MST - Minimum Spanning Tree

[#mst](https://www.spoj.com/problems/tag/mst)

Find the minimum spanning tree of the graph.

Input

On the first line there will be two integers N - the number of nodes and M - the number of edges. (1 <= N <= 10000), (1 <= M <= 100000)  
M lines follow with three integers i j k on each line representing an edge between node i and j with weight k. The IDs of the nodes are between 1 and n inclusive. The weight of each edge will be <= 1000000.

Output

Single number representing the total weight of the minimum spanning tree on this graph. There will be only one possible MST.

Example

**Input:**

4 5

1 2 10

2 3 15

1 3 5

4 2 2

4 3 40

**Output:**

17

Prims algorithm:

#include<bits/stdc++.h>

#define int long long int

#define pii pair<int,int>

#define pb push\_back

using namespace std;

struct ss{

int u,v,w;

};

struct comp{

bool operator()(ss const& p1,ss const& p2)

{

return p1.w > p2.w;

}

};

int n;

vector<pii> adj[10001];

int cnt[10001];

vector<ss> mst;

void prims()

{

ss a;

priority\_queue<ss,vector<ss>,comp> pq;

for(auto child: adj[1])

{

a.u=1,a.v=child.first,a.w=child.second;

pq.push(a);

}

cnt[1]=1;

while(!pq.empty())

{

a=pq.top();

pq.pop();

if(cnt[a.v])

continue;

cnt[a.v]=1;

mst.pb(a);

if(mst.size()==n-1)

return;

for(auto child: adj[a.v])

{

ss b,c;

b.u=a.v,b.v=child.first,b.w=child.second;

if(!cnt[child.first])

pq.push(b);

}

}

}

main()

{

int m;

cin>>n>>m;

for(int i=0;i<m;i++)

{

int u,v,w;

cin>>u>>v>>w;

adj[u].pb({v,w});

adj[v].pb({u,w});

}

prims();

int ans=0;

for(auto it:mst)

{

ans+=it.w;

}

cout<<ans<<"\n";

}