进阶计划开营仪式
& 摸底考试讲评

洛谷网校
离散小波变换°
2023年1月

感谢所有报名计划的同学

小调查：同学现在是几年级？

发在评论区就可以了。
防止出现混淆，初一是七年级（六三学制）或者六年级（五四学制），以此类推。

课程安排

寒假期间，一周两节直播课。

直播课：讲解新的算法、数据结构以及其他知识点。

作业：每节直播课后布置，含必做作业和补充作业。

比赛是专题模拟赛。指定时间，使用在线提交的方式。

直播授课主要是讲解知识点和例题。

错过比赛或者直播，可以之后找时间补。

但最好按时参加。

洛谷

课程安排

1月						
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	春
22	23	24	25	26	27	28
29	30	31				

摸底测试1	讲评&开营式
进阶算法思想1	进阶算法思想2
进阶搜索	
字符串算法	

2月						
日	一	二	三	四	五	六
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

专题模拟赛1/讲评	基础数据结构1
基础数据结构2	
图论1	
图论2	
专题模拟赛2/讲评	

(绿)授课时间：14:00-17:00

(黄)比赛时间：14:00-18:00

讲评时间：19:00开始

* 时间和内容安排可能会少量调整

共计：11次直播授课（可回放），3次模拟比赛及讲评

以及 刷题任务布置与指导+学习答疑

报名要求重申

1

课程面向已经熟练掌握基础算法的学生，
报名课程前，应当符合以下要求：

CSP-J 组拿到 200 分以上，或者同等水平；完成初中数学的学习。
熟练掌握《NOI 大纲》五级内算法知识点；
能完成部分“普及/提高-”难度的题目；这个难度及以上刷题量不少于60题

2

参与本计划学员应当具备以下能力：

可以比较熟练的分析算法时间与空间复杂度。
对于已经学会的算法，仅阅读文字题解就可以独立编写代码。
能够自行对程序查错（本计划不协助调试代码错误）。
有良好的自制力，可独立参与线上课程的学习，主动提问、参与讨论交流。

3

如果学生以上知识/能力有所欠缺，
在学习本计划的过程中会遇到较大的困难。
遇到这种情况，建议选【基础提高衔接计划】、【基础算法计划】

报名要求重申

关于难度

- 完成复习题有困难，建议更换为更简单的课程。
- 同学可能有缺漏需要自己补上。基础知识缺漏用《深入浅出基础篇》学习。
- 学生水平有差异，不需要太焦虑。
- 超越自我，不需要太在意排名。

退课政策

各计划开始后（或者报名后）完成 5 次直播课程（包括正课和模拟比赛讲评课）前，允许主动退出该计划。每上完一次直播课收取 300 元的学费后，退还剩余所有费用。

计划如何帮助同学提升

以下不仅是洛谷可以提供的清单，也对同学训练过程提出了要求。

- 授课：以知识点讲解为主，包含原理和例题、习题。
- 作业：非常重要，从简单到困难的题目集合，会进行竞争排名。
- 模拟比赛：帮助同学积累经验，查缺补漏，也会排名。大家尽量按时参加，赛后需要总结、订正。
- 答疑：帮助同学少走弯路。及时解决同学的问题。不协助调试代码，同学应当有自己调试代码的能力。
- 其他：以调试需要，向助教要一个题目的一个测试点（一次一个测试点，仅限公开题目）。以及可以寻求一些洛谷社区上的便利（改用户名等）。

同学们尽可能合理安排时间，尽可能多的参加这些内容。

计划提供的资源

- 视频授课（直播或者录播/回放）在网校课程页面
- 课程团队在右侧可以看到。
- QQ 群和加群方式也在右边可以看到。

关于专题模拟赛

定时比赛，使用线上提交形式。

注意：

- 不使用文件输入输出；
- 考察的范围包含先前课程的知识点；
- 上传到洛谷进行评测。

比赛结束后，题目上传到团队中，可以自主练习。

正式比赛请务必遵守考场要求。

关于专题模拟赛


进阶算法计划【冬春前期 / 2023】
团队设置
概览
题目
作业
题单
比赛
成员
文件
团队比赛
未开始
进阶算法计划摸底考试
IOI
团队内部赛

计划纪律

1. 认真参与课程，学习所有的直播。
2. 无特殊情况，参加每一场模拟赛，不建议大片课程缺席。
3. 准备好草稿本，笔记本（也可以记录电子版笔记）。
4. 不抄袭题解，不对照题解代码抄写，不背诵题解代码，少参考题解（对于一个题目，半小时思路没有进展再参考题解，只参考思路，不参考题解代码）。
5. 认真完成作业，作业题会多少分写多少分，不一定要写满分。遇到少量没学过的知识不用害怕，可以去学，也可以跳过。
6. 模拟赛后复盘，可以写总结（建议使用洛谷博客）。

模拟赛总结

1. 模拟比赛每道题的得分 如：0 + 100 + 50 +0。
2. 比赛过程中时间安排 如：第 2 题花了 2 小时，导致后面的题没有时间做。第 4 题花了 2 小时尝试完成 100 分，但是没有调试出来，导致本题只有 0 分。
3. 比赛过程中犯的错误 如：第 1 题因为忘记使用文件输入输出，导致 0 分，第 3 题把 m 和 n 写反了。
4. 调整比赛策略 如：第 4 题应当先完成暴力分 30 分，再尝试 100 分的做法。

学术诚信与作业要求

1. 一定要按时完成作业。根据实际能力和时间安排练习补充作业。
2. 作业不可以复制或改编题解或对照题解抄写，或者上课样例代码。
3. 需要在 IDE 里面写好、测试样例，再提交。
4. 有问题及时寻求帮助。
5. 抄袭题解者，第一次警告。第二次棕名并通告家长。
6. 在模拟比赛中作弊者，将直接公开批评。

代码调试与答疑

1. 学习录播课程 调试与对拍。
2. 仔细审题，在草稿纸上记录题目中的重要信息，大致思路。
3. 养成良好代码习惯，变量名应当有实际意义，不使用读入优化，不使用宏定义（define）。
4. 代码中添加适量注释，帮助自己和他人理解。
5. 协助以提示为主，如提供错误样例。
6. 如需询问思路是否正确，请使用文字描述思路，而不是直接粘贴代码。
7. 互帮互助。在群中讨论，可能可以更快解决。

对于家长的建议

监督+信任

监督：督促学生完成作业/比赛，但尽力而为。

信任：相信学生的努力和智慧。

不要焦虑。比赛中个别试题、补充作业试题同学写不出来是非常正常的。看看能不能找到原因。

家长有问题，多在微信群沟通。可以寻求助教的建议。

如果学生在学习过程中遇到较大困难，建议更换至更加基础的课程。

家长需要提供一些协助。文化课方面的权衡、学习练习时间的安排、家校沟通。

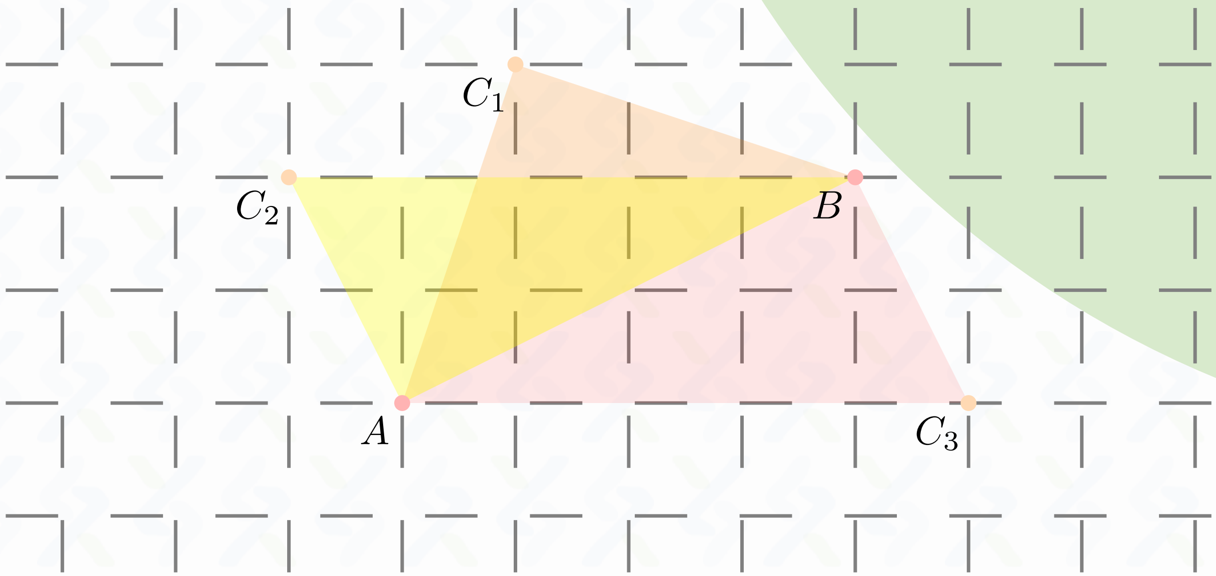
感谢大家报名课程

有问题可以多多提出。

后面是摸底考试讲评。

T1 冰雪聪明

题意：给定两个整点坐标，要求求出任意一个整点，使得这三个整点可以形成直角三角形。



数据范围：$-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$。找到的点的坐标同样应该符合该范围。

T1 冰雪聪明

分类讨论:

- 若 AB 不平行于坐标轴，则 A, B 可构成矩形，取矩形另外两个顶点之一即可符合条件。
- 若 AB 平行于坐标轴，可取 A, B 中的任意一点往垂直于 AB 方向移动一格构成直角三角形。

移动时要注意不能移动出网格图。

构造方案不唯一。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a1, b1, a2, b2;
    scanf("%d%d", &a1, &b1);
    scanf("%d%d", &a2, &b2);
    if(a1 == a2) printf("%d %d\n", a1 == 1e9 ? a1 - 1 : a1 + 1,
b1); else
    if(b1 == b2) printf("%d %d\n", a1, b1 == 1e9 ? b1 - 1 : b1
+ 1); else
    printf("%d %d\n", a1, b2);
    return 0;
}
```

T2 脑袋空空

题意：计算出 n 个数字所有重排方案组成的大整数的和，结果对 998,244,353 取模。

数据范围：$1 \le n \le 10^6$。

部分分：

- 对于前 50% 数据，$1 \le n \le 10$；
- 对于另外 20% 数据，$a_i = 1$。

T2 脑袋空空 / 50 分

使用搜索。直接枚举所有重排方案，计算组成的大整数的值，将结果相加即可。

```
const int MAXN = 100 + 3;
const int MOD = 998244353;
int n, A[MAXN], ans; bool V[MAXN];
void dfs(int u, int w){
    if(u == n){
        ans = (ans + w) % MOD;
    } else {
        up(1, n, i) if(!V[i]){
            V[i] = true, dfs(u + 1, (w * 10ll + A[i]) % MOD),
V[i] = false;
        }
    }
}
```

T2 脑袋空空 / 70 分

注意到 $a_i = 1$，重排后的大整数只能是 $11\cdots1$。同时重排方案一共有 $n!$ 种，所以答案是 $11\cdots1 \times n! \bmod 998244353$。

怎么计算 $11\cdots1 \bmod 998244353$？只要先取 $s = 0$，不断执行操作 $s \leftarrow (s \times 10 + 1) \bmod 998244353$ 即可。

T2 脑袋空空 / 70 分

考虑解决另外 20% 的特殊性质部分分。

注意到 $a_i = 1$，重排后的大整数只能是 $11\cdots1$。同时重排方案一共有 $n!$ 种，所以答案是 $11\cdots1 \times n! \bmod 998244353$。

怎么计算 $11\cdots1 \bmod 998244353$？只要先取 $s = 0$，不断执行操作 $s \leftarrow (s \times 10 + 1) \bmod 998244353$ 即可。

T2 脑袋空空 / 100 分

考虑 a_i 对答案造成的贡献。

a_i 可以出现在第 $1,2,\cdots,n$ 上的每一位。当 a_i 出现在第 j 位上时，一共出现了 $(n-1)!$ 次。在这一位上贡献为 $a_i \times 10^{j-1}$。

于是 a_i 对答案的总贡献为 $a_i \times (10^0 + 10^1 + \cdots + 10^{n-1}) \times (n-1)!$。于是答案为：

$$(a_1 + a_2 + \cdots + a_n) \times (10^0 + 10^1 + \cdots + 10^{n-1}) \times (n-1)!$$

```
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
typedef long long i64;
const int INF = 2147483647;
const int MOD = 998244353;
int n, s1 = 1, s2 = 1, t = 0;
int main(){
    scanf("%d%d", &n, &t);
    up(2, n, i){
        int x; scanf("%d", &x), t = (t + x) % MOD;
        s1 = (1ll * s1 * 10 + 1) % MOD;
        s2 =  1ll * s2 * (i - 1) % MOD;
    }
    printf("%d\n", 1ll * t * s1 % MOD * s2 % MOD);
    return 0;
}
```

T3 爷是阿咔

题意：给定一棵有根树，点有点权和状态 $0/1$。有 m 次操作，每次操作选择一个点：

- 若状态为 0，则置为 1；
- 若状态为 1，则置为 0，同时将儿子节点的状态置为 1。

操作前以及每次操作后，输出所有状态为 1 的节点权值最大值。

数据范围：$1 \le n, m \le 2 \times 10^5$。

部分分：

- 对于前 60% 数据，$1 \le n \le 5000$；
- 对于另外 10% 数据，原树是一条链；
- 对于另外 10% 数据，原树是菊花图。

T3 爷是阿咔 / 60 分 / 70 分 / 80 分

对于前 60% 的数据，直接进行模拟。不过在模拟前需要对整棵树跑一遍 dfs 确定每个节点有哪些儿子。查询时只需要扫描所有点统计答案即可。

对于另外 10% 链的数据，注意到每个节点最多只有一个儿子，于是每次只会最多有两个节点状态反转。可以使用堆来维护最大值。

对于另外 10% 菊花数据，

- 对花瓣的修改操作只会影响一个点，继续用堆维护；
- 对花心的修改操作只会影响花瓣上状态为 0 的点，只要存哪些点状态为 0 并对这些点修改即可。

T3 爷是阿吽 / 100 分

每次修改最多将 1 个点的状态置为 0。

我们只要存储，对于每个节点它哪些儿子的状态可能是 0 即可。可以使用 vector。

- 当把某个节点状态由 1 改成 0，需要往父亲节点的 vector 里加入该节点序号。同时将该节点状态为 0 的儿子节点设为 1，并清空 vector。
- 当把某个节点状态由 0 改成 1，不需要从父亲节点的 vector 里查找并删除。只需要在父亲节点清空 vector 时检查这些曾经状态是 0 的节点现在状态是否为 1。

```cpp
#include<bits/stdc++.h>
#define up(l, r, i) for(int i = l, END##i = r;i <= END##i;++ i)
#define dn(r, l, i) for(int i = r, END##i = l;i >= END##i;-- i)
using namespace std;
typedef long long i64; const int MAXN= 2e5 + 3;
vector <int> V[MAXN], P[MAXN];
int n, m, W[MAXN], F[MAXN]; bool S[MAXN]; void dfs(int u, int
f){
    for(auto &v : V[u]) if(v != f){
        F[v] = u; if(S[v] == 0) P[u].push_back(v);
        dfs(v, u);
    }
}
using pii = pair<int, int>;
priority_queue <pii > Q;
void add(int x){ S[x] = 1, Q.push({W[x], x}); }
void del(int x){ S[x] = 0; }
int  get(){
    while(S[Q.top().second] == 0) Q.pop();
    return Q.top().first;
}
```

```
int main(){
    n = qread(), m = qread(), W[0] = -1, add(0);
    up(1, n - 1, i){
        int u = qread(), v = qread();
        V[u].push_back(v);
        V[v].push_back(u);
    }
    up(1, n, i) W[i] = qread();
    up(1, n, i) S[i] = qread(), S[i] ? add(i) : void();
    dfs(1, 0);
    printf("%d\n", get());
    up(1, m, i){
        int x = qread();
        if(S[x] == 0) S[x] = 1, add(x); else {
            del(x), P[F[x]].push_back(x);
            for(auto &s : P[x]) if(S[s] == 0) add(s);
P[x].clear();
        }
        printf("%d\n", get());
    }
    return 0;
}
```

T4 这寺豪德

题意：给定边长为 n 的三角形迷宫。询问从最上层三角形出发，可以到达多少个小三角形。

数据范围：$1 \le n \le 10^3$。

部分分：

- 对于前 30%的数据，$1 \le n \le 3$。
- 对于另外 10% 的数据，整个三角形只有外轮廓。

T4 这寺豪德 / 30 分 / 40 分

- 对于前 30% 的数据，可以试试手动模拟。但我没试过。
- 对于 10% 的特殊数据，容易发现可以到达大三角形内任意一个小三角形。于是答案就是 n^2。

T4 这寺豪德 / 100 分

考虑使用 bfs。

原题已经给正三角形进行标号。但为了方便，我们还应该给倒三角形进行标号。为了方便：

- 我们将第 i 行第 j 列正三角形记为 $A(i,j)$；
- 我们将第 i 行第 j 列正三角形正下方的倒三角形记为 $B(i,j)$。

处理时，开两个数组 A,B 将 $A(i,j),B(i,j)$ 映射到唯一序号。

记 $P(x,0/1/2)$ 表示序号为 x 的三角形直接相连的三个三角形的序号。这是容易预处理的。 接着就可以愉快 bfs 了。

```
const int MAXN = 1e3 + 3;
const int MAXM = 4e6 + 3;
int A[MAXN][MAXN], B[MAXN][MAXN], M[MAXM], P[MAXM][3], o, n;
void pre(){
    up(1, n, i){
        up(1, i, j){
            M[A[i][j]] = qread();
            P[A[i][j]][0] = B[i - 1][j    ];
            P[B[i - 1][j    ]][0] = A[i][j];
            P[A[i][j]][1] = B[i    ][j    ];
            P[B[i    ][j    ]][1] = A[i][j];
            P[A[i][j]][2] = B[i - 1][j - 1];
            P[B[i - 1][j - 1]][2] = A[i][j];
        }
    }
    up(1, n, i){
        up(1, i, j){
            int x = B[i][j];
            M[x] = (M[P[x][0]] & 1) | (M[P[x][1]] & 2) |
(M[P[x][2]] & 4);
        }
    }
}
```

```
bool V[MAXM];
void bfs(){
    int cnt = 0; queue <int> Q; Q.push(A[1][1]), V[A[1][1]] =
true;
    while(!Q.empty()){
        int u = Q.front(); ++ cnt; Q.pop();
        up(0, 2, i) if(M[u] & (1 << i)){
            int v = P[u][i];
            if(!V[v]) V[v] = true, Q.push(v);
        }
    }
    printf("%d\n", cnt);
}
int main(){
    up(1, n    , i) up(1, i, j) A[i][j] = ++ o;
    up(1, n - 1, i) up(1, i, j) B[i][j] = ++ o;
    n = qread(), pre(), bfs();
    return 0;
}
```

T5 提桶跑路

题意：给定大小为 $n \times m$ 的矩阵。要求选出一个子矩阵，并且从子矩阵找到通向大矩阵顶点的路径，使得所有涉及到的权值之和最大。

数据范围：$1 \leq n \leq 400$。

T5 提桶跑路

鉴定为二合一。

首先需要预处理出，从 (i,j) 开始，到左上角、左下角、右上角、右下角可以取得的最大权值。这里以到左上角为例：

- 记 $lu_{i,j}$ 表示从 $(1,1)$ 只能向右或者向下走到 (i,j) 的最大权值。
- 容易得出状态转移方程式：

$$lu_{i,j} = \max\left(lu_{i-1,j}, lu_{i,j-1}\right) + w_{i,j}$$

其他三个方向同理：

$$ld_{i,j} = \max\left(ld_{i+1,j}, ld_{i,j-1}\right) + w_{i,j}$$
$$ru_{i,j} = \max\left(ru_{i-1,j}, ru_{i,j+1}\right) + w_{i,j}$$
$$rd_{i,j} = \max\left(rd_{i+1,j}, rd_{i,j+1}\right) + w_{i,j}$$

T5 提桶跑路 / 40 分 / 70 分 / 100 分

接着是第二部分，可以枚举这个子矩阵。

对于 $n \le 50$，直接枚举矩阵的四角，再枚举计算矩阵内元素和，复杂度 $O(n^6)$，但是评测机很快且常数较小，可以通过这部分。

对于 $n \le 80$，可以用二维前缀和快速计算子矩阵元素和。复杂度下降到 $O(n^4)$。

对于 $n \le 400$，则需要特别处理：

- 枚举子矩阵的第一行与最后一行的行号 p_1, p_2；
- 计算出行号在 p_1, p_2 内，每一列元素的和 s_i。得到关于列号 q_1, q_2 的表达式：

$$
\begin{aligned}
val = s_{q_2} - s_{q_1-1} + lu_{p_1,q_1} + ld_{p_2,q_1} + ru_{p_1,q_2} + rd_{p_2,q_2} \\
-a_{p_1,q_1} - a_{p_1,q_2} - a_{p_2,q_1} - a_{p_2,q_2}
\end{aligned}
$$

T5 提桶跑路 100 分

$$val = s_{q_2} - s_{q_1-1} + lu_{p_1,q_1} + ld_{p_2,q_1} + ru_{p_1,q_2} + rd_{p_2,q_2}$$
$$-a_{p_1,q_1} - a_{p_1,q_2} - a_{p_2,q_1} - a_{p_2,q_2}$$

整理后为:

$$val = \left(s_{q_2} + ru_{p_1,q_2} + rd_{p_2,q_2} - a_{p_1,q_2} - a_{p_2,q_2}\right) +$$
$$\left(-s_{q_1-1} + lu_{p_1,q_1} + ld_{p_2,q_1} - a_{p_1,q_1} - a_{p_2,q_1}\right)$$

第一个括号内只和 q_2 有关，第二个括号内只和 q_1 有关。

于是可以枚举 q_2，计算所有可能的 q_1 对应的价值的最大值。更新答案即可。

```cpp
const int MAXN= 400 + 3;
int n, m, A[MAXN][MAXN];
i64 S[MAXN][MAXN], T[MAXN], L[MAXN], R[MAXN], ans = -1e18;
i64 LU[MAXN][MAXN], LD[MAXN][MAXN];
i64 RU[MAXN][MAXN], RD[MAXN][MAXN];
void pre(){
    up(1, n, i) up(1, m, j) S[i][j] = S[i - 1][j] + A[i][j];
    up(0, n + 1, i) up(0, m + 1, j)
        LU[i][j] = LD[i][j] = RU[i][j] = RD[i][j] = -INF;
    LU[1][1] = A[1][1], LD[n][1] = A[n][1];
    RU[1][m] = A[1][m], RD[n][m] = A[n][m];
    up(1, n, i) up(1, m, j) if(i != 1 || j != 1)
        LU[i][j] = max(LU[i - 1][j], LU[i][j - 1]) + A[i][j];
    dn(n, 1, i) up(1, m, j) if(i != n || j != 1)
        LD[i][j] = max(LD[i + 1][j], LD[i][j - 1]) + A[i][j];
    up(1, n, i) dn(m, 1, j) if(i != 1 || j != m)
        RU[i][j] = max(RU[i - 1][j], RU[i][j + 1]) + A[i][j];
    dn(n, 1, i) dn(m, 1, j) if(i != n || j != m)
        RD[i][j] = max(RD[i + 1][j], RD[i][j + 1]) + A[i][j];
}
```

```
void calc(){
    up(1, n, a) up(a + 1, n, b){
        up(1, m, i){
            T[i] = S[b][i] - S[a - 1][i] + T[i - 1];
        }
        up(1, m, i)
    L[i] = -T[i - 1] + LU[a][i] + LD[b][i] - A[a][i] - A[b][i],
    R[i] =  T[i    ] + RU[a][i] + RD[b][i] - A[a][i] - A[b][i];
        i64 w = -INF;
        up(1, m, i){
            if(i > 1) ans = max(ans, R[i] + w);
            w = max(w, L[i]);
        }
    }
}
int main(){
    n = qread(), m = qread();
    up(1, n, i) up(1, m, j) A[i][j] = qread();
    pre(), calc();
    printf("%lld\n", ans);
    return 0;
}
```

T6 醋溜便当

题意：给定带权无向图。对每个结点询问，是否存在回路（可多次经过同一条边），长度在 $[x, kx]$ 内。

数据范围：$1 \le n, m \le 2 \times 10^5$，$1 < k \le 10^9$，$1 \le x \le 10^9$，$1 \le w_i \le 10^9$。

T6 醋溜便当

容易发现，如果从 x 出发有一条长度为 a 的回路，那么可以不断绕回路走，走出长度为 $2a, 3a, \cdots$ 的回路。

题设范围是 $[x, kx]$。如果从 x 出发有一条长度不超过 x 且大于 0 的的回路，那么总可以不断重复，直到回路长度在 $[x, kx]$ 内。

于是问题转化为了，对每个点找一条最短的回路。

- 由于原图是无向图，因此总可以走到第一个正权边后，原路返回。于是若回路上有两种不同的正权，一定不优。
- 由于原图有零权边，因此有可能整个过程就经过一次正权边。

可以用并查集维护。

```
const int MAXN = 2e5 + 3;
int F[MAXN], U[MAXN], V[MAXN], W[MAXN], A[MAXN];
int getfa(int x){return x == F[x] ? x : F[x] = getfa(F[x]);}
int main(){
    int n = qread(), m = qread(), x = qread(), k = qread();
    up(1, n, i) A[i] = INF, F[i] = i;
    up(1, m, i)
        U[i] = qread(), V[i] = qread(), W[i] = qread();
    up(1, m, i) if(W[i] == 0){
        int u = getfa(U[i]), v = getfa(V[i]);
        if(u != v) F[u] = v;
    }
    up(1, m, i){
        int u = getfa(U[i]), v = getfa(V[i]), w = W[i];
        if(w != 0){
            if(u == v) A[u] = min(A[u], w);
                else A[u] = min(A[u], 2 * w), A[v] = min(A[v],
2 * w);
        }
    }
    up(1, n, i) printf("%c ", A[getfa(i)] <= x * k ? '1' :
'0');
    return 0;
}
```