# Concurso de Programación
# AdaByron 2021

http://www.ada-byron.es

# Cuadernillo de problemas

Patrocinado por

accenture idealista

ThoughtWorks autentia atsistemas

Realizado en la **Escuela Técnica Superior de Ingeniería Informática (URJC)**
23 y 24 de abril de 2021

URJC Escuela Técnica Superior de Ingeniería Informática

*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**

23 y 24 de abril de 2021

# Índice



Autores de los problemas:

- Alejandra Casado Ceballos (Universidad Rey Juan Carlos)
- Antonio González Pardo (Universidad Rey Juan Carlos)
- Daniel Alejandro Ampuero Anca (HyperScience)
- Dixon David Morán González (ThoughtWorks)
- Isaac Lozano Osorio (Universidad Rey Juan Carlos)
- Iván Martín de San Lázaro (Universidad Rey Juan Carlos)
- Jakub Jan Luczyn (Universidad Rey Juan Carlos)
- Jesús Sánchez-Oro Calvo (Universidad Rey Juan Carlos)
- Leonardo Antonio Santella Dambrosio (Amazon)
- Raúl Martín Santamaría (Universidad Rey Juan Carlos)
- Sergio Cavero Díaz (Universidad Rey Juan Carlos)
- Sergio Pérez Peló (Universidad Rey Juan Carlos)

Tiempo: 1 segundo

# A
# Dickie y las subsecuencias

¡A Dickie le encantan las **subsecuencias crecientes!**

Es por eso que le han asignado la simple tarea de contar subsecuencias. Sin embargo, Dickie tiene una resaca cerebral a causa de quedarse resolviendo problemas hasta altas horas de la madrugada. ¡Es por eso que necesita tu ayuda!

Dado un array de $n$ **elementos distintos**, y un entero $k$, vuestra labor será la de contar el número de **subsecuencias estrictamente crecientes** de $k$ elementos que hay en el array.

Está garantizado que el número de subsecuencias no decrecientes es menor a $8 \times 10^{18}$.

*Una subsecuencia es una secuencia que puede ser derivada eliminando elementos del array.* Por ejemplo siendo: $A = [1, 6, 8, 2, 7, 91, 12]$, una subsecuencia de 4 elementos sería: $[1, 2, 7, 91]$; sin embargo $[2, 7, 91, 1]$ no es una subsecuencia válida, ya que no se puede obtener eliminando elementos de $A$.

## Entrada

La primera línea de la entrada será $n$ y $k$, donde $n$ es el número de elementos distintos en el array, y $k$ la longitud de las subsecuencias que debéis buscar. Después, seguirán $n$ líneas que contendrán un entero $a_i$ que representa el valor almacenado en la posición del array.

## Salida

Imprimir el número de **subsecuencias estrictamente crecientes** de longitud $k$ presentes en el array.

## Entrada de ejemplo

```
8 3
1
5
3
7
10
17
12
```

## Salida de ejemplo

```
12
```

## Límites

- $1 \leq n \leq 10^5$
- $1 \leq k \leq 10$
- $1 \leq a_i \leq 10^{18}$

Tiempo: 1 segundo

# B
# Byron y las cervezas

El gran Byron se encuentra, como de costumbre, en su bar favorito junto a su gran amigo Dickie. Los dos lo pasan genial, ya que a Dickie le gusta retar a Byron, y a este último le gusta ganar a Dickie en cualquier tipo de reto.

El bar en el que se encuentran tiene una lista con $N$ tipos de cerveza, y conocen la cantidad de calorías, $c_i$, que cada cerveza aporta a la persona que las toma.

En esta ocasión, Dickie ha retado a Byron a tomarse $Q$ cervezas. Pero cada cerveza se debe encontrar en un intervalo específico que Dickie le tiene que indicar: $L$ y $R$.Por lo que Byron deberá tomar una cerveza que se encuentre dentro de dicho intervalo de la lista de cervezas.

El problema es que Byron desea ganar la menor cantidad de peso posible, ya que el verano está cerca y tiene pensado ir a visitar el mar. Por esta razón, Byron necesita de tu ayuda para encontrar la cerveza que le aporte menos calorías y que se encuentre dentro del intervalo dado por Dickie.

## Entrada

La primera línea de la entrada sera $N$ y $Q$. En la segunda línea, habrá $N$ enteros, $c_i$, separados por un único espacio. Y finalmente, aparecerán $Q$ líneas que contienen los valores para $L$ y $R$.

## Salida

Imprimir $Q$ líneas, cada una de ellas deberá imprimir $min\ (c_L, c_{L+1}, ..., c_R)$.

## Entrada de ejemplo

```
8 2
1 5 3 7 10 17 12 15
0 7
5 7
```

## Salida de ejemplo

```
1
12
```

## Entrada de ejemplo

```
8 2
100 55 31 7 103 170 120 110
0 7
4 7
```

## Salida de ejemplo

```
7
103
```

## Límites

- $1 \leq N \leq 10^5$ $\quad 1 \leq Q \leq 10^5$ $\quad 1 \leq c_i \leq 10^{18}$
- $0 \leq L, R \leq N-1$ $\quad L \leq R$

Tiempo: 1 segundo

# C
# Bar BBTOtra

En época de pandemia, los pobres estudiantes no pueden ir a casi ningún bar, porque todos están lejos y tendrían que saltarse el cierre perimetral. Para que los estudiantes puedan tener un sitio donde socializar, hemos decidido abrir nuevas franquicias del *Bar BBTOtra*. Cada estudiante irá siempre al bar más cercano, y nos hemos asegurado de que nunca habrá problemas de aforo.

Dado un número de franquicias que queremos abrir y las localizaciones de los pisos de estudiantes, queremos elegir las localizaciones que minimicen la distancia del estudiante más lejano a su bar más cercano. Si se abre un bar en la misma localización de un piso de estudiantes, la distancia entre el bar y el piso será 0.

## Entrada

La primera línea contiene tres enteros $N$, $P$ y $M$ que indican el número de localizaciones donde podemos abrir un bar, el número de bares que vamos a abrir, y el número de líneas con distancias que vienen a continuación (siendo estas las distancias entre todas las posibles localizaciones). Las siguientes $M$ líneas contienen tres enteros $U$, $V$, y $D$ que indican que la distancia entre las localizaciones $U$ y $V$ es $D$.

## Salida

Se imprimirá por consola la distancia máxima que tiene que recorrer un estudiante para ir a su bar más cercano en el mejor de los casos posibles.

## Entrada de ejemplo

```
5 3 10
0 1 266
0 2 298
0 3 702
0 4 932
1 2 701
1 3 822
1 4 870
2 3 912
2 4 191
3 4 668
```

## Salida de ejemplo

```
266
```

## Límites

- $5 \leq N \leq 30$
- $3 \leq P \leq 5$
- $1 \leq D \leq 1000$

Tiempo: 3 segundos

# D
# Lluvia de documentos - Problema Accenture

En la ciudad de Byron ha llegado el momento de recaudar impuestos: todos sus ciudadanos deben presentar papeles para realizar la declaración de la renta. El alcalde se encargaba, hasta ahora, de procesar todos los documentos a mano, pero esto provocaba que, a veces, se recibiesen documentos y se procesaran dos veces o más. Lo que suponía un calvario para el ciudadano al que le reclamaban varias veces su cotización. Para evitar conflictos, este año el alcalde ha decidido informatizar el sistema, asignando un identificador único a cada ciudadano, además de incorporar el nombre de la persona que lo envía. Tu misión es detectar si alguien se ha despistado y ha enviado el documento dos veces. Como cada ciudadano tiene su identificador, no importa que dos personas compartan nombre y apellido.

## Entrada

La entrada comenzará con el número $N$ de documentos que se han recibido en la sede de Hacienda de Byron. A continuación, vendrán $N$ líneas con el identificador $I$ de un ciudadano junto a su nombre $T$. El identificador será un número de 1 a 100000 y el nombre consistirá en una sola cadena de caracteres con el formato *NombreApellido1Apellido2* que no superará los 30 caracteres.

## Salida

Por cada caso de prueba, se debe imprimir "`OK`" si todos los ciudadanos han entregado su documento una sola vez y "`ERROR`" si existen dos documentos iguales.

## Entrada de ejemplo

```
4
1 JuanRodriguezPaez
2 SergioRamosRamos
3 AlaricoChisdasvintoNarvaez
4 SergioRamosRamos
```

## Salida de ejemplo

```
OK
```

## Límites

- $1 \leq N \leq 1000000$
- $1 \leq I \leq N$

Tiempo: 1.5 segundos

# E
# Flamingo airlines - Problema Autentia

Los clientes de la nueva compañía *Flamingo Airlines* están esperando en la puerta de embarque del aeropuerto. Evidentemente, estos clientes son flamencos, esperando a embarcar en el avión. En un primer momento, la compañía creó un sistema de embarque basado en prioridades, pero los flamencos han llegado allí y se han puesto en la cola sin tenerlo en cuenta.

Ignorar el sistema de prioridades ha generado un nuevo problema. Cada flamenco, orgulloso como nadie, va a montar un escándalo si ve un flamenco de menor edad embarcar antes que él. No solo eso, sino que va a montar un escándalo tan largo como la diferencia de edad entre ellos. La aerolínea no puede continuar con el embarque hasta que no se calmen todos los flamencos de la cola (¡Solo imagina intentar montar un flamenco en un avión con otros 3 organizando un albedrío!), así que cada vez que embarque uno, habrá que esperar hasta que la situación se calme.

Así, por ejemplo, si hay 3 flamencos en la cola, de edades 4, 5 y 6, y el primer flamenco embarca, los otros dos flamencos montarán un escándalo. El flamenco de edad 5 tardará 1 minuto en calmarse, mientras que el otro tardará 2. Esto retrasará el avión 2 minutos (Ya que los dos flamencos montarán a la vez el escándalo), y otro minuto más, cuando embarque el flamenco de edad 5 y el de edad 6 monte otro escándalo. En este ejemplo, el avión saldría con un total de 3 minutos de retraso.

Para la compañía, es más barato embarcar a los flamencos siguiendo el orden que ya haya en la cola, y ver todos y cada uno de los escándalos montarse y calmarse, que reordenar la cola, porque eso les haría perder dinero o recibir quejas de algunos flamencos. A *Flamingo Airlines* le gustaría saber ahora cuánto tiempo va a perder en embarcar a todos los pasajeros en el avión. Sabiendo las edades de los flamencos que hay en la cola, y el tiempo original en el que tendría que haber despegado el avión, ¿podrías calcular con cuánto retraso irá el vuelo?

*Nota: Se puede asumir que el tiempo que tarda un flamenco en embarcar es despreciable. Por lo tanto, si embarca el flamenco más viejo de la cola y no se forma escándalo, el siguiente flamenco embarca sin perder tiempo.*

## Entrada

La primera línea contiene un número entero $C$, que especifica el número de casos. A continuación seguirá la descripción de $C$ casos, cada uno descrito en dos líneas. La primera línea contendrá el número $F$ de flamencos esperando en la cola, y un número $T$ indicando el tiempo original de salida del avión. La segunda línea va a contener $F$ números, indicando la edad $E$ de cada flamenco en la cola.

## Salida

Por cada caso de prueba, se debe imprimir el tiempo de retraso con el que despegará el avión. Si tras todos los escándalos el avión consigue despegar a tiempo (o antes), se imprimirá "`En tiempo`".

## Entrada de ejemplo

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

## Salida de ejemplo

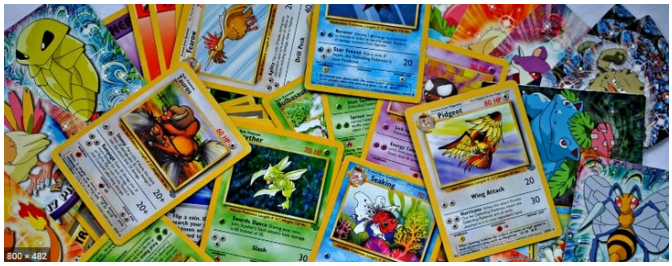
```
9
En tiempo
```

## Límites

- $1 \leq F, T, E \leq 10^6$

Tiempo: 1.5 segundos

# F
# Las cartas Pukemon



A nuestro amigo Navi le encanta coleccionar cartas Pokémon, y tiene barajas que son la envidia de la universidad.

Un día, los malvados becarios del 117 le deciden esconder las cartas en el despacho, pero por suerte Navi estaba sobre aviso. El día anterior escondió en su mazo un chip localizador que, junto a una aplicación móvil que él mismo ha desarrollado, es capaz de saber aproximadamente la dirección donde están sus cartas. Cada vez que su móvil realice una petición, el chip rastreador le responde con la dirección relativa a su posición actual, en la que se encuentran las cartas. La respuesta del rastreador puede ser $+$, $-$ o $=$, dependiendo de la posición de las cartas respecto a la posición actual de Navi. Por ejemplo, si Navi está en la posición $(7, 3, 5)$, y las cartas están en $(3, 5, 5)$, el chip rastreador respondería $(-, +, =)$.

Navi empieza en la posición $(0, 0, 0)$, y puede desplazarse a cualquier punto antes de usar el localizador.

Sin embargo, el chip tiene batería limitada, y cada vez que se comunica con él, se reduce la duración de la batería en una unidad. De manera que cuando llega a 0, se apaga. Se considera que Navi recupera sus cartas en el momento en el que el localizador devuelve $(=, =, =)$.

Sabiendo la batería del chip, ¿podrá Navi recuperar sus cartas?

## Entrada

La entrada comienza con una línea con dos números: $N$, el número de casos, y $B$, el nivel de la batería inicial del chip rastreador. En cada una de las siguientes $N$ líneas, aparecen 3 números, $X$, $Y$, $Z$ (largo, ancho, alto), el tamaño de la habitación donde están escondidas las cartas.

## Salida

Para cada caso de prueba de la entrada se pide: escribir "`PUKECARTAS`" si Navi consigue encontrar sus cartas usando una estrategia inteligente, o "`SIN BATERIA`" si es posible que se quede sin batería antes de encontrar sus cartas. El rastreador empieza con la batería cargada en cada uno de los casos de prueba.

## Entrada de ejemplo

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

## Salida de ejemplo

```
PUKECARTAS
SIN BATERIA
```

## Límites

- $1 \leq X, Y, Z \leq 2^{63} - 1$
- $1 \leq N \leq 500000$
- $1 \leq B \leq 100$

Tiempo: 1 segundo

# G
# El alineador

Idealmente, un entrenador de un equipo de fútbol es un líder, es decir, una persona que conoce y comprende las necesidades de su equipo. Además, tiene la capacidad de transmitir sus ideas y pensamientos en el juego de su equipo. Es muy difícil encontrar entrenadores capaces de llevar a cabo todas esas tareas, y, cuando lo encuentras, se ve comprometido por los resultados de su equipo, más que por el juego que desempeñan. El entrenador se sienta cada partido en una silla de cristal que depende de un grupo de jugadores, de forma que cuando ganan, ganan todos; pero si pierden, pierde el entrenador.

El Rayo Vaticano acaba de cesar a su séptimo entrenador tras 20 jornadas de la liga. Cansados de buscar, contratar y despedir entrenadores, han decidido invertir en un software que les permita ahorrarse la figura de entrenador en su equipo. Este software valida la alineación de cualquier persona, sin necesidad de que tenga conocimientos de fútbol. Dada la plantilla del equipo y un esquema de juego o formación (número de porteros, defensas, mediocentros y delanteros en una alineación), un alineador determina qué jugadores juegan, y el software valida que la alineación es correcta. Un problema menos para el Rayo Vaticano.

## Entrada

La entrada de este problema consta de múltiples líneas. La primera de las ellas contiene 5 números. El primero, J, representa el número de jugadores que tiene la plantilla del equipo. Los siguientes 4 números se corresponden el esquema de juego o formación: el primer número, P, es el número de porteros en la alineación; el segundo, D, el número de defensas; el tercero, M, el número de mediocentros; y, por último, F, el número de delanteros. La segunda línea del programa se corresponde con la alineación determinada por el alineador. Esta línea contiene P+D+M+F números. Cada uno de los números se corresponde con el dorsal de un jugador de la plantilla. Los números de esta línea en ningún caso estarán repetidos. Tras estas dos líneas vienen $3 \times J$ líneas, donde cada 3 líneas consecutivas se corresponden con la información relativa a un jugador. La primera de las tres líneas es el dorsal del jugador; la segunda de ellas, el nombre del jugador (representado con caracteres en inglés, espacios, mayúsculas y minúsculas) que puede estar formado por más de una palabra; y, por último, la tercera línea informa sobre la posición del jugador. La posición del jugador puede ser "P", "DF", "MC", "DL" que se corresponde con portero, defensa, mediocentro y delantero, respectivamente. En ningún caso habrá jugadores con dorsales repetidos.

## Salida

La salida de este programa depende de si la alineación establecida por el alineador es correcta o no. Se considera que una alineación es incorrecta cuando el número de jugadores por posición establecidos por el alineador no coincide con los de la formación o esquema de juego. En el caso de que la alineación sea incorrecta se imprimirá por pantalla "`Error en la alineacion`".

Por otro lado, si la alineación es correcta se deberán imprimir 4 líneas. La primera línea escribe el nombre de los porteros de la alineación separados por espacios. De la misma manera, en la segunda, tercera y cuarta línea se escribe el nombre de los defensas, mediocentros y delanteros separados por espacios respectivamente. El orden de los nombres de los jugadores en cada una de las líneas será el mismo orden con el que se leyeron de la entrada. Si no hay jugadores para una determinada posición se imprimirá una línea vacía.

## Entrada de ejemplo

```
5 1 1 1 1
3 1 9 7
9
Castolo
DL
5
Bautista
MC
3
Facu
DF
7
Ziderm
MC
1
Kakasilla
P
```

## Salida de ejemplo

```
Kakasilla
Facu
Ziderm
Castolo
```

## Entrada de ejemplo

```
5 1 1 0 1
4 1 9
9
Castolo
DL
5
Bautista
MC
3
Facu
DF
1
Van de Vij
DF
4
Frank de Mole
DF
```

## Salida de ejemplo

```
Error en la alineacion
```

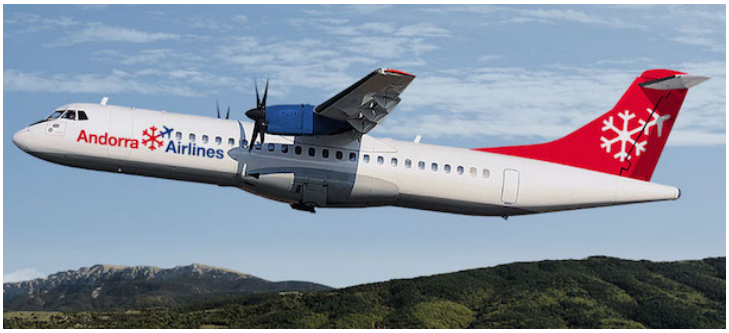
## Límites

- $0 \leq P, D, M, F \leq 250$
- $0 \leq J \leq 1000$

Tiempo: 1 segundo

# H
# Nos vamos de viaje

Durante este año, mudarse a Andorra ha sido uno de los temas más polémicos. Sin embargo, esto no es nuevo, se lleva haciendo mucho tiempo.




Un famoso youtuber está buscando un nuevo sitio para sus proyectos. Para ello, necesita hacer una gran mudanza y está interesado en reducir al máximo el coste (tiene demasiado material para organizar los directos, y es muy caro como para comprarlo de nuevo). Además, sabe que cada euro que pueda ahorrar en la mudanza lo puede invertir en otro tipo de material, lo que a la larga le puede reportar mayores beneficios. La empresa de transporte que utilizará para la mudanza solo le va a cobrar los gastos del peaje a cambio de que consiga repercusión en redes sociales de su empresa. La ruta la debe decidir el youtuber, ya que la empresa no quiere responsabilizarse del material (saben que muchos fans pueden hacer cualquier cosa para robar cosas del youtuber y es demasiado riesgo). Para informarse de las carreteras con peajes que existen, este youtuber ha buscado un mapa de la región, el cual muestra dónde puede moverse en cualquier dirección (arriba, abajo, derecha, izquierda y en diagonal). Por si fuera poco, nuestro youtuber quiere ahorrarse hasta el último céntimo, así que sólo utilizará esta empresa si es capaz de pagarles con el valor exacto.

El youtuber sabe cuántos billetes tiene de cada tipo. ¿Puedes ayudarle a saber si con los billetes de los que dispone puede pagar de forma exacta el precio de la mudanza? En caso de que pueda, ¿podrías decir cuántos billetes de cada tipo debería pagar para que el número total de billetes sea lo más pequeño posible?

## Entrada

La entrada comenzará con 2 números enteros, $N$ que define el número de filas, y $M$ que se refiere al número de columnas. Estos números determinan el tamaño del mapa del youtuber. Por cada fila $N$, encontraremos una línea con $M$ números, $C$, que marcarán el coste de pasar por esta ciudad (el punto de salida vendrá representado con la letra "$S$" y el del final de trayecto vendrá representado con la letra "$F$", con un coste de 0). Seguirá un número $E$, que muestra el total de billetes que dispone el youtuber. Finalmente, aparecerán dos líneas. En la primera línea vendrá el valor $V$ de cada uno de los $E$ billetes que tiene el youtuber, y en la segunda línea vendrán $E$ números mostrando la cantidad de billetes $B$ que existe por cada uno de los anteriores.

## Salida

La salida del programa será la distancia mínima a recorrer seguido de un "`SI`" en caso de que se pueda pagar de manera exacta, mostrando el número mínimo de billetes a utilizar de cada tipo separados por un espacio. En caso de que no pueda pagar con el dinero exacto a la compañía de mudanzas y le toque buscar otra ruta alternativa para cumplir ese requisito se imprimirá "`NO`". Si existen varias soluciones que utilicen el mínimo número de billetes posibles, se escribirá aquella que utilice más billetes de los tipos que aparecen antes en la entrada.

## Entrada de ejemplo

```
5 5
5 20 3 F 1
2 3 4 50 20
40 2 3 45 22
11 2 0 0 1
S 0 2 5 8
4
1 5 10 50
10 2 5 4
```

## Salida de ejemplo

```
6
SI 1 1 0 0
```

## Entrada de ejemplo

```
5 5
5 20 3 F 1
2 3 4 50 20
40 2 3 45 22
11 2 0 0 1
S 0 2 5 8
4
1 5 10 50
5 0 1 1
```

## Salida de ejemplo

```
6
NO
```

## Límites

- $1 \leq N, M \leq 1{,}000$
- $1 \leq C \leq 100{,}000$
- $0 \leq$ La suma del camino más corto $\leq 30{,}000$
- $1 \leq E, B, V \leq 50$

Tiempo: 1 segundo

# I
# First Dates - Problema URJC


En la nueva temporada de First Dates van a establecer un nuevo modelo de cita. En ella, se dispone de una planta con habitaciones cuadradas del mismo tamaño, y en cada habitación habrá una persona dispuesta a tener una cita. A cada persona se le asignará un ticket con un número diferente. La persona con el menor ticket estará en la habitación de la parte superior izquierda; y la del ticket de mayor valor, en la inferior derecha. El resto de habitaciones estarán ordenadas de manera ascendente según el valor del ticket de izquierda a derecha y de arriba abajo.

Cada nuevo participante solicitará una cita diciendo un número y el programa le indicará la habitación donde está la cita con el valor del ticket solicitado, teniendo en cuenta que la primera habitación es la $(0, 0)$. En caso de no haber nadie con el valor del ticket solicitado, se le asignará la habitación con ticket de valor más cercano, pero mayor (si no hay nadie con el ticket mayor se le asignará la última habitación). Dado un conjunto de participantes, el programa nos ha pedido que asignemos la habitación que le corresponde de la forma más rápida posible, ya que tod@s l@s solter@s están impacientes por tener su cita.

## Entrada

La primera línea contiene dos enteros $N$ y $M$ que indican el número de filas de habitaciones y el número de habitaciones por fila, respectivamente. Las siguientes $N$ líneas contienen $M$ enteros $A$ que indican el valor del ticket del participante que estará en la habitación. Finalmente, se recibirá un entero $P$ en cada línea indicando el valor del ticket que solicitan los nuevos participantes, terminando con -1.

## Salida

Por cada habitación solicitada, se imprimirá en una nueva línea "`TU CITA ESTA EN LA HABITACION I J`", donde $I$ y $J$ representan la fila y la columna de la habitación elegida.

## Entrada de ejemplo

```
5 5
18 33 37 47 54
57 59 66 74 75
136 164 166 170 175
187 190 205 216 217
222 227 232 241 247
110
-1
```

## Salida de ejemplo

```
TU CITA ESTA EN LA HABITACION 2 0
```

## Límites

- $1 \leq N, M \leq 1000$
- $1 \leq A, P \leq N \cdot M \cdot 10$

Tiempo: 1 segundo

# J
# La agenda del recuerdo

Jesús es un señor muy... digamos, clásico. Le encanta contar historias y comparar el mundo y la época actual con cómo se hacían las cosas habitualmente antes. "*Cuando yo tenía tu edad...*", "*En mi época...*" son, entre otras frases cliché, sus preferidas.

La edad nos alcanza a todos, eso es un hecho irrefrenable. Jesús ya no tiene la misma memoria que antes y, a veces, se le olvidan las cosas. Para ello, ha empezado a escribir las palabras que habitualmente se le olvidan, para luego buscarlas en algún diccionario físico, de esos con más de 500 páginas.

Sin embargo, tiene un gran problema: no puede acordarse de la palabra que estaba buscando, recuerda solo los prefijos de la misma. Para recordar, por ejemplo, 'retiro', estuvo unos cinco minutos dándole vueltas al asunto. "Re, Re, Re, Hmm no recuerdo", esto fue lo que escuchó su familia aquel día.

Sin más, se ha decidido a pedir ayuda al Ada Byron para que le ayuden en esta difícil tarea. ¿Podrías recordarle a Jesús las palabras que ha escrito sabiendo algún prefijo de ellas?

## Entrada

La primera línea de entrada será un número $N$ denotando las palabras escritas por Jesús, luego, $N$ líneas que contienen una palabra cada una.

Después, un entero $Q$ que representa el número de palabras que Jesús quiere recordar, seguido de $Q$ líneas con una palabra en cada línea.

## Salida

Imprimir, por cada $Q_i$ la frase "`Caso #x:`" donde $x$ es el número del caso (de 1 hasta $Q$) que se está contestando. En la siguiente línea, la frase "`Sin combinacion`" (sin acento) si no es posible encontrar alguna palabra que obtenga ese prefijo, de lo contrario, todas las palabras que contengan ese prefijo, una por línea.

## Entrada de ejemplo

```
5
retiro
remar
remitente
latente
risa
2
re
sol
```

## Salida de ejemplo

```
Caso #1:
remar
remitente
retiro
Caso #2:
Sin combinacion.
```

## Límites

- $1 \leq N \leq 25,000$
- $1 \leq Q \leq 25,000$
- Todas las letras de la palabra están en minúsculas y no tienen espacios.

Tiempo: 1 segundo

# K
# Coleccionando cómics

El nuevo fascículo de tu serie de cómic 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: los colocan en varias pilas y el cliente solamente puede elegir 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 tiene disponible en el momento de entrar a la tienda, tienes que calcular el puesto de la cola que debes ocupar el día del lanzamiento para hacerte con el mejor ejemplar de la tienda.

## Entrada

La entrada empezará con el número de pilas de cómics $N$. A continuación, vendrán $N$ líneas que contienen con un número $K$ que indica la cantidad de cómics en la pila correspondiente, seguido de $K$ números positivos $I$, los identificadores, ordenados desde la cima de la pila hasta la base.

## Salida

Imprimir la posición que tienes que ocupar en la cola a la tienda para llevarte el mejor ejemplar.

## Entrada de ejemplo

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

## Salida de ejemplo

```
6
```

## Entrada de ejemplo

```
3
1 2
3 4 5 6
7 8 9 10 11 12 13 14
```

## Salida de ejemplo

```
1
```

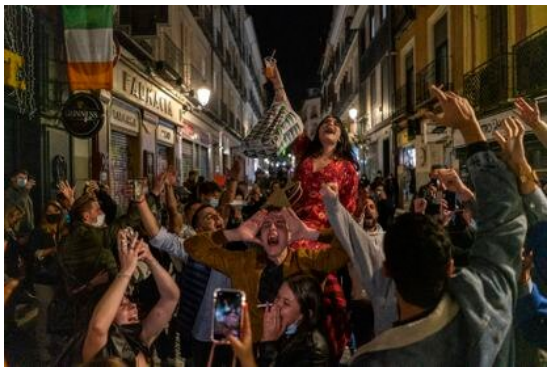
## Límites

- $1 \leq N \leq 10^5$ $\quad$ $1 \leq K \leq 10^5$
- $1 \leq I \leq 10^8$ $\quad$ $1 \leq \sum_{i=1}^{N} K_i \leq 2{,}5 \cdot 10^5$

Tiempo: 0.25 segundos

# L
# Fiesta madrileña

Recientemente, jóvenes de diferentes partes de Europa vienen a ciudades como Madrid, o Barcelona, para pasar el fin de semana y de paso, disfrutar de las fiestas nocturnas que se organizan en estas ciudades. Debido a la situación actual por la COVID-19, la policía nacional busca la manera de evitar que se produzcan esas fiestas, pero necesitan de vuestra ayuda para conseguirlo.



Por cada avión que aterriza en el aeropuerto, la policía pregunta a los turistas sospechosos por las personas de contacto con las que van a pasar el fin de semana (pueden indicar tantas personas de contacto como quieran siempre y cuando, como mínimo, indiquen 1). Además, la policía sabe que en un fin de semana se va a organizar mínimo una fiesta, pero desconocen el número exacto.

Con toda esta información, vuestra labor es la de ayudar a la policía a desarticular las diferentes fiestas clandestinas. Y para realizar esta tarea se necesita conocer por un lado el número exacto de fiestas que se planean celebrar, y por otro, el nombre de los turistas que se deben controlar para desarticular cada fiesta.

## Entrada

La primera línea contiene dos enteros $N$ y $M$ que representan el número de personas sospechosas y el número de personas de contacto en total, respectivamente. Las siguientes $M$ líneas contendrán dos enteros $r_1$ y $r_2$ que representan que el turista $r_1$ ha identificado a $r_2$ como persona de contacto.

## Salida

La salida de vuestro programa mostrará diferente información. En la primera línea aparecerá un número entero $F$ que indicará el número de fiestas diferentes que se van a celebrar. A continuación, deberán seguir $P$ líneas que contendrán (cada una de ellas) el número identificativo de los turistas que se necesita controlar. Hay que tener en cuenta que, si hay varios turistas clave, estos se deberán imprimir en orden ascendente.

## Entrada de ejemplo

```
12 14
6 5
9 10
4 3
0 5
8 10
1 4
2 3
9 11
0 6
3 1
7 4
8 9
1 2
5 4
```

## Salida de ejemplo

```
2
4
5
9
```

## Límites

- $1 \leq N \leq 10^6$
- $1 \leq M \leq 10^6$
- $0 \leq r_1, r_2 \leq N$
- $1 \leq P \leq N$

Tiempo: 1.5 segundos

# M
# La fortaleza

Espartaco ha llegado al monte del Vesubio con esclavos, liberados de sus cadenas, y sus hermanos gladiadores después de varios enfrentamientos con los romanos. Han logrado escudarse en un templo en ruinas, en una posición estratégica que les permite vigilar si los romanos vienen por los flancos o de frente. Sin embargo, Espartaco sabe que el ataque es inminente y ha decidido ponerse manos a la obra para defender lo máximo posible esta posición tan estratégica.

El templo en ruinas tiene algunas columnas que siguen en pie. Espartaco, hábil, ha decidido reutilizar estas columnas para construir muros donde pueda salvaguardar su posición. Su propósito es utilizarlas de tal forma que formen tantos círculos concéntricos como sea posible, de tal forma que los romanos tengan más difícil la entrada al templo.

Espartaco ha decidido nombrar a estos muros *Gladius*, en honor a la espada que representa la libertad por la que ellos pelean. Espartaco está interesado en saber cuántas Gladius se podrán construir dada la configuración de las columnas del templo.

## Entrada

La primera línea de entrada contiene un número $T$ denotando el número de pruebas que querrá realizar Espartaco.

Por cada caso, tendrá un número $N$ representando el número de columnas en el templo, luego, $N$ líneas denotando un par $x_i$ y $y_i$ , con hasta dos dígitos decimales, representando la posición en la columna del templo.

## Salida

Imprimir la frase "`Caso #i:`", donde $i$ es el número del caso, seguido por un espacio y el número de Gladius que Espartaco puede construir.

## Entrada de ejemplo

```
2
3
0 0
10 0
20 0
10
0 0
0 20
20 20
20 0
5 5
5 15
15 15
15 5
10 10
10 11
```

## Salida de ejemplo

```
Caso #1: 0
Caso #2: 2
```

## Límites

- $T \leq 100$
- $1 \leq N \leq 1000$
- -1000 $\leq x, y \leq$ 1000
- No habrá ningún par de columnas en el mismo punto