. Valyrio

- $f(x) = fc(x) + fv(x)$
- $fc(x) = c * fv(x-1)$
- $fv(x) = v * f(x-1)$

Simplificando:

$fc(x) = c * v * f(x-2)$

$f(x) = (c * v * f(x-2) + (v * f(x-1))$

8. Valyrio

Programación dinámica:

$$f(x) = \begin{cases} 1 & \text{if } x = 0 \\ v & \text{if } x = 1 \\ (c * v * f(x-2) + (v * f(x-1)) & \text{else} \end{cases}$$

Tener en cuenta las propiedades de los módulos (se puede realizar el módulo a cada operación intermedia). Para saber si es aceptado o rechazado, puede llevarse una memorización auxiliar indicando si ya se ha realizado un módulo previamente.

9. Free Robot

Los casos son muy pocos, por tanto puede precalcularse y enviarse como solución un arreglo con los valores para todos los casos.

El precalculo puede hacer con un algoritmo de DFS que limpie la estructura de visitados al finalizar una ruta (dado que cualquier posición puede repetirse mas adelante en otra ruta).

10. Acrobat

Encontrar ruta mas corta en grafos ponderados: Algoritmo de Dijkstra.

Es necesario ejecutar dijkstra con cola de prioridad para cumplir el tiempo. Tener en cuenta que en lugar de un valor, el grafo almacena al tiempo dos valores. La cola de prioridad debe dar prioridad al valor de subida, y solo en caso de empates, definir por el de bajada.

11. Lázaro System

El problema es una recursión. Se puede seguir el esquema planteado en el problemset, teniendo en cuenta de que forma los caracteres se invierten al encontrar un ángulo.

Iniciar con una matriz de puntos lo suficientemente grande para contener todo el dibujo.

12. Attractive Subsequence

Almacenar los acumulados de los valores (es decir, para cada posición i almacenar la suma desde el valor 0 hasta el i).

Por cada consulta, recorrer los acumulados y para cada valor en los acumulados realizar un upperbound y lowerbound para conocer el rango en el cual los acumulados cumplen la suma. Almacenar la cantidad de valores en cada rango.