TP2: Indiana Jones en búsqueda de la complejidad esperada (cont...)

Fecha de entrega: viernes 7 de octubre, hasta las 18:00 hs.

Este trabajo práctico consta de varios problemas y para aprobar el mismo se requiere aprobar todos los problemas. La nota final será un promedio ponderado de las notas finales de los ejercicios y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. De ser necesario (o si el grupo lo desea), el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega. Lo grupos deben ser de tres o cuatro personas.

Para cada ejercicio se pide encontrar una solución algorítmica al problema propuesto y desarrollar los siguientes puntos:

1. Describir detalladamente el problema a resolver dando ejemplos del mismo y sus soluciones.
2. Explicar de forma clara, sencilla, estructurada y concisa, las ideas desarrolladas para la resolución del problema. Para esto se pide utilizar pseudocódigo y lenguaje coloquial combinando adecuadamente ambas herramientas (**¡sin usar código fuente!**). Se debe también justificar por qué el procedimiento desarrollado resuelve efectivamente el problema.
3. Deducir una cota de complejidad temporal del algoritmo propuesto (en función de los parámetros que se consideren correctos) y justificar por qué el algoritmo desarrollado para la resolución del problema cumple la cota dada. Utilizar el modelo uniforme salvo que se explicite lo contrario.
4. Dar un código fuente claro que implemente la solución propuesta. El mismo no sólo debe ser correcto sino que además debe seguir las *buenas prácticas de la programación* (comentarios pertinentes, nombres de variables apropiados, estilo de indentación coherente, modularización adecuada, etc.). Se deben incluir las partes relevantes del código como apéndice del informe impreso entregado.
5. Realizar una experimentación computacional para medir la performance del programa implementado. Para ello se debe preparar un conjunto de casos de test que permitan observar los tiempos de ejecución en función de los parámetros de entrada. Deberán desarrollarse tanto experimentos con instancias aleatorias (detallando cómo fueron generadas) como experimentos con instancias particulares (de peor/mejor caso en tiempo de ejecución, por ejemplo). Se debe presentar **adecuadamente** en forma gráfica una comparación entre los tiempos medidos y la complejidad teórica calculada y extraer conclusiones de la experimentación.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación. **Solo se permite utilizar 1 lenguaje para todo el TP.**

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Por otro lado, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP 2: Apellido_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

Problema 1: Laberinto

Indiana Jones continúa en su expedición en la fortaleza de alguna civilización antigua. Mientras llenaban las mochilas con los tesoros que más les convenian encontraron un mapa peculiar. El mapa se parece mucho a un laberinto salvo por el hecho de que no estan conectados todos los puntos.

En el mapa hay un punto que parece indicar el lugar en donde se encuentran juntando tesoros, y hay un lugar que esta indicado con una crúz. Intrigados por saber lo que se encuentra en ese lugar, se proponen como objetivo ir hacia ahi.

Como nuestro equipo vino equipado con pico y pala, pueden derribar algunas paredes. Por otro lado el equipo ya se encuentra cansado y no desea recorrer mucha distancia ni esforzarse rompiendo paredes.

Entonces, nos pidieron ayuda con lo siguiente: Teniendo un mapa indicando un punto de origen y un punto de destino, quieren caminar lo menos posible desde el origen al destino rompiendo a lo sumo una cierta cantidad de paredes.

Formato de entrada: La primer línea consta de tres enteros positivos F,C y P, donde F es la cantidad de filas en el mapa, C es la cantidad de columnas y P es la cantidad máxima de paredes que van a derribar. Las siguientes F filas contienen C caracteres. Un "." representa un lugar caminable, "#" representa una pared, "o" es el punto de origen, y "x" es el destino.

```
F C P
F_1
F_2
...
F_n
```

Formato de salida: Un número indicando la distancia que deben recorrer. Si no es posible ir del origen al destino devolver -1. La salida tendrá el siguiente formato:

```
D
```

Aclaraciones:
Se debe modelar el problema sobre un grafo y utilizarlo para resolver el problema.
Los movimientos son horizontales y verticales, no se pueden mover en diagonal.
Los "bordes" del mapa tendran un "#".
Pueden suponer las siguientes cotas para la entrada:
$4 \leq F, C \leq 10^3$
$0 \leq P \leq 100$
La complejidad temporal deberá ser a lo sumo $\mathcal{O}(FCP)$.

Ejemplos:

Entrada:
```
5 9 2
#########
#o#.#.#x#
#.#.#.#.#
#.#.#...#
#########
```
Salida: `10`

Entrada:
```
5 9 3
#########
#o#.#.#x#
#.#.#.#.#
#.#.#...#
#########
```
Salida: `6`

Problema 2: Juntando piezas

En su destino hacia la cruz en el mapa recorrieron varias salas distintas al atravezar las paredes. Se dieron que en cada sala se iban encontrando con un fragmento de una tabla con un mansucrito antiguo. Por lo tanto, antes de continuar, quieren juntar todos los fragmentos.

A medida que se iban abriendo camino se dieron cuenta que cada pared requiere cierto esfuerzo para abrir la pared. Teniendo un mapa con el esfuerzo que requieren las paredes, quieren encontrar cual es el mínimo esfuerzo que deben hacer para acceder a todas las salas.
Formato de entrada: La primer línea consta de dos enteros positivos F y C, donde F es la cantidad de filas en el mapa, y C es la cantidad de columnas. Las siguientes F filas contienen C caracteres. Un "." representa un lugar caminable, los números del 1 al 9 indican el esfuerzo en romper esa pared, un "#" representa una pared indestructible.

```
F C
F_1
F_2
...
F_n
```

Formato de salida: Un número indicando el esfuerzo que debe hacer el equipo para acceder a todas las salas. Si no es posible recorrer todas las salas imprimir un -1. La salida tendrá el siguiente formato:
`E`

Aclaraciones:
Al igual que el ejercicio anterior, las celdas no son vecinas diagonalmente.
Las paredes con más de 2 paredes adyacentes son indestructibles (contendran un "#").
Las paredes con menos de 2 paredes adyacentes son indestructibles (contendran un "#"). Dos salas se pueden conectar si existe una pared adyacente a las dos salas que se puede destruir (Es decir, no quedan conectadas si se rompen muchas parede adyacentes).Ver ejemplo 3.
Los "bordes" del mapa tendran un "#".
Pueden suponer las siguientes cotas para la entrada:
$1 \leq F, C \leq 10^4$
La complejidad temporal deberá ser a lo sumo $\mathcal{O}(FC \times log(FC))$.

Ejemplos:

Entrada:
```
5 9
#########
#.1.#.#.#
#.#.3.5.#
#.2.#.#.#
#########
```
Salida: `9`

Entrada:
```
5 9
#########
#.#.5.1.#
#.1.#2#.#
#.#.1.5.#
#########
```
Salida: `5`

Entrada:
```
6 9
#########
#...1.#.#
#...#.#.#
#...11..#
#..#.#..#
#########
```
Salida: `3`

Problema 3: Escapando

Luego de colectar todas las piezas Indy se encuentra conforme con lo obtenido, aunque todavía no comprende lo que dice, el material encontrado lo ayudará a conseguir subsidios para continuar con la investigación.

Ya se encuentran en el lugar donde estaba la cruz en el mapa, hay unos carritos sobre vias que parecen dirigirse hacia afuera de la fortaleza. De repente se esuchan unos ruidos muy fuertes desde adentro del

laberinto. Luego de romper todas las paredes internas buscando las partes de la tabla se dieron cuenta que se esta desplomando toda la estructura, por lo que deben hacer un escape rápido.

Al lado del carrito se encuentra un gráfico con la red de las vias. Hay puntos numerados que parecen indicar lugares donde los carritos pueden hacer paradas (estaciones). La estación 1 es donde se encuentran, y la última estación es donde quieren llegar. Las estaciones estan unidas con vías. Al lado de cada vía hay flechas con un número indicando el tiempo que tarda en moverse entre dos estaciones.

Ayudemos a escapar cuanto antes a todos antes de que mueran aplastados.

Formato de entrada: El formato de entrada contiene dos numeros. La cantidad de estaciones N, y la cantidad de vías M. Luego siguen M líneas, cada una contiene tres enteros A,B y C que indican que ir de A a B tarda C segundos. Las estaciones estan númeradas de 1 a N.

```
N M
A_1 B_1 C_1
...
A_M B_M C_M
```

Formato de salida: La primera línea deberá tener un entero T indicando el mínimo tiempo para ir de la estación 1 a la estación N. Si no es posible, devolver -1. En caso de que sea posible, la segunda línea deberá tener un entero S indicando la cantidad de estaciones que debe recorrer y la tercera línea contendra S enteros que indican la forma de escapar lo más rapido posible. La salida tendrá el siguiente formato:

```
T
S
E_1 E_2 ... E_S
```

Aclaraciones:

Pueden suponer las siguientes cotas para la entrada:

$2 \leq N \leq 10^4$

$1 \leq M \leq 10^6$

$1 \leq C \leq 10^3$

La complejidad temporal deberá ser a lo sumo $\mathcal{O}(N^2 log^2(N))$.

Ejemplos:

Entrada:
```
5 5
1 2 1
2 3 4
3 4 5
4 5 3
2 4 2
```
Salida:
```
6
4
1 2 4 5
```

Entrada:
```
9
1 2 8
1 3 4
2 4 2
2 5 1
3 5 1
4 5 1
5 4 10
4 6 1
5 6 7
```
Salida:
```
11
4
1 2 4 6
```

Entrada:
```
3 2
1 2 1
2 1 1
```
Salida:
```
-1
```