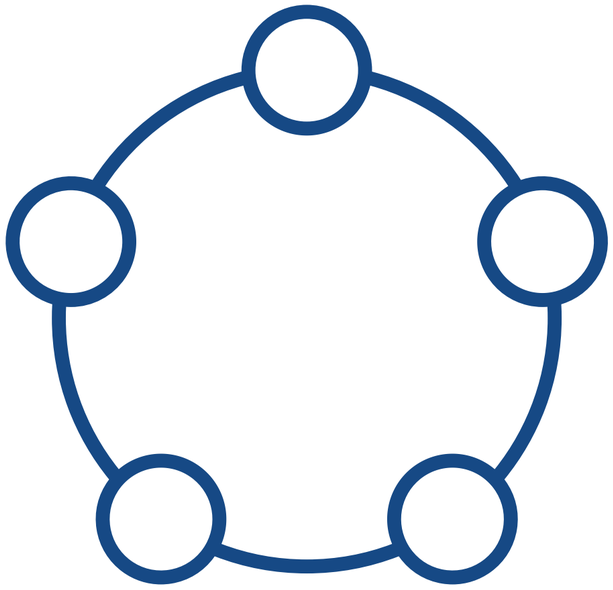


Semana 6

Teoria de Grafos

Ana Luiza Sticca, Gabriel Antônio, Guilherme Cabral e João Pedro Marques

Problemas

03 Anti Brute Force Lock

12 Audiophobia

03 Anti Brute Force Lock

Anti Brute Force Lock

Ultimamente, há um problema sério com o Panda Land Safe Box: vários cofres foram roubados! Os cofres usam a antiga combinação de fechadura rolante de 4 dígitos (você só precisa rolar o dígito, para cima ou para baixo, até que todos os quatro correspondam à chave). Cada dígito é projetado para rolar de 0 a 9. Acumular em 9 fará com que o dígito se torne 0, e rolar para baixo em 0 fará com que o dígito se torne 9. Como existem apenas 10.000 chaves possíveis, de 0.000 a 9.999, qualquer pessoa pode tentar todas as combinações até que o cofre seja desbloqueado.

O que está feito está feito. Mas para desacelerar o futuro ataque de ladrões, a Panda Security Agency (PSA) planejou uma nova fechadura mais segura com múltiplas chaves. Em vez de usar apenas uma combinação de teclas como chave, a fechadura agora pode ter até N chaves que devem ser todas desbloqueadas antes que o cofre possa ser aberto. Esses bloqueios funcionam da seguinte maneira:

Anti Brute Force Lock

- Inicialmente os dígitos estão em 0000.
- As chaves podem ser desbloqueadas em qualquer ordem, definindo os dígitos na fechadura para corresponderem à chave desejada e em seguida, pressionando o botão DESBLOQUEAR.
- Um botão mágico JUMP pode transformar os dígitos em qualquer uma das teclas desbloqueadas sem rolar.
- O cofre será desbloqueado se e somente se todas as chaves forem desbloqueadas em uma quantidade total mínima de rolando, excluindo JUMP (sim, esse recurso é o mais legal).
- Se o número de rolagens for excedido, os dígitos serão redefinidos para 0000 e todas as teclas serão bloqueado novamente. Em outras palavras, o estado do bloqueio será redefinido se a quebra falhar.

Anti Brute Force Lock

Entrada: A primeira linha de entrada contém um inteiro T, seguido pelo número de casos de teste. Cada caso começa com um inteiro N (1 ≤ N ≤ 500), o número de chaves. As próximas N linhas, cada uma contendo exatamente quatro dígitos número (zero à esquerda permitido) representando as chaves a serem desbloqueadas.

Saída: Para cada caso, imprima em uma única linha o número mínimo de rolos necessários para desbloquear todas as chaves. Explicação para o 2º caso:

- Transforme 0000 em 1111, resultado: 4
- Transforme 1111 em 1155, resultado: 8
- Salte 1155 para 1111, podemos fazer isso porque 1111 já foi desbloqueado antes.
- Transforme 1111 em rolos de 5511: 8

Total de lançamentos = 4 + 8 + 8 = 20

Sample Input

```
4
2 1155 2211
3 1111 1155 5511
3 1234 5678 9090
4 2145 0213 9113 8113
```

Sample Output

```
16
20
26
17
```

Anti Brute Force Lock (versão 1)

```
typedef struct Edge
{

    int u, v, w;

}Ed;

int vet[NMAX];

int pai[NMAX];
int tam[NMAX];
```

```
int getDist(int s1, int s2)
{

    int a, b, x, r, i;

    r = 0;

    for(i = 0;i < 4;i++)
    {

        a = s1 % 10;
        b = s2 % 10;

        x = abs(a - b);

        r += min(x, 10 - x);

        s1 /= 10;
        s2 /= 10;
```

```cpp
int findPai(int u)
{

    if(pai[u] == u) return u;

    return pai[u] = findPai(pai[u]);

}

bool Join(int u, int v)
{

    u = findPai(u);
    v = findPai(v);

    if(u == v) return false;

    if(tam[u] > tam[v])
        swap(u, v);

    if(tam[u] > tam[v])
        swap(u, v);

    pai[u] = v;
    tam[v] += tam[u];

    return true;

}

bool cmp(Edge a, Edge b)
{

    return a.w < b.w;

}
```

```cpp
int main()
{

    int tt, n, u, v, w, resp, i, j;

    vector< Ed > edges;

    cin >> tt;

    while(tt--)
    {

        edges.clear();
        resp = 0;

        cin >> n;

        for(i = 1;i <= n;i++) pai[i] = i;
        for(i = 1;i <= n;i++) tam[i] = 1;

        for(i = 1;i <= n;i++) cin >> vet[i];
```

```cpp
for(i = 1;i <= n;i++)
{

    for(j = i + 1;j <= n;j++)
    {

        edges.push_back({i, j, getDist(vet[i], vet[j])});

    }

}

sort(edges.begin(), edges.end(), cmp);

for(auto cur : edges)
{

    u = cur.u;
    v = cur.v;
    w = cur.w;

    if(Join(u, v) == false)
        continue;

    resp += w;
```

```
    w = getDist(vet[1], 0);

    for(i = 2;i <= n;i++)
        w = min(w, getDist(vet[i], 0));

    resp += w;

    cout << resp << endl;

}

return 0;
```

Anti Brute Force Lock (versão 2)

```java
public class Dupla implements Comparable<Dupla> {
    int first;
    int second;

    public Dupla(int a, int b) {
        this.first = a;
        this.second = b;
    }

    public int compareTo(Dupla duplaAlt) {
        return Integer.compare(this.first, duplaAlt.first);
    }
}
```

```java
public class Code {
    int a, b, c, d;

    public Code(String number){
        a = Character.getNumericValue(number.charAt(0));
        b = Character.getNumericValue(number.charAt(1));
        c = Character.getNumericValue(number.charAt(2));
        d = Character.getNumericValue(number.charAt(3));
    }
    public Code(){
        a = 0;
        b = 0;
        c = 0;
        d = 0;
    }
```

```java
    public int codeDist(Code ot){
        int r = 0;
        r += mendist(a, ot.a);
        r += mendist(b, ot.b);
        r += mendist(c, ot.c);
        r += mendist(d, ot.d);
        return r;
    }
    public int mendist(int a, int b){
        int menor = 0;
        int m1 = Math.abs(a-b);
        int m2 = 10-m1;
        return Math.min(m1, m2);
    }
}
```

```
    visited[0] = true;

    lPrior.addAll(adj[0]);

    while(!lPrior.isEmpty()) {
        Dupla a;
        a = new Dupla(lPrior.peek().first, lPrior.poll().second);
        if(!visited[a.second]){
            visited[a.second] = true;
            s += a.first;
           //System.out.println("Em " + dic[a.second].a + dic[a.second].b + dic[a.
            lPrior.addAll(adj[a.second]);
        }
    }
    //adicionamos agora uma aresta 0000 ao caminho
    Code zero = new Code();
    for(int g = 0; g < m; g++){
        lPrior.add(new Dupla(zero.codeDist(dic[g]), 0));
    }
    //a menor aresta de 0000 ate um dos codigos digitados é somada ao resultado
    results[i] = s + lPrior.poll().first;
    lPrior.clear();
}
```

Exemplos rodados

```
> java -classpath .:target/depe
4
2 1155 2211
3 1111 1155 5511
3 1234 5678 9090
4 2145 0213 9113 8113
16
20
26
17
>
```

12

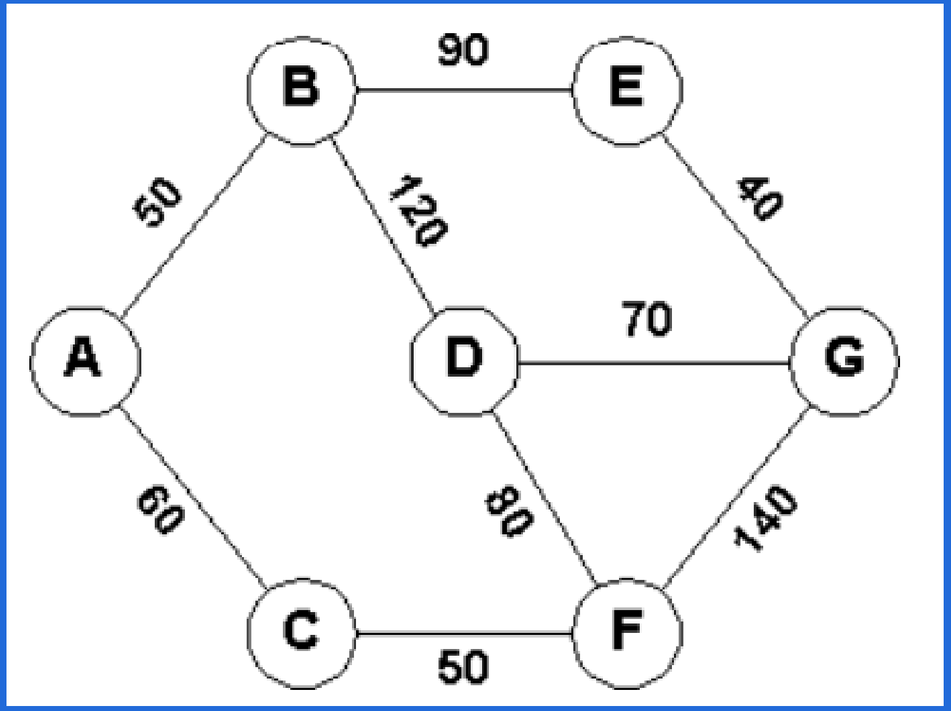
Audiophobia

Audiophobia

Considere-se com sorte! Considere-se sortudo por ainda estar respirando e se divertindo participando deste concurso. Mas tememos que muitos dos seus descendentes possam não ter esse luxo. Pois, como você sabe, somos moradores de uma das cidades mais poluídas do planeta. A poluição está em toda parte, tanto em no ambiente e na sociedade e a nossa falta de consciência está simplesmente a agravar a situação.

Porém, por enquanto, consideraremos apenas um tipo de poluição – a poluição sonora. O volume ou nível de intensidade do som é geralmente medido em decibéis e o som tem nível de intensidade 130 decibéis ou mais é considerado doloroso. O nível de intensidade de uma conversa normal é de 6.065 decibéis e o do tráfego pesado é de 7.080 decibéis. Considere o seguinte mapa da cidade onde as arestas referem-se às ruas e os nos cruzamentos. O número inteiro em cada borda é o nível médio de intensidade do som (em decibéis) na rua correspondente.

Para ir do cruzamento A ao cruzamento G poderá seguir o seguinte caminho: A-C-F-G. Nesse caso, você deve ser capaz de tolerar uma intensidade sonora de até 140 decibéis. Para os caminhos A-B-E-G, A-B-D-G e A-C-F-D-G você deve tolerar respectivamente 90, 120 e 80 decibéis de intensidade sonora. Existem outros caminhos também. No entanto, está claro que A-C-F-D-G é o caminho mais confortável, uma vez que não exige que você tolere mais de 80 decibéis. Neste problema, dado um mapa da cidade, é necessário determinar o nível mínimo de intensidade sonora você deve ser capaz de tolerar para poder passar de uma determinada travessia para outra.



Audiophobia

Entrada: A entrada pode conter vários casos de teste. A primeira linha de cada caso de teste contém três inteiros C(≤ 100), S(≤ 1000) e Q(≤ 10000) onde C indica o número de cruzamentos (os cruzamentos são numerados usando números inteiros distintos variando de 1 a C), S representa o número de ruas e Q é o número de consultas. Cada uma das próximas S linhas contém três inteiros: c1, c2 e d indicando que o som médio o nível de intensidade na rua que liga os cruzamentos c1 e c2 (c1/= c2) é d decibéis. Cada uma das próximas Q linhas contém dois inteiros c1 e c2 (c1/= c2) solicitando o som mínimo nível de intensidade que você deve ser capaz de tolerar para passar do cruzamento c1 ao cruzamento c2. A entrada terminará com três zeros de C, S e Q.

Audiophobia

Saída: Para cada caso de teste na entrada, primeiro produza o número do caso de teste (começando em 1), conforme mostrado na saída de amostra. Então, para cada consulta na entrada, imprima uma linha fornecendo o nível mínimo de intensidade sonora (em decibéis) você deve ser capaz de tolerar para ir da primeira à segunda travessia no consulta. Se não existir nenhum caminho entre eles basta imprimir a linha “no path”. Imprima uma linha em branco entre dois casos de teste consecutivos.

Sample Input

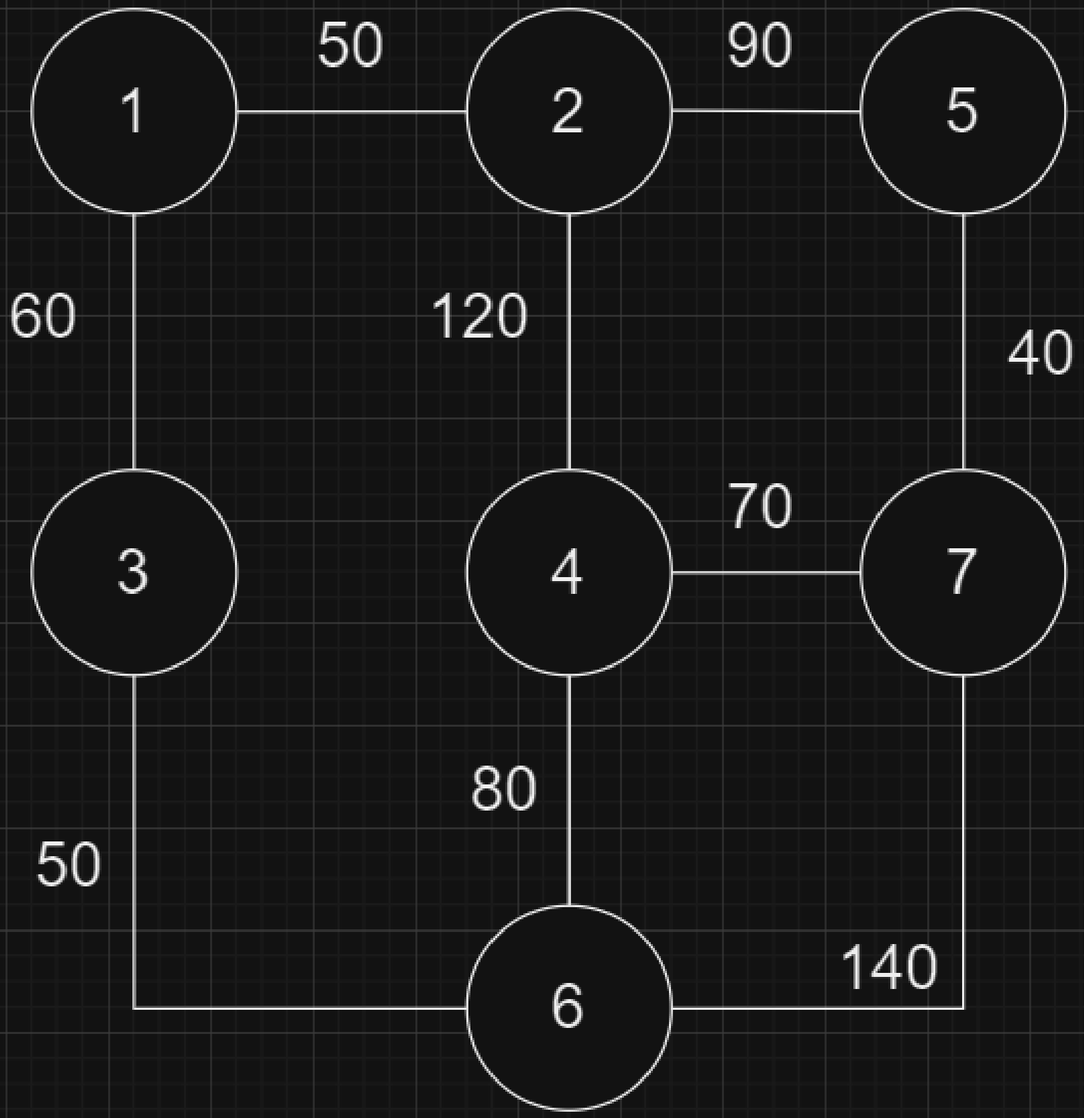
```
7 9 3
1 2 50
1 3 60
2 4 120
2 5 90
3 6 50
4 6 80
4 7 70
5 7 40
6 7 140
1 7
2 6
6 2
7 6 3
1 2 50
1 3 60
2 4 120
3 6 50
4 6 80
5 7 40
7 5
1 7
2 4
0 0 0
```

Sample Output

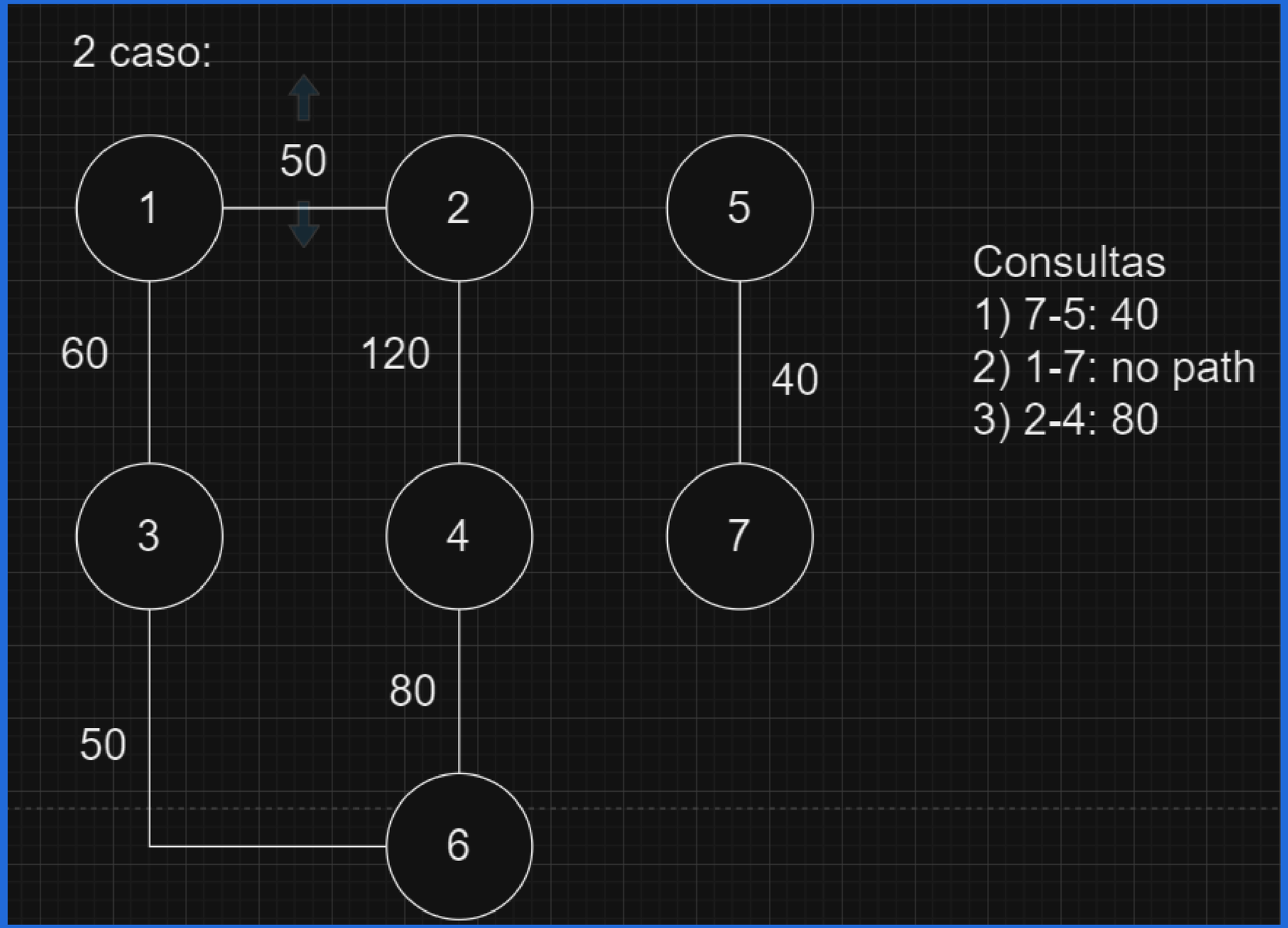
```
Case #1
80
60
60

Case #2
40
no path
80
```

1 caso:



Consultas
1) 1-7: 80
2) 2-6: 60
3) 6-2: 60


2 caso:
50
1
2
5
60
120
40
3
4
7
80
50
6
Consultas
1) 7-5: 40
2) 1-7: no path
3) 2-4: 80

Audiophobia

Estrutura de dados: Listas de Adjacência e matriz de adjacência

Vértices: Cruzamento entre as ruas

Arestas: Ruas

Audiophobia (versão 1)

```
typedef struct Edge
{

    int u, v, w;

}Ed;

vector< pair< int, int > > grafo[NMAX];

int pai[NMAX];
int tam[NMAX];

int findPai(int u)
{

    if(pai[u] == u) return u;

    return pai[u] = findPai(pai[u]);
```

```
bool Join(int u, int v)
{

    u = findPai(u);
    v = findPai(v);

    if(u == v) return false;

    if(tam[u] > tam[v])
        swap(u, v);

    pai[u] = v;
    tam[v] += tam[u];

    return true;

}
```

```cpp
int getMax(int u, int dest, int pai, int ma)
{

    if(u == dest) return ma;

    int v, w, x;

    for(auto viz : grafo[u])
    {

        w = viz.first;
        v = viz.second;

        if(v == pai) continue;

        x = getMax(v, dest, u, max(ma, w));

        if(x != -1) return x;

    }

    return -1;
```

```cpp
bool cmp(Edge a, Edge b)
{

    return a.w < b.w;

}

int main()
{

    int n, m, q, u, v, w, caso = 1, i, j;

    vector< Ed > edges;

    while(cin >> n >> m >> q)
    {

        if(n == 0 && m == 0 && q == 0) break;

        if(caso != 1) cout << endl;
```

```
edges.clear();
for(i = 1;i <= n;i++) grafo[i].clear();

for(i = 1;i <= n;i++) pai[i] = i;
for(i = 1;i <= n;i++) tam[i] = 1;

while(m--)
{

    cin >> u >> v >> w;

    edges.push_back({u, v, w});

}

sort(edges.begin(), edges.end(), cmp);

for(auto cur : edges)
{

    u = cur.u;
    v = cur.v;
    w = cur.w;
```

```
    if(Join(u, v) == false)
        continue;

    grafo[u].push_back({w, v});
    grafo[v].push_back({w, u});

}

cout << "Case #" << caso++ << endl;

while(q--)
{

    cin >> u >> v;

    w = getMax(u, v, u, -1);

    if(w == -1)    cout << "no path" << endl;
    else           cout << w << endl;
```

Audiophobia (versão 2)

```
while (scanner.hasNext()) {
    int n = scanner.nextInt();
    int m = scanner.nextInt();
    int q = scanner.nextInt();

    if (n == 0) {
        break;
    }

    int[][] f = new int[101][101];
    int i, j, k, x, y, b;

    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            f[i][j] = oo;
        }
    }

    while (m-- > 0) {
        x = scanner.nextInt();
        y = scanner.nextInt();
        b = scanner.nextInt();
        f[x][y] = Math.min(f[x][y], b);
        f[y][x] = f[x][y];
    }
```

```
    for (k = 1; k <= n; k++) {
        for (i = 1; i <= n; i++) {
            for (j = 1; j <= n; j++) {
                f[i][j] = Math.min(f[i][j], Math.max(f[i][k], f[k][j]));
            }
        }
    }

    if (cases > 0) {
        System.out.println();
    }
    System.out.printf("Case #%d%n", ++cases);

    while (q-- > 0) {
        x = scanner.nextInt();
        y = scanner.nextInt();
        if (f[x][y] != oo) {
            System.out.println(f[x][y]);
        } else {
            System.out.println("no path");
        }
    }
```

Exemplos rodados