7-2　畅通工程之最低成本建设问题　（30　分）

某地区经过对城镇交通状况的调查，得到现有城镇间快速道路的统计数据，并提出“畅通工程”的目标：使整个地区任何两个城镇间都可以实现快速交通（但不一定有直接的快速道路相连，只要互相间接通过快速路可达即可）。现得到城镇道路统计表，表中列出了有可能建设成快速路的若干条道路的成本，求畅通工程需要的最低成本。

输入格式:

输入的第一行给出城镇数目N （$1<N\leq1000$）和候选道路数目$M\leq3N$；随后的M行，每行给出3个正整数，分别是该条道路直接连通的两个城镇的编号（从1编号到N）以及该道路改建的预算成本。

输出格式:

输出畅通工程需要的最低成本。如果输入数据不足以保证畅通，则输出“Impossible”。

输入样例1:

```
6 15
1 2 5
1 3 3
1 4 7
1 5 4
1 6 2
2 3 4
2 4 6
2 5 2
2 6 6
3 4 6
3 5 1
3 6 1
4 5 10
4 6 8
5 6 3
```

输出样例1:

```
12
```

输入样例2:

```
5 4
1 2 1
2 3 2
3 1 3
4 5 4
```

输出样例2:

```
Impossible
```

```
#include<stdio.h>

#define MAX 100
```

```
#define MAXLEN 555555


typedef struct{
    int vernum,arcnum;
    int arcs[MAX][MAX];
}Graph;

int init(Graph *graph);

int init(Graph *graph){
    int vernum,arcnum;
    scanf("%d %d",&vernum,&arcnum);
    graph->vernum=vernum;
    graph->arcnum=arcnum;
    for(int i=1;i<vernum+1;i++){
        for(int j=1;j<vernum+1;j++){
            graph->arcs[i][j]=MAXLEN;
        }
        graph->arcs[i][i]=0;
    }

    int start,end,fare;
    for(int i=1;i<arcnum+1;i++){
        scanf("%d %d %d",&start,&end,&fare);
        graph->arcs[start][end]=fare;
        graph->arcs[end][start]=fare;
    }

}

typedef struct{
    int adjver[MAX];
    int fare[MAX];
}Help;

int prim(Graph graph){
    Help help;
    for(int i=1;i<graph.vernum+1;i++){
        help.fare[i]=graph.arcs[1][i];
        help.adjver[i]=1;
    }
    int count=0;

    for(int i=1;i<graph.vernum+1;i++){
        int min=MAXLEN;
        int current;
```

```
        for(int j=1;j<graph.vernum+1;j++){
            if(min>help.fare[j]&&help.fare[j]!=0){
                min=help.fare[j];
                current=j;
            }
        }


        if(min!=MAXLEN){
        count+=min;
        }
        help.fare[current]=0;

        for(int j=1;j<graph.vernum+1;j++){
            if(help.fare[j]>graph.arcs[current][j]){
                help.fare[j]=graph.arcs[current][j];
                help.adjver[j]=current;
            }
        }
    }
    return count;
}

int BFS(Graph *graph){
    int queue[graph->vernum];
    int dear,top;
    dear=top=0;
    int count=0;
    int visited[graph->vernum+1];
    for(int i=1;i<graph->vernum+1;i++){
        visited[i]=0;
    }
    queue[top++]=1;
    visited[1]=1;
    count++;
    while(dear!=top){
        int temp=queue[dear++];
        for(int i=1;i<graph->vernum+1;i++){
            if(visited[i]==0&&graph->arcs[temp][i]!=MAXLEN){
                queue[top++]=i;
                visited[i]=1;
                count++;
            }
        }
    }
    return count;
}
```

```
int main(){
	Graph graph;
	init(&graph);
	int count=prim(graph);
	int count1=BFS(&graph);
	if(count1!=graph.vernum){
		printf("Impossible");
	}else
	printf("%d",count);
}
```