ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2021-20. Proyecto
Entrega: Lunes, Diciembre 13 de 2021, 10:00 (por Bloque Neon)
Trabajo por parejas (de la misma sección)

0 OBJETIVOS

- Diseñar soluciones computacionales para problemas.
- Estimar costos de las soluciones planteadas.
- Implementar soluciones.

Se premiarán las mejores soluciones y se castigarán las peores, en cuanto a eficiencia en tiempo y espacio.

1 CONDICIONES GENERALES

Hay tres problemas para resolver mediante soluciones implementadas en *Java* o *Python*.

Para cada problema se pide:
- Análisis temporal y espacial.
- Una implementación en Java o Python

2 PROBLEMAS

A Convolución ponderada

Las reservas económicas de Pisalia cumplen unas leyes de variación que los científicos han logrado establecer con claridad. De hecho, se ha encontrado que `r`, la reserva económica de Pisalia, varía de año a año como sigue (los años se modelan con números naturales). Los parámetros `A`, `B`, `C`, `D` son números reales:

```
r(0)      = A
r(1)      = B
r(k+2)    = C*r(k) + D*r(k+1), k ≥ 0
```

Los economistas del gobierno están interesados en estimar `cp`, la *convolución ponderada* de la reserva en el tiempo n, definida como

```
cp(r,n) = (+k| 0≤k≤n: r(k)*r(n-k)), n ≥ 0.
```

Problema

Se quiere desarrollar un programa que calcule la convolución ponderada `cp(r,n)`, dados `A`, `B`, `C`, `D` y `n`.

El énfasis de este problema está en minimizar el costo espacial.

Ejemplo:

Para

```
A = 0, B = 1, C = 1, D = 1
```

la función `r` coincide con `F`, la sucesión de Fibonacci. Para `n = 5`, se tiene que:

```
     cp(r,5)
=
     cp(F,5)
=
     0*5 + 1*3 + 1*2 + 2*1 + 3*1 + 5*0
=
     10.
```

B Grafos BC

Un grafo *bipartito* es un grafo `G(V,E)` no dirigido tal que `V` se puede partir en dos conjuntos disyuntos `X` y `Y`, de modo que todo arco en `E` una un vértice de `X` con un vértice de `Y`. El *diferencial* de `G` se define como la diferencia (un número entero no negativo) entre el número de nodos en `X` y el número de nodos en `Y`.

Un grafo no dirigido es *conexo*, si existe un camino entre cada par de vértices.

Una grafo *BC* es un grafo bipartito conexo.

Un grafo BC `G` es una *BC-suma* de grafos BC G_1, G_2, …, G_n, n>0, si cada G_k es un subgrafo[1] de `G`. Cada uno de los G_k's es un *BC-sumando*. Una BC-suma puede agregar arcos que no estén en los BC-sumandos, pero todo vértice en `G` debe estar en algún BC-sumando.

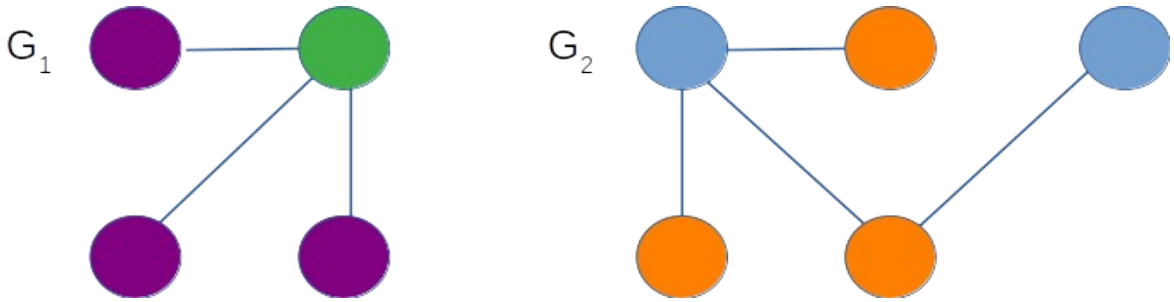
Problema

Se quiere desarrollar un programa que construya `G(V,E)`, un grafo BC a partir de `n` grafos BC, $G(V_k,E_k)$, $0<k \leq n$. Los conjuntos de vértices V_k son disyuntos por pares.

La construcción de `G` debe tener diferencial mínimo, dentro de las posibles BC-sumas.

Ejemplo:

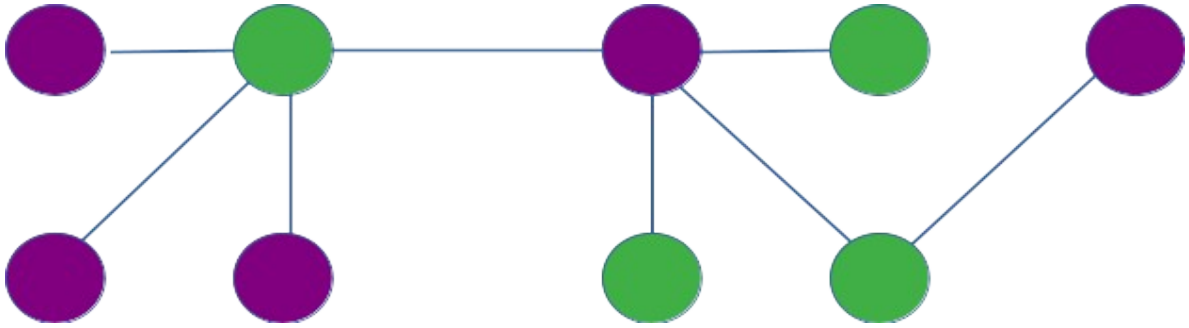
Considérense los grafos BC



1 Para dos grafos `G(V1,E1)`, `G(V2,E2)` se dirá que `G(V1,E1)` es *subgrafo* de `G(V2,E2)` si `V1 V2 y E1={(a,b):E2| a,b V1}`.

(los colores destacan cómo se parten los vértices en cada uno, de manera que solo hay arcos entre nodos de diferente color)

El grafo `G`:



resulta ser una BC-suma de `G1` y `G2`. El diferencial de `G` es 1 (= 5 morados – 4 verdes), que es, en este caso el mínimo posible.

C Texto mínimo reconstruible

El Servicio Secreto tiene una manera de transmitir mensajes de una fuente a un receptor que funciona curiosamente. Cuando la fuente tiene un texto para enviar al receptor, en realidad manda muchos mensajes parciales de un tamaño fijo `k`, `k>0`, que sean *subcadenas* del texto que se quiere enviar. El receptor debe entender como mensaje un texto que tenga tamaño mínimo del que, efectivamente, sean subcadenas todos los mensajes parciales recibidos.

El Servicio Secreto sabe que, de esta manera, el mensaje que se entienda no va a ser necesariamente el mismo originó la transmisión de la fuente. Sin embargo, confía en que sus expertos puedan descubrir el mensaje pretendido a partir de una posible reconstrucción minimal.

Problema

Dadas `n` cadenas de tamaño `k`, `n,k>0`, encontrar un texto de tamaño mínimo tal que toda cadena dato sea subtexto del texto respuesta.

Ejemplo:

Suponga que `n=4`, `k=3` y que se reciben las cadenas `"aab"`, `"baa"`, `"aaa"` y `"bbb"`.

Aunque el texto `"baabbbaaa"` tiene cada cadena recibida como subtexto, tiene 9 caracteres y no es minimal en longitud. Los textos `"baaabbb"` y `"bbbaaab"` tienen 7 caracteres cada uno, y son minimales.

3 ENTRADA / SALIDA DE DATOS

En todas las soluciones que se presenten, la lectura de los datos de entrada se hace por la entrada estándar; así mismo, la escritura de los resultados se hace por la salida estándar.

Puede suponer que ninguna línea de entrada tiene espacios al principio o al final, y que los datos que se listan en cada línea están separados por exactamente un espacio.

El fin de la entrada de cada problema se identifica porque no hay más casos para leer.

A continuación, para cada problema, se establecen parámetros que definen su tamaño y formato de lectura de los datos, tanto de entrada como de salida.

3.1 Problema A: Convolución ponderada

- Tamaño del problema: n (el índice de la convolución ponderada que debe calcularse)
- Condiciones de los casos de prueba: $0<n<10^2$, $-1 \leq A, B, C, D \leq 1$.

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba se describe con una línea de texto de la forma

`n A B C D`

donde n es un número natural y A, B, C y D son números reales. Los números reales que pueden ser datos son números racionales expresados en notación decimal, con un punto separando la parte entera de la fraccionaria; la parte fraccionaria tiene entre 1 y 4 decimales.

El final de la entrada se indica con una linea que solo contenga el número 0

Descripción de la salida

Por cada caso para resolver, imprimir una línea de respuesta.

La línea de respuesta debe ser de la forma
X

donde X es el valor de la convolución ponderada `cp(r,n)` correspondiente redondeado a 4 cifras decimales. El redondeo debe ser al número racional de 4 cifras más cercano, v.gr., `34.454638` redondea a `34.4546`, `34.454688` redondea a `34.4547`, `34.454999…` redondea a `34.4550`. En caso de dos números racionales más cecanos de 4 cifras, se escoge el mayor, v.gr. `34.4555500…` redondea a `34.4556`.

Ejemplos de entrada / salida

Entrada	Salida

`10 0.0 1.0 1.0 1.0`
`15 0.3677 -0.7157 0.3738 -0.1088`
`0`
	`235.0000`
	`-0.0202`

3.2 Problema B: Grafos BC

- Tamaño del problema: n (el número de grafos a sumar)
- Condiciones de los casos de prueba: $0<n<10^2$, $1 \leq |V_k|, |E_k| \leq 10^3$

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso contiene varias líneas. La primera línea contiene un valor entero n, el número de grafos BC que se deben sumar.

A continuación, la entrada contiene n líneas, cada una describiendo un grafo para sumar. La línea k, $1 \le k \le n$, tiene una lista de números enteros

v e x_1 y_1 … x_e y_e

que se deben entender así:

v : número de vértices del grafo ($|V_k|$ para el grafo G_k), $1 \le v \le 10^3$

e : número de arcos del grafo ($|E_k|$ para el grafo G_k)

x_i : un número entero que identifica el vértice inicial del i-arco de G_k , $1 \le i \le e$; $0 \le x_i < v$.

y_i : un número entero que identifica el vértice final del i-arco de G_k , $1 \le i \le e$; $0 \le y_i < v$.

Se puede suponer que cada línea describe un grafo BC. Si bien algunos de los identificadores de los vértices en dos grafos diferentes pueden coincidir (v.gr. hay un vértice identificado 23 en dos grafos diferentes), debe entenderse que se trata de vértices diferentes.

El final de la entrada se indica con una linea que solo contenga el número 0

Descripción de la salida

Para cada caso de prueba, imprimir una línea con un número entero no negativo, correspondiente al diferencial de un grafo G que sea una BC-suma de los grafos de entrada y que tenga diferencial minimal.

Ejemplo de entrada / salida

Entrada	Salida
2 4 3 0 1 1 2 2 3 5 4 0 1 0 2 0 3 2 4 3 1 0 1 0 2 1 0 1 0	1 0

3.3 Problema C: Texto mínimo reconstruible

- Tamaño del problema: <n,k> donde n es el número de subcadenas y k es el tamaño de cada cadena
- Condiciones de los casos de prueba: `0<n<1000 y 0<k<100`

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba empieza con una línea de texto de la forma

n k

a la cual siguen n líneas de texto, cada una con una cadena de caracteres ASCII de tamaño k.

El final de la entrada se indica con una linea que solo contenga el número 0

Descripción de la salida

Por cada caso de prueba para resolver, imprimir una línea de respuesta que contenga un texto reconstruido, de tamaño minimal.

Ejemplo de entrada / salida

Entrada	Salida
4 3 aab baa aaa bbb 6 4 nfid conf cial denc onfi enci 0	bbbaaab confidencial

4 ENTREGABLES

El proyecto puede desarrollarse por grupos de uno o dos estudiantes de la misma sección. La entrega se hace por bloque neon (una sola entrega por grupo de trabajo).

El grupo debe entregar, por bloque neon, un archivo de nombre `proyectoDAlgo.zip`. Este archivo es una carpeta de nombre `proyectoDAlgo`, comprimida en formato `.zip`, dentro de la cual hay archivos fuente de soluciones propuestas y archivos que documentan cada una de las soluciones.

4.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* o en Python

Para el problema `X`, siendo `X {A,B,C}`:

- Entregar un archivode código fuente en *Java* (`.java`) o python (`.py`) con su código fuente de la solución que se presenta.
- Incluir como encabezado de cada archivo fuente un comentario que identifique el (los) autor(es) de la solución.
- Denominar `ProblemaX.java` o `ProblemaX.py` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* o Spyder durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo por cada solución. El archivo debe poderse compilar y ejecutar independientemente (sin depender de ninguna estructura de directorios, librerías no estándar, etc.).

4.2 Archivos que documentan soluciones propuestas

Toda solución propuesta debe acompañarse de un archivo que la documente, con extensión `.pdf`. El nombre del archivo debe ser el mismo del código correspondiente. Por ejemplo, si incluyó un archivo `ProblemaB.java`, como solución para el problema B, debe incluirse un archivo `ProblemaB.pdf` que lo documente.

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Nombre de autor(es)
Identificación de autor(es)

1 Algoritmo de solución
Explicación del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó.
Deseable:
Anotación (contexto, pre-, poscondición, …) para cada subrutina o método que se use.

2 Análisis de complejidades espacial y temporal
Cálculo de complejidades y explicación de las mismas. Debe realizarse un análisis para cada solución entregada.

3 Comentarios finales
Comentarios al desempeño observado de la solución.

Téngase en cuenta que los análisis de 2 tienen sentido en la medida que la explicación de 1 sea clara y correcta. No se está exigiendo formalismo a ultranza, pero sí que, como aplicación de lo estudiado en el curso, se pueda describir un algoritmo de manera correcta y comprensible.

No describa un algoritmo con código GCL a menos que lo considere necesario para explicarlo con claridad. Y, si lo hace, asegúrese de incluir aserciones explicativas, fáciles de leer y de comprender.