Técnicas algorítmicas en ingeniería del software

Grado en Ingeniería del Software (UCM)

Examen final de febrero — Curso 2017/2018

Normas de realización del examen

1. Debes desarrollar e implementar soluciones para cada uno de los tres ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc/domjudge/team`.

2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.

3. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.

4. Puedes descargar el fichero `http://exacrc/19672.zip` que contiene material que puedes utilizar para la realización del examen (transparencias de clase, implementación de las estructuras de datos, una plantilla de código fuente y ficheros de texto con los casos de prueba de cada ejercicio del enunciado).

5. Los ficheros con las implementaciones de las estructuras de datos están instalados en el juez, por lo que no es necesario subirlos como parte de tu solución (y conviene no hacerlo).

6. Los ejercicios están identificados con el nombre del tema de la asignatura en el que habrían aparecido si hubieran sido propuestos como parte de los ejercicios de la evaluación continua. Para obtener la máxima puntuación, las soluciones deberán seguir los criterios exigidos a los ejercicios de ese tema durante el curso (en cuanto a encapsulación, eficiencia, simplicidad, análisis de costes, etc.).

7. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.

8. Las notas de los ejercicios suman 8 puntos. La calificación obtenida en este examen será sumada a la obtenida por la evaluación continua (sobre 2 puntos) para obtener la calificación final de la asignatura.

9. Al terminar el examen, dirígete al puesto del profesor y rellena con tus datos la hoja de firmas que él tendrá. Muéstrale tu documento de identificación.

Primero resuelve el problema. Entonces, escribe el código.

— John Johnson

Comentar el código es como limpiar el cuarto de baño;
nadie quiere hacerlo, pero el resultado es siempre
una experiencia más agradable para uno mismo y sus invitados.

— Ryan Campbell

Ejercicio 1. Colas con prioridad y montículos (2 puntos)

Los calendarios o agendas electrónicas controlan nuestra vida diaria. Para personas como yo, que no se nos da bien la multitarea, es importante tener como mucho una tarea planificada para cualquier minuto de nuestra vida.

En mi calendario hay dos tipos de tareas: tareas únicas y tareas periódicas. Las tareas únicas tienen un instante de comienzo y otro de finalización, ambos incluidos en el tiempo de realización de la tarea. Las tareas periódicas tienen además de los instantes de comienzo y finalización de su primera aparición, un periodo de repetición. Se supone que estas tareas se repiten para siempre sin fin. Para simplificar las cosas, todos los tiempos se expresan en minutos desde un tiempo inicial 0. Por ejemplo, una tarea periódica con tiempo de inicio 5, tiempo de finalización 8 y periodo de repetición 100 ocurriría en los intervalos de tiempo [5..8), [105..108), [205..208), etc...

Tu trabajo consiste en averiguar si mi calendario está libre de conflictos durante un periodo de mi vida. Se considera que dos tareas están en conflicto si y solo si sus intervalos de tiempo se superponen, por ejemplo [2..5) y [4..6) se superponen, pero no [2..8) con [9..10), o [2..8) con [8..10).

Entrada

La entrada está formada por una serie de casos de prueba, ocupando cada uno de ellos varias líneas.

En la primera línea aparecen tres números: N, que representa el número de tareas únicas; M, que representa el número de tareas periódicas; y T, que representa el número de minutos en los que quiero averiguar si hay o no conflictos. A continuación, aparecen N líneas, cada una con los instantes de comienzo y finalización de una tarea única. A estas les siguen M líneas más, cada una con tres números: el instante de comienzo y finalización de la primera aparición de una tarea repetitiva, y el periodo de repetición.

Siempre hay al menos 1 tarea y nunca más de 10.000. Todos los tiempos son números entre 0 y 10^8. Se garantiza que todas las tareas tardan al menos un minuto y que los intervalos de una misma tarea periódica no solapan entre sí.

Salida

Para cada caso de prueba el programa escribirá `SI` si hay conflictos entre algunas tareas en el periodo de tiempo $[0..T)$, y `NO` en caso contrario.

Entrada de ejemplo

```
2 0 10
2 5
4 6
0 2 100
1 4 8
5 7 8
2 1 10
8 20
1 5
6 7 10
```

Salida de ejemplo

```
SI
NO
NO
```

Ejercicio 2. Grafos y estructuras de partición (3 puntos)

Somos una empresa de transporte y hemos decidido renovar parte de nuestra flota de camiones de reparto. A nosotros nos conviene que los camiones sean anchos, porque así se puede repartir y colocar mejor la mercancia. Pero claro, hay ciudades con calles muy estrechas, por donde no todos los camiones pueden pasar.

Tenemos mapas actualizados de las ciudades donde trabajamos, donde hemos señalado para cada calle cuál es la anchura máxima que puede tener un camión para poder transitar por ella.

¿Nos ayudas a decidir si un camión de una anchura determinada puede circular por una ciudad para llegar desde un punto concreto a otro?

Entrada

La entrada está formada por una serie de casos de prueba. En cada caso, primero se describe una ciudad. La primera línea contiene el número V de intersecciones de la ciudad (numeradas de 1 a V) y la segunda el número E de calles entre intersecciones. A continuación aparecen E líneas, cada una con tres números: las intersecciones que une esa calle, y la anchura máxima que puede tener un camión que transite por ella. Todas las calles son de doble sentido.

Tras la descripción de la ciudad, aparece un número K de consultas, seguido de K líneas, cada una con tres números: dos intersecciones distintas, el origen y el destino, y la anchura de un camión, del que estamos interesados en saber si podría viajar desde el origen hasta el destino.

Todos los casos cumplen que $2 \leq V \leq 10.000$, $0 \leq E \leq 100.000$ y $1 \leq K \leq 10$. Todas las anchuras son números entre 1 y 1.000.000.

Salida

Para cada caso de prueba se escribirán K líneas, una por consulta. La respuesta a una consulta será `SI` si un camión de la anchura correspondiente podría recorrer un camino que le llevara del origen al destino, y `NO` en caso contrario.

Entrada de ejemplo

```
5
5
1 2 10
1 3 30
2 4 20
3 4 15
4 5 12
3
1 5 8
1 4 12
1 5 15
```

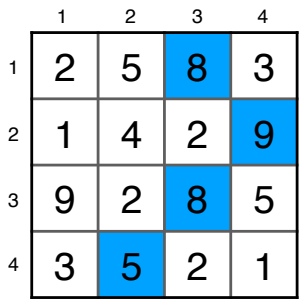
Salida de ejemplo

```
SI
SI
NO
```

Ejercicio 3. Programación dinámica (3 puntos)

Queremos jugar a un juego en el que tenemos un tablero $N \times N$ donde en cada casilla aparece un número natural. El juego consiste en elegir una casilla de la última fila (la N) y movernos desde ella hasta una casilla de la primera fila (la 1), donde los únicos movimientos legales consisten en moverse, sin salirse del tablero, de una casilla a una de las tres casillas de la fila superior alcanzables en vertical o en diagonal (a izquierda o derecha). La cantidad ganada será la suma de los valores en las casillas del tablero por las que pasamos en este recorrido.

Por ejemplo, para conseguir la mayor suma en el siguiente tablero, tendríamos que comenzar en la casilla (4,2), y después pasar por las casillas coloreadas, obteniendo una ganancia total de 30.



Dado un tablero, queremos saber cuál es la máxima cantidad que se puede conseguir y cuál es la casilla por la que habría que comenzar.

Entrada

La entrada está formada por una serie de casos de prueba. Para cada caso, en la primera línea aparece un número N (entre 3 y 500) indicando la dimensión del tablero. Luego aparecen N líneas, cada una con N números (entre 1 y 10.000) separados por espacios, que representan el contenido del tablero.

Salida

Para cada caso de prueba, el programa escribirá una línea que contendrá el máximo valor que se puede obtener y la columna (numeradas de 1 a N) de la celda de la última fila desde donde hay que comenzar el recorrido. En caso de que haya varias preferimos la que esté más a la izquierda (con un número menor).

Entrada de ejemplo

```
4
2 5 8 3
1 4 2 9
9 2 8 5
3 5 2 1
3
1 1 1
1 1 1
1 1 1
```

Salida de ejemplo

```
30 2
3 1
```