

Enunciados de Problemas

Nivel Paradigmas

1 de noviembre de 2024

1. La afición de Hugo

Tiempo límite: 1 segundo
Memoria límite: 256 megabytes

Código SPOJ: TM24LADH

Hugo es un aficionado a las funciones HASH. Su última creación es un algoritmo que, a partir de una entrada, obtiene un número entero (positivo o negativo).

Inicialmente, la entrada es dividida en palabras. Cada palabra es una cadena de caracteres separada de otra palabra por un espacio. De cada palabra, w, se obtiene un número, $n(w)$, que se calcula sumando el valor ASCII de todos los caracteres que la componen. Finalmente, el HASH de la entrada se obtiene sumando los $n(w)$ de las palabras de longitud par y los $(-1) \cdot n(w)$ de las palabras de longitud impar.

Caracteres ASCII

		48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n		
47	/	63	?	79	O	95	_	111	o		

Tabla ASCII

Entrada

Una única línea compuesta de palabras separadas por un espacio (sin espacio al final de la línea). La cantidad de palabras de la entrada estará en el rango de **0 a 20.000**, la longitud de cada palabra estará en el rango de **1 a 20** caracteres, y el valor ASCII de cada uno de los caracteres de una palabra estará en el rango de **33 a 125**.

Salida

La salida es una única línea con el número correspondiente al HASH de la entrada.

Ejemplos

entrada estándar	salida estándar
Tecnomate	-928
el ASCII de # es 35.	80

2. Té extraño

Tiempo límite: 1 segundo
Memoria límite: 256 megabytes

Código SPOJ: TM24TE

La prestigiosa asociación Hot Tea and Machine Learning (mejor conocida como HTML), es la entidad encargada de regular y controlar, con especial dedicación y tenacidad, todo té preparado por programadores alrededor del mundo. Recientemente, esta asociación ha decidido estandarizar las preparaciones de la infusión a nivel global.

Según el estricto ente, para que una bebida sea considerada té, las hojas de la planta *Camellia Sinensis* deben ser tostadas a temperaturas específicas. Dependiendo de este factor, una infusión de hojas con agua caliente puede clasificarse como té blanco, verde, amarillo, oolong, negro o, directamente, no ser considerada té.

	Blanco	Verde	Amarillo	Oolong	Negro
Temperatura (C°)	65-75	76-80	81-85	86-95	96-110

Tipos de té según la temperatura a la que son tostadas las hojas.

Adicionalmente, todo té puede ser enriquecido con aromatizantes. Los únicos ingredientes permitidos por HTML para los aromatizantes son bergamota, lavanda, canela, anís y jengibre. Algunas combinaciones de estos ingredientes han sido oficialmente nombradas por la asociación, que exige su correcta identificación. Según las combinaciones de ingredientes, un aromatizante puede ser Gingergrey, Karma, Police, Golang o no ser una combinación estandarizada.

	Bergamota	Lavanda	Canela	Anís	Jengibre
Gingergrey	✓	✓	✓		✓
Karma	✓	✓	✓	✓	✓
Police		✓		✓	
Golang	✓		✓	✓	

Tipos de aromatizantes estandarizados según los ingredientes que lo componen.

Se le solicita que escriba un programa que permita a HTML determinar la nomenclatura correcta de una infusión.

Entrada

La entrada consta de múltiples líneas.

La primera línea contiene un entero T ($40 \leq T \leq 130$), que representa la temperatura a la que las hojas son tostadas.

Las siguientes cinco líneas contienen una cadena de caracteres mayúsculos s_i ($s_i \in$ {"SI", "NO"}), indicando si el aromatizante utilizado contiene o no al ingrediente i-ésimo y donde $i = 1$ corresponde a bergamota, $i = 2$ a lavanda, $i = 3$ a canela, $i = 4$ a anís y $i = 5$ a jengibre.

Salida

La salida consta de una línea. Si la infusión es un té, la línea contiene la frase "Te", seguida del tipo de tostado y, en caso de ser necesario, el tipo de aromatizante estándar utilizado, separadas por espacios. Si la bebida no es considerada té, la asociación le exige que el programa imprima únicamente la frase "Te extrano".

Ejemplos

entrada estándar	salida estándar
40 SI SI SI SI SI	Te extrano
105 SI SI SI NO SI	Te Negro Gingergrey
75 NO NO SI NO NO	Te Blanco

3. Mensajería Privada

Tiempo límite:	1 segundo
Memoria límite:	256 megabytes

Código SPOJ: TM24MP

En una época de comunicaciones constantes, un grupo de amigos valora más que nunca la **privacidad** de sus mensajes. Ante la creciente presencia de espías digitales y la posibilidad de que alguien pueda interceptar sus charlas, deciden crear su propio sistema de **mensajería privada**. Para ellos, no basta con simplemente esconder el mensaje: deben encriptarlo de forma que, aun si alguien más lo leyera, no tenga manera de entenderlo.

Inspirados por técnicas antiguas y algunos métodos modernos de encriptación, diseñan un sistema basado en rotaciones y sustituciones especiales. Cada letra del mensaje se desplaza un cierto número de posiciones en el alfabeto, y algunas letras se reemplazan por símbolos secretos. Solo aquellos que conocen el esquema completo pueden descifrar los mensajes.

Ahora, el grupo de amigos te ha encargado la tarea de implementar este sistema de encriptación para facilitar su uso. Su objetivo es comunicarse sin temor a ser entendidos por nadie más. Con este sistema, esperan mantener sus conversaciones verdaderamente privadas en un mundo en el que la privacidad es cada vez más escasa.

¿Puedes ayudarlos a construir esta capa extra de seguridad en su mensajería?

Enunciado:

1. **Rotación de Letras**: Cada letra del mensaje es desplazada hacia adelante en el alfabeto un número específico de posiciones, indicado por un parámetro de rotación. Esta rotación es cíclica, es decir, después de z vuelve a empezar desde a.
2. **Sustitución Especial**: Ciertas letras se reemplazan por símbolos para dificultar aún más el entendimiento del mensaje.
3. **Eliminación de espacios**: para que el envío sea más seguro se reemplazarán los espacios por \

Especificaciones

- Las sustituciones especiales son las siguientes: a/@, e/3, i/!, o/0 (cero), s/$, t/7, p/?, b/8. Considerar las mismas sustituciones para las mayúsculas.
- Las letras mayúsculas, deberán mantenerse en mayúscula una vez transformadas.
- Para simplificar el trabajo y considerando que esto es una versión preliminar se considera que el largo máximo del mensaje será de 500 caracteres.
- La entrada solo estará compuesta por letras, sin signos de puntuación ni caracteres especiales ni letras acentuadas ni $\tilde{\text{N}}$.

Entrada

Primera línea: mensaje, una cadena de texto.

Segunda línea: rotación, un entero positivo entre 0 y 25 que indica el número de posiciones de rotación.

Salida

La salida es el mensaje encriptado después de aplicar la rotación y la sustitución especial.

Ejemplos

entrada estándar	salida estándar
Hola 3	Kr0d
mensaje Secreto 5	rj$xf0j\Xjhwjy7

4. De Minería

Tiempo límite: 1 segundo
Memoria límite: 256 megabytes

Código SPOJ: TM24DM

Lucas es un joven explorador que vive en un pueblo minero. Un día, mientras acompañaba a su padre en una expedición, se adentraron en una mina subterránea en busca de minerales valiosos. Lucas, fascinado por la estructura de la mina, comenzó a explorar y notó que las vetas de minerales formaban patrones únicos en las paredes. Intrigado, decidió contar no solo la cantidad de vetas de minerales que encontraba, sino también cuántas de éstas formaban agrupaciones y cuál era la más grande. Su padre, al ver el entusiasmo de Lucas, te ha pedido ayuda para diseñar un algoritmo que pueda analizar la mina y responder a sus preguntas.

Enunciado:

Imagina la mina como una cuadrícula de N filas y M columnas, donde cada celda puede estar vacía (representada por .) o contener una veta de mineral (representada por *). Dos vetas de minerales se consideran parte del mismo grupo si están conectadas en direcciones horizontales, verticales o diagonales.

Tu tarea es determinar dos cosas:

- La cantidad total de grupos de vetas de minerales en la mina.
- El tamaño del grupo de minerales más grande que encuentres.

Entrada

La primera línea contiene dos enteros, N y M ($2 \leq N, M \leq 28$), que representan las dimensiones de la mina. Las siguientes N líneas contienen una cadena de M caracteres cada una, donde ***** representa una veta de mineral y **.** una celda vacía.

Salida

La salida debe contener una línea con dos enteros:

- Un entero que indica el número total de grupos de vetas de minerales.
- Un entero que representa el tamaño del grupo de minerales más grande encontrado.

Ejemplos

entrada estándar	salida estándar
4 5 .*... .**.. ..*..	1 4
3 3 *** .*. ***	1 7
5 5 *.*.* .*.*. ..... .*.*. *.*..	2 5
3 3 *.* ... *.*	4 1

Nota

Las vetas de minerales se consideran adyacentes si están conectadas en cualquiera de las ocho direcciones posibles: vertical, horizontal o diagonalmente.

Un grupo de vetas de minerales es un conjunto de vetas conectadas de esta forma.

Puedes asumir que la entrada estará correctamente formateada, sin caracteres adicionales ni errores.

5. Cuenta Impagable

Autor:	Lucio Rivera
Tiempo límite:	5 segundos
Memoria límite:	256 megabytes

Código SPOJ: TM24CI

El infame BurpChein, CEO de una compañía de ciberseguridad, ha cometido un grave error. En un acto de total imprudencia, BurpChein adquirió una licencia de software para su empresa en nombre de la universidad, que ahora debe pagar el costo de este imperdonable y, no precisamente barato, descuido.

Desafortunadamente para BurpChein, el departamento de sistemas, hará valer los 400 dólares que debió pagar por la licencia y ha decidido someter al vil estudiante a un juego denominado *secuencia 1 a 9*. El departamento determinó que, si BurpChein logra vencer a todos los profesores de sistemas en este juego, se le perdonará su error y podrá continuar estudiando en la universidad.

En una partida del juego, saldrá victorioso el jugador que, a partir de un vector de números enteros positivos indexados en cero, logre encontrar la secuencia de operaciones de menor costo total que transforme el vector dado en el vector [1, 2 , ... , 8, 9]. Las operaciones permitidas son las siguientes:

1. *Reemplazar*, el número v_i en la posición i por $v_i \ mod \ 10^{\phi(v_i)-k}$. Donde la función $\phi(x)$ es la cantidad de dígitos de un número entero positivo x y $0 \leq k < \phi(v_i)$. Costo: k.
2. *Eliminar*, el último dígito del número v_i en la posición i siempre y cuando $\phi(v_i) > 1$. Costo: 1.
3. *Eliminar*, el número v_i en la posición i. Costo i.
4. *Intercambiar*, el número v_i en la posición i con el número v_j en la posición j. Costo: 0.
5. *Copiar*, el número v_i en la posición i siempre y cuando v_i no haya sido alterado anteriormente por las operaciones 1 o 2. La copia es insertada al final del vector. Costo: i.

BurpChein sabe que, por algún motivo que aún desconoce, muchos profesores son imbatibles en este juego. Aterrorizado por la posibilidad de ser expulsado de la universidad, le solicita a usted que escriba un programa que le permita encontrar el costo total mínimo posible para asegurarse siempre salir victorioso en el desafío. BurpChein le promete un cargo en su compañía si logra ayudarlo, aunque le advertimos que quizás este sea un motivo más para no ayudarlo.

Entrada

La entrada consta de múltiples líneas.

La primera línea contiene un entero n ($2 \leq n \leq 10^5$), que representa el tamaño del vector inicial.

La segunda línea contiene n números enteros v_i ($1 \leq v_i \leq 10^5$), representando el número en la posición i del vector.

Se garantiza que todos los dígitos de 1 a 9 aparecen como mínimo una vez en algún número. Se puede demostrar que siempre existe una secuencia de operaciones ganadora.

Salida

La salida consta de una única línea, el costo mínimo posible para ganar la partida.

Ejemplos

entrada estándar	salida estándar
9 1 2 3 4 5 6 7 8 9	0
4 12 334 675 8889	18

6. Torneito

Autor:	Fernando Sebastián Lomazzi y Lucio Rivera
Tiempo límite:	2 segundos
Memoria límite:	256 megabytes

Código SPOJ: TM24TO

El increíble estadio de Lusail en Qatar tiene el privilegio de alojar a la prestigiosa copa mundial de programación competitiva, donde reconocidos programadores como Julian Arrayvarez, Rodrigo DeBug y Lionel Prezi encantan al mundo con sus increíbles habilidades para la codificación. Como era de esperarse, un espectáculo de tal magnitud ha opacado por completo a la copa mundial de fútbol, que ahora debe llevar a cabo sus partidos prácticamente sin público y en plazas circundantes, pues la gran mayoría de los espectadores prefieren ver el mundial de programación competitiva.

En esta competencia, los enfrentamientos son de eliminación directa, es decir que el ganador de un enfrentamiento permanece en la contienda y el perdedor es directamente eliminado. En cada día de competencia, se celebra una *ronda de enfrentamientos*, donde cada equipo restante es emparejado aleatoriamente con otro equipo. Para definir qué equipos se enfrentarán en cada partido y en qué horario lo hará cada uno a lo largo de la ronda, los organizadores llevan a cabo un espectacular sorteo; sin embargo, por motivos históricos, la localía de los enfrentamientos se define justo antes de empezar un partido arrojando una moneda de oro. En este formato, resulta ganador aquel equipo que permanezca en contienda una vez que el resto de los equipos hayan sido eliminados a lo largo de todas las rondas de enfrentamientos necesarias.

Recientemente, el futbolista Kylian Empappé, enajenado por cómo el mundial de programación acapara todo el protagonismo, ha destrozado los servidores de la organización. Si bien el programa escrito para determinar el ganador sobrevivió al ataque, las listas que almacenaban el orden cronológico de los enfrentamientos se vieron completamente desordenadas.

La organización acude a usted, sorprendida al enterarse de que no se encuentra compitiendo en el mundial, para solicitarle que escriba un programa que le permita determinar el ganador de la prestigiosa competencia dados los enfrentamientos llevados a cabo sin ningún orden particular.

Entrada

La entrada consta de múltiples líneas.

La primera línea contiene un entero n ($2 \leq n \leq 2^{10}$), que es una potencia de 2 y representa la cantidad de competidores.

Las siguientes $n-1$ lineas representan los enfrentamientos dados y contienen una cadena de caracteres a_i ($1 \leq |a_i| \leq 20$), un entero p_i^a ($0 \leq p_i^a \leq 10$), un carácter "-", un entero p_i^b, ($0 \leq p_i^b \leq 10$) y una cadena de caracteres b_i ($1 \leq |b_i| \leq 20$), representando el nombre del competidor local, su puntaje, un separador, el puntaje del competidor visitante y el nombre del competidor visitante en el enfrentamiento i respectivamente, separados por espacios.

El nombre de los competidores está compuesto por letras mayúsculas y minúsculas del alfabeto inglés; el primer carácter es una mayúscula, mientras que el resto son minúsculas.

Se garantiza que no hay enfrentamientos que terminen en empate.

Salida

La salida consta de una línea, el ganador de la competencia.

Ejemplos

entrada estándar	salida estándar
4 Espana 1 - 0 Holanda Francia 3 - 2 Inglaterra Francia 1 - 2 Espana	Espana
8 Francia 2 - 4 Argentina Croacia 4 - 2 Brasil Argentina 3 - 0 Croacia Inglaterra 1 - 2 Francia Francia 2 - 0 Marruecos Holanda 3 - 4 Argentina Marruecos 1 - 0 Portugal	Argentina

Nota

Nótese como, en el segundo caso de prueba, Francia obtiene el segundo lugar. Cualquier correspondencia con otra competencia es pura coincidencia

7. Fuga de memoria

Autor:	Lucio Rivera
Tiempo límite:	1 segundo
Memoria límite:	256 megabytes

Código SPOJ: TM24FM

Fernando es un programador muy famoso en su ciudad. La velocidad a la que este prodigioso joven programa es motivo de envidia para todos sus colegas, quienes tiemblan de miedo cada vez que lo ven acercarse —con teclado RGB y mouse óptico en mano— a las competencias de programación que se organizan en su ciudad. Si bien Fernando es un programador extremadamente veloz, la legibilidad de su código y la prevención de bugs no son aspectos que se le den muy bien.

En su tiempo libre, Fernando suele inspeccionar con especial orgullo el código que escribe en su trabajo. En una de estas inspecciones, se percató de que uno de sus programas tenía un peligroso bug: Fernando quería acceder k veces a la posición i-ésima de un vector indexado en cero; pero, incorrectamente, terminaba evaluando la expresión $v[_1\, v[_2 \cdots v[_{k-1}\, v[_k i]]]]$.

Mientras Fernando estaba haciendo la edición anterior de TecnoMate, descubrió un teorema que revolucionó el mundo del álgebra lineal y que lo llevo a convertirse en multimillonario. Por este motivo, ahora se encuentra vacacionando en *Miami*, lo que le impide corregir su error. Fernando le solicita a usted, a cambio de varios millones de dólares, que escriba un programa idéntico al defectuoso que codificó; pero que, dado un vector de entrada, advierta a sus colegas cuando se produzca una fuga de memoria.

Entrada

La entrada consta de múltiples líneas.

La primera línea contiene un entero n ($1 \leq n \leq 1000$), que representa el tamaño del vector.

La siguiente línea contiene n números enteros v_i ($0 \leq v_i \leq 10^5$), representando el elemento en la posición i del vector.

La tercera línea contiene un entero q ($1 \leq q \leq 10$), representando la cantidad de consultas a responder. Cada consulta consiste en dos enteros k e i ($1 \leq k \leq 100$, $0 \leq i \leq 10^5$), la cantidad de veces y el índice inicial en el que el vector es evaluado, respectivamente.

Salida

Para cada consulta, deberá imprimir el resultado de la expresión descrita, si para su evaluación no se requiere acceder a una posición inválida del vector. Caso contrario deberá impirmir la frase *"Mi primera fuga de memoria"*.

Ejemplos

entrada estándar	salida estándar
1 0 1 1 0	0
6 2 0 4 3 5 7 4 1 5 3 1 2 3 5 0	7 4 3 Mi primera fuga de memoria

Nota

En la segunda consulta del caso de prueba, se debe evaluar la expresión $a[\,a[\,a[\,1]]]$. En la primera indexación, se obtiene $a[\,a[\,0]]$; en la segunda, $a[\,2]$ y finalmente en la tercera, se accede al número 4 ubicado en la tercera posición.

NOTA PARA Nivel Algoritmos y Libres Si está utilizando el compilador integrado en *ZinjaI*, es posible que no pueda compilar su programa con $n = 10^6$. Si este es el caso, realice pruebas en su entorno con un valor menor, como $n = 10^5$, pero envíe la solución al juez con el valor correcto.

8. Batalla Continuada

Autor:	Lucio Rivera
Tiempo límite:	5 segundos
Memoria límite:	256 megabytes

Código SPOJ: TM24BC

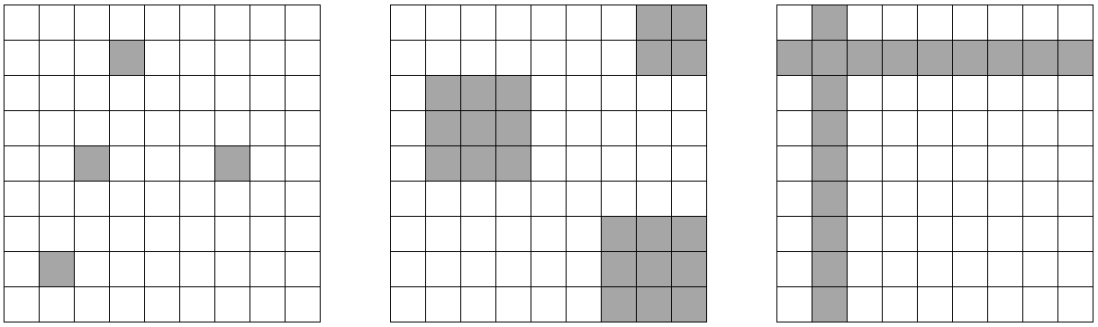
La épica batalla apocalíptica entre patos y burbujas continúa en esta edición de la TecnoMate. Desafortunadamente, en la edición anterior de la competencia, ningún humano pudo ayudar al patógrafo Pating Turing a desencriptar el Código Misterio. Esta situación obligó al reconocido experto a pedirle ayuda al novedoso chatbot DuckGPT, desarrollado por sus *compatotas*, para resolver el problema.

En una maniobra de increíble creatividad, las audaces burbujas lograron acceder al código fuente de DuckGPT y lo alteraron para que los patos no puedan usar la herramienta a su favor. Afortunadamente, el chatbot no fue eliminado en su totalidad y aún conserva algunos recuerdos de su pasado.

Como resultado de una alucinación, DuckGPT accede a ayudar a los patos, pero solo si las aves logran vencerlo en su juego favorito. Este juego comienza con el chatbot creando una matriz cuadrada, donde el elemento en la fila i y la columna j puede ser una mina explosiva mortal o, en su defecto, agua completamente inofensiva. La idea del juego, es que los oponentes no sepan dónde se encuentran las minas.

El objetivo de los patos es destruir, sin conocer su ubicación exacta, todas las minas de la matriz. DuckGPT permite a sus rivales realizar tres tipos de ataques:

- *Ataque 1*: Destruye una única mina en la celda ubicada en la fila i y la columna j.
- *Ataque 2*: Destruye todas las minas dentro de un cuadrado de 3x3 cuyo centro está ubicado en la fila i y la columna j.
- *Ataque 3*: Destruye todas las minas en la fila i y en la columna j.



Los tipos de ataques descritos. En la primera imagen, se visualizan múltiples ataques de tipo 1; en la segunda imagen, se visualizan múltiples ataques de tipo 2; y en tercera imagen se visualiza un único ataque de tipo 3.

Los patos le han concedido una última oportunidad de ser liberado; para lograrlo, usted deberá escribir un programa que, dada una matriz inicial y los ataques realizados, determine la cantidad de minas que quedan sin ser destruidas. Con su invención, los patos pretenden practicar para su enfrentamiento con el chatbot.

Entrada

La entrada consta de múltiples líneas.

La primera línea contiene un entero n ($1 \leq n \leq 100$), que representa el tamaño de la matriz creada por el chatbot.

Las siguientes n líneas contienen n caracteres mayúsculos c_{ij} ($c_{ij} \in$ {"M","A"}), representando el tipo de elemento ubicado en la intersección entre la fila i y la columna j, donde el carácter "M" indica que el elemento es una mina y el carácter "A" indica que es agua.

La siguiente línea contiene un entero q ($1 \leq q \leq 100$), que representa la cantidad de ataques realizados.

Cada ataque consiste en tres enteros a, i y j ($1 \leq a \leq 3$, $1 \leq i, j \leq n$), indicando el tipo de ataque y su ubicación.

Salida

La salida consta de una línea, la cantidad de minas que permanecen sin ser destruidas después de los ataques.

Ejemplos

entrada estándar	salida estándar
6 M M M M A M M M M A A M M M M A A M A A A M A M A A A A A M M M M M M M 3 1 1 4 2 2 2 3 6 6	1
9 A M A A A A A M M M M M M M M M M M A M M M A A A A A A M M M A A A A A A M M M A A M A A A M A A A A A A A A M A A A A M M M A M A A A A M M M A M A A A A M M M 7 1 8 2 1 5 3 1 2 4 2 4 3 2 1 9 2 8 8 3 2 2	1

Nota

NOTA PARA Nivel Algoritmos y Libres

Si está utilizando el compilador integrado en *ZinjaI*, es posible que no pueda compilar su programa con $n = 1000$. Si este es el caso, realice pruebas en su entorno con valores menores, como $n = 100$, pero envíe la solución al juez con el valor correcto.

9. Seguridad en el Casino

Tiempo límite: 2 segundos
Memoria límite: 512 megabytes

Código SPOJ: TM24SC

Este problema no se puede resolver eficientemente con el lenguaje Smalltalk.

El Casino de Santa Fe está teniendo muchos problemas de robos. Y aunque existen cámaras de seguridad en el mismo, los ladrones suelen escaparse sin mucho problema. Cansado de este problema, Daniel (el dueño del casino) decide contratar muchos guardias de seguridad. Pero no los distribuye de manera muy inteligente a través de la superficie del casino. Afortunadamente, la ubicación de los guardias se puede describir con coordenadas enteras en un plano 2D. Como los guardias de seguridad no están distribuidos de manera uniforme, es difícil para el supervisor de seguridad identificar que guardia está más cerca de un incidente. Y es aún más complicado teniendo en cuenta que el espacio en el casino está restringido por las máquinas y juegos. Estás limitaciones hacen que un guardia se mueva de una ubicación a otra de a pasos. En cada paso, un guardia puede cambiar cada una de sus coordenadas por *1, 0 o -1*. La distancia entre dos ubicaciones es igual a la cantidad mínima de pasos que tiene que hacer un guardia de un lugar a otro.

Teniendo la ubicación de los guardias y un conjunto de ubicaciones de incidentes, computar para cada incidente la menor distancia a un guardia. Esto ayudará a mejorar la seguridad en el casino.

Entrada

La primera línea de la entrada contiene dos enteros N y Q ($1 \leq N, Q \leq 3 \cdot 10^5$), el número de guardias y el número de incidentes. Luego le siguen N líneas. Cada una de esas líneas contiene dos enteros X e Y ($0 \leq X, Y \leq 5000$), que describen la ubicación de un guardia. Siguiente, Q líneas. Cada una de esas líneas contiene dos enteros A y B ($0 \leq A, B \leq 5000$), que describen la ubicación de un incidente de seguridad.

Salida

Para cada uno de los Q incidentes mostrar en la salida una línea conteniendo la distancia más corta a un guardia, medido en pasos.

Ejemplos

entrada estándar	salida estándar
2 3 0 1 4 0 5 0 4 3 1 2	1 3 1
2 4 0 0 3 3 1 1 0 3 1 2 3 3	1 3 2 0

Nota

Utilizar fast input/output