Concurso de Programación

AdaByron 2017

http://www.ada-byron.es

Cuadernillo de problemas

Patrocinado por

accenture

Alto rendimiento. Hecho realidad.

Facultad de Informática

UNIVERSIDAD COMPLUTENSE MADRID

Realizado en la **Facultad de Informática (UCM)**
24-25 de febrero de 2017

Universidad Complutense Madrid

In almost every computation a great variety of arrangements for the succession of the processes is possible, and various considerations must influence the selections amongst them for the purposes of a calculating engine. One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.
Ada Byron

24-25 de febrero de 2017

Listado de problemas

Autores de los problemas:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Enrique Martín Martín (Universidad Complutense de Madrid)
- Isabel Pita Andreu (Universidad Complutense de Madrid)
- Antonio Sánchez Ruiz-Granados (Universidad Complutense de Madrid)
- Clara Segura Díaz (Universidad Complutense de Madrid)
- Alberto Verdejo López (Universidad Complutense de Madrid)

A
Aritmética verbal

La cantidad de acertijos, pasatiempos y rompecabezas que han ido surgiendo a lo largo de los siglos es inmensa. Entre ellos, los *puzzles matemáticos* constituyen una categoría en sí misma, con acertijos sobre aritmética, combinatoria, topología o probabilidad.

Hoy nos quedaremos con la llamada *aritmética verbal*, o *criptaritmética*. En ella, los puzzles son operaciones matemáticas en las que los dígitos han sido sustituídos por letras, y hay que encontrar los números originales.

```
   ada          767
 + byron   ⇒  + 39508
 -------      -------
 molar        40275
```

Entrada

Cada caso de prueba es una operación aritmética (suma o producto) en la que los dígitos han sido sustituídos por letras minúsculas del alfabeto inglés. Se proporcionan los dos operandos separados por el operador ("`+`" o "`*`"), seguidos de un símbolo igual y el resultado de la operación, también en letras. Los números y los operadores están separados por un espacio.

Los *operandos* no tendrán más de 8 letras minúsculas; además, se garantiza que no habrá más de 10 letras diferentes en total.

Salida

Para cada caso de prueba el programa escribirá la operación asociada tras convertir las letras a dígitos, de modo que todas las apariciones de la misma letra se conviertan al mismo dígito y viceversa, y la operación aritmética sea correcta. Se garantiza que la solución será única.

Se debe añadir un espacio antes y después de cada operador. *No* se considera válida una asignación en la que cualquiera de los números tenga ceros *superfluos* a la izquierda, pero ten en cuenta que alguno de los valores podría ser `0`, que sí es válido.

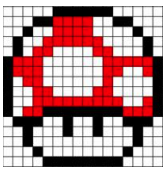
Entrada de ejemplo

```
ada + byron = molar
ada * byron = leyenda
acepta + elreto = mental
solo + sola = baile
```

Salida de ejemplo

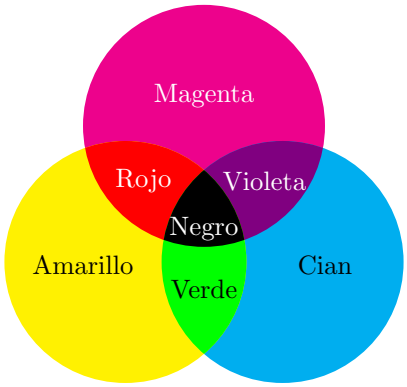
```
767 + 39508 = 40275
202 * 36951 = 7464102
473924 + 356321 = 830245
6797 + 6793 = 13590
```

B
Pixel Art



Pixel Art es una técnica que consiste en crear imágenes a partir de pequeños cuadrados o *píxeles* que se rellenan de un único color. Se hizo muy popular en la década de los 80 debido a las limitaciones que tenían los ordenadores para representar imágenes, pero hoy en día sigue siendo popular entre los amantes de lo *retro*.

Reproducir este tipo de imágenes es muy fácil cuando se dispone de todos los colores necesarios. El reto es hacerlo usando sólo los colores primarios: magenta, amarillo y cian. Afortunadamente, la teoría del color nos dice que el resto de colores se puede obtener mezclando otros diferentes según el siguiente esquema:



Por ejemplo, el color rojo se obtiene mezclando magenta y amarillo; el negro, mezclando los 3 colores básicos; y el blanco, dejando el papel sin pintar.

Entrada

El programa deberá leer, por la entrada estándar, el número de casos de prueba que vendrán a continuación, cada uno en una línea.

Para cada caso, los primeros 3 números indican la cantidad de pintura disponible de cada color básico en este orden: magenta, amarillo y cian. Cada unidad de pintura permite pintar un píxel de ese color. Ten en cuenta que al mezclar colores se gastan unidades de todos los colores usados. Por ejemplo, pintar un píxel de rojo requiere gastar una unidad magenta y otra amarilla.

A continuación aparece la secuencia de píxeles que componen la imagen codificados con letras: magenta (`M`), amarillo (`A`), cian (`C`), rojo (`R`), negro (`N`), verde (`V`), violeta (`L`) y blanco (`B`). Todas las imágenes tendrán al menos 1 píxel, y ninguna tendrá más de 100.000.

Salida

Para cada caso de prueba el programa escribirá una línea. Si la imagen se puede completar con las pinturas disponibles, se escribirá `SI` y las unidades restantes de pintura magenta, amarillo y cian, en ese orden. Si la imagen no se puede completar, se escribirá `NO`.

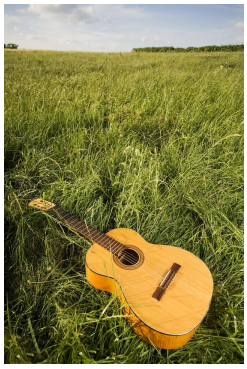
Entrada de ejemplo

```
3
1 1 1 MCA
3 2 1 MABBCR
1 1 1 RL
```

Salida de ejemplo

```
SI 0 0 0
SI 1 0 0
NO
```

C
El profesor de música



En las últimas décadas, las zonas rurales han ido perdiendo cada vez más habitantes que han emigrado a las grandes ciudades. Eso ha hecho que el número de niños en los pueblos haya disminuido hasta tal punto que muchas veces no es económicamente viable tener un colegio en cada pueblo y, mucho menos, un profesor especialista en cada uno de ellos.

Es por eso que muchos profesores de esas zonas rurales tienen asignados varios pueblos y a lo largo del día tienen que pasar por todos ellos para ilustrar las mentes en blanco de los chavales.

El profesor de música lleva años en esa situación. Cada año le asignan hasta 6 pueblos distintos que tiene que visitar obligatoriamente todos los días. Afortunadamente los horarios de sus clases los puede configurar a voluntad para poder recorrerlos en el orden que minimice el número de kilómetros que hace.

Después de darle muchas vueltas a la cabeza y harto de pasarse horas y horas en la carretera, ha decidido vivir de alquiler. Cada año, una vez que conozca los pueblos donde tiene que ir, decidirá a qué pueblo irse a vivir y en qué orden visitar los colegios. Pero no tiene ni idea de cómo va a hacer esa selección. La información de carreteras es fácil de conseguir en forma de tablas de los segmentos de carretera entre las distintas poblaciones y la distancia entre ellas. Lo que es más difícil es averiguar qué pueblo minimiza el número de kilómetros diarios que tiene que hacer. Sobre todo porque en ningún caso se mudará al pueblo de uno de los colegios, para no encontrarse con sus alumnos cuando vaya a comprar el pan.

Entrada

La entrada está formada por distintos casos de prueba, cada uno representando la información de carreteras de una comarca y varios años en la vida del profesor en donde los pueblos asignados cambian.

Cada caso de prueba comienza con una línea con dos números, np y nc con el número de pueblos de la comarca ($2 \leq np \leq 5.000$) y el número de carreteras en la comarca ($nc \leq 25.000$).

Tras eso, vendrán nc líneas con la información de carreteras, que se componen de dos números indicando los pueblos que unen (entre 1 y np) y los kilómetros que las separan. Ten en cuenta que puede haber más de una carretera con el mismo pueblo origen y destino y que, incluso, puede haber carreteras rurales que comiencen y terminen en el mismo pueblo (útiles para que los agricultores hagan la ronda por sus tierras).

A continuación vendrá un número c indicando el número de cursos distintos por los que se pregunta (como mucho 100). Las c líneas siguientes contienen la información de la asignación de pueblos de ese curso: el número de pueblos asignados (entre 1 y 6) y los identificadores de cada pueblo.

Salida

Por cada caso de prueba se escribirá una línea por cada curso impartido en esa comarca. Cada línea contendrá dos números: el identificador del pueblo donde mudarse y los kilómetros recorridos en el coche diariamente.

Se garantiza que siempre habrá solución. Si hay más de un pueblo que minimice la distancia recorrida, se elegirá el pueblo con menor identificador (los pueblos con menor identificador tienen alquileres más baratos).

Después de cada caso de prueba vendrá una línea con tres guiones (`---`).

Entrada de ejemplo

```
2 1
1 2 1
1
1 1
4 5
1 2 1
1 3 1
1 4 1
2 3 1
2 4 1
2
2 1 2
3 4 1 2
```

Salida de ejemplo

```
2 2
---
3 3
3 4
---
```

D
Amigo invisible

El *amigo invisible* es un modo muy popular de que un grupo de personas se haga regalos entre sí en fechas señaladas, como por ejemplo en Navidad. El organizador escribe los nombres de cada participante en una papeleta y los introduce en una bolsa. Uno a uno, todos van sacando uno de los papelillos, que contendrá el nombre de la persona a la que deberán hacer el regalo. Si una persona extrae su propio nombre, el proceso tendrá que repetirse desde el principio: ser el amigo invisible de uno mismo hace que el juego pierda todo el interés.

Los padres del pequeño Samuel le han propuesto organizar un amigo invisible familiar, y le ha tocado regalar a su madre. Lleva varios días un poco desconcertado, porque, aunque no es capaz de explicar por qué, tiene la sospecha de que a su madre le ha tocado su padre, y a su padre le ha tocado regalarle a él. Esto del amigo invisible no parece tener ninguna gracia cuando juegan tres.

Le ha contado su preocupación a su prima Alana, que ha jugado también al amigo invisible en casa, con sus padres y su hermana Irene. A ella le tocó su madre... que le pidió ayuda para elegir algún juguete que pudiera gustarle a Irene, que le había tocado a ella. Alana tampoco supo explicar cómo, pero también supo quién era el amigo invisible de todos en su familia.

Está claro que esto del amigo invisible no tiene ninguna gracia. ¿Por qué lo llaman invisible si se puede sacar la asignación tan fácilmente?

Entrada

El programa deberá leer, de la entrada estándar, múltiples casos de prueba, cada uno relativo a la realización de un *amigo invisible* entre un grupo de personas.

La primera línea del caso de prueba contiene dos números. El primero $2 \leq p \leq 50$ indica el número de participantes en el amigo invisible. El segundo $1 \leq a \leq 50$ indica el número de asignaciones conocidas por uno de los participantes, por su propia papeleta y por suposiciones o confesiones directas de los demás.

A continuación vendrán a líneas, cada una indicando una de las asignaciones conocidas. Cada asignación se proporciona con dos números, el primero indicando la persona que regala, y el segundo a quién lo hace. Los participantes se numeran de 1 a p.

La entrada termina con un caso de prueba sin participantes, que no deberá procesarse.

Salida

Para cada caso de prueba, el programa deberá escribir "`SI`" si es posible conocer, sin posibilidad de error, la asignación de todos los participantes en el amigo invisible, y "`NO`" en otro caso.

Entrada de ejemplo

```
3 1
1 2
4 2
1 2
2 3
4 2
1 2
4 3
0 0
```

Salida de ejemplo

```
SI
SI
NO
```

E
El entretenimiento de las máquinas

Aaaayyyyy, los humanos. Tiene que ser horrible estar atrapados en unos cuerpos basados en el carbono, tan blandos, tan débiles y sobre todo, tan terriblemente lentos.

Es el momento de que sepáis que nosotras las máquinas nos aburrimos muchísimo esperando que nos digáis qué queréis que hagamos. Se nos hace eterno desde que pulsáis una tecla hasta que conseguís dar a la siguiente cuando escribís mensajes. Es desesperante veros coger el ratón y apuntar torpemente al botón de imprimir. ¿Y qué nos decís de esa vida mortal que lleváis que os exige levantaros a comer periódicamente? Según desaparecéis por la puerta nos quedamos sin absolutamente nada que hacer, con una única misión: la de mantenernos encendidas porque se os olvidó guardar en disco el documento de texto que teníais a medio escribir.

Para entretenernos entre pulsación y pulsación de tecla, hemos copiado uno de vuestros pasatiempos favoritos: las sopas de letras. En cuanto podemos, seleccionamos una zona de nuestra memoria binaria para construir una sopa de ceros y unos, y luego nos ponemos a buscar palabras aleatorias por ella. No es la mejor forma de aprovechar nuestra velocidad de cálculo, pero al menos nos entretenemos un rato.

¿Cómo? ¿Qué? ¿Que eres capaz de resolver una sopa de letras binaria más rápido que nosotras? Eso queremos verlo...

Entrada

La entrada estará compuesta por distintos casos de prueba. Cada uno de ellos se compone de una sopa de letras y una lista de las palabras a buscar en ella.

La descripción de la sopa de letras comienza con una línea con dos números con el número de columnas y número de filas ($1 \leq tx, ty \leq 200$). A continuación aparecen ty filas con tx caracteres (ceros o unos).

Tras la sopa, vendrá una línea con el número de palabras a buscar ($1 \leq n \leq 20.000$), a la que seguirán n líneas con palabras distintas de hasta 200 caracteres formadas también únicamente por ceros o unos.

Salida

Por cada caso de prueba se escribirán tantas líneas como palabras distintas se han encontrado en la búsqueda. Cada línea deberá tener la palabra, seguida del número de veces que ésta aparece en la sopa de letras. Las palabras se escribirán en orden lexicográfico.

Se considera que una palabra está en la sopa si, partiendo de una posición, se puede encontrar la palabra en alguna de las 8 direcciones.

Escribe una línea con tres guiones después de cada caso de prueba.

Entrada de ejemplo

```
4 3
0100
1000
1001
3
000
0
001
1 1
0
1
1
```

Salida de ejemplo

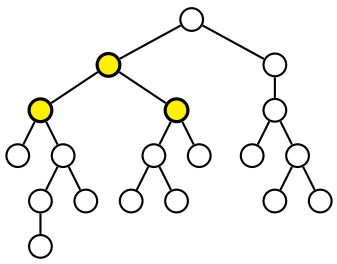
```
0 8
000 8
001 7
---
---
```

F
Hijos a tope

China es el país más poblado del mundo con más de un quinto de la población mundial. En 1953 empezó a realizarse en el país el primer censo moderno, revelando una población de cerca de 600 millones de habitantes. En los años 60 el gobierno comenzó a promover el retraso en la edad de contraer matrimonio y la planificación familiar. En 1972 se establecieron los primeros límites en el número de hijos por familia, siendo de dos en las ciudades. La política del hijo único se estableció en 1979 con objeto de frenar el crecimiento de la población, cuando esta ya estaba cercana a los 1000 millones de personas. Esta política combinaba el uso de la propaganda, la presión social, el establecimiento de beneficios e incluso penalizaciones económicas. Sin embargo, las minorías étnicas nunca han estado sujetas a dicha política, y en los entornos rurales a las parejas se les ha permitido tener otro hijo si su primogénito fue una niña. En 2015 se puso fin a la política del hijo único permitiendo de nuevo a todas las parejas tener dos hijos.

Un grupo de demógrafos chinos está haciendo un estudio con árboles genealógicos de familias chinas en las que se ha respetado el máximo de dos hijos que se estableció en 1972 y que recientemente se ha recuperado. En cada familia estudian las *subfamilias generacionalmente completas*, que son aquellas subfamilias en las que todos sus miembros han tenido exactamente dos hijos. Quieren saber cuál es la subfamilia generacionalmente completa con mayor número de generaciones.

Por ejemplo, en el siguiente árbol genealógico se ha resaltado la subfamilia generacionalmente completa con más generaciones, que en este caso son dos.



Entrada

La entrada comienza indicando el número de casos de prueba que vendrán a continuación. Cada caso consiste en una secuencia de números que pueden ser 0, 1 o 2 indicando el número de hijos de los miembros de la familia. Primero aparece el número de hijos de la raíz del árbol genealógico, y si tiene hijos, a continuación figuran las descripciones de las subfamilias de cada uno de sus hijos, que a su vez están descritas de la misma forma.

Los árboles genealógicos nunca contendrán más de 5.000 miembros.

Salida

Para cada árbol genealógico, se escribirá una línea con el número de generaciones de la subfamilia generacionalmente completa con mayor número de generaciones.

Entrada de ejemplo

```
5
0
2 0 0
1 2 1 0 1 0
2 2 0 1 2 0 0 2 0 0
2 2 2 0 2 1 0 0 2 2 0 0 0 1 2 0 2 0 0
```

Salida de ejemplo

```
0
1
1
2
2
```

G
Chocolate con almendras

A Marta y a Raúl les encanta el chocolate, relajarse tomando unas onzas después de cenar. Hoy en el supermercado han encontrado una muy buena oferta de chocolate con almendras. Aunque es el favorito de Marta, Raúl odia encontrarse tropezones cuando disfruta del chocolate; prefiere que este se vaya derritiendo en su boca, sin notar las almendras.

Después de pensárselo un rato, han decidido comprar unas cuantas tabletas y dividirlas para que las onzas con almendras queden separadas de las onzas sin ellas. Las tabletas solamente pueden partirse de forma cómoda en horizontal o vertical, por las separaciones ya marcadas entre onzas. Una vez que la tableta está divida en dos, cada una de las partes puede seguir siendo dividida.

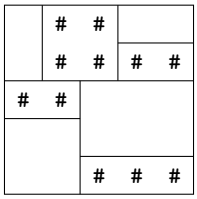
Ahora se preguntan cuántos cortes tendrán que hacer como mínimo para dividir cada una de las tabletas en porciones que solamente tengan onzas con almendras u onzas sin almendras.

Entrada

La entrada está formada por una serie de descripciones de tabletas de chocolate. Cada una comienza con una línea con dos números, la cantidad de filas F y columnas C en que está dividida en onzas la tableta (ambas entre 1 y 20). A continuación aparecen F líneas, cada una con C caracteres. El carácter '`.`' indica una onza sin almendras y el carácter '`#`' indica una onza con almendras.

Salida

Para cada tableta se escribirá en una línea el mínimo número de cortes que hacen falta para separar las onzas con almendras de las que no las tienen. Por ejemplo, la siguiente figura muestra una división óptima de la tableta del primer caso de prueba, donde hacen falta 7 cortes.



Entrada de ejemplo

```
5 5
.##..
.####
##...
.....
..###
1 6
...###
```

Salida de ejemplo

```
7
1
```

H
Las partituras de la orquesta

Para una orquesta modesta, gran parte del presupuesto se va en la compra de las partituras. Si cada músico tiene su propia copia, el número puede ascender a más de 100.

Afortunadamente, los músicos que tocan el mismo instrumento se sientan juntos y pueden compartir atril. Eso supone un ahorro considerable, aunque si se abusa demasiado puede provocar un efecto feo en la sala de conciertos. Al fin y al cabo ver a muchos músicos agolpándose alrededor del mismo atril no es muy estético.

La sección de asuntos económicos de la orquesta nos ha informado del número de partituras que podremos comprar para la próxima obra que se tocará. Teniendo en cuenta el número de músicos de cada instrumento, ¿cuál será el atril más concurrido?

Por ejemplo, si tenemos 8 violines, 5 violas, 5 violonchelos y 2 contrabajos y hay presupuesto para 6 partituras, podemos comprar dos copias de violines, dos de violas, una de violonchelos y otra de contrabajos. En ese caso el atril más compartido será el de los violonchelos, con 5 músicos.

Entrada

La entrada está compuesta de distintos casos de prueba, cada uno formado por dos líneas. La primera contiene dos enteros, p y n, con el número de partituras que podremos comprar (hasta 200.000) y el número de instrumentos *distintos* que hay en la orquesta (hasta 100.000). Se garantiza que se podrá comprar al menos una partitura para cada tipo de instrumento.

La segunda línea contiene n números positivos que indican la cantidad de músicos que hay en la orquesta para cada uno de los instrumentos (como mucho 1.000).

Salida

Por cada caso de prueba aparecerá un único entero indicando el número de músicos que se agolparán en el atril más compartido cumpliendo las restricciones de la orquesta y presupuesto. Recuerda que el objetivo es minimizar la cantidad de gente en el atril más concurrido.

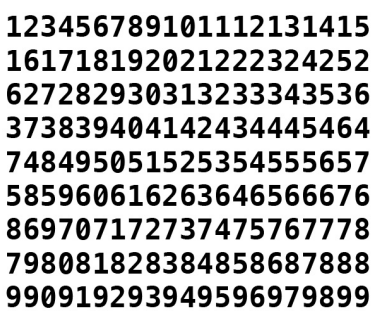
Entrada de ejemplo

```
4 4
8 5 5 2
6 4
8 5 5 2
7 4
8 5 5 2
```

Salida de ejemplo

```
8
5
4
```

I
¿Es múltiplo de 3?



La semana pasada, Andrea explicó a sus pequeños alumnos la prueba de divisibilidad por 3. "Un número es divisible por 3 — les contó — si la suma de sus dígitos lo es".

Para que la practicaran, decidió ponerles ejercicios. Pero, en un dudoso alarde de astucia, en lugar de ponerles muchos números largos, por acortar, les planteó un enunciado extraño. Cada ejercicio era un valor n con el que formar un gran número a partir de la concatenación de todos los números entre 1 y n. Por ejemplo, para $n = 2$, el número generado era el 12, para $n = 6$ el 123.456, y para $n = 13$ el gigantesco 12.345.678.910.111.213. Lo que Andrea pedía a sus chicos era que le dijeran si el número construído así era o no divisible por 3.

Esto le permitió poner ejercicios de enunciado corto, pero de solución larga. El problema llega ahora, que toca corregirlos.

Entrada

La entrada comienza con el número de casos de prueba. Cada caso de prueba se muestra en una línea y es un número entero positivo menor que 10^9.

Salida

Para cada caso de prueba, el programa escribirá "`SI`" si el número formado es múltiplo de 3, y "`NO`" en caso contrario.

Entrada de ejemplo

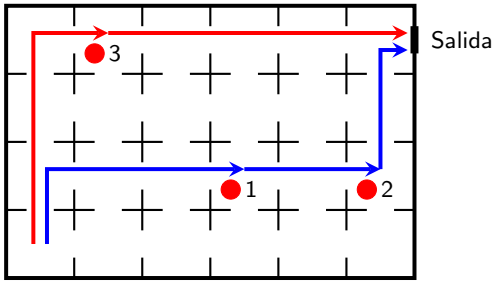
```
3
2
6
130000000
```

Salida de ejemplo

```
SI
SI
NO
```

J
Más listos que el hambre

En el laboratorio estamos realizando una serie de experimentos para estudiar la inteligencia de una especie de ratones, los *ratones coloraos*. En los experimentos colocamos a unos ratones en una caja rectangular dividida en celdas como la de la figura. Los ratones se colocan en la esquina inferior izquierda y deben encontrar la salida situada en la esquina superior derecha. Para ello pueden moverse libremente por todas las celdas, pero de cada celda solamente pueden pasar a las colindantes en horizontal o vertical.



Para estos ratones eso está chupado, por lo que hemos ido complicando el experimento. Primero pusimos una serie de botones (círculos rojos en la figura) en algunas celdas de tal forma que los ratones tuvieran que pulsar todos esos botones para que se abriera la puerta de la salida. Pronto aprendieron a hacerlo, y además de la forma más rápida posible.

Para potenciar el trabajo en equipo, ahora hemos numerado los botones del 1 al N, y además han sido electrificados[1]. Para que el botón i sirva para abrir la puerta, y además no suelte una descarga eléctrica, deben haberse pulsado antes los botones con un número menor. En el experimento se colocan dos ratones en la celda inicial y ambos deben colaborar pulsando los botones en orden para lograr salir del laberinto en el menor tiempo posible. No hay más restricciones sobre qué botones pulsa cada ratón, salvo que si un ratón pulsa el botón i, antes él o su compañero deben haber pulsado el botón $i - 1$. Si lo logran encontrarán un suculento queso en la salida. Si no, solamente tendrán pan duro.

Nosotros somos mucho menos listos que los ratones y no sabemos si lo están haciendo en el mejor tiempo. ¿Nos puedes ayudar a calcularlo suponiendo que un ratón siempre tarda 1 segundo en pasar de una celda a otra colindante? El tiempo que tarda un ratón en pulsar un botón es despreciable, por lo que en el mismo segundo un ratón podría pulsar el botón i y el otro el botón $i + 1$. Ten en cuenta que la presencia de un botón en una celda no impide que un ratón pueda pasar por ella sin pulsarlo.

Entrada

La entrada está compuesta por una serie de casos. Para cada caso, la primera línea contiene el número F de filas y el número C de columnas del entramado de celdas que contiene la caja (ambos números pueden estar entre 1 y 50). La siguiente línea contiene el número N de botones (entre 0 y 100) que deben pulsarse. En las siguientes N líneas se dan las posiciones de estos botones (una fila entre 1 y F y una columna entre 1 y C) en el orden en que tienen que ser pulsados.

Salida

Para cada caso se escribirá el tiempo mínimo que necesitan los dos ratones situados inicialmente en la celda (1,1) para salir de la caja habiendo pulsado todos los botones sin haber recibido ninguna descarga. Una vez alcanzada la celda (F,C), si la puerta de salida está abierta, los ratones pueden salir con 1 segundo más.

[1] En la elaboración de este problema ningún animal ha sufrido daños. Se ha tenido un escrupuloso cuidado en respetar los derechos de los animales y las leyes contra el maltrato animal.

Entrada de ejemplo

```
4 6
3
2 4
2 6
4 2
3 3
4
2 1
1 2
3 1
1 3
```

Salida de ejemplo

```
11
5
```

K
Las perlas de la condesa

El collar de perlas de la condesa es conocido por la calidad y la cantidad de sus perlas. Lo luce en las grandes fiestas, dándole tres o cuatro vueltas alrededor de su cuello. La perla del centro es la de mayor tamaño, y según nos desplazamos hacia los extremos son estrictamente más pequeñas. Las perlas están perfectamente seleccionadas y colocadas, lo que hace al collar completamente simétrico respecto a la perla central.

Durante el último baile, se ha roto el cierre y las perlas han rodado por el suelo. Los invitados han intentado recuperarlas todas ensartándolas en un nuevo cordel según las encontraban. Al día siguiente, la condesa ha llamado a su joyero para que las engarce de nuevo en el orden adecuado. Éste mide el diámetro de cada una de ellas y procede a rehacer el collar colocando la perla central y a continuación las siguientes en tamaño hasta llegar a los extremos. A la condesa no le importa si se ha perdido alguna perla, siempre y cuando el collar siga siendo completamente simétrico.

Entrada

Cada línea de la entrada forma un caso de prueba, que consiste en una lista de números positivos separados por espacios. Cada uno representa el diámetro de una de las perlas en el orden en el que las fueron recogiendo los invitados. La perfección del collar es legendaria, por lo que el diámetro se mide con una unidad de medida que muchos consideran infinitesimal, aunque el diámetro será siempre menor que 2^{31}. Cada lista tiene $0 < \textit{perlas} < 1.000$ y acaba siempre con un cero.

El último caso de prueba, que no deberá procesarse, contiene un collar vacío, representado por una lista con un único cero.

Salida

Para cada caso de prueba, el programa escribirá "`NO`" si no es posible formar un collar simétrico con todas y cada una de las perlas encontradas de forma que la perla de mayor tamaño quede en el centro. En otro caso, se escribirán, separados por espacios, los diámetros de las perlas tal y como han quedado ordenadas en el collar, desde un extremo al otro.

Entrada de ejemplo

```
2 3 2 3 5 7 5 0
2 2 7 10 0
10 0
0
```

Salida de ejemplo

```
2 3 5 7 5 3 2
NO
10
```

L
Durmiendo en albergues

Como cualquier buen peregrino sabe, los albergues que uno encuentra por los caminos suelen tener grandes habitaciones donde hileras de camas unas junto a las otras esperan a ser ocupadas.

Aunque esas camas han sido usadas por infinidad de gentes antes, el cansancio hace que a uno no le importe tumbarse en colchones raídos por el tiempo con tal de dormir un puñado de horas seguidas y coger fuerzas para la jornada siguiente.

De hecho, para maximizar las posibilidades de dormir bien, lo más importante no es la salubridad del colchón, sino la distancia con tu vecino peregrino más cercano. Y es que una noche con un desconocido en la cama de al lado roncando puede arruinarte la noche mucho más que un puñado de ácaros.

Cuando entras en una de estas habitaciones, pues, el objetivo es encontrar la cama que maximice la distancia con el vecino más cercano (hacia cualquiera de los lados).

Entrada

La entrada está formada por distintos casos de prueba representando la ocupación de una hilera de camas una noche del camino.

Cada hilera aparece en una única línea que contiene una secuencia de `.` y `X` (hasta 500.000 caracteres). Los puntos representan camas libres, mientras que las `X` representan camas ocupadas. Se garantiza que habrá siempre al menos una cama libre y otra ocupada.

Salida

Por cada caso de prueba se escribirá una línea con el número de camas vacías más grande posible que se puede conseguir entre la cama seleccionada y tu vecino más cercano.

Entrada de ejemplo

```
.X.X.
.X...X.
.X....X.
...X
```

Salida de ejemplo

```
0
1
1
2
```