Day 5 练习题（图论进阶）

By PaperCloud

（请选手仔细阅读此页内容）

比赛时间：**2023-7-13**

试题名称	环城旅行	阳光运动	分头行动	逆向生长
时间限制	1000ms	1000ms	2000ms	10000ms
内存上限	512MB	512MB	512MB	1024MB
评测方式	全文比较	全文比较	全文比较	全文比较
测试点数目	10	20	20	20

⚠️ **注意事项：**

1. 四道题难度差异不大，都可以尝试；
2. 尽可能把可以拿到的部分分都拿到；
3. 有问题找出题人；
4. 注意时间分配，题目可能有点多。
5. 推荐一种快速的输入输出方法，以（A+B problem 为例）：

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0);cin.tie(0);
    int a, b;
    cin >> a >> b;
    cout << a + b << "\n";
    // 使用 "\n" 比 std::endl 更快。
    // 关同步后，cin/cout 和 scanf/printf 不要混用。
}
```

1 环城旅行

1.1 题目描述

2022 年 11 月 20 日，是 Barisore 与好朋友出去游玩的日子。

H 市有着各种各样的景点，景点之间道路交通发达，构成了著名的文化风景区。

H 市共有 n 个景点和 m 条道路。第 i 条道路连接景点 x_i 和 y_i。

因为大家更愿意享受过程而非结果，所以他们调查出了每条道路的看点，并用权值 w_i 来表示。

Barisore 想要选择一个景点作为她与好朋友们的集合地，然后，她希望能够从这个景点开始，沿着道路游览若干的景点之后回到集合地。

Barisore 讨厌重复，所以她不想经过同一条道路，但是可以重复经过一个景点。她还希望他们的旅程尽可能的丰富多彩。具体地，她希望最大化自己经过的所有道路的最大权值和最小权值之差。

1.2 输入格式

第一行两个正整数 n, m，表示景点数与道路数。

接下来 m 行每行三个正整数 x_i, y_i, w_i，表示一条道路。

输入保证图中至少存在一条回路，Barisore 不会面临无法游览的境地。

1.3 输出格式

一行一个整数，表示旅程中经过的所有道路的最大权值和最小权值之差的最大值。

1.4 样例输入与输出

1.4.1 样例输入

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

1.4.2 样例输出

```
4
```

1.4.3 样例解释

沿路径 $4 \xrightarrow{5} 5 \xrightarrow{9} 7 \xrightarrow{8} 4$ 旅行，经过的所有道路的最大权值和最小权值之差为 4。容易发现没有答案更大的回路。

1.5 数据范围与提示

对于全部测试数据：$1 \le n \le 10^5$，$1 \le m \le 2 \times 10^5$，$1 \le x_i, y_i \le n$，$1 \le w_i \le 10^9$。

测试点编号	$n \le$	$m \le$	$w_i \le$	特殊性质
1	10	20	100	0
2	100	200	1000	2
3	100	200	1000	1
4	100	200	1000	0
5	500	1000	10^6	1
6	500	1000	10^6	0
7	5000	10^4	10^9	1
8	5000	10^4	10^9	0
9	10^5	2×10^5	10^9	1
10	10^5	2×10^5	10^9	0

特殊性质：

- 0：无
- 1：图中仅有唯一一条回路
- 2：图中任意两点间存在至少两条边不重合的路径

提示：本题考查边双连通分量的性质，可以发现，一个环路上的所有点必然在一个边双连通分量里。一个简易求边双的代码如下：

```
vector<pair<int,int>> G[MN];
void tarjan(int x, int par) {
    low[x] = dfn[x] = ++cnt;
    st[++top] = x;
    for(auto &[y, w] : G[x]) if(y != par) {
        if(dfn[y]) low[x] = min(low[x], dfn[y]); // 返祖边
        else {
            tarjan(y, x);
            low[x] = min(low[x], low[y]); // 树边
        }
    }
    if(dfn[x] == low[x]) { // 达到边双连通分量条件
        ++bcc; // 新建一个边双
        while(st[top + 1] != x) bel[st[top]] = bcc, --top; // 出栈
    }
}
```

2 阳光运动

2.1 题目描述

在偌大的 **Z** 大，拥有一辆自行车作为代步工具是必须的……

Barisore 是一个刚刚进入校园的新生，这天，她装好了自己的新车，从自己的宿舍楼出发，准备游览这片未知的校园。早在报道日，她就领到一个手绘版的校园地图，上面标志着 n 个她想要去参观的地点，校园的道路都是双向的，为了让交通更加便利，两个地点之间常常有不止一条道路。Barisore 想从学校出发，经过所有景点**至少**一次并最终回到宿舍休息。

Barisore 想要自己骑得尽兴，但是又不希望自己太累，所以她想控制自己的路程在$[L,R]$之间。

Barisore 想知道自己骑行方案的数量。两种方案不同，当且仅当它们经过道路的数量不同或顺序不同。

2.2 输入格式

第一行四个整数 n,m,L,R 。

以下 m 行，每行三个整数 x,y,z ，表示 x 号目标地点和 y 号目标地点之间有一条长为z的双向道路。如果 x 或 y 为 0 ，则表示寝室和目标地点之间的双向道路。

2.3 输出格式

一行一个整数，方案数。由于方案数可能很大，你只需要输出它对 998244353 取模的值即可。

2.4 样例输入与输出

2.4.1 样例输入 1

```
1 2 3 4
0 1 1
0 1 2
```

2.4.2 样例输出 1

```
4
```

2.4.3 样例解释 1

4 条路径分别为：

```
0 -(1)-> 1 -(2)-> 0;
0 -(2)-> 1 -(1)-> 0;
0 -(2)-> 1 -(2)-> 0;
0 -(1)-> 1 -(1)-> 0 -(1)-> 1 -(1)-> 0.
```

2.4.4 样例输入 2

```
3 5 5 10
0 1 2
0 2 3
1 2 1
1 3 2
2 3 3
```

2.4.5 样例输出 2

```
8
```

2.4.6 样例解释 2

8 条路径分别为:

```
0 -> 1 -> 3 -> 2 -> 0, L = 10;
0 -> 2 -> 3 -> 1 -> 0, L = 10;
0 -> 1 -> 2 -> 3 -> 1 -> 0, L = 10;
0 -> 1 -> 3 -> 1 -> 2 -> 0, L = 10;
0 -> 1 -> 3 -> 2 -> 1 -> 0, L = 10;
0 -> 2 -> 1 -> 3 -> 1 -> 0, L = 10;
0 -> 1 -> 2 -> 1 -> 3 -> 1 -> 0, L = 10;
0 -> 1 -> 3 -> 1 -> 2 -> 1 -> 0, L = 10.
```

2.4.7 样例输入 3

```
5 10 50 100
0 1 5
0 2 7
0 3 4
1 3 6
1 4 4
2 3 7
2 4 7
3 4 4
1 5 6
4 5 9
```

2.4.8 样例输出 3

```
606168943
```

2.5 数据范围与提示

对于前 10% 的数据，$n \le 2, m \le 2$；

对于前 20% 的数据，$n \le 2$；

对于前 40% 的数据，$n \le 5, m \le 10, 1 \le L \le R \le 50$，保证答案取模前不会超过 10^6；

对于全部数据，$1 \le n \le 14, 1 \le m \le 50, 0 \le x, y \le n, 1 \le z \le 10, 1 \le L \le R \le 200$。

提示：考虑哈密顿回路的一般计数方法，这题加入了**距离的限制**，请尝试使用状态压缩dp解决这个问题。**注意本题一个点可以经过多次，因此不再需要判断转移点是否已经走过。**你应该尝试为 dp 数组增加一维用来存路径长度。

下附一般哈密顿回路计数问题代码：

```
int S=(1<<n)-1;
dp[0][1] = 1; // 起点为 0，S 为空，值为 1，因为不考虑起点，这样计算更方便。
for(int s = 1; s <= S; ++s) // 一定要先枚举集合
    for(int i = 0; i < n; ++i) if(dp[i][s])
        for(auto &j : G[i]) if(!((s>>j)&1)) dp[j][s|(1<<j)] += dp[i][s];
// 如果 j 不在 s 中，就用 dp[i][s] 更新 dp[j][s|(1<<j)]
int ans = dp[0][all];
```

3 分头行动

3.1 题目描述

Bardisk 和 Barisore 热爱探索，这天他们相约来到地处白国的边缘的一座古老的海岛——南林岛。

岛上有 n 个打卡点，编号为 $1, 2, \cdots, 2$，打卡点之间通过 $n-1$ 条双向通行的道路连通。Bardisk 和 Barisore 各自有自己想要去打卡的地方，具体的，Bardisk 需要到达的打卡点有 m_1 个，分别为 $a_1, \cdots, a_{m_1}$；而 Barisore 需要到达的打卡点有 m_2 个，分别为 $b_1, \cdots, b_{m_2}$。他们可以以任意顺序访问所有他们需要到达的打卡点，且他们沿着一条道路从一个打卡点到达另一个打卡点均花费一个单位时间。

由于南林岛位置偏僻且地形复杂，他们两人决定采取一个比较安全的探索规则：每个单位时间，只有一人可以沿着道路去往另一个打卡点，而另一个人则呆在自己当前的打卡点不动，以便于及时接收对方发来的信号；此外，为了保证在突发情况下两人可以及时碰头，在任意时刻，两人之间的距离不能超过 d，这里对于距离的定义为两人之间的路径上需要经过的道路的数量。

目前两人都位于 1 号打卡点，在探索结束之后，两人均需要返回 1 号打卡点（返回的时候同样采取上述的探索规则），请问他们最少需要多少时间完成本次探索。

3.2 输入格式

第一行四个整数 n, m_1, m_2, d。表示打卡点的个数，Bardisk 和 Barisore 各自需要到达的打卡点个数，两人之间的最大距离。

第二行 m_1 个整数，表示 Bardisk 需要到达的打卡点编号。

第三行 m_2 个整数，表示 Barisore 需要到达的打卡点编号。

接下来 $n-1$ 行，每行两个整数 u, v 表示 u 号打卡点和 v 号打卡点之间有一条双向通行道路。

3.3 输出格式

一行一个整数，表示 Bardisk 和 Barisore 完成探索的最短时间。

3.4 样例输入与输出

3.4.1 样例输入 1

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

3.4.2 样例输出 1

```
6
```

3.4.3 样例解释 1

Bardisk 和 Barisore 的一种探索方式如下：

1. Barisore 不动，Bardisk 路线为 $1 \to 2 \to 3 \to 2 \to 1$，共花费时间 4；
2. Bardisk 不动，Barisore 路线为 $1 \to 4 \to 1$，共花费时间 2。

3.4.4 样例输入 2

```
6 1 2 1
4
3 6
1 2
2 3
2 4
1 5
5 6
```

3.4.5 样例输出 2

```
14
```

3.4.6 样例解释 2

Bardisk 和 Barisore 的一种探索方式如下：

1. Bardisk 从 1 移动到 2；
2. Barisore 从 1 移动到 2；
3. Bardisk 从 2 移动到 4；
4. Bardisk 从 4 移动到 2；
5. Barisore 从 2 移动到 3；
6. Barisore 从 3 移动到 2；
7. Bardisk 从 2 移动到 1；
8. Barisore 从 2 移动到 1；
9. Bardisk 从 1 移动到 5；
10. Barisore 从 1 移动到 5；
11. Barisore 从 5 移动到 6；
12. Barisore 从 6 移动到 5；
13. Bardisk 从 5 移动到 1；
14. Barisore 从 5 移动到 1。

3.5 数据范围与提示

一共有 **20** 组数据：

对于 **1 ~ 4** 组数据，$m_1, m_2 \leq n \leq 12$；

对于 **5 ~ 9** 组数据，$n \leq 1000, m_1, m_2 \leq 5$**;**

对于 **10 ~ 12** 组数据，$d = 1$；

对于 **13 ~ 16** 组数据，$d = n$；

对于 **17 ~ 20** 组数据，$1 \leq m_1, m_2 \leq n \leq 5 \times 10^5$。

提示:

1. 如果 **A** 要到节点 x，且 x 的深度大于等于 d，那么 **B** 一定要到达 x 的 d 级祖先，想想为什么。**d** 级别祖先应该怎么求?
2. 如果只有一个人，给定一个必须要走的集合，你是否可以求出它的最少行走步数?
3. 是否可以把两个人的行走路径分开来考虑，使用 **(2.)** 的方法计算?

4 逆向生长

4.1 题目描述

你平时会怎样画一棵树？我想你会先画出根节点，接着向下延伸出几条边，连向它的儿子节点，然后继续向下，画出孙子节点……

如果这一切颠倒过来，会发生什么？

现在你有一棵逆向生长的树，这棵树有 n 个叶子节点，编号依次为 $a_1, a_2, \cdots, a_n$。一开始，你只有这 n 个叶子节点，你将**依次**处理 q 个操作，操作 q_j 分为如下两种：

- $opt_i = 1$：表示新增一个节点 x_i，它有 s_i 个儿子，分别为 $b_{i,1}, b_{i,2}, \cdots, b_{i,s_i}$，由于树是逆向生长的，此时 x_i 的儿子节点都已经存在；
- $opt_i = 2$：表示一次询问，求出 x_i, y_i 在树上的 **LCA**（最近公共祖先节点），如果此时 x_i, y_i 不连通，请输出 `0`。

注意：在 q 次操作之后，所有节点不一定连通。

本题输入采用强制在线，具体细节请见【输入格式】。

4.2 输入格式

第一行一个整数 n（$1 \le n \le 2 \times 10^5$）表示树叶子节点的个数。假设这棵树一共有 N（$n \le N \le 2 \times 10^5$）个节点。

第二行 n 个整数，$a_1, a_2, \cdots, a_n$（$1 \le a_i \le N, i = 1, \cdots, n$）。

第三行一个整数 q, A, B（$1 \le q \le 2 \times 10^5$，$1 \le A, B \le 888$），表示操作个数，$A, B$ 为计算系数。定义函数 $F(x) = x \oplus (A \times \text{lastans} + B) \oplus \text{lastans}$，其中 lastans 为上次 $opt_i = 2$ 的操作输出的答案，初始情况 $\text{lastans} = 0$。

接下来 n 行，每行表示一个操作，其中第 i 行第一个数为 $opt_i \in \{1, 2\}$：

若 $opt_i = 1$，则接下来为两个整数 x_i^0, s_i（$0 \le s_i \le N$），然后为 s_i 个整数 $b_{i,1}^0, \cdots, b_{i,s_i}^0$；

- 该操作为新增一个节点，其编号为 $x_i = F(x_i^0)$；
- s_i 表示 x_i 的儿子节点的个数；
- x_i 的儿子节点为 $b_{i,1}, b_{i,2}, \cdots, b_{i,s_i}$，其中 $b_{i,j} = F(b_{i,j}^0), j = 1, \cdots, s_i$；

若 $opt_i = 1$，则接下来为两个整数 x_i^0, y_i^0；

- 该操作为询问 x_i, y_i 当前的 **LCA** 节点编号，其中 $x_i = F(x_i^0), y_i = F(y_i^0)$；
- 如果此时 x_i, y_i 不连通，请输出 `0`；
- 设当前询问的输出结果为 z_i，则当前操作之后，将 lastans 修改为 z_i。

对于所有输入中的 $x_i^0, b_{i,j}^0, y_i^0$ 等，保证其取值范围为 $[0, 2^{30} - 1]$。

4.3 输出格式

对于每个 $opt_i = 2$ 的询问，输出一行，其中包含一个整数，表示该询问的答案。

4.4 样例与输出与输出

4.4.1 样例输入

```
4
4 10 7 5
10 1 0
1 2 1 7
2 10 4
1 3 1 2
1 1 4 4 10 3 5
2 7 7
1 9 1 1
2 9 10
1 8 1 9
1 6 1 8
2 6 7
```

4.4.2 样例输出

```
0
7
9
6
```

4.5 数据范围与提示

对于 **30%** 的数据：$1 \le N, q \le 5 \times 10^3$；

对于 **50%** 的数据：$1 \le N, q \le 3 \times 10^4$；

另有 **25%** 的数据：$A = 1, B = 0$。（即 $A \times y + B = y$，那么 $x \oplus (A \times y + B) \oplus y = x$，可以发现，该部分输入不强制在线）；

对于 **100%** 的数据：$1 \le N, q \le 2 \times 10^5$。

提示：由于输出量过大，请不要使用过慢的输入输出方式（如未关同步的 **cin/cout**）。

提示： 如果没有强制在线，本题为普通的 **LCA** 问题，尝试为每一个点记录下它出现的时间，使用离线方法求解；对于强制在线的问题，尝试倍增的做法。

倍增数组没有办法在一开始就求出来：尝试动态维护倍增数组，你需要知道加入一个点之后，它会更新哪些点的倍增祖先。