車同軌

問題敘述

這是小高與他的好朋友，秦始皇：

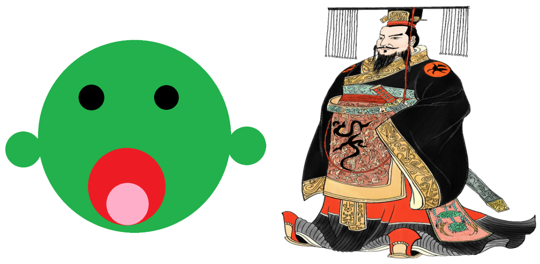

Figure 1: 小高與秦始皇

眾所周知秦始皇平時喜歡騎北極熊遊歷各地：

Figure 2: 秦始皇騎北極熊

為了能夠更高效率的移動，秦始皇想要在各個城市之間建造道路，但礙於大部分的資源都拿去修長城了，所以需要更謹慎的規劃資源的消耗。

這時候，他想起了他的好朋友小高就讀的是國立成功大學資訊工程學系，於是他便找上小高，希望小高能夠幫他解決這個問題。

小高作為秦始皇的好朋友當然是義不容辭，於是便接下了這個任務。

然而，就在小高將秦國的地理資訊導入系統之時，突然想起了一陣急促的敲門聲。

.

.

.

還不等小高起身，那扇門便被砸開了，而在門外站著的人，是服學阿姨。

原來，小高在過去兩年間，喜歡在打掃時間跑回宿舍大叫，導致他的服學（一）及服學（二）都被當掉了，即使在重修之後仍沒有通過因而引起了服學阿姨的注意。

小高本想著，明年重修就好，但沒想到未來的新生不用再修服學了，這也就代表小高喪失了重修服學的機會，因此才會被服學阿姨找上門來。

在這般情況下，小高自然沒有辦法繼續幫秦始皇規劃道路，於是在他臨終走前，他找上了你，並希望你能完成他的遺願程式。

小高告訴你，他已經完成了這支程式的大部分內容，也就是輸入與輸出，你只需要完成他剩下來的部分即可。

簡單來說

這邊是定義：

```c
// mst.h
#define MAX_NODES 100000
#define MAX_EDGES 1200000

typedef struct Edge {
    struct Node *u;
    struct Node *v;
    int w;
    char keep;
    void *var;
} Edge;

typedef struct Node {
    int id;
    int edge_count;
    struct Edge **edges;
    void *var;
} Node;

void generate_mst(Node *node);
```

你需要實作一個函式如下：

```c
#include "mst.h"

void generate_mst(Node *node) {};
```

傳入的 Node 會是整張圖中的其中一個節點，Node 中所附帶的參數說明如下：

- id: 節點的 ID，一個 ID 只會對應到一個節點，不會重複
- edge_count: 該節點所擁有的邊的數量
- edges: 一維的陣列，裡面儲存了該節點所擁有的邊
- var: 一個指針，可以任意使用，如果有必要，你可以自行定義結構 (struct) 並在運行途中進行轉型 (Type Casting)，來儲存任何你想要儲存的數據，會先初始化為 NULL

Edge 的參數說明如下：

- u, v: 路徑兩端的 Node
- w: 權重
- keep: 是否要保留該條路徑，會初始化為 0
- var: 同 Node 中的 var

總之，你只會得到一個起始節點，然後想辦法用最少的成本將所有節點相連，並且將所需要的路徑透過將 keep 設為 1 進行標記。

輸入說明

不斷輸入直至 EOF 為止。

每行有三個正整數 u, v, w，表示在節點 u 與節點 v 之間有一條權重為 w 的無向邊。

共有 n 個節點與 m 條邊。

輸出說明

共三個正整數，分別代表最小生成樹的節點數、邊數以及權重和。

測資限制

- $8 \leq n \leq 10^5$
- $7 \leq m \leq 1.2 \times 10^6$
- $1 \leq u, v < n$
- $1 \leq w \leq m \times 0.4$

如何繳交

請將函式獨立於一個檔案中，並且**一定要包含** `#include "mst.h"`，繳交至 Judge 時，只需要上傳該檔案即可。

例如，這是你的資料夾：

```
├ main.c
├ mst.h
└ hw.c // 或其他你自己命名的名字
```

你需要上傳的只有 `hw.c`，該檔案的內容大致如下:

```c
// hw.c
#include "mst.h"     // 必須要有這行
#include <stdlib.h> // 其他你用到的函式庫
.
.
.
void generate_mst(Node *node) {
    // 在這裡實作你的程式
}
```

檔案中可以包含你其他會用到的東西，如 struct、function 之類的。

請注意，**不要修改** `main.c` 及 `mst.h` 的內容，這兩個檔案會在評分時**由 Judge 系統生成**，請不要在你的電腦上改完之後又來問：**為什麼在 Judge 上會 Compile Error**。

範例測資

範例輸入 1

```
0 1 1
0 4 4
0 5 4
1 3 3
1 5 4
1 6 3
2 3 4
2 4 5
2 7 2
3 6 1
4 7 2
5 6 4
5 7 1
6 7 1
```

範例輸出 1

```
8 7 11
```

範例說明 1

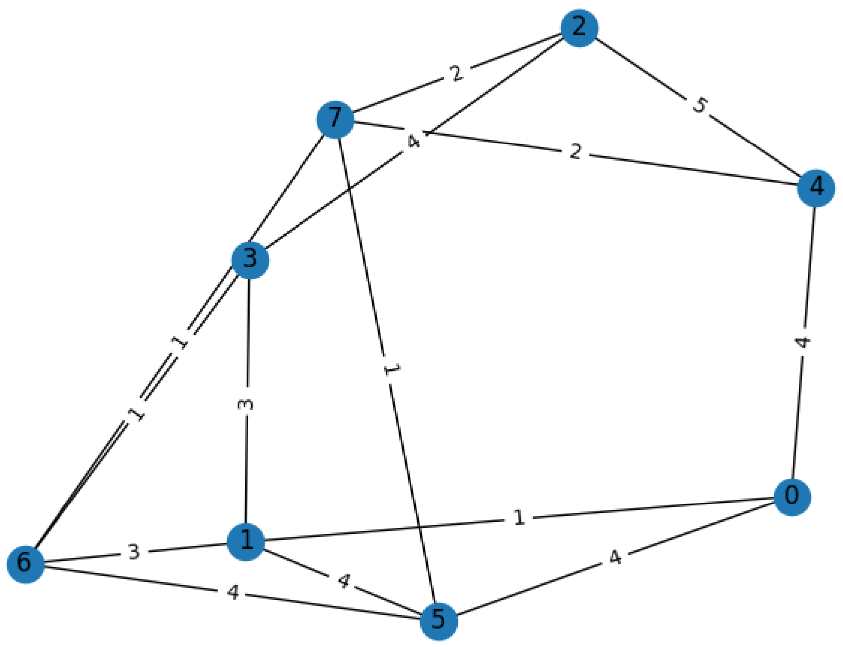

Figure 3: 範例輸入 1

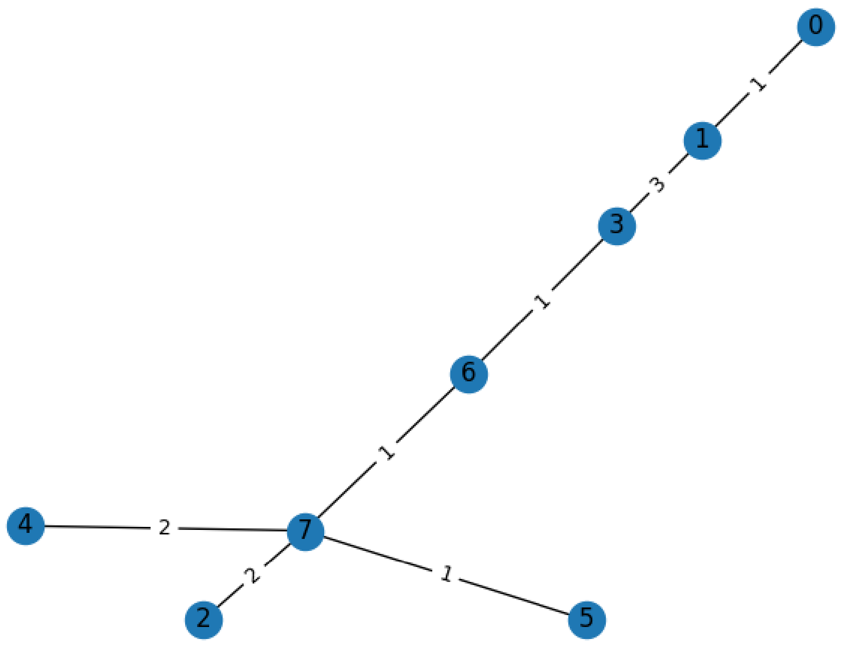

Figure 4: 範例輸出 1

附件

View On Github Gist

A. mst.h

```c
// mst.h
#define MAX_NODES 100000
#define MAX_EDGES 1200000

typedef struct Edge {
    struct Node *u;
    struct Node *v;
    int w;
    char keep;
    void *var;
} Edge;

typedef struct Node {
    int id;
    int edge_count;
    struct Edge **edges;
    void *var;
} Node;

void generate_mst(Node *node);
```

B. main.c

```c
// main.c
#include "mst.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    int u, v, w, edge_count = 0;
    Node *node_u, *node_v;
    int *discovered = (int *)malloc(sizeof(int) * MAX_NODES);
    Node **nodes = (Node **)malloc(sizeof(Node *) * MAX_NODES);
    Edge **edges = (Edge **)malloc(sizeof(Edge *) * MAX_EDGES);

    memset(discovered, 0, sizeof(int) * MAX_NODES);
    memset(nodes, 0, sizeof(Node *) * MAX_NODES);
    memset(edges, 0, sizeof(Edge *) * MAX_EDGES);

    while (scanf("%d%d%d", &u, &v, &w) == 3) {
        node_u = nodes[u];
        if (node_u == NULL) {
            node_u = (Node *)malloc(sizeof(Node));
            node_u->id = u;
            node_u->edge_count = 0;
            node_u->edges = NULL;
            nodes[u] = node_u;

            node_u->var = NULL;
        }
        node_v = nodes[v];
        if (node_v == NULL) {
            node_v = (Node *)malloc(sizeof(Node));
            node_v->id = v;
            node_v->edge_count = 0;
            node_v->edges = NULL;
            nodes[v] = node_v;

            node_v->var = NULL;
        }

        node_u->edge_count++;
        node_v->edge_count++;

        Edge *new_edge = (Edge *)malloc(sizeof(Edge));
        new_edge->u = node_u;
        new_edge->v = node_v;
        new_edge->w = w;
        new_edge->keep = 0;
        new_edge->var = NULL;

        edges[edge_count++] = new_edge;
    }

    for (int i = 0; i < edge_count; i++) {
        Edge *edge = edges[i];
        node_u = edge->u;
        node_v = edge->v;

        if (node_u->edges == NULL) {
            node_u->edges =
                (Edge **)malloc(sizeof(Edge *) * node_u->edge_count);
            node_u->edge_count = 0;
        }
        node_u->edges[node_u->edge_count++] = edge;

        if (node_v->edges == NULL) {
            node_v->edges =
                (Edge **)malloc(sizeof(Edge *) * node_v->edge_count);
            node_v->edge_count = 0;
        }
        node_v->edges[node_v->edge_count++] = edge;
    }

    generate_mst(nodes[0]);

    int node_count = 0;
    int keep_edge_count = 0;
    unsigned long long total_weight = 0;
    for (int i = 0; i < edge_count; i++) {
        Edge *edge = edges[i];
        if (edge->keep) {
            total_weight += edge->w;
            keep_edge_count++;
```

```c
            u = edge->u->id;
            v = edge->v->id;
            if (!discovered[u]) {
                discovered[u] = 1;
                node_count++;
            }
            if (!discovered[v]) {
                discovered[v] = 1;
                node_count++;
            }
        }
    }
    printf("%d %d %llu\n", node_count, keep_edge_count, total_weight);
}
```

Unified Vehicle Tracks

Problem Description

This is Xiao Gao and his good friend, Qin Shi Huang:

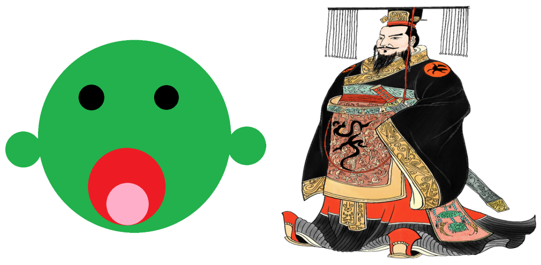

Figure 5: Xiao Gao and Qin Shi Huang

As is well known, Qin Shi Huang usually likes to ride polar bears to travel around:

Figure 6: Qin Shi Huang Riding a Polar Bear

In order to be able to move more efficiently, Qin Shi Huang wants to build roads between various cities, but because most resources are used to build the Great Wall, it is necessary to plan resource consumption more carefully.

At this time, he remembered that his good friend Xiao Gao is studying in the Department of Computer Science and Information Engineering at National Cheng Kung University, so he approached Xiao Gao, hoping that Xiao Gao could help him solve this problem.

As Qin Shi Huang's good friend, Xiao Gao naturally took on the task without hesitation.

However, just as Xiao Gao was importing the geographical information of the Qin state into the system, he suddenly heard a rapid knocking sound.

.

.

.

Before Xiao Gao could get up, the door was smashed open, and the person standing outside was the Service Study Auntie.

It turned out that over the past two years, Xiao Gao liked to run back to his dormitory and shout during cleaning time, causing him to fail both Service Study (I) and Service Study (II). Even after retaking the courses, he still did not pass, which attracted the attention of the Service Study Auntie.

Xiao Gao originally thought that he could retake the course next year, but he did not expect that future freshmen would no longer need to take Service Study. This meant that Xiao Gao lost the opportunity to retake Service Study, which is why the Service Study Auntie came to his door.

In this situation, Xiao Gao naturally could not continue to help Qin Shi Huang plan the roads. So before he left, he found you and hoped that you could fulfill his last wish program.

Xiao Gao told you that he has already completed most of the content of this program, which is the input and output. You only need to complete the remaining part.

In simple terms

Here are the definitions:

```c
// mst.h
#define MAX_NODES 100000
#define MAX_EDGES 1200000

typedef struct Edge {
    struct Node *u;
    struct Node *v;
    int w;
    char keep;
    void *var;
} Edge;
```

```
typedef struct Node {
    int id;
    int edge_count;
    struct Edge **edges;
    void *var;
} Node;

void generate_mst(Node *node);
```

You need to implement a function as follows:

```
#include "mst.h"

void generate_mst(Node *node) {};
```

The passed `Node` will be one of the nodes in the entire graph. The parameters attached to the `Node` are explained as follows:

- `id`: The `ID` of the node. Each `ID` corresponds to only one node and will not be duplicated.
- `edge_count`: The number of edges this node has.
- `edges`: A one-dimensional array storing the edges this node has.
- `var`: A pointer that can be used arbitrarily. If necessary, you can define your own structure and perform type casting during runtime to store any data you want. It will be initialized to `NULL`.

The parameters of `Edge` are explained as follows:

- `u`, `v`: The `Node`s at both ends of the path.
- `w`: The weight.
- `keep`: Whether to keep this path. It will be initialized to 0.
- `var`: Same as `var` in `Node`.

In short, you will only be given a starting node. Then, you need to connect all nodes with the minimum cost and mark the required paths by setting `keep` to 1.

Input Format

Keep reading input until EOF.

Each line has three positive integers u, v, w, indicating an undirected edge with weight w between node u and node v.

There are n nodes and m edges.

Output Format

Three positive integers, representing the number of nodes, the number of edges, and the total weight of the minimum spanning tree, respectively.

Constraints

- $8 \leq n \leq 10^5$
- $7 \leq m \leq 1.2 \times 10^6$
- $1 \leq u, v < n$
- $1 \leq w \leq m \times 0.4$

How to Submit

Please put the function in a separate file and **must include** `#include "mst.h"`. When submitting to Judge, only upload this file.

For example, this is your folder:

```
├─ main.c
├─ mst.h
└─ hw.c // or any other name you choose
```

You only need to upload `hw.c`. The content of this file should roughly be:

```c
// hw.c
#include "mst.h"     // This line must be included
#include <stdlib.h> // Other libraries you use
.
.
.
void generate_mst(Node *node) {
    // Implement your program here
}
```

The file can include other things you need, such as structs, functions, etc.

Please note, **do not modify** the content of `main.c` and `mst.h`. These two files will be **generated by the Judge system** during grading. Do not modify them on your computer and then ask: **Why does it show Compile Error on Judge?**.

Example Test Case

Sample Input 1

```
0 1 1
0 4 4
0 5 4
1 3 3
1 5 4
1 6 3
2 3 4
2 4 5
2 7 2
3 6 1
4 7 2
5 6 4
5 7 1
6 7 1
```

Sample Output 1

```
8 7 11
```

Explanation of Sample 1

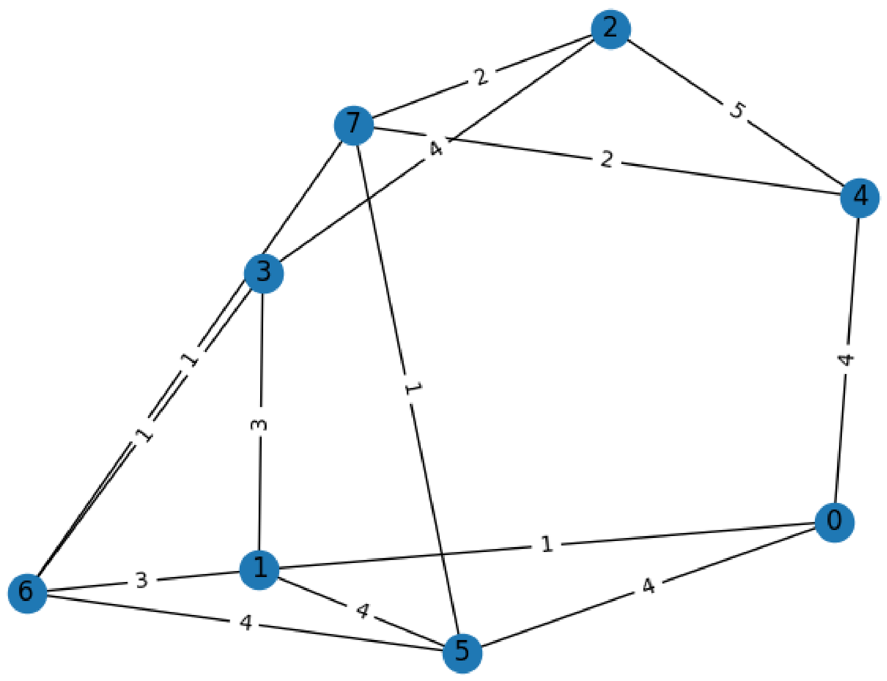

Figure 7: Sample Input 1

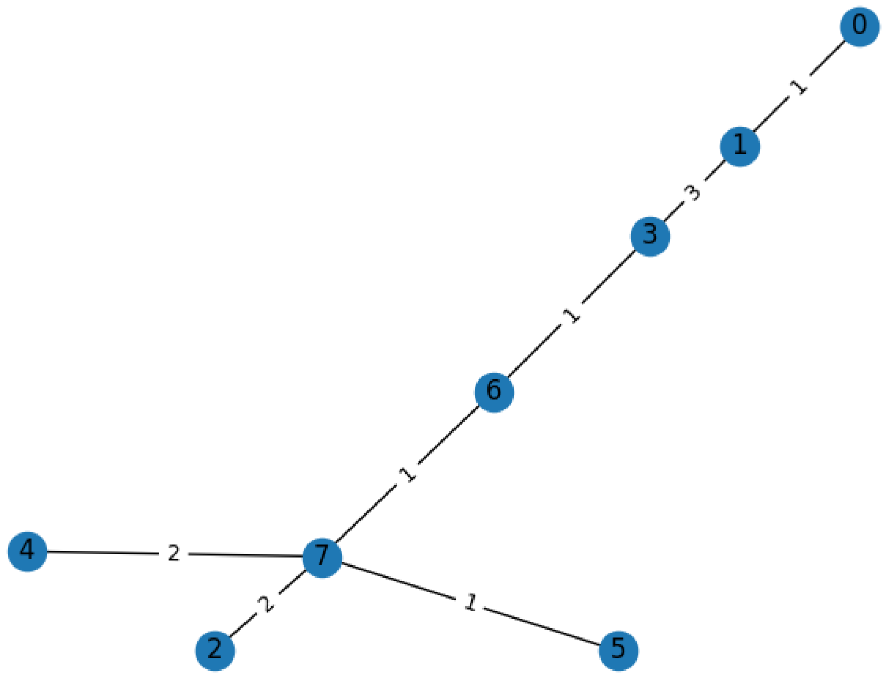

Figure 8: Sample Output 1

Appendix

View On Github Gist

A. mst.h

```c
// mst.h
#define MAX_NODES 100000
#define MAX_EDGES 1200000

typedef struct Edge {
    struct Node *u;
    struct Node *v;
    int w;
    char keep;
    void *var;
} Edge;

typedef struct Node {
    int id;
    int edge_count;
    struct Edge **edges;
    void *var;
} Node;

void generate_mst(Node *node);
```

B. main.c

```c
// main.c
#include "mst.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
    int u, v, w, edge_count = 0;
    Node *node_u, *node_v;
    int *discovered = (int *)malloc(sizeof(int) * MAX_NODES);
    Node **nodes = (Node **)malloc(sizeof(Node *) * MAX_NODES);
    Edge **edges = (Edge **)malloc(sizeof(Edge *) * MAX_EDGES);

    memset(discovered, 0, sizeof(int) * MAX_NODES);
    memset(nodes, 0, sizeof(Node *) * MAX_NODES);
    memset(edges, 0, sizeof(Edge *) * MAX_EDGES);

    while (scanf("%d%d%d", &u, &v, &w) == 3) {
        node_u = nodes[u];
        if (node_u == NULL) {
            node_u = (Node *)malloc(sizeof(Node));
            node_u->id = u;
            node_u->edge_count = 0;
            node_u->edges = NULL;
            nodes[u] = node_u;

            node_u->var = NULL;
        }
        node_v = nodes[v];
        if (node_v == NULL) {
            node_v = (Node *)malloc(sizeof(Node));
            node_v->id = v;
            node_v->edge_count = 0;
            node_v->edges = NULL;
            nodes[v] = node_v;

            node_v->var = NULL;
        }

        node_u->edge_count++;
        node_v->edge_count++;

        Edge *new_edge = (Edge *)malloc(sizeof(Edge));
        new_edge->u = node_u;
        new_edge->v = node_v;
        new_edge->w = w;
        new_edge->keep = 0;
        new_edge->var = NULL;

        edges[edge_count++] = new_edge;
    }

    for (int i = 0; i < edge_count; i++) {
        Edge *edge = edges[i];
        node_u = edge->u;
        node_v = edge->v;

        if (node_u->edges == NULL) {
            node_u->edges =
                (Edge **)malloc(sizeof(Edge *) * node_u->edge_count);
            node_u->edge_count = 0;
        }
        node_u->edges[node_u->edge_count++] = edge;

        if (node_v->edges == NULL) {
            node_v->edges =
                (Edge **)malloc(sizeof(Edge *) * node_v->edge_count);
            node_v->edge_count = 0;
        }
        node_v->edges[node_v->edge_count++] = edge;
    }

    generate_mst(nodes[0]);

    int node_count = 0;
    int keep_edge_count = 0;
    unsigned long long total_weight = 0;
    for (int i = 0; i < edge_count; i++) {
        Edge *edge = edges[i];
        if (edge->keep) {
            total_weight += edge->w;
            keep_edge_count++;
            u = edge->u->id;
            v = edge->v->id;
            if (!discovered[u]) {
                discovered[u] = 1;
                node_count++;
            }
            if (!discovered[v]) {
                discovered[v] = 1;
                node_count++;
            }
        }
    }
    printf("%d %d %llu\n", node_count, keep_edge_count, total_weight);
}
```