Técnicas algorítmicas en ingeniería del software

Grado en Ingeniería del Software (UCM)

Examen extraordinario de julio — 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/julio24816.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 (2.5 puntos)

En el Hospital ACR (*Aquí Curamos Rápido*) se han puesto a mejorar las Urgencias para que los enfermos que llegan con dolencias más graves sean atendidos antes que los demás. Para eso, han comprado una UCM (*Unidad de Catalogación Médica*) que es capaz de valorar instantáneamente el estado de un paciente con un número entero entre 0 y 1.000.000, donde 0 indica que su dolencia es menor (quizá incluso inexistente y sea un mero hipocondríaco) y 1.000.000 indica que el enfermo está casi caminando hacia la luz.

Por desgracia, la afluencia de enfermos es tal que incluso así es muy complicado saber rápidamente quién debería ser el próximo en ser atendido. No hacen más que entrar pacientes nuevos a la vez que los más graves son atendidos, y no es fácil mantenerlos ordenados. Cuando un médico queda libre, debe ser atendido el enfermo a la espera con una valoración más grave. Si hay dos pacientes evaluados con la misma gravedad, deberá ser atendido el que más tiempo lleve esperando.

Para ayudar en la tarea de decidir quién es el próximo, desde ACR se ha hecho un llamamiento para buscar ayuda entre los mejores programadores. ¿Eres tú uno de ellos?

Entrada

La entrada está formada por diversos casos de prueba. Cada caso comienza con una línea indicando el número n de eventos que ocurrirán (como mucho 200.000), y a continuación aparecen n líneas cada una con un evento. Un evento puede ser la llegada de un paciente nuevo, o la atención por parte de un médico que ha quedado libre del paciente más urgente. Los ingresos de pacientes nuevos se indican de la forma `I nombre gravedad`, donde `nombre` es una cadena de como mucho 20 caracteres (sin espacios) y `gravedad` es un número entre 0 y 1.000.000 con su estado (0 leve, 1.000.000 muy grave). Los eventos en los que se atiende al siguiente paciente se indican con el carácter `A`. Se garantiza que no habrá nunca eventos de tipo `A` si no quedan pacientes esperando.

La entrada termina cuando el número de eventos es 0.

Salida

Para cada evento de tipo `A` de cada caso de prueba se escribirá el nombre del paciente que es atendido en ese momento. Se atiende primero al paciente más grave y, en caso de igualdad entre dos o más pacientes, se elegirá entre ellos al que más tiempo lleve esperando.

Al finalizar el tratamiento de cada caso se escribirá una línea más con cuatro guiones (`----`).

Entrada de ejemplo

```
9
I Alberto 4000
I Pepe 3000
A
I Rosa 2000
I Laura 5000
A
I Sara 3000
A
A
0
```

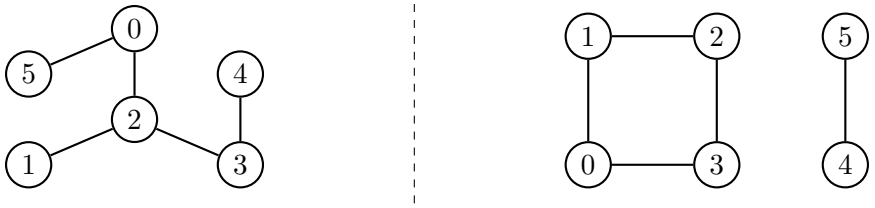
Salida de ejemplo

```
Alberto
Laura
Pepe
Sara
----
```

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

Se dice que un grafo no dirigido es un *árbol libre* si es acíclico y conexo (o dicho de otra manera, todo par de vértices está conectado por exactamente un camino).

De los dos grafos siguientes, solamente el de la izquierda es árbol libre.



El problema consiste en decidir si un grafo no dirigido es árbol libre o no.

Entrada

La entrada está compuesta por diversos casos de prueba. Para cada caso, la primera línea contiene el número de vértices del grafo, V (entre 1 y 10.000), y la segunda el número de aristas, A (entre 0 y 100.000). A continuación aparecen A líneas, cada una con dos enteros que representan los extremos de cada una de las aristas (valores entre 0 y $V - 1$). Los grafos no contienen aristas de un vértice a sí mismo ni más de una arista que conecte un mismo par de vértices.

Salida

Para cada caso de prueba se escribirá `SI` si el grafo es árbol libre y `NO` en caso contrario.

Entrada de ejemplo

```
6
5
0 5
0 2
2 1
2 3
4 3
6
5
0 1
1 2
2 3
3 0
4 5
```

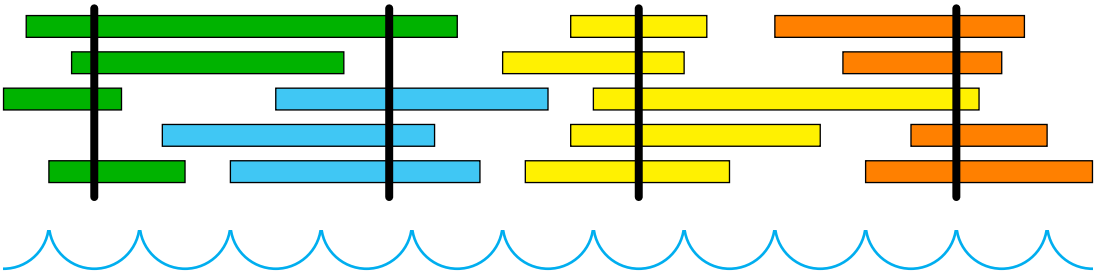
Salida de ejemplo

```
SI
NO
```

Ejercicio 3. Algoritmos voraces (2.5 puntos)

Ya nadie se cree, cuando un apartamento veraniego es anunciado con un gran *¡En primera línea de playa!*, que vaya a ser cierto. Por eso, los dueños de varios edificios de apartamentos (paralelos a la playa pero no en primera línea) han decidido construir pasadizos subterraneos (perpendiculares a la playa) que conecten todos los edificios con la arena. Así creen que los clientes estarán más safisfechos.

Como construir estos pasadizos no es barato, primero quieren saber cuántos túneles como mínimo serían necesarios. Por ejemplo, para la configuración de edificios de la figura (donde se han omitido los edificios en primera línea) son necesarios 4 túneles.



Entrada

La entrada consta de una serie de casos de prueba. Cada uno comienza con una línea con el número N de edificios ($1 \leq N \leq 100.000$). A continuación aparecen N líneas cada una con dos enteros que representan el extremo más occidental (W_i) y el más oriental (E_i) de cada edificio, con $W_i < E_i$, medidos en metros desde el extremo más occidental de la playa. Todas estas medidas son números enteros entre 0 y 10^9.

La entrada terminará con un caso sin edificios, que no debe procesarse.

Salida

Para cada caso de prueba se escribirá una línea con el mínimo número de pasadizos que es necesario construir. Los pasadizos deben ser de 1 metro de ancho y para ser útiles a un edificio deben estar completamente debajo de él cuando lo atraviesan.

Entrada de ejemplo

```
4
1 4
6 15
2 10
12 20
2
1 4
4 8
2
1 4
3 8
0
```

Salida de ejemplo

```
2
2
1
```