Problem A. Salvando el cine

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Aquellos que no saben de películas suelen confundir las sagas de Star Wars y Star Trek, lo cuál es ofensivo porque Star Wars es ampliamente superior. Esomer ya está harto de estas ofensas, por lo que quiere hacer un programa que, dado un personaje, responda si pertenece a la saga de Star Wars o a la de Star Trek. Por desgracia, él no sabe programar del todo bien, así que necesita tu ayuda.

Input

La entrada consistirá de una sola palabra s, que será una de las siguientes tres opciones (sin las comillas): "Yoda", "Spock" o "Frodo".

Output

En el caso de que s sea "Yoda", debes imprimir: "Pertenece a Star Wars.".

En el caso de que s sea "Spock", debes imprimir: "Pertenece a Star Trek.".

En el caso de que s sea "Frodo", debes imprimir: "No pertenece ni a Star Wars ni a Star Trek.".

Todas ellas sin las comillas.

Examples

standard input	standard output
Spock	Pertenece a Star Trek.
Yoda	Pertenece a Star Wars.
Frodo	No pertenece ni a Star Wars ni a Star Trek.

Note

—

Idea del problema: Esomer

Preparación del problema: Esomer

Ocurrencias: Novato 1, Avanzado 1

Problem B. Operación

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

José acaba de aprender las operaciones aritméticas básicas (suma, resta, multiplicación y división). Como se le da bastante bien, su amigo Óscar, que está jugando a un videojuego nuevo que ha comprado, pide ayuda a José con los deberes que le han mandado. El profesor le da dos números enteros positivos a y b y le pide que al aplicar una operación de las 4 mencionadas anteriormente obtenga el mayor resultado posible. Es decir, le pide que encuentre el valor máximo de $a?b$, donde ? puede ser suma, resta, multiplicación o división.

Input

Se te dan dos números enteros a y b. ($1 \leq a, b \leq 100$).

Output

Tienes que imprimir un número, el valor máximo tras aplicar una operación a a y b.

Examples

standard input	standard output
6 3	18
8 1	9

Note

—

Idea del problema: Hectorungo18

Preparación del problema: Hectorungo18

Ocurrencias: Novato 2

Problem C. Máximo beneficio

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Rafa y Jan son como los porros, que buenos que están pero que malos que son.

Hay un banco con una pila de n cheques, cada uno con un valor específico en dólares.

El banco tiene una política única: permite a sus clientes canjear una cantidad limitada de cheques por día.

Tu tarea es ayudar a los clientes del banco a determinar el valor total máximo que pueden canjear en un **solo** día.

Input

La primera línea contiene dos números enteros n ($1 \leq n \leq 10^4$) y k ($1 \leq k \leq n$), que representan el número de cheques y el número máximo de cheques que permite canjear por día el banco, respectivamente.

La segunda línea contiene n enteros $c_1, c_2, ..., c_n$ ($1 \leq c_i \leq 10^4$), que representan los valores de cada cheque.

Output

Escriba un único número entero, el valor total **máximo** que se puede canjear en un solo día eligiendo la combinación óptima de k cheques.

Example

standard input	standard output
3 2 11 5 10	21

Note

La estrategia óptima es tomar el primer y tercer cheque para obtener una suma de 21 dólares.

—

Problem idea: danx

Problem preparation: danx

Ocurrences: Novice 3

Problem D. jbum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	128 megabytes

Javier, el culturista natty que tira 1073741824 kg en press de banca, está entrenando press de banca.

Para poder llegar al peso necesario en el levantamiento, Javier tendrá que utilizar el *duplicator*. Si introduce x discos en el *duplicator*, pasado 1 minuto tendrá el doble. Para volver a usar la máquina tiene que sacar todos los discos y volver a introducir los que quiera. Como va hasta arriba de creatina y preentreno no tarda nada en esta acción. Inicialmente solo tiene un disco que le ha regalado su amigo Bumstead. También es importante saber que la barra que utiliza para el levantamiento es mágica, por lo que no pesa nada.

Javier necesita n discos para realizar su ejercicio. Debes ayudarle a saber en cuántos minutos como mínimo puede llegar a tener el peso deseado. Si el mínimo de minutos es m, también tienes que decirle cuantos discos tiene que meter en el i-ésimo minuto en el *duplicator*, para poder llegar en el mínimo de minutos posibles.

Si hay varias formas de meter discos en el *duplicator* para llegar al peso n en m minutos, debes imprimir la que sea lexicográficamente menor.

Una secuencia de números a es lexicográficamente menor a otra b de la misma longitud, si se cumple que existe un índice i tal que $a_i < b_i$ y para todo j tal que $1 \leq j < i$ se cumple que $a_j = b_j$.

Input

La única línea contiene un entero n ($2 \leq n \leq 10^9$), el peso que quiere levantar Javier.

Output

La primera línea de la salida debe contener un entero m, el mínimo número de minutos que necesita Javier para poder hacer su serie con el peso deseado.

La segunda línea debe contener m enteros, $x_1, x_2, \ldots, x_m$, siendo x_i el número de discos que mete Javier en el *duplicator* la i-ésima vez. Recuerda que si hay varias formas posibles de llegar al peso en m segundos, debes imprimir la secuencia que sea menor lexicográficamente.

Examples

standard input	standard output
1024	10 1 2 4 8 16 32 64 128 256 512
33	6 1 1 2 4 8 16

Note

—

Idea del problema: Hectorungo18

Preparación del problema: Hectorungo18

Ocurrencias: Novato 4

Problem E. Buscando palindromios

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Incluso el palíndromo más pequeño puede cambiar el curso del futuro

— Galadriel, o tal vez Javier

Un palíndromo es una cadena de caracteres que se lee igual de izquierda a derecha que de derecha a izquierda. Javier, siendo el chico despistado que es, ha estado pensando que "palindromio" es la forma correcta de escribirlo ¡desde hace casi un año! Para hacerle un favor, creemos una nueva definición antes de que se dé cuenta de su error.

Un "palindromio" es una cadena de dígitos del 0 al 9 que está exactamente a una posición de ser un palíndromo. 101 y 22 son palíndromos, mientras que 100 y 20 son "palindromios". Insatisfecho con esto, Javier ahora quiere saber cuántos "palindromios" se pueden crear usando exactamente n dígitos del 0 al 9. Ayúdalo a satisfacer su curiosidad.

Input

La primera línea contiene un entero t, $(1 \leq t \leq 100)$, el número de preguntas que Javier te hará. Cada una de sus preguntas consistirá en un solo entero n, la longitud del "palindromio", $(1 \leq n \leq 50000)$.

Output

Para cada pregunta, debes imprimir el número de "palindromios" de longitud n. Dado que este número puede ser muy muy grande, Javier quedará satisfecho si lo imprimes $\mod 10^9 + 7$. Esta operación se realiza utilizando % en C++ y Python. Ten en cuenta que $(a+b) \mod P$ es lo mismo que $(a \mod P + b \mod P) \mod P$, y algo similar se aplica con el producto, por lo que puedes aplicar la operación de módulo después de cada operación que realices para evitar desbordamientos. Si tienes preguntas sobre módulos, las responderemos en las aclaraciones.

Example

standard input	standard output
4 2 3 2023 13	90 900 616533557 540000000

Note

—

Idea del problema: Javi

Preparación del problema: Javi

Ocurrencias: Novato 5

Problem F. Harry potter en CMS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Harry está participando en la Olimpiada Informática de Hogwarts. En ella hay 1 problema con muchas subtareas. Sin embargo, Voldemort ha lanzado un conjuro sobre los jueces, haciéndoles creer que Harry ha copiado en algunos de sus envíos. Para saber cómo le ha ido en la competición teniendo en cuenta el encantamiento de Voldemort, Harry te hará q consultas.

Las consultas pueden ser de 3 tipos:

- 1 k: Harry te da información sobre un envío. Se te dan k números $x_1, x_2, \ldots, x_k$, cada x_i representa el índice de una subtarea que Harry ha obtenido como correcta en este envío. Todos los x_i son diferentes. Ten en cuenta, que Harry puede obtener como correcta una subtarea que ya tenía como correcta gracias a un envío anterior.
- 2 i: Harry te dice que los jueces le han anulado el envío que él te había especificado en la consulta i. Está garantizado que la consulta i es de tipo 1 y que Harry ya te había hecho esa consulta. También se garantiza que ese envío no se había anulado todavía.
- 3: Debes imprimir la cantidad de envíos que están incrementando la puntuación de Harry actualmente (teniendo en cuenta tan solo las consultas anteriores). Un envío incrementa la puntuación de Harry si, para alguna de las subtareas que obtiene como correcta, es el primer envío que te ha especificado Harry y aún no se ha anulado con esa subtarea correcta.

Input

La primera línea contiene un entero t, el número de casos a responder. ($1 \leq t \leq 100$).

La primera línea de cada caso contiene un entero q, el número de consultas que te hace Harry. ($2 \leq q \leq 2 \cdot 10^5$).

Las siguientes q líneas contienen las consultas. La línea j contiene un entero $type_j$, indicando el tipo de la consulta ($1 \leq type_j \leq 3$). Si $type_j = 1$, le seguirá un entero k_j y k_j enteros $x_{j_1}, x_{j_2}, \ldots, x_{j_k}$, las subtareas que obtiene Harry en ese envío, ($1 \leq x_{j_i} \leq 2 \cdot 10^5$). Si $type_j = 2$, le seguirá un entero i_j, el índice de la consulta especificando el envío que queda anulado.

Está garantizado que tanto la suma de q como la suma de $\sum k_j$ sobre todos los casos son como mucho $2 \cdot 10^5$.

Output

Para cada caso, debes imprimir el número de envíos que están incrementando la puntuación de Harry por cada consulta del tipo 3 que recibes.

Example

standard input	standard output
2 10 1 3 3 2 5 3 1 1 3 1 1 2 2 1 3 2 3 3 2 4 3 11 1 2 2 3 1 1 2 1 1 3 1 1 5 3 2 1 3 1 2 2 3 2 2 2 3 3	1 2 1 0 2 3 2

Note

—

Idea del problema: Esomer

Preparación del problema: Esomer, Hectorungo18

Ocurrencias: Novato 6

Problem G. XOR + Constructivo = Amor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Litusiano ha encontrado, una vez más, otro enunciado de 8 páginas de largo. Disgustado con los creadores de problemas que crean enunciados tan largos, ha creado un problema con un enunciado muy corto. Ahora te lo ha regalado para que puedas disfrutar de un enunciado corto, por una vez. Desafortunadamente, Esomer es muy malvado y ha interceptado el regalo de Litusiano, agregando este párrafo completamente inútil para molestar a Litusiano.

Se te da un arreglo a de longitud 30 y necesitas construir un nuevo arreglo b de longitud mínima tal que:

- Hay al menos a_i elementos con el bit i establecido en su representación binaria, para $0 \leq i < 30$.
- La suma de todos los elementos es s.
- El XOR bit a bit[†] de todos los elementos es exactamente x.

Si no existe tal arreglo, imprime -1 en su lugar.

[†]Si no sabes qué es el XOR bit a bit, recuerda que puedes usar internet!.

Input

La primera línea de la entrada contiene un solo entero t, el número de casos de prueba. ($1 \leq t \leq 10^4$).

En la primera línea de cada caso de prueba hay dos enteros, s y x, la suma requerida de todos los elementos en b y el XOR requerido de todos los elementos en b, respectivamente. ($0 \leq s \leq 10^{18}$; $0 \leq x < 2^{30}$).

Luego viene una línea con 30 enteros, $a_1, a_2, \ldots, a_{30}$, donde cada a_i denota el número mínimo de elementos en b que deben tener el i-ésimo bit establecido en su representación binaria. ($0 \leq a_i \leq 10^9$).

Output

Para cada caso de prueba, si no existe ningún arreglo que cumpla las condiciones, imprime -1. De lo contrario, imprime la longitud mínima de dicho arreglo.

Examples

standard input	standard output
2 9 5 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 6 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2 -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3 9 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2147483648 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 104 10 5 2 3 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	2 -1 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Note

—

Idea del problema: Esomer

Preparación del problema: Esomer

Ocurrencias: Novato 7, Avanzado 2

Problem H. Hormigas menorquinas

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Dani es uno de los mejores programadores competitivos del mundo, novio de la jambilla, mejor amigo de Innokentiy y culturista.

Un día en Menorca dejó unas galletas en su cama y aparecieron un montón de hormigas.

Hugo y Luis le dijeron que tirara la cama por la ventana y Dani lo hizo, pero Ilya y el funador se enfadaron mucho con ellos.

Ahora Dani necesita ayuda con el siguiente problema:

Hay n tipos de hormigas, y a_i hormigas del tipo i-ésimo, tienes que encontrar el **menor** número de movimientos para lanzar **al menos** p hormigas por la ventana.

Ten en cuenta que en cada movimiento, Dani puede lanzar **como máximo** m hormigas y k hormigas del **mismo** tipo.

Input

Cada caso contiene múltiples casos de prueba. La primera línea contiene el número de casos de prueba t ($1 \le t \le 10^4$). La descripción de los casos de prueba sigue a continuación.

La primera línea consta de tres números enteros n, m, k ($1 \le n \le 10^6; 1 \le m, k \le 10^{18}$) — el número de tipos de hormigas, el número máximo de hormigas que se pueden lanzar en un movimiento y el número máximo de hormigas del mismo tipo que se pueden lanzar en un movimiento, respectivamente.

Luego siguen n enteros $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^{12}$) — el número de hormigas de cada tipo.

La última línea consta de un número entero p ($1 \le p \le \sum_{i=1}^{n} a_i$) — el número mínimo de hormigas que se deben lanzar.

Se garantiza que la suma de n en todos los casos de prueba no exceda 10^6.

Output

Para cada caso de prueba, devuelva un único número entero: el número más pequeño de movimientos posibles.

Example

standard input	standard output
4 4 3 2 3 7 2 1 13 3 2 2 2 2 3 6 2 3 2 10 1 10 1 1 1 1 1	5 3 5 1

Note

La respuesta para el primer caso de prueba podría ser 1-er movimiento: $(3, 2, 2)$, 2-do movimiento: $(1, 2, 2)$, 3-er movimiento: $(1, 2, 1)$, 4-to movimiento: $(2, 3, 4)$, 5-to movimiento: (2).

Se puede demostrar que se necesitan al menos 5 movimientos.

La respuesta para el segundo caso de prueba podría ser 1-er movimiento: $(1, 1)$, 2-do movimiento: $(2, 3)$, 3-er movimiento: $(3, 3)$.

Se puede demostrar que se necesitan al menos 3 movimientos.

Problem I. Billetes falsos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

El banco Monopoly ha estado muy preocupado recientemente por la posibilidad de un robo, ya que no quieren perder sus billetes ~~falsos~~. Para reforzar su seguridad, han implementado un sistema de cámaras de seguridad frente a su bóveda. Ahora están preocupados de que pueda tener fallas, por lo que han pedido tu ayuda.

La habitación donde se encuentra la bóveda se puede representar como una cuadrícula de tamaño $n \times n$, donde la celda $(1, 1)$ es la entrada, y la bóveda está en la celda (n, n). Hay m cámaras en la habitación, cada una de ellas ocupando exactamente una celda. Las cámaras pueden apuntar en 4 direcciones diferentes: arriba, abajo, izquierda y derecha. Cada cámara apunta inicialmente en una dirección, que puede ser diferente entre cámaras. Cuando una cámara apunta en una dirección, cubre todas las celdas en esa dirección, incluyendo la celda en la que se encuentra.

Más formalmente, si la cámara está en la celda (r, c), donde r representa la fila desde arriba y c la columna desde la izquierda, entonces:

- Si la cámara apunta hacia arriba, para todo $1 \leq i \leq r$, la cámara cubre la celda (i, c).
- Si la cámara apunta hacia abajo, para todo $r \leq i \leq n$, la cámara cubre la celda (i, c).
- Si la cámara apunta hacia la izquierda, para todo $1 \leq j \leq c$, la cámara cubre la celda (r, j).
- Si la cámara apunta hacia la derecha, para todo $c \leq j \leq n$, la cámara cubre la celda (r, j).

Si un ladrón está en una celda cubierta por una cámara, será capturado.

Además, como los dueños de Monopoly son muy paranoicos, cada segundo las cámaras girarán 90° en sentido **antihorario** (es decir, si una cámara apunta hacia abajo, apuntará hacia la derecha; si apunta hacia la derecha, apuntará hacia arriba; y así sucesivamente). Los ladrones que intenten entrar en la bóveda tendrán que comenzar en la celda $(1, 1)$ y llegar a la celda (n, n) sin ser vistos por las cámaras. Cada segundo, los ladrones pueden hacer una de dos cosas:

- Quedarse en la celda en la que se encuentran.
- Moverse a una celda adyacente (es decir, una celda que comparte un lado con la celda en la que se encuentran).

Ten en cuenta que **las cámaras y los ladrones se mueven simultáneamente**, por lo que un ladrón puede moverse a una celda que esté actualmente cubierta por una cámara, siempre y cuando esté descubierta en el siguiente segundo. Consulta la sección de ejemplos para ver un ejemplo de esto.

Tu tarea es determinar si un ladrón puede llegar a la bóveda.

Input

La primera línea contiene un solo entero t, que denota el número de casos de prueba. $(1 \leq t \leq 100)$.

La primera línea de cada caso de prueba contiene dos enteros n y m, que representan el tamaño de la cuadrícula y el número de cámaras, respectivamente. $(1 \leq n \leq 1000; 0 \leq m \leq \max(0, n^2 - 2))$.

A continuación, siguen m líneas, cada una con dos enteros r y c, que representan la fila y la columna de la cámara, y un carácter d que será uno de 'U', 'D', 'L', 'R', que representa la dirección inicial de la cámara. $(1 \leq r, c \leq n)$.

Se garantiza que ninguna cámara comparte la misma celda, que ninguna cámara cubre inicialmente la celda $(1, 1)$ y que no hay una cámara en la celda (n, n).

Además, se garantiza que la suma de n en todos los casos de prueba es como máximo 1000.

Output

Para cada caso de prueba, debes imprimir "YES" si un ladrón puede llegar a la bóveda o "NO" si no puede.

Example

standard input	standard output
3 3 1 2 2 U 3 3 2 2 U 3 2 R 1 3 U 3 2 1 3 U 2 3 U	YES NO NO

Note

—

Idea del problema: Esomer

Preparación del problema: Esomer

Ocurrencias: Novato 9

Problem J. cPerturbación en la Fuerza

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Los impactos de la creación de "palindromio" por Javier y sus amigos ha alcanzado a Esomer (quien es capaz de medir cuando las fuerzas del orden se desequilibran en el universo).

Esomer es capaz de sentir que la Fuerza se ha desequilibrado en un entero x. Él, siendo el sabio ocupado que es, no tiene tiempo para restablecerla y necesitará hacer uso de sus n estudiantes para restaurar la Fuerza. Cada estudiante tiene un poder de a_i, y elegirlo reducirá la diferencia de x a $x - a_i$. Cada día que pasa todos sus estudiantes aumentan su poder en 1.

Ayúdale a remediar esta perturbación lo antes posible, es decir, imprime el mínimo numero de días en el que la diferencia x puede reducirse a **exactamente** 0 usando a sus estudiantes.

Una vez Esomer haya usado a uno de sus estudiantes, debido a su cansancio no podrá volverlo a usar.

Input

La primera línea contiene un entero t, $(1 \leq t \leq 10)$, el número de casos que a responder.

Cada uno de los casos consistirá en dos enteros n, la cantidad de estudiantes que posee $(1 \leq n \leq 200)$ y x la desigualdad que se ha causado en la fuerza $(1 \leq x \leq 200)$.

Le seguirán n enteros, la secuencia $a_1, a_2, ..., a_n$, donde a_i representa la fuerza del estudiante i-ésimo $(1 \leq a_i \leq x)$.

Está garantizado que las sumas de n y x sobre todos los casos no superarán 200.

Output

Para cada caso, debes imprimir un único entero, el mínimo número de días que han de pasar para que Esomer pueda reducir x a 0.

Example

standard input	standard output
1 3 4 1 1 1	1

Note

—

Idea del problema: Javi

Preparación del problema: Javi

Ocurrencias: Avanzado 3

Problem K. Óscar y su batalla

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Óscar está jugando a un videojuego. Su objetivo es conseguir el mayor número de monedas posible. Para ello tiene que elegir a uno de los n personajes disponibles. El i-ésimo personaje tiene un ataque de a_i puntos y una defensa de b_i puntos. Para conseguir las monedas deberá eliminar monstruos (hay m monstruos). El j-ésimo monstruo tiene un ataque de c_j puntos y una defensa de d_j puntos y otorga e_j monedas tras ser eliminado.

Para poder eliminar a un monstruo, Óscar necesita que el ataque del personaje sea **mayor o igual** a la defensa del monstruo y que su defensa sea **mayor o igual** al ataque monstruo. Es decir, el personaje i puede matar al monstruo j si y solo si $a_i \geq d_j$ y $b_i \geq c_j$.

Óscar solo puede elegir un personaje, pero puede matar a tantos monstruos como quiera. Una vez que un monstruo está muerto, no se le puede volver a matar. ¿Cuál es el máximo número de monedas que Óscar puede conseguir?

Input

La primera línea de la entrada contiene un entero t, el número de casos. ($1 \leq t \leq 100$).

La primera línea de cada caso contiene dos números enteros n y m ($1 \leq n, m \leq 2 \cdot 10^5$).

Las siguientes n líneas contienen dos números cada una, a_i y b_i; el ataque y la defensa de cada personaje, respectivamente. ($1 \leq a_i, b_i \leq 10^9$).

Las siguientes m líneas contienen tres números cada una, c_j, d_j y e_j; el ataque, la defensa y las monedas que otorga el monstruo j. ($1 \leq c_j, d_j, e_j \leq 10^9$).

Se garantiza que la suma de n y m en todos los casos de prueba es como máximo $2 \cdot 10^5$.

Output

Por cada caso, debes imprimir el máximo número de monedas que puede conseguir Óscar con un único personaje.

Examples

standard input	standard output
1 4 5 3 1 4 5 4 2 2 2 3 5 1 2 1 1 1 1 1 4 4 1 3 3 1	4
1 6 6 2 7 2 5 4 1 7 5 1 2 7 5 6 8 1 4 4 6 8 1 1 3 6 3 7 2 4 8 4 8	9

Note

—

Idea del problema: Hectorungo18

Preparación del problema: Esomer

Ocurrencias: Novato 10, Avanzado 5

Problem L. Intervalos aleatorios

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Dani tiene dos números enteros n y m y un botón que hace lo siguiente al presionarse:

- Genera n intervalos **aleatorios** de la forma (l, r) donde $1 \le l \le r \le m$.

Dani ha presionado el botón **dos** veces, pero solo te ha dado los n intervalos generados aleatoriamente después de la primera pulsación. Por favor, ayuda a Dani a encontrar la probabilidad de que **ningún** par de intervalos interseque.

Un par de intervalos (i, j) no interseca si y solo si $r_i \le l_j$ o $r_j \le l_i$. Ten en cuenta que no está garantizado que después de presionar el botón por primera vez no interseque ningún par de intervalos. Para obtener más información, consulta la sección de notas debajo de los casos de ejemplo.

Se puede demostrar que la probabilidad se puede representar como una fracción simple P/Q donde Q es coprimo con 998244353. Encuentra el valor de $P \cdot Q^{-1}$ módulo 998244353.

Input

Cada caso contiene múltiples casos de prueba. La primera línea contiene el número de casos de prueba t ($1 \le t \le 69$). La descripción de los casos de prueba sigue a continuación.

La primera línea contiene dos números enteros n, m ($1 \le n \le 269; 1 \le m \le 6.9 \cdot 10^6$) — el número de intervalos generados después de presionar el botón y el límite de los extremos de los intervalos, respectivamente.

Luego siguen n líneas, cada una consta de dos números enteros l_i, r_i ($1 \le l_i \le r_i \le m$) — los extremos de los intervalos generados.

Se garantiza que la suma de n en todos los casos de prueba no es superior a 269.

Output

Para cada caso de prueba, devuelve un único número entero: la probabilidad de que ningún par de intervalos interseque módulo 998244353.

Example

standard input	standard output
8 1 3 2 2 3 4 3 3 1 3 3 3 2 4 1 1 3 4 4 10 5 7 2 2 3 4 9 9 2 15 1 3 2 5 2 69 8 11 9 10 1 10 1 1 2 11 3 3 3 3	831870295 726721889 259543532 10510086 0 0 1 501184665

Note

En el primer caso de prueba, hay 6 distribuciones posibles, pero solo 5 son válidas. Si el intervalo generado fuera $(1, 3)$, los intervalos $(2, 2)$ y $(1, 3)$ intersecarían.

En el cuarto caso de prueba, hay 100 distribuciones posibles, pero solo 22 son válidas.

En el quinto caso de prueba, la respuesta es 0 porque algunos de los intervalos generados después de la primera pulsación del botón intersecan.

Problem M. La batalla del abismo de Helm

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

El mundo cambia y todo aquello que alguna vez parecía invencible hoy es inseguro. ¿Cómo podrá una torre resistir a fuerzas tan numerosas y a un odio tan implacable?

— Théoden, Rey de Rohan

Hay n torres en el abismo de Helm, cada una de ellas tiene un poder de a_i y una fortaleza de b_i. Los estrategas de Rohan determinan que los orcos llegarán en q oleadas y que en la j-ésima oleada x_j orcos atacarán la torre y_j. En esa oleada, la torre y_j recibirá un daño de $max(0, x_j - a_{y_j} \cdot p_{y_j})$, en que p_{y_j} es el número de soldados en la torre y_j. Una vez una torre i ha recibido un daño de al menos b_i, será conquistada. Si algunos orcos atacan una torre que ya está conquistada, simplemente volverán confundidos a sus líneas (los orcos no son muy inteligentes). Cuando una torre es conquistada, todos los soldados en ella morirán. Al inicio de cada oleada (antes de que el ataque de esa oleada ocurra), las murallas internas recibirán un daño igual al número de torres conquistadas en ese momento. Si las murallas internas reciben mucho daño, Rohan caerá y toda esperanza para los humanos se desvanecerá, por lo que quieres minimizar el daño que estas han recibido después de las q oleadas.

Rohan cuenta con m soldados, y te ha pedido que les ayudes a posicionarlos estratégicamente. Debes crear el mencionado arreglo p de longitud n, que debe satisfacer las siguientes condiciones:

- Para toda i, $0 \leq p_i \leq m$.
- $\sum p_i \leq m$.

Sea d el mínimo daño que pueden recibir las murallas internas sobre todas las posibilidades de p. Debes determinar el valor de d y el arreglo **lexicográficamente más pequeño**[†] p que haga que el daño total recibido por las murallas internas es exactamente d.

[†] Un arreglo a es lexicográficamente más pequeño que un arreglo diferente b del mismo tamaño si, en la primera posición i en la que difieren, $a_i < b_i$.

Input

La primera línea de la entrada contiene un entero t, el número de casos. ($1 \leq t \leq 100$).

La primera línea de cada caso contiene tres enteros n, m y q, el número de torres, soldados y oleadas, respectivamente. ($1 \leq n \leq 1000$; $0 \leq m \leq 1000$; $1 \leq q \leq 50000$).

Cada una de las siguientes n líneas contiene dos enteros, a_i and b_i, el poder y la fortaleza de la i-ésima torre. ($1 \leq a_i, b_i \leq 10^9$).

Después hay q líneas, cada una con dos enteros x_j and y_j, el número de orcos y la torre a la que atacan en la j-ésima oleada, respectivamente. ($1 \leq x_j \leq 10^9$; $1 \leq y_j \leq n$).

Está garantizado que ni la suma de n ni la suma de m sobre todos los casos excederán 1000, así cómo que la suma de q sobre todos los casos no excederá 50000.

Output

Por cada caso, imprime un entero d, el mínimo daño que pueden recibir las murallas internas.

Después, imprime n enteros, los elementos del arreglo p.

Example

standard input	standard output
2 5 15 7 2 3 4 5 3 3 4 1 1 1 3 1 4 5 1 3 3 2 100 1 2 3 9 4 5 9 4 1 1 1 1 1 1 1 1 1 1 3 5 4 5 5 5 6 5	2 1 0 1 0 4 0 0 0 0 0 5

Note

—

Idea del problema: Carlos Villagordo Espinosa

Preparación del problema: Carlos Villagordo Espinosa

Ocurrencias: Avanzado 7

Problem N. El naranjo de Omer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Omer es un matemático al que le encanta la programación y los naranjos, cuando termina el horario de oficina le encanta jugar con su precioso y amado naranjo.

Este naranjo esta formado por n naranjas numeradas de 1 a n, conectadas por $n-1$ ramas bidireccionales de modo que **todas** las naranjas están conectadas entre ellas por alguna secuencia de ramas. Además, este naranjo tiene como **raíz** la naranja 1, y cada naranja tiene un peso w_i $(1 < w_i <= n)$ donde todos los w_i son diferentes, ten en cuenta que w es una permutación de longitud n.

Mientras juega con su naranjo, Omer se hace algunas preguntas de la siguiente forma:

Dados u, a, b calcula el valor $\sum_{i=a}^{b} f(u,i)$ donde $f(u,i)$ = número de naranjas con peso divisible entre i en el subárbol que tiene como raíz la naranja u.

Como está muy ocupado cuidando de sus hijos, necesita que le ayudes con su naranjo y que respondas a todas sus preguntas.

Input

Cada caso contiene múltiples casos de prueba. La primera línea contiene el número de casos de prueba t $(1 \le t \le 1000)$. La descripción de los casos de prueba sigue a continuación.

La primera línea contiene dos números enteros n, q $(2 \le n, q \le 2 \cdot 10^5)$ — el número de naranjas y el número de preguntas, respectivamente.

La segunda línea consiste en una permutación de longitud n, $w_1, w_2, ...w_n$ — los pesos de las naranjas.

Luego siguen $n-1$ líneas, cada una consta de dos números enteros u_i, v_i $(1 \le u_i, v_i \le n)$ — indicando que hay una rama entre las naranjas u y v.

Finalmente siguen q líneas, cada una contiene tres números enteros u_i, a_i, b_i $(1 \le u_i \le n; 1 \le a_i\ leqb_i \le n)$ — las preguntas formuladas por Omer.

Se garantiza que la suma de n en todos los casos de prueba no excede $2 \cdot 10^5$.

Se garantiza que la suma de q en todos los casos de prueba no excede $2 \cdot 10^5$.

Output

Para cada caso de prueba, devuelve q enteros: la respuesta a las preguntas.

Example

standard input	standard output
2 5 4 1 5 3 4 2 2 1 2 4 3 5 3 2 1 2 4 2 3 5 1 1 5 3 2 2 2 6 2 1 1 2 1 1 1 1 1 2 1 2 2 2 1 2 2 1 1 2 2 2	4 3 10 1 2 3 1 1 1 0

Note

—

Idea del problema: danx

Preparación del problema: danx

Ocurrencias: Avanzado 8

Problem O. Bea la maximizadora

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

A Bea le gustan las cosas grandes

Todos lo saben

Bea tiene dos arrays a y b, ambos de n enteros.

Una permutación de longitud n es un array de longitud n donde cada número entre 1 y n aparece exactamente una vez.

Tu tarea es encontrar una permutación p de longitud n que **maximice** el siguiente valor:

$(a_1 + b_{p_1})$ & $(a_2 + b_{p_2})$ & ... & $(a_n + b_{p_n})$.

& es el operador bitwise AND, si no sabes como funciona recuerda que puedes buscarlo en internet ;)

Como puede haber muchas soluciones, Bea también quiere encontrar una que **minimice** la distancia máxima desde la posición de un valor en la permutación elegida hasta su posición original en la permutación ordenada; en otras palabras, Bea quiere encuentrar una que minimice el siguiente valor: $\max_{1 \le i \le n}(|i - pos_i|)$ donde pos_i es la posición de i en la permutación elegida y $|x|$ es la valor absoluto de x.

Por favor, ayuda a Bea :(

Input

Cada caso contiene múltiples casos de prueba. La primera línea contiene el número de casos de prueba t ($1 \le t \le 100$). La descripción de los casos de prueba sigue a continueación.

La primera línea contiene un número entero n ($2 \le n \le 1500$) — el tamaño de los arrays.

La siguiente línea consiste en n números enteros a_i ($0 \le a_i \le 10^9$) — los valores de a.

Finalmente siguen n números enteros más, b_i ($0 \le b_i \le 10^9$) — los valores de b.

Se garantiza que la suma de n en todos los casos de prueba no será superior a 1500.

Output

Para cada caso de prueba, escriba dos números enteros, el valor máximo posible eligiendo la permutación de manera óptima y la mínima distancia máxima posible desde la posición de un valor en la permutación elegida hasta su posición original en la permutación ordenada, de forma que se continúe obteniendo el valor máximo.

Example

standard input	standard output
4 5 5 13 4 10 0 3 0 2 11 15 3 7 1 2 5 8 1 3 1 2 3 5 4 6 2 1000000000 1000000000 1000000000 1000000000	8 1 2 1 7 2 2000000000 0

Note

—

Problem idea: danx

Problem preparation: danx

Ocurrences: Advanced 9

Problem P. Pistas de esquí

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A Esomer le gusta mucho ir a esquiar. Sin embargo, él odia los telesillas, puesto que son muy largos y siempre hace mucho frío en ellos. Decidido a no tener que sufrir ni un momento más, ha intentado idear una ruta con el mínimo tiempo de telesilla posible. Por desgracia, él no se orienta muy bien con el mapa de las pistas, así que te ha pedido ayuda.

Las pistas de esquí a las que va Esomer se pueden representar como un grafo dirigido y acíclico de n nodos y m aristas (las pistas). Además, hay k telesillas, que conectan un nodo a con otro nodo b. Curiosamente, todos los telesillas cumplen la condición de que siempre se puede llegar a a desde b sin usar ningún telesilla. Recorrer una pista o usar un telesilla le lleva 1 minuto a Esomer.

Esomer está interesado en cuánto tiempo tiene que estar como mínimo en los telesillas si quiere estar x minutos en total en la estación de esquí, sin quedarse quieto en un nodo ni un segundo, empezando el recorrido en el nodo y. Esomer puede acabar su recorrido en cualquier nodo e irse para casa en cuanto ha estado los x segundos.

Por favor, ayuda a Esomer.

Input

En la primera línea hay un entero t, el número de casos. ($1 \leq t \leq 100$).

En la primera línea de cada caso hay cuatro enteros n, m y k. ($2 \leq n \leq 10^5$; $1 \leq m \leq \min(10^5, \frac{n \cdot (n-1)}{2})$; $1 \leq k \leq 10^2$).

Luego hay m líneas, cada una con dos enteros, u y v, indicando que hay una arista de u a v. ($1 \leq u, v \leq n$).

Está garantizado que el grafo formado por las m aristas es acíclico.

Siguen k líneas, cada una con dos enteros, a y b, indicando que hay un telesilla de a a b. ($1 \leq a, b \leq n$; $a \neq b$).

Está garantizado que se puede llegar a a desde b sin usar ningún telesilla.

Finalmente, hay una línea conteniendo dos números enteros x e y, el número de minutos que Esomer quiere pasar en la estación de esquí y el nodo desde el cual comienza, respectivamente. ($1 \leq x \leq 10^9$; $1 \leq y \leq n$). Se garantiza que Esomer puede llegar a un telesilla desde el nodo y.

Está garantizado que la suma de n y m sobre todos los casos es como mucho 10^5, mientras que la suma de k sobre todos los casos es, como mucho, 10^2.

Output

Para cada caso deberás imprimir un entero s, el mínimo número de minutos que tiene que estar Esomer en telesillas, si quiere estar x minutos en total en la estación.

Examples

standard input	standard output
1 6 5 2 2 4 5 4 6 2 6 4 3 4 2 6 4 6 4 2	1
1 10 13 5 3 4 9 4 5 8 9 2 4 2 1 10 10 9 9 8 7 5 5 2 9 3 10 2 3 2 8 10 2 9 4 3 4 1 5 7 12 7	2

Note

—

Idea del problema: Carlos Villagordo Espinosa

Preparación del problema: Carlos Villagordo Espinosa

Ocurrencias: Avanzado 10