

Nivel Libres
Problem Set

11 de Noviembre de 2022

Problema A: Souvenirs

Timelimit: 1 s

En la UTN Santa Fe contamos con una impresora 3D, y queremos usarla para imprimir algunos souvenirs para los participantes de TecnoMate. Sin embargo, imprimir un souvenir demora 1 hora. Entonces, para intentar agilizar la producción de souvenirs, podemos usar una impresora para imprimir otra impresora 3D, y este trabajo de imprimir una nueva impresora también demora 1 hora.

Cada impresora podemos volver a usarla inmediatamente después de que termine un trabajo de impresión, para imprimir otra cosa, ya sea un nuevo souvenir u otra impresora.

¿Cuál es la mínima cantidad de horas necesarias para imprimir **N** souvenirs?

Entrada

La entrada contiene un único entero **N (1 <= N <= 10 000)**, el número de souvenirs que se necesita imprimir.

Salida

Imprima un único entero indicando la mínima cantidad de horas necesarias para imprimir **N** souvenirs.

Ejemplos

Entrada

```
1
```

Salida

```
1
```

Entrada

```
3
```

Salida

```
3
```

Entrada

```
6
```

Salida

```
4
```

Adaptado de: https://kthchallenge17.kattis.com/contests/dyf9q8/problems/3dprinter

Problema B: Propuesta de matrimonio **Timelimit: 5 s**

Pepita vuelve a su casa desde el trabajo todos los días usando las líneas del subte. Pepito está muy enamorado de ella y hoy la quiere sorprender en alguna de las estaciones de subte de camino a casa y proponerle matrimonio.

Pepita no siempre viaja a casa exactamente por el mismo camino, pero siempre optimiza su tiempo, viajando por uno de los caminos más cortos.

Pepito necesita tu ayuda para que le digas todas las estaciones en las que puede estar seguro que se va a encontrar con Pepita si él decide esperarla ahí.

Entrada

La primera línea de entrada contiene dos enteros **N** y **M (1 <= N, M <= 100 000)**, indicando respectivamente el número de estaciones de subte y el número de vías de subte que conectan dos estaciones. Luego siguen **M** líneas, donde cada una contiene tres enteros **A, B, K (0 <= A,B < N, 1 <= K <= 1 000 000 000)**, indicando que hay un subte que viaja de la estación **A** hacia la estación **B** (pero no en sentido contrario), y demora **K** segundos en hacer dicho viaje. Tenga en cuenta que puede haber más de una conexión entre un mismo par de estaciones.

La última línea de entrada contiene dos enteros **T** y **C (0 <= T, C < N)**, indicando respectivamente la estación más cercana al trabajo de pepita, y la estación más cercana a su casa.

Salida

Imprima una línea con una lista de enteros separados por espacios, indicando todas las estaciones **X** tal que todo camino mínimo desde **T** hacia **C** pasa por **X**, en orden creciente.

Ejemplos

Entrada

```
4 4
0 1 7
1 3 7
0 2 7
2 3 7
0 3
```

Salida

```
0 3
```

Entrada

```
7 8
0 1 21
```

```
1 3 21
3 4 21
4 6 21
5 6 21
3 5 21
2 3 21
0 2 21
0 6
```

Salida

```
0 3 6
```

Adaptado de:
https://kthchallenge14.kattis.com/contests/fbmwtxaqtuus/problems/intercept

Problema C: Ofertas Mundialistas **Timelimit: 1 s**

Quedan 9 días para el Mundial de Fútbol, y las empresas de venta de electrodomésticos, aprovechando esta temporada, organizaron un día de rebajas en la venta de televisores.

El problema es que se esperan multitudes, y no se quiere que los locales se llenen de personas y que los empleados no los puedan atender a todos.

Para lograr que no se llenen los locales, se decidió limitar la cantidad de clientes que pueden entrar en el local. Para hacer esto, dividen a todas las personas en la entrada del local en grupos de tamaño **N** y los procesan de la siguiente manera:

- Los **N** participantes tiran un dado y reportan el resultado $A_1, A_2, A_3, ..., A_N$.
- Para prevenir que hagan trampa, no se elige al que tiene el número más alto, sino al que tiene el mayor valor único.
- Todos los que no se seleccionan, se mueven hacia atrás.
- Si no gana nadie, se repite el proceso.

Por ejemplo, si hay 3 clientes y los resultados son 5, 5, 4. Gana el tercero, ya que aunque el primero y segundo cliente tienen un número mayor, pero que no es único.

Trabajas en el departamento de sistemas de una de estas empresas y te piden desarrollar un programa que seleccione al jugador ganador

Entrada

La primera línea de entrada contiene un entero **N (1 <= N <= 100)**, indicando la cantidad de personas en el grupo. La segunda línea tiene **N** enteros $A_1, A_2, A_3, ..., A_N$ **(1 <=** A_k **<= 6** para todo **1 <= k <= N)**, el resultado de cada tiro de dado.

Salida

Imprima el índice del cliente que tiene el mayor único resultado o **"none"** si ninguno tiene un resultado único.

Ejemplos

Entrada

```
8
1 3 1 5 6 6 4 1
```

Salida

```
4
```

Entrada

```
4
5 5 5 5
```

Salida

```
none
```

Adaptado de: https://kthchallenge15.kattis.com/contests/kth2015/problems/blackfriday

Problema D: Campus Universitario **Timelimit: 2 s**

Un grupo de estudiantes de Ingeniería Civil de la UTN está armando una propuesta para construir un nuevo gran campus universitario. Este nuevo campus se puede modelar como un tablero de **NxN** casillas, donde cada una de las casillas del tablero corresponde a una zona dentro del campus universitario, y cada una de estas zonas puede ser de alguno de los siguientes tipos:

- Una zona de dormitorios o alojamiento para estudiantes (identificadas con el número 1 en el tablero)
- Una zona de aulas o laboratorios (identificadas con el número 2)
- Una zona de comedores o almacenes (identificadas con el número 3)

El tablero entonces nos permite modelar las distancias entre las distintas zonas del campus. A cada fila del tablero se le asigna un número de 1 a **N** desde arriba hacia abajo, y a cada columna se le asigna un número de 1 a **N** de izquierda a derecha. Un estudiante parado en una casilla en la fila **F** y columna **C**, puede moverse a las 4 casillas adyacentes a la misma, si es que existen: **(F-1, C), (F+1, C), (F, C-1) y (F, C+1)**. Cualquiera de estos 4 movimientos le lleva 1 minuto a dicho estudiante. No es posible moverse fuera del tablero.

Los estudiantes encargados de esta propuesta están preocupados por las distancias de los dormitorios a los comedores (obviamente que la comida es más importante que el estudio!) y necesitan tu ayuda para evaluar distintos diseños del campus.

Dado un diseño del campus, encontrar cuál es la mayor distancia (en minutos) que un estudiante necesita viajar para llegar de una zona de dormitorios a la zona de comedores más cercana a ese dormitorio. Dicho en otras palabras, encontrar cuál es la mínima distancia **D** que nos garantiza que toda zona de dormitorios tiene al menos una zona de comedores a distancia menor o igual que **D**.

Entrada

La primera línea de entrada contiene un entero **N (2 <= N <= 1 600)**, indicando el tamaño del tablero. Luego siguen **N** líneas con **N** caracteres cada una, describiendo el tablero que modela un diseño del nuevo campus universitario. Se garantiza que el tablero solo contiene los números 1, 2 y 3, y que cada uno de estos números aparece al menos una vez en el tablero.

Salida

Imprima un único entero, indicando la mayor distancia que un estudiante necesita viajar para llegar de una zona de dormitorios a la zona de comedores más cercana a ese dormitorio.

Ejemplos

Entrada

```
2
33
21
```

Salida

```
1
```

Entrada

```
4
2123
3122
3212
3221
```

Salida

```
3
```

Adaptado de: https://kthchallenge16.kattis.com/contests/s6ksft/problems/zoning

Problema E: Carrera de naves espaciales **Timelimit: 5 s**

Es el año 3210 y tanto los viajes en el espacio como algunos mecanismos de teletransportación son algo muy normal. En esta época existe un nuevo deporte muy conocido, que se parece un poco a las carreras de autos de hoy en día.

Este nuevo deporte es una carrera de naves espaciales en línea recta, donde la pista tiene un cierto largo y cada nave corre a su propia máxima velocidad. Por cuestiones de seguridad, no se les permite a los corredores adelantar a otras naves, entonces cuando un corredor más rápido alcanza a otro más lento, el más rápido queda a la par del más lento hasta llegar al final de la pista.

Sin embargo, al final de la pista hay una máquina que teletransporta instantáneamente a las naves hacia el inicio de la pista, y entonces estas naves siguen corriendo. Cuando múltiples naves entran al teletransportador al mismo tiempo, todas aumentan su velocidad al máximo inmediatamente (y aquí entonces los corredores más rápidos adelantan a los más lentos).

Todos los corredores largan al mismo tiempo desde el inicio de la pista a su máxima velocidad. Es decir, que en el instante de la largada, los corredores más lentos no afectan a los más rápidos.

Se considera que una nave completa una vuelta cuando la misma es teletransportada desde el final de la pista, nuevamente hacia el inicio.

Dada la cantidad de vueltas que debe realizar cada corredor y la velocidad máxima de su nave medida en el tiempo que le lleva completar una vuelta, calcular en qué tiempo finalizará cada uno.

Entrada

La primera línea de entrada contiene un entero **N (1 <= N <= 5 000)**, indicando la cantidad de corredores. Luego siguen **N** líneas, donde la **j**-ésima de ellas contiene 2 enteros T_j y K_j **(1 <=** T_j **<= 1 000 000, 1 <=** K_j **<= 1 000)**, que indican respectivamente la velocidad máxima (medida en el tiempo que le lleva completar una vuelta) y la cantidad de vueltas que debe completar al **j**-ésimo corredor.

Los corredores están ordenados en orden no-creciente según su velocidad. Es decir que T_1 **<=** T_2 **<= ... <=** T_N (recuerde que menor tiempo en dar una vuelta significa mayor velocidad).

Salida

Imprima **N** líneas, tal que la **j**-ésima línea contenga un entero indicando en qué tiempo completará la carrera el **j**-ésimo corredor.

Se puede demostrar que todos los tiempos de finalización serán enteros positivos.

Ejemplos

Entrada

```
2
4 8
7 6
```

Salida

```
36
42
```

Adaptado de: https://kthchallenge16.kattis.com/contests/s6ksft/problems/racetrack

Problema F: Una Tremenda Nave

Timelimit: 1 s

Gavin Belson es un excéntrico billonario, y para no ser igual a Elon Musk, le gusta mandar cosas a Venus. Gavin tiene un problema, no puede mandar cosas a Venus cuando lo desee, los ingenieros aeroespaciales le dicen que lo óptimo es hacerlo cuando Venus y la tierra están más cercanas.

Los ingenieros han calculado que esa ventana de tiempo en la que es ideal hacer el envío a Venus, ocurre cada 26 meses. Sabemos que hubo una ventana en **abril de 2018**.

Ahora, Gavin necesita saber cuándo será la siguiente ventana más cerca, para poder calcular si su equipo de construcción podrá terminar Una Tremenda Nave que llegue a Venus.

Escribe un programa para Gavin que dado un año, determine si existe en ese año una ventana óptima de tiempo para realizar un lanzamiento.

Entrada

La única línea de input contiene un entero **N (2018 <= N <=10 000)**, el año a determinar.

Salida

Imprime "yes" si existe una ventana de tiempo óptima para realizar el lanzamiento, si no existe en ese año, imprime "no".

Ejemplos

Entrada

```
2018
```

Salida

```
yes
```

Entrada

```
2019
```

Salida

```
no
```

Entrada

```
2020
```

Salida

```
yes
```

Entrada

```
2028
```

Salida

```
no
```

Adaptado de:
https://kthchallenge18.kattis.com/contests/kthchallenge18/problems/marswindow

Problema G: La conquista de Uruguay **Timelimit: 2 s**

Es de conocimiento general que los canguros quieren conquistar Uruguay desde hace cientos de años, y el océano que los separa es lo único que los detenía para lograr su objetivo.

Pero luego de poder conseguir barcos, los canguros tienen la posibilidad de enfrentarse al pueblo uruguayo y poder conquistar ese territorio.

Toda persona y/o animal, tiene una fuerza de lucha que se puede cuantificar.

Los canguros no van a revolucionarse sin analizarlo previamente, y quieren saber si su fuerza unida es capaz de ganarle a la fuerza unida de los uruguayos.

Los canguros son muy astutos así que deben analizar si luego de la batalla quedará alguno en pie, para poder organizarse después de la misma, por lo que no solo quieren saber si es posible ganarles a los humanos, sino que también quieren saber cual es la cantidad mínima necesaria de animales para ganarles a todos los humanos.

La fuerza de un ejército de canguros es igual a la suma de las fuerzas individuales de cada uno. De forma similar se define también la fuerza de un ejército de uruguayos. Además, específicamente los canguros ganarán la batalla si su ejército no es más débil que el de los uruguayos.

Los canguros se contactan con programadores argentinos y deciden dejarlos abrir cuentas en dólares en el país oriental, si los ayudan a con los datos proporcionados decir si es posible el triunfo en el enfrentamiento y cual es la cantidad mínima de canguros necesario para lograrlo.

Entrada

En la primera línea recibirás dos enteros **N** y **M** separados por un espacio. Siendo **N** la cantidad de canguros, y **M** la cantidad de uruguayos **(1 <= N, M <= 100 000)**.

En la segunda línea habrá **N** enteros positivos separados por espacios, que representan la fuerza de cada canguro. Cada uno de estos enteros no será mayor a **10 000**.

En la tercera línea habrá **M** enteros positivos separados por espacios, que representan la fuerza de cada habitante de Uruguay. Cada uno de estos enteros no será mayor a **10 000**.

Salida

En la misma línea deberás retornar dos enteros **P** y **Q.**

- **P** tiene que ser 0 o 1, donde 0 significa que no se le puede ganar al pueblo uruguayo. Y 1 que es posible el triunfo de los canguros.
- **Q** es un entero que representa la cantidad mínima de animales necesarios para ganarles a los humanos. Si no es posible la victoria animal, **Q = 0**.

Ejemplos

Entrada

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

Salida

```
1 2
```

Entrada

```
4 2
4 4 4 4
10 21
```

Salida

```
0 0
```

Entrada

```
4 2
4 4 4 4
10 6
```

Salida

```
1 4
```

Problema original por Matías Ramos

Problema H: Cartas de colores **Timelimit: 10 s**

Nano estaba aburrido y se le ocurrió un nuevo juego con su mazo de cartas de colores. El primer paso del juego es mezclar las cartas y ponerlas arriba de la mesa una al lado de la otra formando una fila, y de tal forma que sea posible ver el color que tiene cada carta. Luego el juego continúa en rondas.

En cada ronda, el jugador debe elegir exactamente dos cartas que tengan el mismo color. Si no hay dos cartas del mismo color, el jugador no suma más puntos y el juego finaliza inmediatamente. En caso contrario, el jugador suma 1 punto y debe eliminar de la fila de cartas las dos cartas elegidas, y también debe eliminar todas las cartas que no se encuentren entre medio de ellas dos en la fila. Es decir, si en una ronda el jugador elige las cartas de igual color que se encuentra en las posiciones **i** y **j (i < j)**, entonces en la siguiente ronda solo quedarán las cartas en las posiciones **i+1, i+2, ..., j-1**.

Dada una distribución inicial de la fila de cartas, ¿Cuál es el máximo puntaje que se puede obtener?

Entrada

La entrada contiene múltiples casos de prueba.

Cada caso de prueba comienza con una línea que contiene un entero **N (1 <= N <= 5000)**, indicando la cantidad de cartas que hay inicialmente en la fila. La segunda línea contiene **N** enteros positivos, donde el **k**-ésimo de ellos C_k **(1 <= C_k <= 10 000)** indica el color de la **k**-ésima carta.

Salida

Por cada caso de prueba, imprimir un único entero, indicando la máxima cantidad de puntos que puede obtener un jugador si comienza a jugar con la fila de cartas provista en la entrada.

Ejemplos

Entrada

```
3
5 5 5
10
91 3 17 8 25 144 2 11 20 38
12
9 5 3 5 6 2 9 5 1 4 1 3
7
1 8 2 8 1 7 2
4
3 5 3 8
11
1 2 4 8 16 32 16 8 4 2 1
6
```

```
1 2 3 1 2 3
```

Salida

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

Adaptado de: https://dmoj.ca/problem/ctuopen2017g

Problema I: Mesas **Timelimit: 5 s**

Juancito pronto abrirá un nuevo bar y necesita tu ayuda a la hora de definir cuántas mesas comprar. Juancito ya tiene definidos muchos lugares en su bar donde podría colocar una mesa, sin embargo, para garantizar la comodidad de sus clientes, él quiere asegurarse de que cualquier par de mesas estén a una distancia de al menos **1.3** metros.

Ayuda a Juancito a saber cuántas mesas comprar, diciéndole cuál es la máxima cantidad de mesas que va a poder colocar, cumpliendo con la condición mencionada previamente.

Entrada

La entrada contiene múltiples casos de prueba.

Cada caso de prueba comienza con una línea que contiene un entero **N (1 <= N <= 2000)**, indicando la cantidad de posibles lugares dónde se podría colocar una mesa. Luego siguen **N** líneas, donde cada línea contiene dos enteros **X** e **Y (0 <= X, Y <= 1 000 000 000)**, indicando las coordenadas cartesianas en metros en las que se puede poner cada mesa.

Restricción extra: como Juancito quiere poner muchas mesas, los posibles lugares para las mesas no están dispersos, si no que están cerca entre sí. Más formalmente, si decimos que dos puntos están conectados cuando su distancia es 1, se garantiza que los puntos de la entrada forman un único grupo conectado, es decir que se puede viajar entre cualquier par de puntos usando estos saltos de distancia 1.

Salida

Por cada caso de prueba, imprimir una línea con un único entero, indicando la máxima cantidad de mesas que Juancito puede colocar en su bar, garantizando que no habrá dos a una distancia menor que **1.3** metros.

Ejemplos

Entrada

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

Salida

```
2
4
```

Adaptado de: https://dmoj.ca/problem/ctuopen2017j

Problema J: Escape Room **Timelimit: 5 s**

Tomas y Fernando trabajan en una sala de escape, donde los jugadores van avanzando en un pasillo de habitaciones y cada habitación tiene escondida una llave para avanzar a la siguiente habitación.

Pero ocurrió un error y las llaves se encuentran mezcladas, en habitaciones incorrectas.

Para poder acomodarlas y evitar errores, Tomás y Fer van a trabajar en turnos, donde en un turno Tomás elige la llave de una habitación X y la mueva hacia otra habitación Y, y al mismo tiempo Fer toma la llave de la habitación Y y la mueve hacia la habitación X. Básicamente en cada turno intercambian las llaves entre un par de habitaciones. Luego de un número de intercambios lograrán ordenar las llaves.

Hacer estos intercambios es mucho trabajo, por lo que Tomás y Fer quieren saber cuál es el menor número de intercambios necesarios para lograr ordenar las llaves, dejando cada una en la habitación que debe ir.

Entrada

Puede haber más de un caso de prueba por entrada.

Cada caso de prueba consiste en dos líneas.

La primera línea tendrá un entero **N (1 <= N <= 200 000)** especificando el número de habitaciones. Las habitaciones y sus respectivas llaves se identifican con los números **1,2,3,...,N**. La llave 1 va en la habitación 1, la llave 2 en la habitación 2 y así respectivamente.

La segunda línea tiene **N** enteros separados por espacios **(a1, a2, ..., aN)** donde **a1** indica que llave se encuentra inicialmente en la habitación 1, **a2** indica que llave se encuentra en la habitación 2, y así sucesivamente.

Salida

Para cada caso de prueba imprime un entero indicando el mínimo número de turnos necesarios para ordenar correctamente las llaves y colocarlas en sus respectivas habitaciones.

Ejemplos

Entrada

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

```
4 3 1 2
5
5 4 1 2 3
10
9 8 10 5 3 2 4 7 6 1
6
5 6 3 4 1 2
```

Salida

```
0
2
3
3
9
2
```

Adaptado de: https://dmoj.ca/problem/ctuopen2017i