青少年软件编程（C 语言）等级考试试卷（八级）

1 旅游规划

有了一张自驾旅游路线图，你会知道城市间的高速公路长度、以及该公路要收取的过路费。现在需要你写一个程序，帮助前来咨询的游客找一条出发地和目的地之间的最短路径。如果有若干条路径都是最短的，那么需要输出最便宜的一条路径。

时间限制：1000
内存限制：65536

输入
输入说明：输入数据的第 1 行给出 4 个正整数 n、m、s、d，其中 n（2 ≤ n ≤ 500）是城市的个数，顺便假设城市的编号为 0~(n-1)；m 是高速公路的条数；s 是出发地的城市编号；d 是目的地的城市编号。随后的 m 行中，每行给出一条高速公路的信息，分别是：城市 1、城市 2、高速公路长度、收费额，中间用空格分开，数字均为整数且不超过 500。输入保证解的存在。

输出
在一行里输出路径的长度和收费总额，数字间以空格分隔，输出结尾不能有多余空格。

样例输入

```
4 5 0 3
0 1 1 20
1 3 2 30
0 3 4 10
0 2 2 20
2 3 1 20
```

样例输出

```
3 40
```

参考代码：

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

struct Edge { int to, dist, cost; };

int main() {
```

```
    int n, m, s, d; cin >> n >> m >> s >> d;
    vector<vector<Edge>> g(n);
    for (int i = 0, a, b, c, f; i < m; ++i) {
        cin >> a >> b >> c >> f;
        g[a].push_back({b, c, f});
        g[b].push_back({a, c, f});
    }
    vector<int> dist(n, INT_MAX), fee(n, INT_MAX);
    priority_queue<tuple<int, int, int>, vector<tuple<int, int, int>>, greater<>> pq;
    dist[s] = 0, fee[s] = 0; pq.emplace(0,  0, s);
    while (!pq.empty())  {
        auto [d, f, u] = pq.top();  pq.pop();
        if (d > dist[u] || (d == dist[u] && f > fee[u])) continue;
        for (auto &[v, dd, ff] : g[u]) {
            if (d + dd < dist[v] || (d + dd == dist[v] && f + ff < fee[v])) {
                dist[v] = d + dd, fee[v] = f + ff;
                pq.emplace(dist[v],  fee[v], v);
            }
        }
    }
    cout << dist[d] << " " << fee[d];
}
```

2 取帽子

拼题 er 们觉得戴帽子会令自己看上去很帅，所以他们不管到哪里都会戴着帽子。有一天他们去到一家餐厅，服务员把他们的帽子收集了堆起来保管。当大家要离开的时候，发

现帽子被像上图那样摞起来了。于是你的任务就是帮他们排好队，使得每个人都能按顺序顺利取到自己的帽子。

已知每顶帽子的大小都不相同，并且帽子的尺寸跟帽子主人的体重有关 —— 越重的人戴的帽子就越大。

时间限制：1000
内存限制：65536

输入
输入第一行给出一个正整数 n (≤ 10^4)，为拼题 er 的人数。随后一行给出 n 个不同的帽子尺寸，为不超过 10^5 的正整数，顺序是从帽子堆的底部向上给出。最后一行给出 n 个不同的体重，顺序对应编号从 1 到 n 的拼题 er。体重是不超过 10^6 的正整数。一行中的数字以空格分隔。

输出
在一行中按照取帽子的顺序输出帽子主人的编号。数字间以 1 个空格分隔，行首尾不得有多余空格。

样例输入

```
10
12 19 13 11 15 18 17 14 16 20
67 90 180 98 87 105 76 88 150 124
```

样例输出

```
3 4 8 6 10 2 1 5 9 7
```

提示
样例说明： 第一顶帽子的尺寸是最大的 20，所以对应第 3 个人的最大体重 180，于是第 3 个人排在最前面。 第二顶帽子的尺寸是第 6 小的 16，对应第 6 小的体重 98，是第 4 个人，于是第 4 个人下一个走。 以此类推。

参考代码：

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

int main() {
    int n; cin >> n;
    vector<int> hats(n), weights(n);
    for (auto &x : hats) cin >> x;
```

```
    for (auto &x : weights) cin >> x;
    vector<pair<int, int>> map(n);
    for (int i = 0; i < n; ++i) map[i] = {hats[i], weights[i]};
    sort(map.begin(),  map.end(),  [&](auto &a, auto &b) {
        return a.first  != b.first  ? a.first  > b.first  : a.second  > b.second;
    });
    vector<int> res;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (weights[j] == map[i].second) {
                res.push_back(j  + 1);
                weights[j] = -1;
                break;
            }
        }
    }
    for (int i = 0; i < n; ++i) cout << res[i] << (i < n - 1 ? ' ' : '\n');
}
```

3 最短工期

一个项目由若干个任务组成，任务之间有先后依赖顺序。项目经理需要设置一系列里程碑，在每个里程碑节点处检查任务的完成情况，并启动后续的任务。现给定一个项目中各个任务之间的关系，请你计算出这个项目的最早完工时间。

时间限制：1000
内存限制：65536

输入
首先第一行给出两个正整数：项目里程碑的数量 N（≤ 100）和任务总数 M。这里的里程碑从 0 到 N-1 编号。随后 M 行，每行给出一项任务的描述，格式为“任务起始里程碑 任务结束里程碑 工作时长”，三个数字均为非负整数，以空格分隔。

输出
如果整个项目的安排是合理可行的，在一行中输出最早完工时间；否则输出"Impossible"。

样例输入

```
样例#1：

9 12

0 1 6
```

```
0 2 4
0 3 5
1 4 1
2 4 1
3 5 2
5 4 0
4 6 9
4 7 7
5 7 4
6 8 2
7 8 4


样例#2：
4 5
0 1 1
0 2 2
2 1 3
1 3 4
3 2 5
```

样例输出

```
样例#1：
18


样例#2：
Impossible
```

参考代码:

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

int main() {
    int N, M; cin >> N >> M;
```

```
    vector<vector<pair<int, int>>> g(N);
    vector<int> in(N, 0), time(N, 0);
    for (int i = 0, a, b, t; i < M; ++i) {
        cin >> a >> b >> t;
        g[a].emplace_back(b, t);
        in[b]++;
    }
    queue<int> q;
    for (int i = 0; i < N; ++i) if (!in[i]) q.push(i);
    while (!q.empty())  {
        int u = q.front();  q.pop();
        for (auto &[v, t] : g[u]) {
            time[v] = max(time[v], time[u] + t);
            if (--in[v] == 0) q.push(v);
        }
    }
    int res = *max_element(time.begin(),  time.end());
    cout << (res ? res : "Impossible");
}
```

4 城市间紧急救援

作为一个城市的应急救援队伍的负责人，你有一张特殊的全国地图。在地图上显示有多个分散的城市和一些连接城市的快速道路。每个城市的救援队数量和每一条连接两个城市的快速道路长度都标在地图上。当其他城市有紧急求助电话给你的时候，你的任务是带领你的救援队尽快赶往事发地，同时，一路上召集尽可能多的救援队。

时间限制：1000
内存限制：65536

输入
输入第一行给出 4 个正整数 n、m、s、d，其中 n（2 ≤ n ≤ 500）是城市的个数，顺便假设城市的编号为 0 ~ (n-1)；m 是快速道路的条数；s 是出发地的城市编号；d 是目的地的城市编号。 第二行给出 n 个正整数，其中第 i 个数是第 i 个城市的救援队的数目，数字间以空格分隔。随后的 m 行中，每行给出一条快速道路的信息，分别是：城市 1、城市 2、快速道路的长度，中间用空格分开，数字均为整数且不超过 500。输入保证救援可行且最优解唯一。

输出
第一行输出最短路径的条数和能够召集的最多的救援队数量。第二行输出从 s 到 d 的路径中经过的城市编号。数字间以空格分隔。

样例输入

```
4 5 0 3
20 30 40 10
0 1 1
1 3 2
0 3 3
0 2 2
2 3 2
```

样例输出

```
2 60
0 1 3
```

参考代码：

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

struct Edge { int to, dist; };

int main() {
    int n, m, s, d; cin >> n >> m >> s >> d;
    vector<int> rescue(n);
    for (auto &x : rescue) cin >> x;
    vector<vector<Edge>> g(n);
    for (int i = 0, a, b, c; i < m; ++i) {
        cin >> a >> b >> c;
        g[a].push_back({b, c});
        g[b].push_back({a, c});
    }
    vector<int> dist(n, INT_MAX), resc(n, 0), prev(n, -1);
        priority_queue<tuple<int, int, int>, vector<tuple<int, int, int>>,
greater<>> pq;
    dist[s] = 0, resc[s] = rescue[s]; pq.emplace(0, -resc[s], s);
    while (!pq.empty()) {
        auto [d, r, u] = pq.top(); pq.pop(); r = -r;
        if (d > dist[u] || (d == dist[u] && r < resc[u])) continue;
        for (auto &[v, w] : g[u]) {
            if (d + w < dist[v] || (d + w == dist[v] && r + rescue[v] > resc[v])) {
                dist[v] = d + w, resc[v] = r + rescue[v], prev[v] = u;
                pq.emplace(dist[v], -resc[v], v);
```

```
            }
        }
    }
    cout << dist[d] << " " << resc[d] << "\n";
    vector<int> path;
    for (int u = d; u != -1; u = prev[u]) path.push_back(u);
    reverse(path.begin(),  path.end());
    for (int i = 0; i < path.size();  ++i) cout << path[i] << (i < path.size()  - 1 ?
' ' : '\n');
}
```