Olimpiada Boliviana de Informatica

Fase Final - Nivel 3

2015

Olimpiada Boliviana de Informatica

7 de octubre de 2015

Las páginas están numeradas desde el 1 al 11 .

Índice

Agua **1**

El escape de Wordo **3**

Himno **5**

Perlas **7**

Robot **10**

Problema A
Agua

Tiempo límite: *1 sec*
Memoria límite: *64 MB*

Dibu ha sido contratado por la empresa municipal de agua Sanagua, y ahora es encargado del suministro de agua a todos los barrios de la ciudad.

La red de agua de la ciudad está compuesta por N unidades de distribución, ubicadas en cada uno de los barrios. Estas unidades de distribución se encargan de recibir el agua para cada barrio y de distribuir lo demás hacia los barrios vecinos. Algo muy importante acá es que por cuestiones de seguridad, cada unidad de distribución sólo puede recibir el agua de una fuente, lo que quiere decir que jamás dos unidades de distribución enviarán agua a la misma unidad.

Uno de los trabajos más importantes de Dibu es estar al tanto de la cantidad de agua que se envía a cada uno de los barrios de la ciudad, para prevenir que se envíe mas agua de la necesaria a un barrio, o al revés, que se envíe menos cantidad de la necesaria

Desafortunadamente para ti, Dibu esta de vacaciones y te toca hacer su trabajo hasta que regrese.

Cada barrio tiene permitido usar X cantidad de litros diarios de agua, sobre esta base, se realizarán dos tipos de operaciones y cada operación va precedida por un número que indica su tipo:

1 X Y: Incrementar en **X** ($1 \leq X \leq 104$) litros a la cantidad de agua que puede usar el barrio **Y** y a la de todos sus vecinos.

2 Z: Preguntar cuál es el barrio que más agua puede recibir de entre todos los vecinos de **Z** (un barrio Y se considera vecino de Z si es que de alguna manera Z le envia el agua a Y) con $Z \leq N$

Todos los barrios están enumerados de 1 a N, sin embargo, la distribución de agua comienza en la empresa Sanagua que es cosiderada el "Barrio 0" y además Sanagua no recibe agua bajo ninguna circunstancia (Sanagua la distribuye)

Entrada

La entrada consiste en un numero N, que es la cantidad de barrios que existen en la ciudad. A continuación existen N líneas. La i-esima línea contiene 2 enteros, la cantidad X ($1 \leq x \leq 104$) de litros de agua que el barrio i puede recibir, y el barrio del cual recibe el agua. En la siguiente línea un número M, que representa la cantidad de operaciones que se realizarán. En las siguientes M líneas habrá una operación, cada operación en una línea, tal cual estan descritas mas arriba.

Salida

Para cada operación de tipo 2, se debe indicar el barrio que más agua recibe, si más de un barrio reciben la máxima cantidad de agua, imprimir el menor.

Ejemplo de entrada

```
10
0 0
5 0
7 1
8 2
2 4
8 1
0 5
8 2
3 4
0 9
5
1 3 2
2 10
2 2
1 3 9
2 1
```

Ejemplo de salida

```
10
4
6
```

Subtareas

Subtarea 1: (20 ptos.)

$1 \leq N \leq 1000$

$1 \leq M \leq 1000$

Subtarea 2: (25 ptos.)

$1 \leq N \leq 10^5$

$1 \leq M \leq 10^5$

Todas las operaciones tipo 1 se realizarán primero.

Subtarea 3: (55 ptos.)

$1 \leq N \leq 10^5$

$1 \leq M \leq 10^5$

Problema B
El escape de Wordo

Tiempo límite: *1 sec*
Memoria límite: *512 MB*

Wordo, el malvado villano intergaláctico, culpable de muchos delitos como robo de lunas y estrellas, ha escapado de la prisión de máxima seguridad de Bitavia.

Esto ha alarmado a toda la comunidad, es por esto que la Organización de Búsquedas Interestelares (OBI) ha puesto en marcha un plan en el que ha ofrecido una gran recompensa por cualquier información proporcionada acerca de la ubicación de Wordo.

Como era de esperarse, muchas personas han afirmado ver a Wordo en distintos lugares de la comunidad interestelar. Como es de esperarse, algunas de estas denuncias son ciertas, y otras no lo son.

La OBI entonces tiene la importante tarea de averiguar si una denuncia es verdadera o no, para esto se basa en los siguientes hechos:

- Una denuncia se compone de una lista de planetas donde ha sido visto Wordo.
- Se ha descubierto una red de tuneles de gusano (mínima) construida por los secuaces de Wordo, que conectan varios planetas, para asi ampliar las posibles rutas de escape de este villano.
- Wordo solo se transportará mediante los tuneles de su red de tuneles de gusano
- Ademas, se sabe que Wordo es un villano muy inteligente, por lo que jamás volvera a un sitio donde ya ha estado antes.

Tu tarea es escribir un programa que ayude a la OBI a decidir cuando una denuncia es verdadera.

Entrada

En la primera línea de entrada habran dos enteros N y Q, la cantidad de ciudades que conecta la red de tuneles de la red de Wordo y la cantidad de denuncias.

Las siguientes $N-1$ líneas describen los tuneles, en cada línea dos enteros A y B ($A \neq B$) que significan que los planetas A y B están unidos por un tunel. A continuación siguen Q denuncias cada denuncia esta formada por un entero K y a continuación siguen K enteros, los planetas que forman parte de la denuncia.

Nota: Una denuncia es verdadera cuando realmente es posible que Wordo haya pasado por todos los planetas de la denuncia.

Salida

Una línea que contenga "SI" (sin las comillas), cuando la denuncia sea verdadera, o "NO" si la denuncia es falsa.

Ejemplo de entrada

```
10 4
3 1
5 7
4 6
10 4
8 9
2 1
3 4
8 5
5 3
3
8 10 3
4
8 7 9 6
3
2 1 6
3
6 2 5
```

Ejemplo de salida

```
SI
NO
SI
NO
```

Subtareas

Subtarea 1: (25 ptos.)

El número de nodos esta entre [1, 100] y la cantidad de denuncias no supera el 100.

Subtarea 2: (15 ptos.)

El número de nodos esta entre [1, 1000] y la cantidad de denuncias es 1000.

Subtarea 3: (60 ptos.)

El número de nodos entre [1, y 10^5] la sumatoria de todos los K no superará 10^6.

Problema C
Himno

Tiempo límite: *1.5 sec*
Memoria límite: *128 MB*

Oh no! Wordo, el malvado villano, ha invadido la ciudad! :′ (

Luego de cuantificar los daños, los inspectores se han dado cuenta de que falta algo muy importante: El himno. Si, Wordo se ha llevado el himno de la ciudad (aún se investiga como es que esto puede ser posible). Afortunadamente tu eres el mejor compositor de la ciudad, y te han dado el trabajo de escribir un nuevo himno para la ciudad. Claro, esto no es gratis, y han ofrecido pagarte una gran comisión, proporcional a la belleza del himno que compongas.

Para poder medir la belleza de tu himno, el alcalde de la ciudad te ha dado la siguiente información:

- Se usará una escala de notas especial, basada en la serie de fibonacci, donde cada nota esta representada por el siguiente valor $F(i)\,\%M$ (M es un número que elegiras tú). Si se usan otras notas la belleza del himno es 0.
- Un himno se considera bello cuando las notas que lo componen están ordenadas, es decir, el himno tiene la forma: $F(0)\,\%M, F(1)\,\%M, F(2)\,\%M ... F(N)\,\%M$. De otra manera, se considera que la belleza del himno es 0.
- El himno no debe repetir juntas un mismo par de notas. Es decir, si en algun momento el himno contiene el par de notas X, Y en ese orden, no puede existir X, Y en ninguna otra parte del himno. Si esto sucede la belleza del himno es 0.
- Si se cumplen las demas condiciones, la belleza del himno es igual a la cantidad de notas que se tocan. Por ejemplo si M es igual a 2, el himno es: $0, 1, 1$, la belleza total será 3

Solo por si no lo recuerdas, la serie de fibonacci es esta: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Entrada

Este problema no tiene entrada.

Salida

En la primera línea de salida deberán estar dos enteros N y M, la cantidad de notas que se tocan en tu himno y el número M que elegiste. En la segunda línea deberás mostrar N números, separados por un espacio, que representa el himno que compusiste.

```
4 2
0 1 1 0
```

Tarea

Este problema tiene una subtarea única que vale 100 puntos. Para conseguir el puntaje completo, tu himno deberá tener una belleza de por lo menos 300.000.

Si la belleza del himno que tu programa produce es de X, tu puntaje será dado por la proporción de X con respecto a 300.000. Formalmente:

$$puntaje = (min(X, 300{,}000)/300{,}000) * 100$$

Problema D
Perlas

Tiempo límite: *2 sec*
Memoria límite: *128 MB*

Addis se fue de viaje unos días a algún lugar que nadie conoce. Como siempre piensa en sus amigos, decidió traer unos souvenirs para ti: pequeñas perlas esféricas.

Antes de dártelas, Addis te propone un reto. Él te prestará una cantidad N de perlas, todas idénticas a simple vista y con el mismo peso. Sin embargo, entre ellas habrá una que es diferente de las demás, porque es más liviana. A esta perla la llamaremos *perla falsa*, y el reto consiste en que deberás encontrarla.

Lamentablemente, la diferencia de peso es muy insignificante como para que la puedas detectar con las manos. ¡Qué suerte que Diego está dispuesto a prestarte su balanza!

La balanza tiene dos platos, y para que funcione correctamente, en cada plato debes poner la misma cantidad de perlas. Como Addis encuentra injusto que uses la balanza, te indica que te regalará más perlas mientras menos veces la utilices. Por esto, tienes que encontrar la perla falsa con la menor cantidad de pesajes.

Resumen de la tarea

Escribe un programa que encuentre la perla falsa entre las N que Addis te ha prestado. Se te pedirá que resuelvas múltiples casos de prueba, donde cada uno corresponde a un conjunto diferente de perlas prestadas.

Interludio: Este ejercicio requiere trabajar con un programa evaluador

Para resolver este problema, querido concursante, sólo necesitarás implementar correctamente una función en C++, ya que el ejercicio trae consigo código que se encarga de realizar la lectura de los datos entrada y emitir la salida necesaria para que te evaluemos. Este código que facilita el trabajo se llama programa evaluador (en inglés, **grader**).

Adjunto a este enunciado, o en tu computadora, deberías tener tres archivos:

- grader.cpp, que es el programa evaluador. No necesitas tocar este archivo.
- Perlas.h, que es un archivo de definición de funciones y constantes, utilizadas por el evaluador. No necesitas tocar este archivo.
- Perlas.cpp, que es el único archivo donde deberás escribir código para resolver el problema.

Para realizar pruebas en tu propio equipo, puedes hacer una de dos cosas: 1) crear un proyecto de CodeBlocks y añadirle los tres archivos ya mencionados, ó 2) compilar mediante la línea de comandos usando la siguiente instrucción:

```
g++ -DEVAL -static -O2 -std=c++11 -o Perlas grader.cpp Perlas.cpp
```

De cualquiera manera, al final tendrás un programa que podrás ejecutar. Este programa leerá datos para los diferentes casos de prueba, y se encargará de evaluar el desempeño del código que escribiste en el archivo Perlas.cpp.

Ya que vas a necesitar usar el grader para probar la función que escribas, el último subtítulo de este documento describe el formato en que este programa lee su entrada y lo que imprime como salida.

Descripción de la tarea

Tú programa deberá implementar la función `find_odd_pearl`, de la cual ya existe una definición a medias en el archivo Perlas.cpp. El grader llamará a `find_odd_pearl(N)` exactamente una vez por cada caso de prueba.

Para propósitos del problema, las perlas están numeradas desde 1 hasta N inclusive.

- `find_odd_pearl(N)`
  - Esta función se llama una vez por cada caso de prueba.
  - Su único argumento N te indica la cantidad de perlas que Addis te prestó, entre las cuales obviamente hay una falsa.
  - Deberás determinar cuál es la perla falsa, realizando múltiples llamadas a una función que el grader pone a tu disposición: `weigh`. Más detalles sobre ella abajo.
  - El resultado de esta función debe ser un entero que indique el número de la perla falsa.
- `weigh(A, B)` - corresponde a utilizar la balanza de Diego
  - A y B son vectores de enteros (`vector<int>`). Es decir, la función se utiliza pasándole dos vectores como argumentos. (No puedes usar arreglos estáticos `int[]`).
  - El contenido de A y B indican qué perlas se han de poner en el plato izquierdo de la balanza y qué perlas en el derecho, respectivamente. Por tanto, los enteros que se pongan en estos vectores deben corresponder a números de perlas (entre 1 y N).
  - La función devuelve uno de los siguientes enteros para indicar el resultado del pesaje:
    - -1 (la constante `WR_LESS`). Indica que el plato izquierdo es el más liviano.
    - 0 (la constante `WR_EQUAL`). Indica que ambos platos pesan lo mismo.
    - 1 (la constante `WR_GREATER`). Indica que el plato izquierdo es el más pesado.
    - 2 (la constante `WR_INVALID`). Indica que los vectores A y B no contienen información correcta para realizar un pesaje. Esto ocurre si no tienen la misma cantidad de perlas, si contienen algún entero que no está entre 1 y N, o si alguna perla aparece más de una vez en la comparación.

Ejemplo

Imaginemos el caso en que Addis nos da $N = 1$ perla.

Claramente, como sólo hay una única perla, esta tiene que ser la perla falsa. Por ello, podemos retornar 1 (uno) desde nuestra función.

Imaginemos el caso en que Addis nos da $N = 2$ perlas.

Sabemos que una perla es falsa, y para determinar cuál vamos a necesitar pesarlas entre sí. Para esto llamamos a `weigh({1}, {2})`, de modo que en el plato izquierdo de la balanza esté la perla 1 y en el plato derecho la perla 2. Si el resultado de `weigh` es `WR_LESS`, esto quiere decir que la perla 1 es la más liviana y por tanto es la perla falsa; caso contrario, debe ser la perla 2.

Subtareas

Para todas las subtareas se cumple que $1 \leq N \leq 1000$.

Subtarea	**Puntos**	**Restricciones**
1	5	Sin límite de llamadas a `weigh`.
2	10	Hasta 500 llamadas a `weigh`.
3	35	Hasta 10 llamadas a `weigh`.
4	50	Hasta 7 llamadas a `weigh`.

Una solución que no encuentre la perla falsa en todas las situaciones posibles recibirá cero puntos, sin importar la cantidad de llamadas a `weigh` que haga.

Formatos de entrada y salida del grader

El grader trabaja leyendo casos de prueba hasta que alcanza el final de archivo (EOF).

Un caso de prueba se describe mediante dos números enteros N y M separados por varios espacios o saltos de línea, donde N es la cantidad de perlas que Addis te está prestando y M es el número de la perla falsa.

Por ejemplo, una entrada que consista de los casos de prueba `5 3` (5 perlas, la tercer perla es la falsa) y `8 1` (8 perlas, la primera perla es falsa) puede escribirse así:

```
5 3
8 1
```

El grader imprimirá una línea con dos enteros por cada caso. El primer entero es el resultado de la llamada a `find_odd_pearl`, el segundo es la cantidad de llamadas a `weigh` realizadas.

Problema E
Robot

Tiempo límite: *1 sec*
Memoria límite: *512 MB*

Lua ha decidido adentrarse en el campo de la robótica y participar en la Olimpiada Boliviana de Robótica (hará una demostración, ya que no esta en colegio y no puede competir :( ). El se ha enterado de que una de las pruebas consiste en hacer que tu robot pueda salir de un laberinto siguiendo una secuencia de movimientos que él debe programar. Los movimientos posibles son **S** avanzar, **L** girar el robot 90 grados a la izquierda y **R** girar el robot 90 grados a la derecha.

Ya que Lua quiere demostrar que "si la gente supiera programar esto sería 1000 veces más fácil", ha decidido hacer un programa universal, una secuencia de movimientos tal que no importa cual sea el laberinto que le pongan (siempre y cuando tenga solución), ni donde comience su robot el siempre salga del laberinto, además ha decidido agregarle un desafío extra el sólo usará una secuencia de movimientos de a lo mucho 10 instrucciones, si se le acaban las instrucciones y no ha logrado salir su programa se volverá a ejecutar una y otra vez hasta que su robot salga del laberinto. Ayuda a Lua a saber cual será la secuencia de movimientos correcta para salir de cualquier laberinto que le propongan tal que no tenga más de 10 movimientos... Ok, es broma es una tarea demasiado difícil para ti aún te falta mucho para lograr algo así, y Lua esta seguro que lo logrará solo.

Entonces te preguntarás para que quiere Lua tu ayuda... La respuesta es simple a el le da pereza asegurarse de que el comité organizador de la OBR proponga laberintos en los que realmente se pueda salir comenzando en cualquier lugar. Entonces Lua te dará la información del tipo de laberinto (un número), el laberinto (una matriz de $N * M$ donde un '.' representa una casilla vacía, una **'X'** un obstáculo, y **'E'** la salida del laberinto) a continuación te dará la longitud de la cadena y la cadena, la secuencia que el programa de Lua halló para salir del laberinto (que según cuenta la leyenda es perfecta). El robot siempre esta mirando arriba, hacia el norte; no olvides que los programas de Lua, si bien tienen solo K instrucciones ($2 \leq K \leq 10$), una vez terminados si el robot no llego a la salida vuelven a ejecutarse.

Tarea

Subtarea 1: (15 ptos.)

Para el tipo de laberinto 1, además de darte las dimensiones de N y M del laberinto, y su secuencia de movimientos se te dará R y C que representan la fila y columna donde estará ubicado siempre el Robot ($0 \leq R < N$ y $0 \leq C < M$). Debes responder si es posible salir del laberinto siguiendo la secuencia de Lua, con un "SI" o un "NO" si es imposible.

Ejemplo de Entrada

```
1
3 4
E...
```

```
X.X.
....
3
SSL
1 2
```

Ejemplo de salida

```
SI
```

Subtarea 2: (25 ptos.)

Si el tipo de laberinto es 2 ($N \leq 10$, $M \leq 10$), debes decir si el robot de Lua logrará salir de toda casilla libre siguiendo la secuencia de movimientos con un "SI", si es que no es posible salir de toda casilla libre, debes decir el número de casillas de las que es posible salir.

Ejemplo de entrada

```
2
3 4
E...
X.X.
....
3
SSL
```

Ejemplo de salida

```
5
```

Subtarea 3: (60 ptos.)

Si el tipo de laberinto es 3 hacer lo mismo que en subtarea 2 sólo que ahora $N \leq 100$, $M \leq 100$

Ejemplo de Entrada

```
3
3 3
E..
...
...
3
RSS
```

Ejemplo de Salida

```
SI
```