目录

abc481 1
E.Most Brilliant Starry Sky 1
F.Evil Shortest Path 4

abc481

E.Most Brilliant Starry Sky

题目描述

在 xy 平面上有 N 颗星星。第 i 颗星星用坐标 (X_i, Y_i) 表示。

画一条经过原点且不经过任何星星的直线，这条直线将平面分成两个区域，选择其中一个区域。

求所选区域内最多能包含多少颗星星。

共有 T 组测试数据，请分别求解。

约束条件

- $1 \leq T \leq 2 \times 10^5$
- $1 \leq N \leq 2 \times 10^5$
- 所有测试数据的 N 之和不超过 2×10^5。
- $-10^9 \leq X_i, Y_i \leq 10^9$
- $(X_i, Y_i) \neq (0, 0)$
- $(X_i, Y_i) \neq (X_j, Y_j)$ $(i \neq j)$
- 任意两颗不同的星星不与原点共线。
- 所有输入均为整数。

输入格式

输入由标准输入给出，格式如下：

```
T
case_1
case_2
...
case_T
```

每个测试数据的格式如下：

```
N
X_1 Y_1
X_2 Y_2
...
X_N Y_N
```

输出格式

输出 T 行。第 i 行输出第 i 组测试数据的答案。

样例

1.in / 1.out

```
3
4
```

```
3   2 1
4   2 -1
5   -1 2
6   -2 -3
7   3
8   1 0
9   2 1
10  3 -2
11  20
12  351656128 486890400
13  43833334 189818465
14  244758369 410609525
15  82390518 959687989
16  37964139 450230904
17  -27725133 788196612
18  392638907 -544184247
19  -305242056 -60014958
20  -928747557 460755669
21  -415081310 638281062
22  307908966 -865172261
23  580339611 459847409
24  42654321 -744472144
25  -286500328 34461456
26  -874976232 -245368444
27  462187336 508372206
28  -229275227 74979882
29  -180126519 740406850
30  877877433 -211582842
31  256315745 -928285605
```

```
1  3
2  3
3  15
```

在第 1 组测试数据中，画直线 $y = -x$ 并选择包含第 1 颗星星的区域，该区域包含第 1、2、3 颗星星。不可能包含全部 4 颗星星，所以答案是 3。

在第 2 组测试数据中，画直线 $x = 0$ 并选择其右侧区域，所有星星都被包含。

题解

考虑一条旋转的直线，扫描维护直线两端的点。 容易说明，旋转一圈之后，所有情况都能考虑到。

这题我在考场上卡了排序，意思是元素必须要有稳定的相对顺序。 什么意思呢？就是像这道题，斜率会有正负无穷的情况，此时我之前忽略了它们，那可能就会影响排序。 所以，以下 `cmp` 函数中的前面部分不能省略，它代表将这些值放在数组的两端。

```cpp
bool cmp(const s &a, const s &b) {
  if (a.x == 0)
    return true;
  if (b.x == 0)
```

```
    return false;

  // a.y/a.x - b.y/b.x < 0;
  // ay bx - ax by < 0 ()rev if ax bx < 0
  i64 r = 1ll * a.y * b.x - 1ll * a.x * b.y;

  if (1ll * a.x * b.x > 0)
    return r < 0;
  else
    return r > 0;
}
```

代码不难，思维也不难，就做一个简单的记录。

```cpp
#include <algorithm>
#include <stdio.h>

#define maxn 200007

typedef long long i64;

struct s {
  int x, y;
} li[maxn];

bool cmp(const s &a, const s &b) {
  if (a.x == 0)
    return true;
  if (b.x == 0)
    return false;

  // a.y/a.x - b.y/b.x < 0;
  // ay bx - ax by < 0 ()rev if ax bx < 0
  i64 r = 1ll * a.y * b.x - 1ll * a.x * b.y;

  if (1ll * a.x * b.x > 0)
    return r < 0;
  else
    return r > 0;
}

int liner = 0, mx;

int T, n;

inline int max(int a, int b) { return a > b ? a : b; }

int main() {
```

```
35    scanf("%d", &T);
36
37    while (T--) {
38      liner = 0;
39      mx = -1;
40
41      scanf("%d", &n);
42      for (int i = 1; i <= n; i++) {
43        scanf("%d%d", &li[i].x, &li[i].y);
44
45        if (li[i].x == 0)
46          liner += (li[i].y > 0);
47        else
48          liner += (li[i].x > 0);
49      }
50      mx = max(mx, max(liner, n - liner));
51
52      std::sort(li + 1, li + n + 1, cmp);
53
54      for (int i = 1; i <= n; i++) {
55        //   printf("(%d, %d) slope = %lf\n", li[i].x, li[i].y,
56        //  1. * li[i].y / li[i].x);
57
58        if (li[i].x > 0) {
59          liner--;
60        }
61
62        if (li[i].x < 0) {
63          liner++;
64        }
65
66        mx = max(mx, max(liner, n - liner));
67      }
68
69      printf("%d\n", mx);
70    }
71    return 0;
72  }
```

F.Evil Shortest Path

题目描述

给定一个有 N 个顶点、M 条边的有向图。顶点编号为 $1, 2, ..., N$。第 i 条边从顶点 A_i 指向顶点 B_i，边上写有一个 1 到 9 之间的数字 C_i。

沿边的方向移动时，将经过的边上写的数字按经过顺序拼接得到的整数称为这次移动的「得分」。例如，若经过的边上数字依次为 $3, 1, 4$，则得分为 314。同一顶点或边可以多次经过。

对于每个 $v = 2, 3, ..., N$，求从顶点 1 到顶点 v 的移动的最小得分，并对 998244353 取模。保证给定的有向图可以从顶点 1 沿边方向到达其他所有顶点。

约束条件

- $2 \le N \le 2 \times 10^5$
- $N-1 \le M \le 2 \times 10^5$
- $1 \le A_i, B_i \le N$
- $1 \le C_i \le 9$
- 所有输入均为整数。
- 可以从顶点 1 沿边方向到达其他所有顶点。

输入格式

输入由标准输入给出，格式如下：

```
N M
A_1 B_1 C_1
A_2 B_2 C_2
...
A_M B_M C_M
```

输出格式

输出 $N-1$ 行。第 $v-1$ 行输出顶点 v 的答案。

样例

1.in / 1.out

```
4 4
1 2 1
1 3 1
2 4 9
3 4 2
```

```
1
1
12
```

- 顶点 2：$1 \to 2$（边权 1）得分为 1。
- 顶点 3：$1 \to 3$（边权 1）得分为 1。
- 顶点 4：$1 \to 2 \to 4$（边权 $1, 9$）得分为 19，$1 \to 3 \to 4$（边权 $1, 2$）得分为 12，所以最小为 12。

因此按行输出 $1, 1, 12$。

2.in / 2.out

```
3 5
1 2 9
1 2 2
2 2 1
1 3 9
2 3 1
```

```
2
9
```

- 顶点 2：$1 \to 2$（边权 9）得分为 9，$1 \to 2$（边权 2）得分为 2，所以最小为 2。
- 顶点 3：$1 \to 3$（边权 9）得分为 9，$1 \to 2 \to 3$（边权 $2, 1$）得分为 21，所以最小为 9。

注意图中可能包含重边和自环。

3.in / 3.out

```
11 11
1 2 9
2 3 9
3 4 8
4 5 2
5 6 4
6 7 4
7 8 3
8 9 5
9 10 3
9 10 2
10 11 9
```

```
9
99
998
9982
99824
998244
9982443
99824435
998244352
998244352
```

注意得分需要对 998244353 取模后输出。

题解

这道题是一个最短路的模板，需要合理转化。

转化什么意思？如果我们有能力存完整个代表路径序列的字符串，那么我们当然可以直接比较字符串。

但是从时间和空间上都不允许，所以考虑转化这个比较。

我们考虑一个“排名”来代表各个点实际的距离字符串的相对大小，那么具体的，增加数字后排名只增不降。

我们用（排名，扩展数字）的二元组来标定顺序，这样的话，我们得到的排名就是单调不降的。

容易说明，在这样的情况下，每个点第一个被处理到的值就是这个点的答案。

```cpp
#include <array>
#include <functional>
#include <queue>
#include <stdio.h>
#include <utility>
#include <vector>
#define maxn 200077

const int mod = 998244353;
```

```cpp
typedef std::pair<int, int> pii;
int dis[maxn];

int n, m, u, v, w;
std::pair<int, int> last;

typedef std::array<int, 4> qobj;
std::priority_queue<qobj, std::vector<qobj>, std::greater<qobj>> q;

std::vector<pii> edge[maxn];
int rel[maxn], rank = 0;

int main() {
    scanf("%d%d", &n, &m);

    for (int i = 1; i <= m; i++)
        scanf("%d%d%d", &u, &v, &w), edge[u].push_back(std::make_pair(v, w));

    for (int i = 1; i <= n; i++)
        dis[i] = -1;
    dis[1] = 0;

    for (auto [v, c] : edge[1])
        q.push({0, c, v, c});

    while (not q.empty()) {
        auto [r, c, v, x] = q.top();
        q.pop();

        if (dis[v] != -1)
            continue;

        if (last != std::make_pair(r, c)) {
            rank++;
            last = {r, c};
        }

        dis[v] = x;
        for (auto [wd, wv] : edge[v])
            if (dis[wd] == -1)
                q.push({rank, wv, wd, int((1ll * x * 10 + wv) % mod)});
    }

    for (int i = 2; i <= n; i++)
        printf("%d\n", dis[i]);

    return 0;
}
```