Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de julio — Curso 2020/2021

Normas de realización del examen

1. Esta parte del examen dura **2 horas**.
2. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`. Utiliza para cada ejercicio el fichero plantilla que te hemos entregado, escribiendo tu solución en los espacios reservados para ello, siempre entre las etiquetas `<answer>` y `</answer>`.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. En la segunda parte del examen (el cuestionario) una pregunta te pedirá que digas el usuario que has usado. Es **muy importante** que la contestes bien.
4. Escribe tu **nombre y apellidos** en el hueco reservado para ello en el fichero plantilla de cada ejercicio.
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). Simplemente utiliza el `#include` correspondiente.
6. Puedes acceder a las diapositivas del curso en la dirección `http://exacrc/diapositivas`. Si necesitas consultar la documentación de C++, está disponible en `http://exacrc/cppreference`.
7. 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, partes a incluir en la explicación de la solución, análisis de costes, etc.).
8. 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.

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 de prioridad y montículos (3.5 puntos)

El nuevo fascículo de tu serie de cómics favorita está a punto de salir a la venta y todos quieren hacerse con una versión en papel del mismo. Además, los ejemplares vendidos por esta editorial vienen con un número identificador único, por lo que cuanto más bajo sea el número, más codiciado es el ejemplar.

En la tienda de cómics en la que sueles comprar tienen una manera muy peculiar de venderlos: colocan todos los ejemplares del nuevo lanzamiento en varias pilas, los clientes se colocan en fila para entrar en la tienda por turno, y el cliente al que le toca el turno solamente puede elegir comprar uno de los ejemplares que se encuentran en la cima de alguna de las pilas.

Como no quieres un ejemplar cualquiera, has decidido hacer un poco de trampa y tienes en tu poder los identificadores de todos los ejemplares y sus posiciones dentro de las pilas. Teniendo en cuenta que cada cliente va a comprar el mejor ejemplar que tenga disponible en el momento de entrar a la tienda, tienes que calcular el puesto de la cola de espera que debes ocupar el día del lanzamiento para hacerte con el mejor ejemplar de la tienda (el que tiene un identificador menor de entre todos los disponibles).

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea que contiene el número N de pilas de cómics. A continuación, aparecerán N líneas con la descripción de cada pila: un número K que indica la cantidad de cómics en esa pila, seguido de K números positivos (entre 1 y 10^8), los identificadores de los ejemplares, ordenados desde el fondo de esa pila hasta su cima (que aparece a la derecha del todo en la descripción).

Para cada caso de prueba, todos los identificadores de ejemplar son distintos, pero no tienen por qué ser consecutivos o comenzar en 1.

Salida

Para cada caso de prueba se escribirá una línea con la posición (numeradas desde 1) que tienes que ocupar en la cola de espera de la tienda para llevarte el mejor ejemplar.

Entrada de ejemplo

```
2
5 5 2 3 1 20
7 9 12 44 13 4 7 8
3
3 4 5 6
2 3 2
4 11 8 9 7
```

Salida de ejemplo

```
6
1
```

Ejercicio 2. Grafos y estructuras de partición (3.5 puntos)

Tras el confinamiento, el mundo se ha vuelto loco: griterío en las calles y en los balcones, lanzamiento de contenedores, manifestaciones clandestinas... No hay quien ande por la ciudad. Con todo, el papel higiénico se ha vuelto un bien preciado, debido a sus múltiples usos. Los manifestantes los usan para lanzarlos sin herir a nadie, los artistas para hacer mejunje, y hay gente que lo utiliza hasta como filtro de café.

Y con la demanda el precio ha empezado a subir. Además, la ciudad está patas arriba, por lo que puede haber numerosas calles cortadas. A lo que se une las restricciones de movilidad.

Por todo ello, has decidido crear una aplicación que dado el conjunto de calles transitables de la ciudad y el de supermercados, así como el precio del rollo de papel higiénico en cada supermercado, calcule cuánto le costaría a un comprador el rollo más barato, dependiendo de dónde se encuentre.

Entrada

La entrada consistirá en un serie de casos de prueba. Cada caso comienza con una línea con el número N de puntos en la ciudad y el número C de calles transitables. A continuación, aparecerán C líneas, cada una con dos números (entre 1 y N), describiendo los dos puntos de la ciudad que une una calle de doble sentido.

Tras la descripción de las calles aparecerá el número S de supermercados en la ciudad, seguido de S líneas con dos números: el punto donde está localizado un supermercado en la ciudad y el precio del rollo de papel higiénico en dicho supermercado. Se garantiza que no habrá dos supermercados en la misma localización.

Finalmente, aparecerá una línea con el número K de consultas, seguido de esas consultas, consistente cada una en el punto de la ciudad donde se encuentra el comprador.

Salida

Para cada caso de prueba se escribirá una línea por cada una de las K consultas con el precio del rollo más barato que se puede comprar desde la localización del comprador. En caso de que no se pueda alcanzar ningún supermercado desde esa localización, se escribirá `MENUDO MARRON` en su lugar.

La salida de cada caso de prueba termina con una línea con tres guiones (`---`).

Entrada de ejemplo

```
7 5
1 2
2 3
3 1
4 5
7 6
3
2 100
3 200
4 50
4
1 3 5 6
```

Salida de ejemplo

```
100
100
50
MENUDO MARRON
---
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de febrero — Curso 2020/2021

Normas de realización del examen

1. Esta parte del examen dura **2 horas**.
2. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`. Utiliza para cada ejercicio el fichero plantilla que te hemos entregado.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. En la segunda parte del examen (el cuestionario) una pregunta te pedirá que digas el usuario que has usado. Es **muy importante** que la contestes bien.
4. Escribe tu **nombre y apellidos** en el hueco reservado para ello en el fichero plantilla de cada ejercicio.
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). Simplemente utiliza el `#include` correspondiente.
6. Puedes acceder a las diapositivas del curso en la dirección `http://exacrc/diapositivas`.
7. 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, partes a incluir en la explicación de la solución, análisis de costes, etc.).
8. 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.

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. Grafos y estructuras de partición (3 puntos)

Un grupo de ratones de laboratorio está siendo entrenado para escapar de un laberinto. El laberinto está compuesto por una serie de celdas, donde cada celda está conectada a otras a través de pasadizos donde hay obstáculos, por lo que los ratones tardan un tiempo conocido en superar cada uno de estos tramos. Además, algunos pasajes permiten a los ratones ir en un solo sentido, pero no al revés.

Todos los ratones han sido muy bien entrenados y, cuando son colocados en una celda arbitraria del laberinto, siguen un camino que los lleva a la celda de salida en un tiempo mínimo.

Vamos a realizar el siguiente experimento: se coloca un ratón en cada celda del laberinto (excepto en la celda de salida) y se inicia un temporizador de cuenta atrás. Cuando el cronómetro se detiene, contamos la cantidad de ratones que han salido del laberinto.

¿Puedes escribir un programa que, dada la descripción del laberinto y el límite de tiempo, prediga la cantidad de ratones que saldrán del laberinto? Puedes suponer que no hay cuellos de botella en el laberinto, es decir, que todas las celdas tienen espacio para un número arbitrario de ratones.

Entrada

La entrada está compuesta por diversos casos de prueba. La primera línea de cada caso contiene 4 números: el número N de celdas del laberinto (numeradas de 1 a N), el número S de la celda donde se encuentra la salida, el número T de segundos con el que se inicia el cronómetro para la cuenta atrás, y el número P de pasadizos. Las siguientes P líneas describen cada una de ellas un pasadizo, dando 3 números: dos números de celda A y B y los segundos que tarda un ratón en llegar de A a B.

Obsérvese que cada conexión es unidireccional, es decir, los ratones no pueden viajar de B a A a menos que haya otra línea que especifique ese pasadizo. Además, el tiempo requerido para viajar en cada dirección puede ser diferente.

Salida

Para cada caso de prueba el programa debe escribir una línea con el número de ratones que alcanzarán la celda de salida S en como mucho T segundos.

Entrada de ejemplo

```
5 5 20 5
1 2 5
1 4 10
2 4 7
3 4 15
4 5 10
3 1 10 2
2 3 5
3 2 6
```

Salida de ejemplo

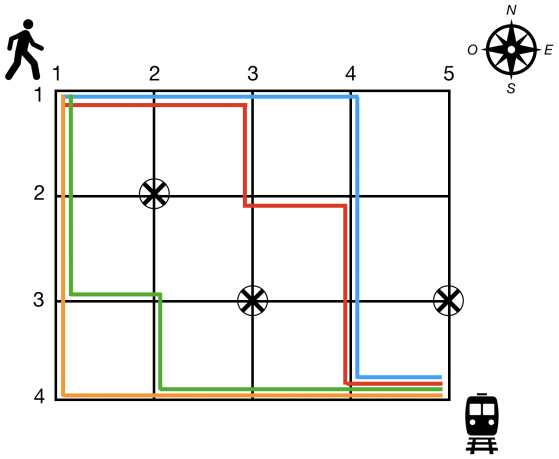
```
3
0
```

Ejercicio 2. Programación dinámica (4 puntos)

Cuadracity es un lugar muy agradable para pasear. Las calles, de dos direcciones, van de norte a sur o de este a oeste, dividiendo la ciudad en manzanas cuadradas regulares. La mayoría de las intersecciones tienen pasos de peatones que los vehículos respetan escrupulosamente. En algunas de ellas, sin embargo, cruzar a pie no es tan seguro y los peatones se ven obligados a utilizar los pasadizos subterráneos disponibles.

Supón que te encuentras en la esquina noroeste de la ciudad y quieres ir a la estación de tren, que se encuentra en la esquina sureste, sin caminar más de lo necesario y evitando los pasadizos, que suponen un retraso extra. ¿De cuántas formas distintas puedes realizar este recorrido satisfaciendo ambas restricciones?

Por ejemplo, la siguiente figura ilustra una ciudad con cuatro calles este-oeste y cinco calles norte-sur. Hay tres intersecciones marcadas como no seguras (con pasadizo subterráneo). El recorrido desde la esquina noroeste a la estación de tren requiere atravesar $3 + 4 = 7$ manzanas, y hay cuatro formas distintas que evitan los pasadizos.



Entrada

La entrada está formada por una serie de casos de prueba. Cada caso comienza con una línea con dos números: el número N de calles este-oeste y el número M de calles norte-sur. Las siguientes N líneas contienen cada una M caracteres, que describen lo que ocurre en cada intersección. El carácter '`.`' significa que la intersección es segura mientras que el carácter '`P`' indica que hay un pasadizo subterráneo.

Salida

Para cada caso de prueba se escribirá una línea con el número de caminos distintos que van de la esquina noroeste a la esquina sureste sin ser más largos de lo necesario (si hay $N \times M$ calles, el camino más corto recorre $(N-1)+(M-1)$ manzanas) y sin pasar por pasadizos. Ese número será siempre menor que 10^9.

Entrada de ejemplo

```
4 5
.....
.P...
..P.P
.....
3 5
.P...
...P.
PPP..
```

Salida de ejemplo

```
4
0
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de septiembre — Curso 2019/2020

Normas de realización del examen

1. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.
2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.
3. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.
4. 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).
5. 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.).
6. 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.
7. Las notas de los ejercicios suman 7 puntos. Se necesita obtener al menos 3.5 puntos para poder aprobar la asignatura. La calificación obtenida en este examen será sumada a la obtenida por la evaluación continua (sobre 3 puntos) para obtener la calificación final.
8. 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 de prioridad y montículos (2 puntos)

Queremos organizar una competición en la que participan N equipos, cada uno de los cuales cuenta inicialmente con S_i seguidores, $1 \leq i \leq N$. Cada vez que se juega un partido, el equipo que gana sigue compitiendo, y el perdedor desaparece de la competición (no hay empates, ni misericordia para el perdedor). Y no hay más reglas para la organización de los partidos (quién debe jugar con quién) que la de tener al final un único vencedor.

Para cada partido, entregaremos una gorra conmemorativa a todos los asistentes. Sabemos que cada vez que un equipo pierde, todos sus seguidores pasan a serlo del equipo que los ha derrotado, y que a cada partido acuden todos los seguidores (actuales) de los dos equipos.

¿Puedes ayudarnos a organizar los partidos de forma que el número de gorras que tengamos que comprar sea lo menor posible?

Entrada

La entrada está compuesta por diversos casos de prueba, ocupando cada uno de ellos dos líneas: la primera contiene el número N de equipos, un entero entre 1 y 200.000, y la segunda contiene N números enteros entre 1 y 1.000.000, que representan los seguidores iniciales de cada equipo.

La entrada termina con un caso sin equipos ($N = 0$), que no debe procesarse.

Salida

Para cada caso de prueba se deberá escribir una línea con el menor número de gorras necesarias para realizar los partidos que lleven a tener un único vencedor

Entrada de ejemplo

```
1
10
3
3 1 4
4
3 4 5 6
0
```

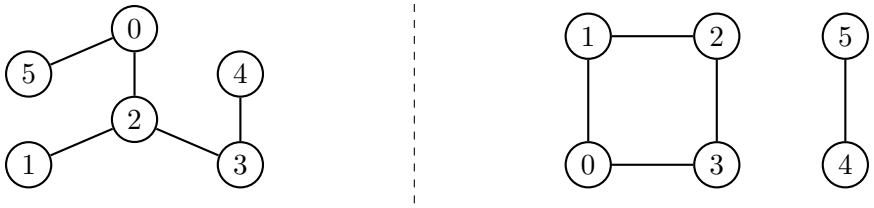
Salida de ejemplo

```
0
12
36
```

Ejercicio 2. Grafos y estructuras de partición (2.5 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.

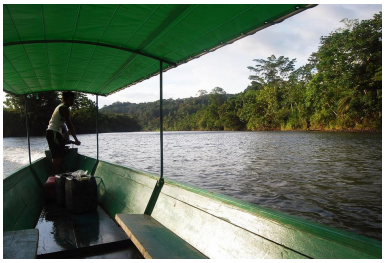
Entrada de ejemplo

```
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. Programación dinámica (2.5 puntos)



India Nayons quiere planificar una aventurilla por el Amazonas. A lo largo del río hay una serie de poblados indígenas, cuyos habitantes, al observar el creciente auge del turismo rural, han ideado un sistema de alquiler de canoas. En cada poblado se puede alquilar una canoa, la cual puede devolverse en cualquier otro poblado que esté a favor de la corriente.

Consultando por Internet los costes de alquileres entre poblados, India ha constatado que el coste del alquiler desde un poblado i hasta otro j puede resultar mayor que el coste total de una serie de alquileres más breves. En tal caso, es más rentable devolver la primera canoa en alguna aldea k entre i y j, y seguir camino en una segunda canoa, sin ninguna penalización por cambiar de canoa.

¿Sabrías calcular el coste mínimo de un viaje en canoa desde todos los posibles puntos de partida i hasta todos los posibles puntos de llegada j?

Entrada

La entrada está compuesta por diversos casos de prueba. Cada uno comienza con una línea con el número N de poblados (al menos 2, y no más de 200). A continuación aparece la información sobre los alquileres. Esta consta de $N-1$ líneas: la primera tiene $N-1$ valores, que representan el coste del alquiler de una canoa para viajar del primer poblado a cada uno de los demás a favor de la corriente; la segunda tiene $N-2$ valores, que representan los costes de los alquileres desde el segundo poblado a todos los demás a favor de la corriente (el tercero, el cuarto, etc.); y así hasta que la última línea tiene un único valor que representa el coste del alquiler de una canoa desde el penúltimo poblado al último. Todos los costes son números entre 1 y 1.000.000.

Salida

Para cada caso de prueba se deben escribir $N-1$ líneas, con el coste mínimo de cada posible viaje. La primera línea contendrá $N-1$ valores, que representarán el coste mínimo de un plan de viaje que comienza en el primer poblado y termina en cada uno de los siguientes a favor de la corriente; la segunda línea contendrá $N-1$ valores que representarán los costes mínimos para viajar desde el segundo poblado a todos los demás a favor de la corriente; etc.

Entrada de ejemplo

```
5
3 10 30 90
5 20 15
10 8
4
```

Salida de ejemplo

```
3 8 18 16
5 15 13
10 8
4
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de enero — Curso 2019/2020

Normas de realización del examen

1. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.
2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.
3. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.
4. 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).
5. 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.).
6. 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.
7. Las notas de los ejercicios suman 7 puntos. Se necesita obtener al menos 3.5 puntos para poder aprobar la asignatura. La calificación obtenida en este examen será sumada a la obtenida por la evaluación continua (sobre 3 puntos) para obtener la calificación final.
8. 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 de prioridad y montículos (2 puntos)

Por fin la agencia espacial ha puesto en funcionamiento la nueva base en Marte. La energía que necesita para su funcionamiento la proporcionan B baterías. La base está plenamente funcional si tiene las B baterías funcionando. Si alguna de ellas se queda sin carga, seguirá siendo habitable, aunque perderá parte de sus funciones. Si no queda ninguna batería con carga, el personal deberá abandonar la base inmediatamente.

Cada batería tiene carga para un tiempo de funcionamiento, que puede ser diferente de unas a otras. Las baterías se pueden recargar de manera instantánea, pero cada vez que se recargan pierden capacidad, disminuyendo su tiempo máximo de funcionamiento en Z unidades de tiempo, respecto a la última vez que se cargó. La disminución es la misma para todas las baterías.

Cuando no se pueden volver a recargar (porque han perdido ya toda su capacidad) es necesario reemplazarlas por una batería nueva. Para ello se tienen baterías de repuesto. Las baterías de repuesto se utilizan en el orden en que han ido llegando a la base.

El personal de mantenimiento tiene orden de recargar las baterías cuando se descargan y si ya no pueden ser recargadas más veces, sustituirlas por una de repuesto (si hay). Las baterías descargadas se revisan por orden de identificador, por lo que si dos de ellas necesitan mantenimiento al mismo tiempo se actúa antes sobre la de menor identificador.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso consta de tres líneas. En la primera se indica el número de baterías que deben estar funcionando al mismo tiempo para el correcto funcionamiento de la base ($0 < B \leq 50.000$), seguido del tiempo que durará cada una de las baterías (los tiempos se dan en el orden en que fueron conectadas las baterías, y estas serán identificadas con los números del 1 al B en ese orden). En la siguiente línea se da el número de baterías de repuesto ($0 \leq R \leq 100.000$) seguido del tiempo que dura cada una de las baterías (en el orden en que llegaron a la base, y numeradas de B+1 a B+R). En la última línea se indica el tiempo de carga $Z > 0$ que se pierde cada vez que se realiza una recarga, que es el mismo para todas las baterías, seguido del tiempo T en que se quiere consultar el estado de las baterías. Todos los tiempos que haga falta calcular serán inferiores a 10^9.

Salida

Para cada caso de prueba se escribirán varias líneas. En la primera se escribe el estado de la base cuando se hayan superado T unidades de tiempo: `CORRECTO` si todas las baterías están en funcionamiento; `FALLO EN EL SISTEMA` si la base está funcionando sin todas las baterías y `ABANDONEN INMEDIATAMENTE LA BASE`, si todas las baterías se han agotado y no pueden ser recargadas. En las siguientes líneas se indicará para cada batería en funcionamiento, su identificador seguido del tiempo hasta el que durará la carga actual de la batería, ordenadas según irían siendo recargadas (o sustituidas).

El caso terminará con una línea con 3 guiones (`---`).

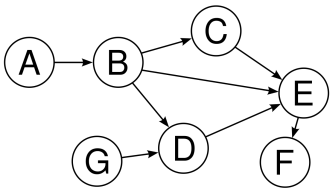
Entrada de ejemplo

```
2 4 6
1 8
4 10
3 6 14 6
0
2 7
1 4
0
4 4
```

Salida de ejemplo

```
FALLO EN EL SISTEMA
3 12
---
CORRECTO
1 10
3 10
2 14
---
ABANDONEN INMEDIATAMENTE LA BASE
---
```

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



A la vuelta de las vacaciones mi jefe tenía preparada una larga lista de tareas que tengo que hacer. Las tareas no son totalmente independientes entre sí, por lo que algunas tareas solamente pueden ser realizadas cuando se hayan terminado ya otras.

Primero he analizado todas las tareas, y he descubierto las relaciones de precedencia directa entre algunos pares de tareas, es decir, cuándo una tarea A tiene que ser realizada antes que otra tarea B.

Ahora me falta decidir un orden en el que hacer todas las tareas de forma que se respeten esas precedencias. O quejarme a mi jefe si eso es imposible.

Entrada

La entrada está formada por una serie de casos de prueba. Cada caso ocupa varias líneas. La primera contiene el número $1 \leq N \leq 10.000$ de tareas a realizar (las tareas están numeradas de 1 a N). La segunda línea contiene el número $0 \leq M \leq 100.000$ de relaciones directas de dependencia existentes. A continuación aparecen M líneas cada una con dos números A y B entre 1 y N, indicando que la tarea A debe realizarse antes que la tarea B.

Salida

La salida contendrá una línea por cada caso de prueba. Si es posible ordenar las tareas de forma que se respeten todas las dependencias, se escribirá una posible ordenación. En caso contrario, se escribirá `Imposible`.

Entrada de ejemplo

```
7
8
1 2
2 3
2 4
3 5
2 5
4 5
5 6
7 4
2
2
1 2
2 1
```

Salida de ejemplo

```
1 2 7 3 4 5 6
Imposible
```

Ejercicio 3. Programación dinámica (2 puntos)

Sergio es un apasionado de la música y su cometido todos los veranos es poder ver al mayor número de sus artistas favoritos en directo. Su único impedimento es el dinero ya que dispone de un presupuesto limitado para comprar las entradas. Por ello ha seleccionado todos los festivales a los que le gustaría ir, averiguando el número de grupos que van al festival (todos distintos entre sí) y el precio de la entrada de este, para poder elegir los festivales a los que irá. Su objetivo es maximizar el número de grupos a los que podrá escuchar tocar.

Tu objetivo es ayudar a Sergio a cumplir su cometido haciendo un programa que le diga a cuántos grupos distintos podrá escuchar como máximo con su presupuesto.

Entrada

La entrada consta de varios casos de prueba. Para cada caso, en la primera línea aparecerán dos enteros: el presupuesto P del que dispone Sergio ($0 \leq P \leq 1.000$) y el número N ($1 \leq N \leq 100$) de festivales distintos que ha seleccionado. A continuación aparecen N líneas con dos enteros que describen cada festival: el primero es el número de grupos que actúan en el festival (entre 1 y 40) y el segundo es el precio de la entrada (entre 10 y 100).

Salida

Para cada caso de prueba se deberá mostrar en una línea el máximo número de grupos que Sergio puede ir a ver.

Entrada de ejemplo

```
60 4
15 50
10 40
5 10
8 20
80 2
10 100
7 90
```

Salida de ejemplo

```
20
0
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de julio — Curso 2018/2019

Normas de realización del examen

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

2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.

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

4. Del enlace **Material para descargar** puedes descargar un archivo comprimido 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 7 puntos. La calificación obtenida en este examen será sumada a la obtenida por la evaluación continua (sobre 3 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.

Ejercicio 1. Colas de prioridad y montículos (2.5 puntos)

Desde hace algún tiempo, quedamos la pandilla para ir a volar drones los sábados. Cada dron necesita dos pilas para poder volar: una de 9 V y otra de 1.5 V. En el club tenemos dos cajas para guardar las pilas, en una tenemos las de 9 V y en otra las de 1.5 V. Cada sábado cogemos las que están más cargadas y las colocamos en los drones. Para aprovechar al máximo el tiempo de vuelo, colocamos siempre las dos pilas más cargadas de cada tipo juntas, ya que los drones solo vuelan mientras las dos pilas tienen carga; después las dos siguientes más cargadas las ponemos en el siguiente dron; y así mientras queden pilas con carga de los dos tipos. Una vez colocadas las pilas, echamos los drones a volar. Cuando todos ellos acaban en el suelo por agotamiento de alguna de sus pilas, volvemos al club y guardamos en las cajas las pilas que todavía no están totalmente gastadas.

Por ejemplo, si a un dron le pusimos una pila de 9 V que permitía volar 5 horas y una pila de 1.5 V que permitía volar 2, el dron habrá volado 2 horas y al volver al club guardaremos la pila de 9 V a la que le quedarán 3 horas de vuelo. La pila de 1.5 V estará agotada y la echaremos al cubo de reciclaje.

Queremos saber cuántas horas de vuelo realizarán entre todos los drones cada sábado que podamos salir a volar, antes de que se agoten las pilas que hay ahora mismo en las cajas. Las pilas las tenemos que colocar en el club, por lo que cada dron solo puede volar una vez cada sábado.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea en que se indica el número N de drones que tenemos ($1 \leq N \leq 1.000$), el número A de pilas de 9 V y el número B de pilas de 1.5 V ($1 \leq A,B \leq 200.000$). En la línea siguiente se indica el número de horas de vuelo que permite la carga de cada una de las pilas de 9 V y en la tercera línea el número de horas de vuelo que permite cada una de las pilas de 1.5 V.

Salida

Para cada caso de prueba se escribirá en una línea el número de horas de vuelo que realizarán los drones cada sábado, mientras se pueda salir a volar algún dron.

Se garantiza que esos números nunca serán mayores que 10^9.

Entrada de ejemplo

```
2 4 2
5 12 7 15
10 10
2 4 3
5 12 7 15
20 20 2
3 3 3
25 15 10
20 20 5
1 4 6
5 9 2 6
7 3 3 1 6 4
```

Salida de ejemplo

```
20
27 12
40 5
7 6 4 2 2 1
```

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

En una red social hay una serie de usuarios que se comunican entre ellos dentro de una serie de grupos de amigos. Queremos analizar el proceso de distribución de noticias entre estos usuarios.

Inicialmente, algún usuario recibe una noticia de una fuente externa. Entonces envía la noticia a sus amigos en la red (dos usuarios son amigos si hay al menos un grupo al que pertenezcan ambos). Los amigos envían a su vez esa noticia a sus amigos y el proceso continua así hasta que no exista una pareja de amigos tal que uno de ellos conozca la noticia y el otro no.

Para cada usuario de la red, queremos saber cuántos usuarios terminarían conociendo la noticia si inicialmente solamente ese usuario la conocía.

Entrada

La entrada está formada por una serie de casos, cada uno de los cuales ocupa varías líneas. En la primera línea de cada caso aparecen dos números: el número N de usuarios de la red y el número M de grupos ($1 \leq N, M \leq 100.000$). A continuación aparecen M líneas describiendo esos grupos. Para cada grupo, la descripción comienza con el número de usuarios del grupo (entre 0 y N) seguido de los identificadores de esos usuarios (todos distintos), números entre 1 y N. La suma de los tamaños de todos los grupos no es mayor que 500.000.

Salida

Para cada caso se escribirá una línea con N números. El número i-ésimo indicará el número de usuarios que terminarían conociendo la noticia si el usuario i fuera quién comenzara a distribuirla.

Entrada de ejemplo

```
7 5
3 2 5 4
0
2 1 2
1 1
2 6 7
4 2
1 1
1 3
```

Salida de ejemplo

```
4 4 1 4 4 2 2
1 1 1 1
```

Ejercicio 3. Programación dinámica (2.5 puntos)

La empresa *Iluminate, S.L.* está especializada en la iluminación de salas de fiesta. Coloca tiras de bombillas en el techo y luces indirectas en las paredes, hasta obtener una iluminación que satisfaga a los organizadores de las fiestas. Para ser más competitiva se ha asociado con una fábrica que le proporciona tantas bombillas como necesite, a bajo coste, pero solo de algunas potencias.

Para que los organizadores queden satisfechos, les pide en el contrato que indiquen la potencia instalada que desean para la sala. Sin embargo, se han dado cuenta de que muchas veces sería más barato realizar una instalación que superase la potencia requerida y piensan que esto no debería molestar a los organizadores, ya que la sala estará más iluminada. Por ello han cambiado el contrato y ahora piden al organizador que les indique una potencia mínima para la sala, pudiendo la empresa instalar una potencia superior siempre que no sobrepase la potencia máxima admitida por la instalación.

¿Sabrías calcular la potencia que debes instalar en la sala, sabiendo la potencia máxima admitida por la instalación y la potencia mínima requerida por los organizadores para que la instalación sea lo más barata posible?

Entrada

La entrada consta de una serie de casos de prueba. Cada caso consta de tres líneas. En la primera línea se indica el número N de tipos de bombillas ($1 \leq N \leq 1.000$), la potencia máxima $PMax$ permitida y la potencia mínima $PMin$ requerida por los organizadores ($1 \leq PMin \leq PMax \leq 1.000$). En la línea siguiente se da la potencia de cada tipo de bombilla que nos proporciona la fábrica y en la tercera línea se indica el coste de cada tipo de bombilla.

Salida

Para cada caso de prueba se escribirá en una línea el coste mínimo de la instalación y la potencia que se debe instalar. Si el coste mínimo se consigue con varias potencias, se escribirá la menor de todas ellas. Si es imposible conseguir ninguna de las potencias permitidas, se escribirá `IMPOSIBLE`.

Entrada de ejemplo

```
2 7 4
5 2
30 10
3 8 6
2 3 5
30 45 50
3 11 11
4 6 4
1 2 3
```

Salida de ejemplo

```
20 4
80 7
IMPOSIBLE
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (UCM)

Examen final de enero — Curso 2018/2019

Normas de realización del examen

1. Debes desarrollar e implementar soluciones para cada uno de los ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc`.
2. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen.
3. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.
4. Del enlace **Material para descargar** puedes descargar un archivo comprimido 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 7 puntos. La calificación obtenida en este examen será sumada a la obtenida por la evaluación continua (sobre 3 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.

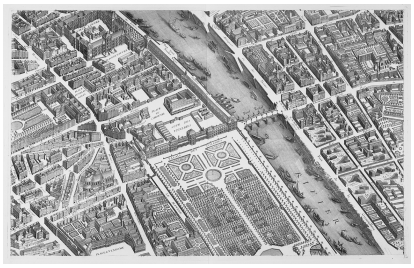
Primero resuelve el problema. Entonces, escribe el código.

— 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.

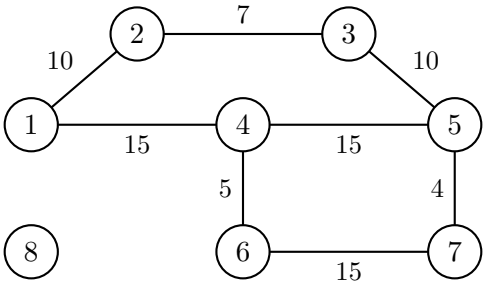
— Ryan Campbell

Ejercicio 1. Grafos y estructuras de partición (4 puntos)



Hace poco me he mudado a una nueva ciudad. Aprovechando que me gusta caminar, siempre que puedo voy andando a los sitios y así voy conociendo la ciudad. Cuando voy lejos y tengo prisa suelo coger las grandes avenidas porque las conozco más y sé que no voy a perderme, pero soy consciente de que, muchas veces, callejeando por calles cortas recorrería menos distancia, aunque tuviese que atravesar muchas de ellas. Me pregunto cuántas veces el camino más corto en distancia también es el que pasa por menos calles.

Por ejemplo, si el siguiente esquema representa la ciudad, con 8 intersecciones y 8 calles, donde junto a cada calle aparece su longitud medida en metros, para ir del punto 1 al punto 5 el camino más corto es el $1 \to 2 \to 3 \to 5$, que recorre 27 metros y atraviesa tres calles, mientras que el camino $1 \to 4 \to 5$, aunque es más largo (30 metros) pasa solamente por dos calles. En cambio, el camino más corto que une los puntos 6 y 5 sí utiliza el menor número de calles (no hay ningún otro camino con menos calles).



Entrada

La entrada consta de varios casos de prueba, ocupando cada uno de ellos varias líneas.

En la primera aparece el número N (entre 1 y 10.000) de intersecciones en la ciudad, y en la segunda el número C (entre 0 y 100.000) de calles (entre intersecciones). A continuación, aparece una línea por cada calle con tres enteros, que indican los números de las intersecciones que une la calle (números entre 1 y N) y su longitud (un valor entre 1 y 5.000) medida en metros. Todas las calles pueden recorrerse en ambos sentidos.

A continuación aparece el número K de consultas (no más de 10) seguido de esas consultas: dos números que representan las intersecciones origen y destino. Se garantiza que para cada consulta, el camino más corto entre origen y destino es único.

Salida

Para cada caso de prueba se escribirá una línea por cada consulta que contendrá la distancia, medida en metros, del camino más corto que conecta el origen con el destino, seguida de la palabra `SI` si ese camino además atraviesa el menor número de calles o `NO` en caso contrario. Si para una consulta no existiera camino que conecte el origen con el destino, entonces se escribiría `SIN CAMINO`.

Después de la salida de cada caso se escribirá una línea con `----`.

Entrada de ejemplo

```
8
8
1 2 10
2 3 7
3 5 10
1 4 15
4 5 15
4 6 5
5 7 4
6 7 15
3
1 5
3 8
6 5
```

Salida de ejemplo

```
27 NO
SIN CAMINO
19 SI
----
```

Ejercicio 2. Programación dinámica (3 puntos)

Para desayunar, mi hermana gemela y yo solemos tomar un bizcocho adornado con frutas escarchadas por encima. Nuestras frutas preferidas son la naranja y el limón. Los mejores días son aquellos en que las dos tomamos cada una un trozo que tenga naranja por encima o bien las dos tomamos trozos con limón, es decir, las dos probamos el mismo sabor. Cada día cortamos dos trozos iguales de bizcocho, pero nuestra madre solamente nos deja cortarlo de alguno de los dos extremos del bizcocho, los dos del mismo lado o cada uno de un lado distinto, pero no podemos partirlo y sacar un trozo del centro. Teniendo en cuenta que nunca hay dos trozos de bizcocho consecutivos que tengan la misma fruta, ¿cuál es el máximo número de días en que podremos tomar dos trozos con la misma fruta por encima?

Entrada

La entrada consta de una serie de casos de prueba. Cada caso comienza con una línea en que se indica el número N, siempre par, de trozos de bizcocho que se pueden cortar en total ($2 \leq N \leq 1.000$). En la línea siguiente aparecen N números, que indican la fruta que tiene encima cada trozo del bizcocho, empezando por el de la izquierda. El valor 0 indica que no tiene fruta, el valor 1 indica que tiene naranja y el valor 2 indica que tiene limón.

Salida

Para cada caso de prueba se escribirá en una línea el número máximo de veces que podemos tomar dos trozos con la misma fruta.

Entrada de ejemplo

```
4
1 2 1 2
8
0 1 2 0 0 2 1 2
12
1 2 0 0 2 0 2 1 0 0 2 0
```

Salida de ejemplo

```
0
2
2
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (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
```

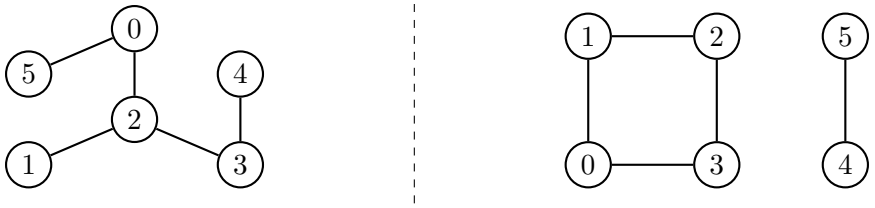
Salida de ejemplo

```
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
```

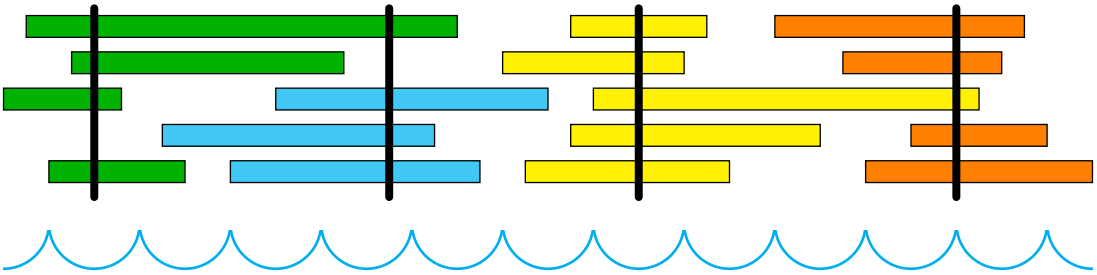
Salida de ejemplo

```
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
```

Métodos algorítmicos en resolución de problemas

Grado en Desarrollo de Videojuegos (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
```

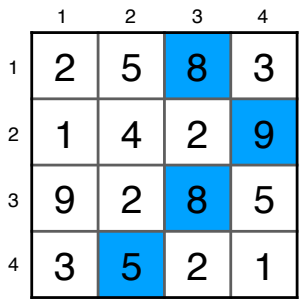
Salida de ejemplo

```
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
```