UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE PRODUCCION Y SERVICIOS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Curso: Laboratorio de Análisis y Diseño de Algoritmos

Aula 10 -

Presentado por:

Tacca Apaza, Nohelia Estefhania

Docente:

Alex Josue Florez Farfan

Grupo-"B"

Arequipa - Perú

Diciembre 2021

Greedy Algorithms

1. Explican cómo llegan a la solución

2. Escriben el código

3. Lo suben a la página

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)

5. Suben el código y la captura de pantalla

Ejercicio 01 - Road Reparation

There are n cities and m roads between them. Unfortunately, the condition of the roads is so poor that they cannot be used. Your task is to repair some of the roads so that there will be a decent route between any two cities.

For each road, you know its reparation cost, and you should find a solution where the total cost is as small as possible.

Input

The first input line has two integers n and m: the number of cities and roads. The cities are numbered 1,2,…,n.

Then, there are m lines describing the roads. Each line has three integers a, b and c: there is a road between cities a and b, and its reparation cost is c. All roads are two-way roads.

Every road is between two different cities, and there is at most one road between two cities.

Output

Print one integer: the minimum total reparation cost. However, if there are no solutions, print "IMPOSSIBLE".

Constraints

$1 \le n \le 10^5$

$1 \le m \le 2 \cdot 10^5$

$1 \le a,b \le n$

$1 \le c \le 10^9$

Example

Input:

5 6

1 2 3

2 3 5

2 4 2

3 4 8

5 1 7

5 4 4

Output:

14

2. Escriben el código (✔)

3. Lo suben a la página (✔)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)

Road Reparation

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Road Reparation
Sender:	Nohelia
Submission time:	2021-12-19 06:45:51
Language:	C++11
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.01 s	»
#5	ACCEPTED	0.01 s	»
#6	ACCEPTED	0.24 s	»
#7	ACCEPTED	0.24 s	»
#8	ACCEPTED	0.24 s	»
#9	ACCEPTED	0.24 s	»
#10	ACCEPTED	0.24 s	»
#11	ACCEPTED	0.12 s	»
#12	ACCEPTED	0.01 s	»
#13	ACCEPTED	0.01 s	»
#14	ACCEPTED	0.01 s	»

5. Suben el código y la captura de pantalla

```cpp
#include<bits/stdc++.h>

#define ll long long int
using namespace std;

struct Edge{
        int a;
        int b;
        ll weight;
};

vector<Edge> edgeList;
int _par[200001];

bool comp(Edge a , Edge b){
        return a.weight < b.weight;
}

int find(int a){
        if(_par[a] == -1) return a;

        return _par[a] = find(_par[a]);
}

bool merge(int a , int b){
        a = find(a);
        b = find(b);

        if(a == b) return false;

        _par[a] = b;
        return true;
}

int main(){
        int n , m;
        ll res = 0;
        int addedEdgeCount = 0;
        Edge temp;

        cin>>n>>m;

        for(int i=1;i<=n;i++) _par[i] = -1;

        for(int i=1;i<=m;i++) cin>>temp.a>>temp.b>>temp.weight , edgeList.push_ba

        sort(edgeList.begin() , edgeList.end() , comp);

        for(Edge e : edgeList){
                if(merge(e.a , e.b)) res += e.weight , addedEdgeCount++;
        }

        if(addedEdgeCount == n-1)
        cout<<res;
        else
        cout<<"IMPOSSIBLE";
}
```

Ejercicio 02 - Shortest Routes I

There are n cities and m flight connections between them. Your task is to determine the length of the shortest route from Syrjälä to every city.

Input

The first input line has two integers n and m: the number of cities and flight connections. The cities are numbered 1,2,…,n, and city 1 is Syrjälä.

After that, there are m lines describing the flight connections. Each line has three integers a, b and c: a flight begins at city a, ends at city b, and its length is c. Each flight is a one-way flight.

You can assume that it is possible to travel from Syrjälä to all other cities.

Output

Print n integers: the shortest route lengths from Syrjälä to cities 1,2,…,n.

Constraints

$1 \le n \le 105$

$1 \le m \le 2 \cdot 105$

$1 \le a,b \le n$

$1 \le c \le 109$

Example

Input:

3 4

1 2 6

1 3 2

3 2 3

1 3 4

Output:

0 5 2

2. Escriben el código (✔)

3. Lo suben a la página (✔)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)

CSES Problem Set

Shortest Routes I

TASK | SUBMIT | RESULTS | STATISTICS

Submission details

Task:	Shortest Routes I
Sender:	Nohelia
Submission time:	2021-12-19 06:52:50
Language:	Java
Status:	READY
Result:	TIME LIMIT EXCEEDED

Test results

test	verdict	time	
#1	ACCEPTED	0.18 s	»
#2	ACCEPTED	0.18 s	»
#3	ACCEPTED	0.18 s	»
#4	ACCEPTED	0.18 s	»
#5	ACCEPTED	0.18 s	»
#6	TIME LIMIT EXCEEDED	--	»
#7	TIME LIMIT EXCEEDED	--	»
#8	TIME LIMIT EXCEEDED	--	»
#9	TIME LIMIT EXCEEDED	--	»
#10	TIME LIMIT EXCEEDED	--	»
#11	TIME LIMIT EXCEEDED	--	»
#12	TIME LIMIT EXCEEDED	--	»
#13	ACCEPTED	0.18 s	»
#14	TIME LIMIT EXCEEDED	--	»
#15	TIME LIMIT EXCEEDED	--	»
#16	TIME LIMIT EXCEEDED	--	»
#17	TIME LIMIT EXCEEDED	--	»
#18	TIME LIMIT EXCEEDED	--	»

5. Suben el código y la captura de pantalla

Code

```java
import java.io.*;
import java.util.*;

public class solution {
        static Scanner s = new Scanner(System.in);
    static class E {
        int j, w;
        E(int j, int w) {
            this.j = j; this.w = w;
        }
    }
    static ArrayList[] aa;
    static final long INF = 0x3f3f3f3f3f3f3f3fL;
    static long[] dd;
    public static void main(String[] args) throws IOException {
        PrintWriter pw = new PrintWriter(System.out);
        int n = s.nextInt();
        int m = s.nextInt();
        aa = new ArrayList[n];
        for (int i = 0; i < n; i++)
            aa[i] = new ArrayList<E>();
        while (m-- > 0) {
            int i = s.nextInt() - 1;
            int j = s.nextInt() - 1;
            int w = s.nextInt();
            aa[i].add(new E(j, w));
        }
        dd = new long[n];
        Arrays.fill(dd, INF);
        dd[0] = 0;
        TreeSet<Integer> pq = new TreeSet<>((i, j) -> dd[i] == dd[j] ? i - j : dd
        pq.add(0);
        Integer i_;
        while ((i_ = pq.pollFirst()) != null) {
            int i = i_;
            ArrayList<E> adj = aa[i];
            for (E e : adj) {
                long d = dd[i] + e.w;
                if (dd[e.j] > d) {
                    if (dd[e.j] != INF)
                        pq.remove(e.j);
                    dd[e.j] = d;
                    pq.add(e.j);
                }
            }
        }
```

```
        for (int i = 0; i < n; i++)
            pw.print(dd[i] + " ");
        pw.println();
        pw.close();
    }
    static class _Scanner {
        InputStream is;
        _Scanner(InputStream is) {
            this.is = is;
        }
        byte[] bb = new byte[1 << 15];
        int k, l;
        byte getc() throws IOException {
            if (k >= l) {
                k = 0;
                l = is.read(bb);
                if (l < 0) return -1;
            }
            return bb[k++];
        }
        byte skip() throws IOException {
            byte b;
            while ((b = getc()) <= 32)
                ;
            return b;
        }
        int nextInt() throws IOException {
            int n = 0;
            for (byte b = skip(); b > 32; b = getc())
                n = n * 10 + b - '0';
            return n;
        }
    }
}
```

Shortest Routes I

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Shortest Routes I
Sender:	Nohelia
Submission time:	2021-12-19 07:14:55
Language:	C++11
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.01 s	»
#5	ACCEPTED	0.01 s	»
#6	ACCEPTED	0.14 s	»
#7	ACCEPTED	0.14 s	»
#8	ACCEPTED	0.14 s	»
#9	ACCEPTED	0.14 s	»
#10	ACCEPTED	0.14 s	»
#11	ACCEPTED	0.10 s	»
#12	ACCEPTED	0.07 s	»
#13	ACCEPTED	0.01 s	»
#14	ACCEPTED	0.07 s	»
#15	ACCEPTED	0.07 s	»
#16	ACCEPTED	0.07 s	»
#17	ACCEPTED	0.07 s	»
#18	ACCEPTED	0.09 s	»

Code

```cpp
#include <bits/stdc++.h>
using namespace std;

#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define trav(a, x) for(auto& a : x)
#define all(x) begin(x), end(x)
#define sz(x) (int)(x).size()
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

int main() {
    cin.sync_with_stdio(0); cin.tie(0);
    cin.exceptions(cin.failbit);

    constexpr ll inf=1e18;
    int n, m;
    cin >> n >> m;
    vector<vector<pii>> g(n+1);
    while (m--) {
        int a, b, c;
        cin >> a >> b >> c;
        g[a].push_back({b, c});
    }
    vector<ll> d(n+1, inf);
    typedef pair<ll, int> T;
    priority_queue<T, vector<T>, greater<T>> q;
    for(q.push({0, 1}); !q.empty(); q.pop()) {
        ll cur=q.top().first;
        int i=q.top().second;
        if (d[i]<inf) continue;
        d[i]=cur;
        trav(p, g[i]) q.push({cur+p.second, p.first});
    }
    rep(i, 1, n+1) cout << d[i] << " \n"[i==n];
    return 0;
}
```

Ejercicio 03 - Shortest Routes II

There are n cities and m roads between them. Your task is to process q queries where you have to determine the length of the shortest route between two given cities.

Input

The first input line has three integers n, m and q: the number of cities, roads, and queries.

Then, there are m lines describing the roads. Each line has three integers a, b and c: there is a road between cities a and b whose length is c. All roads are two-way roads.

Finally, there are q lines describing the queries. Each line has two integers a and b: determine the length of the shortest route between cities a and b.

Output

Print the length of the shortest route for each query. If there is no route, print −1 instead.

Constraints

$1 \leq n \leq 500$

$1 \leq m \leq n2$

$1 \leq q \leq 105$

$1 \leq a,b \leq n$

$1 \leq c \leq 109$

Example

Input:

4 3 5

1 2 5

1 3 9

2 3 3

1 2

2 1

1 3

1 4

3 2

Output:

5

5

8

-1

3

2. Escriben el código (✔)

3. Lo suben a la página (✔)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)

Shortest Routes II

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Shortest Routes II
Sender:	Nohelia
Submission time:	2021-12-19 07:23:32
Language:	Java
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.08 s	»
#2	ACCEPTED	0.08 s	»
#3	ACCEPTED	0.08 s	»
#4	ACCEPTED	0.08 s	»
#5	ACCEPTED	0.08 s	»
#6	ACCEPTED	0.43 s	»
#7	ACCEPTED	0.41 s	»
#8	ACCEPTED	0.41 s	»
#9	ACCEPTED	0.41 s	»
#10	ACCEPTED	0.40 s	»
#11	ACCEPTED	0.40 s	»
#12	ACCEPTED	0.43 s	»
#13	ACCEPTED	0.08 s	»
#14	ACCEPTED	0.48 s	»
#15	ACCEPTED	0.42 s	»

5. Suben el código y la captura de pantalla

```java
import java.io.*;
import java.util.*;

public class main {
	static final long INF = 0x3f3f3f3f3f3f3f3fL;
	public static void main(String[] args) throws IOException {
		_Scanner sc = new _Scanner(System.in);
		PrintWriter pw = new PrintWriter(System.out);
		int n = sc.nextInt();
		int m = sc.nextInt();
		int q = sc.nextInt();
		long[][] aa = new long[n][n];
		for (int i = 0; i < n; i++) {
			for (int j = 0; j < n; j++)
				aa[i][j] = INF;
			aa[i][i] = 0;
		}
		while (m-- > 0) {
			int i = sc.nextInt() - 1;
			int j = sc.nextInt() - 1;
			int w = sc.nextInt();
			if (aa[i][j] > w)
				aa[j][i] = aa[i][j] = w;
		}
		for (int k = 0; k < n; k++)
			for (int i = 0; i < n; i++)
				for (int j = i + 1; j < n; j++) {
					long a = aa[i][k] + aa[k][j];
					if (aa[i][j] > a)
						aa[j][i] = aa[i][j] = a;
				}
		while (q-- > 0) {
			int i = sc.nextInt() - 1;
			int j = sc.nextInt() - 1;
			long a = aa[i][j];
			pw.println(a == INF ? -1 : a);
		}
		pw.close();
	}
	static class _Scanner {
		InputStream is;
		_Scanner(InputStream is) {
			this.is = is;
		}
		byte[] bb = new byte[1 << 15];
		int k, l;
		byte getc() throws IOException {
			if (k >= l) {
				k = 0;
				l = is.read(bb);
```

```
                                l = is.read(bb);
                                if (l < 0) return -1;
                        }
                        return bb[k++];
                }
                byte skip() throws IOException {
                        byte b;
                        while ((b = getc()) <= 32)
                                ;
                        return b;
                }
                int nextInt() throws IOException {
                        int n = 0;
                        for (byte b = skip(); b > 32; b = getc())
                                n = n * 10 + b - '0';
                        return n;
                }
        }
}
```

Ejercicio 04 - Longest Flight Route

Uolevi has won a contest, and the prize is a free flight trip that can consist of one or more flights through cities. Of course, Uolevi wants to choose a trip that has as many cities as possible.

Uolevi wants to fly from Syrjälä to Lehmälä so that he visits the maximum number of cities. You are given the list of possible flights, and you know that there are no directed cycles in the flight network.

Input

The first input line has two integers n and m: the number of cities and flights. The cities are numbered 1,2,…,n. City 1 is Syrjälä, and city n is Lehmälä.

After this, there are m lines describing the flights. Each line has two integers a and b: there is a flight from city a to city b. Each flight is a one-way flight.

Output

First print the maximum number of cities on the route. After this, print the cities in the order they will be visited. You can print any valid solution.

If there are no solutions, print "IMPOSSIBLE".

Constraints

$2 \leq n \leq 10^5$

$1 \leq m \leq 2 \cdot 10^5$

$1 \leq a,b \leq n$

Example

Input:

5 5

1 2

2 5

1 3

3 4

4 5

Output:

4

1 3 4 5

Flight Discount

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Flight Discount
Sender:	Nohelia
Submission time:	2021-12-19 07:28:49
Language:	C++11
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.12 s	»
#5	ACCEPTED	0.18 s	»
#6	ACCEPTED	0.12 s	»
#7	ACCEPTED	0.13 s	»
#8	ACCEPTED	0.20 s	»
#9	ACCEPTED	0.11 s	»
#10	ACCEPTED	0.01 s	»
#11	ACCEPTED	0.01 s	»
#12	ACCEPTED	0.01 s	»
#13	ACCEPTED	0.09 s	»
#14	ACCEPTED	0.08 s	»
#15	ACCEPTED	0.01 s	»
#16	ACCEPTED	0.01 s	»
#17	ACCEPTED	0.01 s	»
#18	ACCEPTED	0.10 s	»
#19	ACCEPTED	0.01 s	»

```cpp
#include <bits/stdc++.h>
using namespace std;
#define endl ("\n")
#define int long long
#define pb push_back
#define mp make_pair
#define ff first
#define ss second
#define all(c) c.begin(), c.end()
#define rep(i,n) for(int i=0;i<n;i++)
#define fast ios_base::sync_with_stdio(false), cin.tie(nullptr), cout.tie(nullptr
#define inf 1e17


int vis[100005];
int dis1[100005],dis2[100005];
void dij(int s, int dis[], vector<pair<int, int>>adj[]){
    priority_queue<pair<int, int>> q;
    rep(i,100005)dis[i+1]=inf;
    dis[s]=0;
    q.push({0,s});
    memset(vis,0,sizeof vis);
    while(!q.empty()){
        int x = q.top().ss; q.pop();
        if (vis[x])continue;
        vis[x]=1;
        for (auto [y,z]: adj[x]){
            if (dis[x]+z<dis[y]){
                dis[y]=dis[x]+z;
                q.push({-dis[y],y});
            }
        }
    }
}

int32_t main(){
    fast
    int t=1;
    // cin>>t;
    while(t--){
        int n,m;cin>>n>>m;
    vector<pair<int, int>>adj1[n+1], adj2[n+1];
    vector<tuple<int,int,int>>e;
    rep(i,m){
        int x, y, z;cin>>x>>y>>z;
        e.pb({x,y,z});
        adj1[x].pb({y,z});
        adj2[y].pb({x,z});
    }
    dij(1,dis1,adj1);
    dij(n,dis2,adj2);
    int ans=inf;
```

```
    dij(n,dis2,adj2);
    int ans=inf;
    for(auto [x,y,z]:e){
        ans=min(ans, dis1[x]+ dis2[y]+ z/2);
    }
    cout<<ans<<endl;
    }
    return 0;
}
```

Ejercicio 05 - Flight Discount

Your task is to find a minimum-price flight route from Syrjälä to Metsälä. You have one discount coupon, using which you can halve the price of any single flight during the route. However, you can only use the coupon once.

Input

The first input line has two integers n and m: the number of cities and flight connections. The cities are numbered 1,2,…,n. City 1 is Syrjälä, and city n is Metsälä.

After this there are m lines describing the flights. Each line has three integers a, b, and c: a flight begins at city a, ends at city b, and its price is c. Each flight is unidirectional.

You can assume that it is always possible to get from Syrjälä to Metsälä.

Output

Print one integer: the price of the cheapest route from Syrjälä to Metsälä.

When you use the discount coupon for a flight whose price is x, its price becomes $\lfloor x/2 \rfloor$ (it is rounded down to an integer).

Constraints

$2 \le n \le 10^5$

$1 \le m \le 2 \cdot 10^5$

$1 \le a,b \le n$

$1 \le c \le 10^9$

Example

Input:

3 4

1 2 3

2 3 1

1 3 7

2 1 5

Output:

2

CSES Problem Set

Longest Flight Route

TASK | SUBMIT | RESULTS | STATISTICS | HACKING

Submission details

Task:	Longest Flight Route
Sender:	Nohelia
Submission time:	2021-12-19 07:49:04
Language:	C++11
Status:	READY
Result:	ACCEPTED

Test results

test	verdict	time	
#1	ACCEPTED	0.01 s	»
#2	ACCEPTED	0.01 s	»
#3	ACCEPTED	0.01 s	»
#4	ACCEPTED	0.01 s	»
#5	ACCEPTED	0.01 s	»
#6	ACCEPTED	0.08 s	»
#7	ACCEPTED	0.09 s	»
#8	ACCEPTED	0.09 s	»
#9	ACCEPTED	0.09 s	»
#10	ACCEPTED	0.09 s	»
#11	ACCEPTED	0.06 s	»
#12	ACCEPTED	0.08 s	»
#13	ACCEPTED	0.01 s	»
#14	ACCEPTED	0.01 s	»
#15	ACCEPTED	0.07 s	»
#16	ACCEPTED	0.01 s	»
#17	ACCEPTED	0.06 s	»
#18	ACCEPTED	0.04 s	»
#19	ACCEPTED	0.01 s	»
#20	ACCEPTED	0.05 s	»

```cpp
#include <bits/stdc++.h>

using namespace std;
const int maxN = 1e5+1;

bool vis[maxN];
int N, M, K, a, b, in[maxN], p[maxN], l[maxN], ans[maxN];
vector<int> G[maxN];
queue<int> Q;

void dfs(int u, int par = 0){
    vis[u] = true;
    for(int v : G[u])
        if(v != par && !vis[v])
            dfs(v, u);
}

int main(){
    scanf("%d %d", &N, &M);
    for(int i = 0; i < M; i++){
        scanf("%d %d", &a, &b);
        G[a].push_back(b);
        in[b]++;
    }

    dfs(1);
    if(!vis[N]){
        printf("IMPOSSIBLE\n");
        return 0;
    }

    fill(l+2, l+maxN, -1);
    for(int i = 1; i <= N; i++)
        if(in[i] == 0)
            Q.push(i);

    while(!Q.empty()){
        int u = Q.front(); Q.pop();
        for(int v : G[u]){
            if(l[u] != -1 && l[v] < l[u]+1){
                l[v] = l[u] + 1;
                p[v] = u;
            }
            in[v]--;
            if(in[v] == 0)
                Q.push(v);
        }
    }

    K = l[N] - l[1];
    printf("%d\n", K+1);
    for(int i = K, u = N; i >= 0; i--){
        ans[i] = u;
        u = p[u];
    }
    for(int i = 0; i <= K; i++)
        printf("%d%c", ans[i], (" \n")[i==K]);
}
```