题目：随机游走

Contents

SE@SJTU 2023

1 简介与背景

假设你是一名很喜欢做研究的出租车司机，在搭载乘客之时总喜欢研究一些看起来稀奇古怪的问题。最近，你听说模拟随机游走可以用来评估出租车在城市中的移动性，于是你决定把随机游走应用到接送乘客的工作中。

2 题目要求

2.1 [步骤 1] 构建道路

函数名：本步骤需要实现在一个名为 `BuildGraph` 的函数中。

一个城市的道路网络可以表示为一个有向图 G，其中节点表示道路交叉口，边表示道路，边的权重表示道路的长度。

标准输入：

你需要从标准输入中读取一个保存了道路等信息的文件的路径名（中间无空格）。

文件输入：

标准输入指定的文件中，保存了道路等信息，具体如下。第一行文本为空格隔开的四个正整数 N、M、Q 和 P，其中 N 和 M 分别表示道路中节点的数量和边的数量，节点编号为从 0 到 N-1。Q 和 P 在后续的步骤中会使用到。第 2 行到第 M+1 行，每行有三个整数 X、Y、W，表示从 X 到 Y 有一条 **单向** 路径，长度为 W。此后的 P 行，每行有一个整数，其作用将在后续的步骤中介绍。

标准输出：

在本步骤中，你需要按上述方式读取数据，构建道路网络，并在标准输出中打印入边最多的节点编号和其入边数，两个数字用一个空格隔开，占一行。如果有多个节点的入边数量相同，打印最小的节点编号。

2.2 [步骤 2] 求最短路

函数名：本步骤需要实现为一个名为 `ComputePath` 的函数。

在本步骤中，你需要计算所有点到所有点之间的最短路。

提示：

```
if (dist[x][v] > dist[x][u] + w[u][v])
      dist[x][v] = dist[x][u] + w[u][v];
```

标准输出：

在本步骤中，你需要在标准输出中打印最长的最短路信息，包含三个数字：x y w。表示在所有节点中，节点 x 到节点 y 的最短路最长，为 w。如果有多个最短路最长，则输出 x 最小的最短路信息，如果 x 相同，则输出 y 最小的最短路信息。数字之间使用一个空格隔开，三个数字占一行。

2.3 [步骤 3] 随机游走

函数名：本步骤需要实现为一个名为 `RandomWalk` 的函数。

你的出租车是一辆定制款，除了作为驾驶员的你之外，最多可以再承载 Q 名乘客。现在你接收到一项任务，需要依次从 P 个节点处接乘客，每个节点处有 1 位乘客，最终把他们全部送到节点 0 处。为了研究随机游走，你驾驶出租车从节点 0 开始出发，在每个节点，从所有出边中随机选择一条出边前进，选择每条边的概率相同。具体选路方法：对于有 n 条出边的节点 v，将其出边按照边另一端的节点编号从小到大排序，然后选择从节点 v 出发的第 (random % n) 条路。

一些其他规则：

1. 乘客不能在节点 0 之外的节点下车；
2. 不能超载，车坐满之后一定要先回到节点 0，才能接下一个乘客；
3. 车到达节点 0 后，所有乘客均会下车，不论车中是否坐满乘客；
4. 一定要按照顺序接乘客。

现在你想知道需要最少走多长的路，才能把 P 名乘客全部送到节点 0 处。

数字 Q、P，与此后的 P 个乘客所在节点编号，均已经在此前的步骤 1 中读取。

在随机选择路时，请使用下述方法生成随机数。为保证正确性，请确认前 10 个生成的随机数如下。

```
1382929239
2595771929
3824335302
1887095479
2525665185
2885523164
543726083
188212084
607055924
3293118934
```

为了方便使用，以下代码也在 `randomcode.txt` 中给出。

```cpp
//
class MersenneTwister {
public:
  MersenneTwister() {
    index = 0;
    mt[0] = 2023; // seed
    for (int i = 1; i < 624; ++i) {
      mt[i] = (0x6C078965 * (mt[i-1] ^ (mt[i-1] >> 30)) + i)
              & 0xFFFFFFFF;
    }
  }

  unsigned int generate() {
    if (index == 0) {
      for (int i = 0; i < 624; ++i) {
        unsigned int y =
           ((mt[i] & 0x80000000) + (mt[(i + 1) % 624] & 0x7FFFFFFF))
          & 0xFFFFFFFF;
        mt[i] = mt[(i + 397) % 624] ^ (y >> 1);
        if (y % 2 != 0) {
          mt[i] ^= 0x9908B0DF;
        }
      }
    }

    unsigned int y = mt[index];
    y ^= (y >> 11);
    y ^= ((y << 7) & 0x9D2C5680);
    y ^= ((y << 15) & 0xEFC60000);
    y ^= (y >> 18);

    index = (index + 1) % 624;
    return y;
  }

  unsigned int mt[624];
  int index;
};
```

标准输出：

在本步骤中，你需要在标准输出中打印一个数字，为接送所有乘客所需要驾驶的路的长度，共占一行。

文件输出：

在本步骤中，你需要在当前工作目录下创建“path.txt”文件，并在其中打印出租车走过的路径。每行一个数字，表示路径中的一个节点。第一行和最后一行均为 0。

2.4 [步骤 4] 路径分析

函数名：本步骤需要实现为一个名为 `PathAnalyze` 的函数。

本步骤中，你需要分析上一步骤的驾驶路径，计算出前 3 个最频繁访问的节点。

标准输出：

在本步骤中，你需要在标准输出中打印三行信息，分别为前 3 个最频繁访问的节点编号，以及这些节点被访问的次数。每行为一个节点，按照从最频繁到最不频繁顺序打印。如果两个节点的访问次数相同，在优先打印编号小的节点。

2.5 [步骤 5] 最优路径

函数名：本步骤需要实现为一个名为 `BestPath` 的函数。

由于随机游走需要走的路太长了，你放弃了实践随机游走的想法。这次你想走最短路将 P 名乘客接到节点 0，你想知道这次的最短路径的长度。要求与步骤 3 相同。

标准输出：

在本步骤中，你需要在标准输出中打印一行信息，包括一个数字，为最短路径长度。

3 输入输出样例

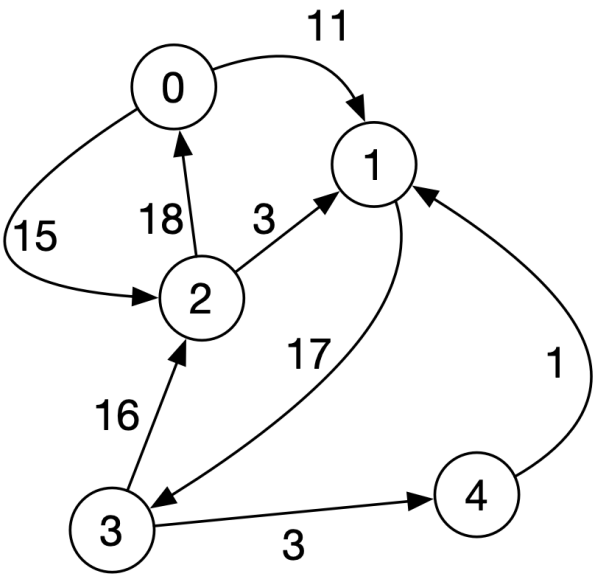
为了方便大家进行调试，我们在 data 目录中给出了样例（sample）、小规模（small）和大规模（large）数据。

给出的每个测试包括四个文本文件，以 sample 为例：

- sample.stdin.txt 为标准输入的内容
- sample.road-pass.txt 为输入文件的内容（对应标准输入中的 `./data/sample.road-pass.txt`）
- sample.stdout.txt 为正确的标准输出的内容
- sample.path.txt 为正确的 path.txt 文件内容

大规模测试的 stdout.txt 和 path.txt 文件不予给出。

为了方便调试，下面给出 sample 中的道路图：



4 提交要求和考核标准

4.1 编码要求

语言不限，但根据题目选择合适的语言并按照要求进行设计和编码。

4.2 评分标准

设计和实现程序，完成前述功能。如果不能完成全部程序功能，也请不要担心，我们会根据各个方面独立评分。具体标准如下：

步骤	分值
[步骤 1] 构建道路	15
[步骤 2] 求最短路	15
[步骤 3] 随机游走	20
[步骤 4] 路径分析	15
[步骤 5] 最优路径	15
通过所有测试数据（每个测试数据运行限时 5 秒）	10
代码规范、命名标准、代码风格、注释和说明等	10

4.3 提交要求

请将程序源代码使用 7z 格式压缩后命名为“姓名.7z”并将其 **放置在 U 盘中**，请确保文件完整性。

压缩包中应仅包含源代码和指定提交的文件，不包含测试用的输入输出、中间文件或可执行文件。压缩包大小原则上不超过 5MB。