**Roads and Libraries**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

class Solution {

public static void Main(String[] args) {

//Scanner in = new Scanner(System.in);

//int q = in.nextInt();

int q = int.Parse(Console.ReadLine());

for(int a0 = 0; a0 < q; a0++){

// int n = in.nextInt();

string[] tokens\_n = Console.ReadLine().Split(' ');

int n = Convert.ToInt32(tokens\_n[0]);

int m = Convert.ToInt32(tokens\_n[1]);

long clib = Convert.ToInt64(tokens\_n[2]);

long croad = Convert.ToInt64(tokens\_n[3]);

Graph g=new Graph(n,false);

//int m = in.nextInt();

//int clib = in.nextInt();

//int croad = in.nextInt();

for(int a1 = 0; a1 < m; a1++){

//int city\_1 = in.nextInt();

//int city\_2 = in.nextInt();

string[] tokens\_city\_1 = Console.ReadLine().Split(' ');

int city\_1 = Convert.ToInt32(tokens\_city\_1[0]);

int city\_2 = Convert.ToInt32(tokens\_city\_1[1]);

g.insertEdge(city\_1-1,city\_2-1);

}

Console.WriteLine(getCost(g,clib,croad));

}

}

private static long getCost(Graph g, long clib, long croad) {

if (clib < croad) {

return clib \* g.v;

}

bool[] visited = new bool[g.v];

long cost = 0;

for (int i = 0; i < g.v; i++) {

if (!visited[i]) {

long x = DFSUtil(g, visited, i);

cost += (x - 1) \* croad;

cost += clib;

}

}

return cost;

}

private static int DFSUtil(Graph g,bool[] visited,int current){

int count=1;

visited[current]=true;

foreach (int a in g.list[current]){

if(!visited[a]){

count+=DFSUtil(g,visited,a);

}

}

return count;

}

}

class Graph

{

public int v;

public List<int>[] list;

private bool directed;

/\*\* Constructor of Graph, takes no. of vertex and boolean value to specify whether graph is directed or not \*/

public Graph(int v, bool directed)

{

this.v = v;

this.directed = directed;

list = new List<int>[v];

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

{

list[i] = new List<int>();

}

}

/\* vertex numbering start with 0 as well as adjacency list no. start with 0 \*/

public void insertEdge(int a, int b)

{

list[a].Add(b);

if (!directed)

{

list[b].Add(a);

}

}

}