ISIS 1105 Diseño y Análisis de Algoritmos
Semestre 2015-10. Proyecto
Prof. Rodrigo Cardoso
Entrega: Lunes, Mayo 25 de 2015, 12:00 (por Sicua+)
Trabajo individual o por parejas.

0 OBJETIVOS

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

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*. Para el primero de ellos se proporciona una solución ingenua que servirá de referencia para las que se entreguen como respuesta..

Para cada problema se pide:
- Análisis temporal y espacial de la solución ingenua que se proporciona.
- Una solución java. Para el primer problema, la solución que se presente debe ser mejor –en complejidad temporal- que la solución ingenua.
- En todas las soluciones, la entrada de datos y la salida de resultados deben hacerse de forma análoga a la que se realiza en la solución ingenua. La idea con esto es resaltar que el interés de las soluciones que se presenten esté en el algoritmo subyacente, más que en la lectura de los datos.

2 PROBLEMAS

A Cuadrados y rectángulos máximos

Un servicio de interpretación de fotografías digitales analiza fotos de tamaño `M×N` pixeles. Cada pixel puede ser blanco o negro. El servicio está interesado en encontrar, para cada foto, el área del cuadrado de pixeles blancos más grande, así como la del rectángulo de pixeles blancos más grande contenidos en la foto.

El siguiente ejemplo muestra una foto de tamaño `4×6`, en la que el cuadrado más grande tiene un área de `4` y el rectángulo más grande tiene un área de `5`.

1	1	0	1	1	0
1	0	1	1	1	0
1	0	0	0	1	1
1	1	1	1	1	0

B Compatibilidades

MatEx es una compañía ofrece el servicio de encontrar parejas *compatibles* o *pasablemente compatibles*, basados en los gustos de cada uno de los posibles candidatos. La idea es sencilla: cada posible cónyuge declara sus preferencias sobre varios criterios de interés común. La idea de MatEx es la de suponer que una pareja será exitosa como matrimonio en la medida que sus preferencias sean compatibles o, al menos, casi compatibles.

Suponga que `X` e `Y` son cualidades que son consideradas en la calificación. Si una persona declara que `X>Y`, significa que esta persona prefiere la cualidad `X` a la cualidad `Y` (no quiere decir que su eventual pareja deba tener alguna cualidad, solo es su opinión sobre `X` e `Y`). Las preferencias son transitivas, i.e., si `X>Y` y `Y>Z`, se entiende que `X>Z`.

Dos personas son *compatibles* si sus preferencias son completamente consistentes. El chequeo de consistencia consiste en poder afirmar que las cualidades calificadas por cada una de las dos personas se pueden mezclar en una sola lista que refleje las preferencias de las dos personas. En este caso, si una persona dijo `X>Y`, la cualidad `X` aparece antes de la cualidad `Y` en la lista mezclada.

Si las personas no son compatibles, todavía pueden ser *pasablemente compatibles*. Esto quiere decir que hay una manera de mezclar en una lista las preferencias de las dos personas, exceptuando exactamente una de las preferencias de alguna de las personas, que refleje las preferencias originales con excepción de la que se desechó.

Si una pareja no es compatible ni pasablemente compatible, se dice que es *no compatible*.

Cada integrante de la posible pareja declara sus preferencias con un conjunto de listas en las que indica qué prefiere y en qié orden. Las cualidades no mencionadas no importan a la persona que califica.

Por ejemplo, Alicia y Bernardo tienen las siguientes cualidades para observar en una posible pareja y calificar:

```
ciclista, culto, entusiasta, gourmet, malabarista, navegante,
películas, organizado, acertijos, rico, teatro, nadador.
```

Y declaran que:
Alicia:

```
organizado > acertijos > rico, nadador > teatro, rico > películas
```

Bernardo:

```
navegante > películas > acertijos, rico > teatro
```

Nótese que Alicia prefiere `acertijos` a `películas`. Por otro lado, no tiene preferencia entre `nadador` y `películas`, y no le importa que su eventual pareja opine sobre `malabarista` o `navegante`.

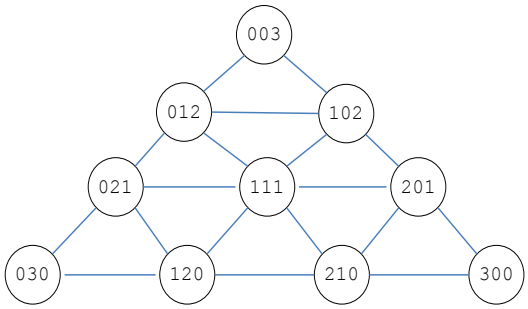
En este caso, Alicia y Bernardo no son compatibles. Para comprobarlo, una lista mezclada debería tener `rico > películas` (por Alicia), `películas > acertijos` (por Bernardo) y `acertijos > rico` (por Alicia). Esto es contradictorio, porque implicaría que `rico > rico`.

Sin embargo, la pareja sí es pasablemente compatible. para ver esto, basta que se elimine la preferencia `rico > películas` (de Alicia) para que ahora se pueda construir la lista mezclada `organizado > navegante > películas > acertijos > rico > nadador > teatro`.

MatEx necesita asistencia computacional para desarrollar un software que determine si una pareja de clients es compatible, pasablemente compatible o incompatible.

C El juego de la Y

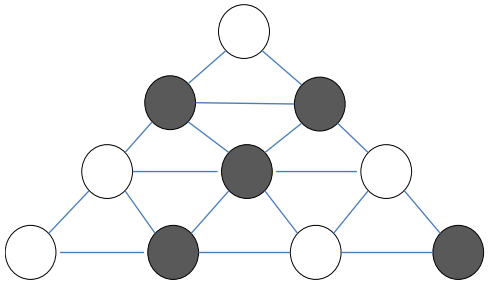
El juego de la Y se desarrolla sobre una `n`-malla triangular (`n` es el *orden* de la malla). Una `3`-malla se muestra en la siguiente figura:



En general, una `n`-malla tiene `(n+1)(n+2)/2` nodos con "coordenadas baricéntricas" no negativas `(x,y,z)`, donde `x+y+z=n`. Las coordenadas son asignadas de modo que, a lo largo de caminos diagonales descendentes de derecha a izquierda, la coordenada `x` se mantiene constante, la coordenada `y` aumenta en `1` y la coordenada `z` disminuye en `1` (nótese que se mantiene invariante `x+y+z=n)`. Una situación simétrica se da sobre diagonales descendentes de izquierda a derecha (donde las `y` permanecen constantes) y sobre horizontales (donde las `z` permanecen constantes).

Se dice que un nodo `(x,y,z)` en una `n`-malla está sobre el lado `x` (respectivamente, `y`, `z`) si y solo si `x=0` (respectivamente `y=0`, `z=0`).

El juego de la Y se realiza entre dos jugadores, Blanco y Negro, que colocan fichas del color de sus nombres en los nodos de la malla, siguiendo unas reglas algo complicadas. Al final, cada nodo de la malla tiene una ficha de color blanco o una de color negro. El *ganador* es el jugador cuyo color haya formado una Y, es decir, sus fichas están colocadas sobre un conjunto conexo de nodos que tiene un punto en cada lado de la `n`-malla. Por ejemplo, la siguiente figura muestra una situación final en la que gana Negro:



El problema que se debe resolver es el de, dada una posición final del juego, decidir quién lo gana.

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. Más aún: el código fuente escrito en la solución ingenua del problema A se debería reutilizar -en lo posible- para implantar la lectura de la entrada / salida en las nuevas soluciones.

A continuación, para cada problema, se establecen parámetros que definen su tamaño y umbrales de desempeño. Estos son medidas de tamaño de los datos que servirán para hacer, eventualmente, pruebas pequeñas, medianas y grandes sobre las souciones que se presenten.

Además de lo anterior, se define el formato que deben cumplir los datos de entrada y de salida.

4.1 Problema A (Cuadrados y rectángulos máximos en una foto)

Umbrales

- Tamaño del problema: `M, N` (dimensiones de la foto).
- Condiciones de los casos de prueba *small* : `1 ≤ M,N ≤ 10`
- Condiciones de los casos de prueba *medium* : `1 ≤ M,N ≤ 100`
- Condiciones de los casos de prueba *big* : `1 ≤ M,N ≤ 1000`

Descripción de la entrada

La entrada contiene varios casos de prueba. Cada caso de prueba comienza con una línea que contiene dos números enteros positivos:
`M N`

que representan el número de filas y columnas de bits de la foto. A continuación siguen `M` líneas cada una con `N` caracteres en `{0,1}`, de modo que el `j`−simo carácter de la fila `i` representa el bit `(i,j)` de la foto.

Puede suponer que ninguna línea de la entrada tiene espacios al principio o al final, y que los bits que se listan no tienen espacios entre sí.

El fin de la entrada se identifica porque se intenta leer un caso de prueba (que no debe analizarse) que empieza con la línea
`0 0`

Descripción de la salida

Una línea con dos números, separados por un blanco:
`cmax rmax`

que representan el área del cuadrado más grande y el área del rectángulo más grande, conformados enteramente por bits en `1`.

4.2 Problema B (Compatibilidades)

Umbrales

- Tamaño del problema: `N` (número de cualidades sobre las que se puede opinar).
- Condiciones de los casos de prueba *small* : `1 ≤ N ≤ 10`
- Condiciones de los casos de prueba *medium* : `1 ≤ N ≤ 50`
- Condiciones de los casos de prueba *big* : `1 ≤ N ≤ 100`

Descripción de la entrada

La entrada contiene varios casos de prueba, cada uno de ellos definido con líneas de datos que establecen las preferencias de una pareja. Para cada caso:
La primera línea contiene dos cadenas no vacías de caracteres, separadas por un blanco, que representan los nombres de las personas de la pareja.
La segunda línea tiene una cadena de la forma

$q_{11}\ q_{12} \cdots q_{1r1}$, $q_{21}\ q_{22} \cdots q_{2r2}$, … , $q_{m1}\ q_{12} \cdots q_{m,rm}$;

donde q_{ij} es el identificador de la j-sima cualidad de la i-sima lista de preferencias que declara la primera persona. Con esta línea se representa el hecho de que esta persona opine que
$q_{11} > q_{12} > \cdots > q_{1r1}$
$q_{21} > q_{22} > \cdots > q_{2r2}$
…
$q_{m1} > q_{12} > \cdots > q_{m,rm}$

La tercera línea tiene un formato igual al de la segunda línea, pero se refiere a la opinión de la segunda persona.

Cada cualidad se representa con una cadena de 1 a 10 caracteres. Cada pareja opina sobre, a lo sumo, 100 cualidades diferentes. Se garantiza que los conjuntos de preferencias de cada persona son libres de ciclos (i.e., cada persona es auto compatible).

Se puede suponer que ninguna línea de la entrada tiene espacios al principio o al final, y que los números que se listan están separados por un solo espacio.

El fin de la entrada se indica con una cadena que contiene dos asteriscos separados por un espacio (un caso que no debe analizarse):
```
* *
```

Descripción de la salida

Para cada caso de prueba se debe imprimir una línea con sólo un carácter, C, P o N, indicando que la pareja es compatible, parcialmente compatible o no compatible.

4.3 Problema C (El juego de la Y)

Umbrales

- Tamaño del problema: n
- Condiciones de los casos de prueba *small* : $1 \leq n \leq 5$
- Condiciones de los casos de prueba *medium* : $1 \leq n \leq 10$
- Condiciones de los casos de prueba *big* : $1 \leq n \leq 20$

Descripción de la entrada

La entrada contiene varios casos de prueba. La primera línea de cada caso de prueba contiene dos números enteros (separados por un blanco):
```
n m
```

donde n es el orden de la malla y m el número de posiciones de la malla que tienen una ficha negra (una ficha de Negro), con $0 \leq n \leq 20$ y $0 \leq m \leq (n+2)(n+1)/2$. Después de esta línea siguen m líneas, cada una con tres valores enteros no negativos x, y, z, representando las coordenadas baricéntricas (x,y,z) de un nodo de la malla que tiene una ficha negra.

El fin de la entrada se identifica porque se intenta leer un caso de prueba (que no debe analizarse) que empieza con la línea
```
0 0
```

Descripción de la salida

Los casos de prueba se deben analizar en el orden en que se presentan en la entrada, Para cada uno se debe imprimir una línea con la letra B o con la letra N, dependiendo de que sea el jugador Blanco o el Negro quien gana, respectivamente.

5 ENTREGABLES

El proyecto puede desarrollarse por grupos de uno o dos estudiantes. La entrega se hace por Sicua+, y debe tener cuidado de inscribir los miembros de cada grupo de acuerdo con los mecanismos provistos por la plataforma.

El grupo debe entregar, por Sicua+, 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.

5.1 Archivos fuente de soluciones propuestas

Todos los programas implementados en *Java* deben compilar en *JDK 8*.

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

- Entregar un archivo *Java* (`.java`) con su código fuente, por cada solución que se quiera presentar.
- Incluir como encabezado de cada archivo fuente un comentario que identifique la sección, el grupo y los autores de la solución.
- Denominar `ProblemaX_1.java` el archivo de la solución que se presente.

Nótese que, si bien puede utilizarse un *IDE* como *Eclipse* durante el desarrollo del proyecto, la entrega requiere incluir solo un archivo `.java` por cada solución. El formato presentado en la solución ingenua debe seguirse como modelo de presentación de las nuevas soluciones.

5.1 Archivos que documentan soluciones propuestas

Toda solución propuesta debe acompañarse de un archivo que la documente, con extensión `.doc`, `.docx` o `.pdf`. El nombre del archivo debe ser el mismo del código *Java* correspondiente. Por ejemplo, si incluyó un archivo `ProblemaB_1.java`, como solución para el problema B, debe incluirse un archivo `ProblemaB_1.docx` que lo documente.

Un archivo de documentación debe contener los siguientes elementos:

0 Identificación
Sección
Grupo Sicua+
Nombre de autor(es)
Identificación de autor(es)

1 Algoritmo de solución
Explicación informal del algoritmo elegido. Si hubo alternativas de implantación diferentes, explicar por qué se escogió la que se implementó.
Deseable:
Descripción GCL del algoritmo de la solución implementada.
Anotación (contexto, pre- y 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
Para el problema A debe analizarse y compararse la solución ingenua con la nueva solución que se entrega. En este caso, determinación de un valor, para el tamaño del problema, por

encima del cual la solución ingenua se desempeña peor que la solución considerada. incluir una justificación de esta respuesta (con gráficas, experimentos, estadísticas, tablas, ...).

3 Comentarios finales

Comentarios al desempeño observado de la solución. Por ejemplo, hacer un análisis de la clase de complejidad a la que pertenece el problema.