El fémur de Juan

Como cada 29 de marzo, Juan sale de casa y se encamina hacia la morada de Jacobo, su mejor amiguete, para conmemorar la fecha por todo lo alto, como cabe esperar. Sin embargo, un malintencionado escalón se interpone en su camino. En un giro inesperado de los acontecimientos, el peldaño provoca la caída de Juan, aparatosa a la par que bochornosa, mientras suelta una risotada pétrea que casa con su papel de archienemigo del protagonista. Fruto del vuelco, Juan se parte el fémur en dos lugares distintos, dando lugar a tres fragmentos de longitudes *a*, *b* y *c*.

Mientras espera a los servicios médicos, Juan se pregunta si podría formar un triángulo no degenerado con estos fragmentos de hueso.

Input Format
La entrada consiste en varios casos. Cada caso consta de tres números naturales *a, b, c* separados por espacios. Cada caso termina con un salto de línea.

Constraints

$0 \leq a,b,c < 10^8$

Output Format
Imprime una línea por caso con la palabra "si" si Juan puede formar un triángulo no degenerado con los fragmentos de hueso y la palabra "no" en caso contrario.

Solución en Python

```python
import sys

for line in sys.stdin:
	line = line.split(' ')
	a, b, c = int(line[0]), int(line[1]), int(line[2])

	if (b > a):
	aux = a
	a = b
	b = aux
	if (c > a):
	aux = a
	a = c
	c = aux

	if (a < b+c): print('si')
	else: print('no')
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```cpp
#include <iostream>

using namespace std;

int main() {
    int a, b, c;
    while (cin >> a >> b >> c) {
        if (b > a) swap(a,b);
        if (c > a) swap(a,c);
        if (a < b + c) cout << "si" << endl;
        else cout << "no" << endl;
    }
}
```

Autor de la solución: Cesc Folch (Miembro del comité organizador)

Juegos burocráticos

La burocracia es sinónimo de aburrimiento, pero no tendrás más remedio que enfrentarte a ella si quieres cobrar la beca antes de que acabe el año. A tal fin, te diriges a las oficinas ministeriales a probar suerte con las gestiones, para ver si consigues el documento acreditativo necesario.

Hay *n* ventanillas de secretaría en fila, una detrás de otra. En cada una de las ventanillas puedes hacer un trámite, que consiste en presentar un documento A para obtener otro documento B. El trámite solo se puede hacer si se dispone del documento A, y este documento se conserva tras el trámite. No obstante, una vez pasada una ventanilla (se realice o no un trámite en ella) no se puede volver atrás, por argucias legales que no vienen al caso. Al pasar la última ventanilla ya no será posible volver a la primera, puesto que las oficinas ya habrán cerrado.

Inicialmente, dispones de un solo documento. Dados el documento inicial y el deseado, ¿cuál es el mínimo número de trámites posible?

Input Format

La entrada empieza con un entero *T*, el número de casos de prueba. Cada caso empieza con *n* y *m*: el número de ventanillas y el número total de documentos que existen. Los documentos se numeran de *0* a *m-1*, siendo *0* el documento inicial y *m-1* el documento deseado. Siguen *n* líneas con dos enteros a_i b_i, indicando que en la *i*-ésima ventanilla se puede presentar el documento a_i para obtener el documento b_i.

Constraints

Caso 1 (5 puntos): $1 \leq n \leq 10^3, m = 2$

Caso 2 (5 puntos): $n = 2, 1 \leq m \leq 10^3$

Caso 3 (36 puntos): $1 \leq n, m \leq 10^3$

Caso 4 (54 puntos): $1 \leq n, m \leq 10^6$

Output Format

Para cada caso, imprimid una línea con el mínimo número de trámites a realizar. Si fuera imposible obtener el documento deseado, imprimid en su lugar "Imposible" (sin las comillas).

Solución en Python

```
ncases = int(input())
for case in range(ncases):
        line = input()
        line = input().split(' ')
```

```
        n, m = int(line[0]), int(line[1])

        V = [int(-1) for i in range(m)]

        V[0] = int(0)
        while (n > 0):
        n -= 1
        line = input().split(' ')
        x, y = int(line[0]), int(line[1])

        if (V[x] != -1 and (V[y] == -1 or V[y] >= V[x] + 1)):
                V[y] = V[x] + 1

        if (V[m-1] == -1): print('Imposible')
        else: print(V[m-1])
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, m;
        cin >> n >> m;
        vector<int> V(m,-1);
        V[0] = 0;
        while (n--) {
                int x,y;
                cin >> x >> y;
                if (V[x] != -1 and (V[y] == -1 or V[y] > V[x] + 1))
                        V[y] = V[x] + 1;
        }
        if (V[m-1] == -1) cout << "Imposible" << endl;
        else cout << V[m-1] << endl;
    }
}
```

Autor de la solución: Cesc Folch (Miembro del comité organizador)

Correo no deseado

Después de recibir propaganda de restaurantes locales, María decide pasar la tarde haciendo su pasatiempo favorito: quitar de los buzones la propaganda no deseada. Por este motivo, decide ir con un grupo de amigos a Badajoz para quitar el correo no deseado. El lugar donde se encuentran los buzones puede ser modelizado como una recta unidimensional. Los buzones se encuentran en las posiciones l_i, mientras que los amigos de María están, inicialmente, en las posiciones p_i. Cada segundo, los amigos de María pueden desplazarse una unidad a la derecha o a la izquierda. Si su posición coincide con la de un buzón, pueden eliminar la publicidad de ese buzón instantáneamente. Vuestro objetivo es encontrar el mínimo tiempo que necesitan María y sus amigos para quitar toda la propaganda si se distribuyen el trabajo óptimamente. En un mismo instante puede haber dos personas en la misma posición y también puede haber más de un buzón en la misma posición.

Input Format

La entrada empieza con un entero *t*, el números de casos. Le siguen los *t* casos y cada uno empieza descrito por dos enteros *n* y *m*, el número de buzones y la cantidad de amigos de María (contándola a ella). Las siguientes *n+m* líneas describen (con un entero cada una) la posición de los buzones (las *n* primeras líneas) y de los amigos de María (las siguientes).

Constraints

Caso 1 (7 puntos): $n = 1,\ 1 \le m \le 10^5,\ 0 \le ti,\ pi \le 10^{18}$

Caso 2 (12 puntos): $1 \le n \le 10^5,\ m = 1,\ 0 \le ti,\ pi \le 10^{18}$

Caso 3 (21 puntos): $1 \le n \le 20,\ 1 \le m \le 20,\ 0 \le ti,\ pi \le 10^9$

Caso 4 (26 puntos): $1 \le n \le 500,\ 1 \le m \le 500,\ 0 \le ti,\ pi \le 10^9$

Caso 5 (34 puntos): $1 \le n \le 10^5,\ 1 \le m \le 10^5,\ 0 \le ti,\ pi \le 10^{18}$

Output Format

Imprimid *t* líneas con un entero cada una, el tiempo mínimo que necesitan María y sus amigos para limpiar los buzones de propaganda.

Solución en Python

```
ncases = int(input())

for case in range(ncases):
        line = input().split(' ')
        n, m = int(line[0]), int(line[1])
```

```
        L = []
        P = []

        for i in range(n):
        innum = int(input())
        L.append(innum)
        for i in range(m):
        innum = int(input())
        P.append(innum)

        L = sorted(L)
        P = sorted(P)

        a = int(-1)
        b = int(2e18)

        while (a+1 < b):
        c = (a + b)//2
        j = 0

        for p in P:
                antj = j
                while (j < n and abs(L[j] - L[antj]) + min(abs(L[j] - p), abs(L[antj] - p))
<= c): j += 1

        if (j == n): b = c
        else: a = c

        print(int(b))
```

Autor de la solución: Javier López-Contreras (Miembro del comité organizador)

Solución en C++

```
#include <iostream>
#include <algorithm>

using namespace std;

typedef long long ll;
typedef vector<ll> vi;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, m;
        cin >> n >> m;
        vi L(n), P(m);
        for (auto& l : L) cin >> l;
        for (auto& p : P) cin >> p;
        sort(L.begin(),L.end());
        sort(P.begin(),P.end());
        ll a = -1;
```

```
        ll b = 2e18;
        while (a + 1 < b) {
                ll c = (a+b)/2;
                int j = 0;
                for (auto p : P) {
                        int antj = j;
                        while (
                            j < n and
                            abs(L[j]-L[antj]) + min(abs(L[j]-p),abs(L[antj]-p)) <= c
                          )
                                ++j;
                }
                if (j == n) b = c;
                else a = c;
        }
        cout << b << endl;
    }
}
```

Autor de la solución: Cesc Folch (Miembro del comité organizador)

La Pachanga

Quieres organizar un partido de fútbol entre *n* jugadores. Tienes que separar los jugadores en dos equipos, y todos los jugadores han convenido que poco les importa el número de jugadores en cada equipo. Sin embargo, sabes que hay personas que no se pueden ni ver, mucho menos jugar en el mismo equipo. De hecho, sabes que *m* parejas de jugadores distintas se odian una cantidad *h* (medida en fobios, unidad habitual del odio). Con esta información, debes encontrar el mínimo número $k \geq 0$ con el que se pueden hacer dos equipos distintos donde el odio entre dos personas cualesquiera de un mismo equipo es menor o igual a *k*.

Input Format

La entrada consiste en distintos casos. La primera línea contendrá un natural *t*, el número de casos.

Para cada caso:

- La primera línea contendrá dos enteros *n, m*, el número de jugadores, y el número de parejas que se odia.
- Las siguientes *m* líneas contendrán tres enteros u_k, v_k, h_k, los dos jugadores y cuánto se odian. No habrá dos parejas u_k y v_k repetidas.

Constraints

$1 \leq t$

$1 \leq uk, vk \leq n, uk \neq vk$

32 puntos: $2 \leq n \leq 2000, 0 \leq m \leq 2000, 1 \leq h_k \leq 2000$

17 puntos: $2 \leq n \leq 2000, 0 \leq m \leq 2000, 1 \leq h_k \leq 10^9$

51 puntos: $2 \leq n \leq 200000, 0 \leq m <= 200000, 1 \leq h_k \leq 10^9$

Output Format

La salida debe ser un natural para cada caso, el mínimo número $k \geq 0$ con el que se pueden hacer dos equipos distintos donde el odio entre dos personas cualesquiera de un mismo equipo es menor o igual a *k*.

Solución en C++

```
#include <iostream>
#include <vector>

using namespace std;
```

```cpp
typedef pair<int, int> pi;
typedef vector<pi> vpi;
typedef vector<vpi> vvpi;
typedef vector<int> vi;

vvpi G;
vi V;

bool dfs(int x, int c, int k) {
    if (V[x] != -1) return (V[x] == c);
    V[x] = c;
    bool res = true;
    for (auto it : G[x]) {
        int y = it.first;
        if (it.second > k) res &= dfs(y,c^1,k);
    }
    return res;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n, m;
        cin >> n >> m;
        G = vvpi(n);
        while (m--) {
               int x,y,k;
               cin >> x >> y >> k;
               --x; --y;
               G[x].push_back(pi(y,k));
               G[y].push_back(pi(x,k));
        }
        int a = -1;
        int b = 1e9+10;
        while (a + 1 < b) {
               int c = (a+b)/2;
               bool res = true;
               V = vi(n,-1);
               for (int i = 0; i < n; ++i) {
                      if (V[i] == -1) res &= dfs(i, 0,c);
               }
               if (res) b = c;
               else a = c;
        }
        cout << b << endl;
    }
}
```

Autor de la solución: Cesc Folch (Miembro del comité organizador)