兰州大学第四届程序设计大赛题解

Solutions of The 4th Lanzhou University Programming Contest

2020.11.14

更多题解代码请转至

https://github.com/LovelyAnQi/LZUACM-S4

赛制	ACM 个人赛
语言	C/C++、Java、Python3
题目	15
时长	4 小时

为了响应教育部提出“新工科”教育战略，培养和展示我校大学生分析、解决问题和计算机编程的能力，鼓励和培养创新思维，丰富校园学术气氛，造就具有综合素质的面向 21 世纪的计算机人才，兰州大学于 2020 年 11 月 14 日举办兰州大学第四届程序设计大赛，面向兰州大学全体本科学生。

(感谢兰州大学信息科学与工程学院马俊老师提供的技术支持)

Accepted Wrong Answer Runtime Error Time Limit Exceeded
Memory Limit Exceeded Compile Error

目录

A 欢迎来到兰州大学第四届程序设计大赛

时间限制： 1000ms
空间限制： 256MB
难度： ★☆☆☆☆ ☆☆☆☆☆

标签： 字符串

题目描述

又到了一年一度的信息科技活动月，作为兰州大学程序设计大赛的工作人员，GJX 很早就来到了比赛的现场，准备为即将到来的参赛选手张贴欢迎标语。遗憾的是，现场的打印机出了一点故障，于是 GJX 决定发动志愿者们手写标语。

现在请你帮助一起完成标语的制作工作，本题目没有输入，并且仅仅需要你输出 `Welcome to the 4th Lanzhou University Programming Contest!`。

输入描述

本题目没有输入

输出描述

一行，一个字符串，输出下列内容（行尾的换行不是必要的）：`Welcome to the 4th Lanzhou University Programming Contest!`

题解

$O(1)$：Accepted 按要求输出一行

参考代码

C

```c
#include <stdio.h>
int main()
{
    printf("Welcome to the 4th Lanzhou University Programming Contest!\n");
    return 0;
}
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    cout << "Welcome to the 4th Lanzhou University Programming Contest!" << endl;
    return 0;
}
```

Java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Welcome to the 4th Lanzhou University Programming Contest!");
    }
}
```

Python3

```python
print("Welcome to the 4th Lanzhou University Programming Contest!")
```

B QQ群保卫战

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ☆☆☆☆☆

标签： 模拟 图论 深度优先搜索DFS 广度优先搜索BFS

题目描述

为了组织竞赛的报名，群主 $LovelyAnQi$ 创建了一个竞赛通知群，并邀请 GJX 成为了通知群的管理员。

通知群每天都有很多申请需要处理，而 GJX 忙于制作宣传标语，无暇管理群内的事务，请你帮一帮 GJX。

通知群内的每件事情都可以用一行若干个数字来表示，每条消息的具体格式像下面这样：

- $0\ Number$：群主 $LovelyAnQi$ 创建了QQ群并邀请 GJX 成为了管理员，$Number$ 是群主的编号，此条消息出现且一定出现在第一行；
- $1\ Number$：编号为 $Number$ 的人搜索了群号并申请加入QQ群；
- $2\ Number$：编号为 $Number$ 的人退出了QQ群；
- $3\ Number1\ Number2$：编号为 $Number1$ 的人尝试邀请编号为 $Number2$ 的人加入群聊；
- $4\ Number$：编号为 $Number$ 的人在QQ群内发送了广告。

如果有一个编号为 n 的人发送了广告，GJX 会把直接或间接涉及 n 的所有人踢出群聊。直接涉及 n 的人被定义为邀请过 n 和被 n 邀请过的人，间接涉及 n 的人被定义为直接涉及一个“直接涉及 n 或间接涉及 n”的人。GJX 完成踢人后，需要公示踢出群聊的人的数量。

如果一个人邀请了一个受到过广告事件牵连的人，那么对这个人的惩罚同发广告的人，并且要输出此次踢出的人的数量。

一般情况下，GJX 会默认批准加群的申请，除非尝试加群（无论是搜索群号还是被邀请）的人是之前被牵连而踢出群的人，特别的，即是一次广告事件中被牵连的人提前退出了群聊，也不能够再次加群。

如果直到最后一条消息处理完毕，QQ群中都没有人发布广告，GJX 会开心的说：`Nice Group Chat`；

如果群主退出了QQ群，GJX 会遗憾的说：`Group Chat Disbanded`；

特别的，群主退群只可能出现在最后一条消息；

如果广告事件中牵连到了群主，GJX 不会踢人，并会主动退出QQ群，然后说：`Incredible Group Chat`；

如果群主或 GJX 退出了群聊，程序立即结束。

输入描述

第一行，一个整数 $n\ (1 \le n \le 10^6)$，表示消息的数量；

接下来的 n 行，每行 $2 \sim 3$ 个整数（取决于第一个整数所代表的消息类型），其中 $0 \le Number \le n-1$；

数据保证不会出现以下无效输入：

1. 已经在群聊中的人重新申请加群或被邀请加入群聊；

2. 不在群聊中的人退群、发出邀请或发出广告；

3. 在群主退出群聊（群聊解散）后还有新消息

输出描述

在 GJX 每次踢人时输出被踢出群聊的人的数量，每个输出占一行；

如果群主退出了QQ群，输出`Group Chat Disbanded`并结束程序；

如果广告事件中牵连到了群主，输出`Incredible Group Chat`并结束程序；

如果直到最后一条消息处理完毕，QQ群中都没有人发布广告，群主也没有退出群聊，输出`Nice Group Chat`

题解

$O(n)$：`Accepted` 选择合适的结构建立一个无向图，将所有有过直接邀请关系的人用无向边连起来。记录每个人是否在群内，是否被广告牵连过。每次有人发广告或邀请了被牵连过的人进群时，通过搜索遍历所有被牵连的对象（无论该对象是否在群内），统计在群内的对象数量并维护状态信息。注意，邀请关系一旦确立不会再被解除

参考代码

C

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define maxn 1000005
#define maxm 2000005
int n, op, a, b, owner, cnt, head[maxn], q[maxm], ql, qr;
bool in[maxn], vis[maxn], flag;
struct Edge{int u, v, next;} edge[maxm];
void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
void Kick(int x) {
    ql = 1, qr = 0;
    q[++qr] = a;
    vis[a] = 1;
    int ans = 0;
    while (ql <= qr) {
        int u = q[ql++];
        if (owner == u) {
            printf("Incredible Group Chat\n");
            exit(0);
        }
        if (in[u]) {
            ans++;
            in[u] = 0;
        }
        for (int i = head[u]; i; i = edge[i].next) {
            int v = edge[i].v;
            if (vis[v]) continue;
```

```
            q[++qr] = v;
            vis[v] = 1;
        }
    }
    printf("%d\n", ans);
    flag = 1;
}
int main()
{
    scanf("%d", &n);
    while (n--) {
        scanf("%d %d", &op, &a);
        if (op == 0) {
            owner = a;
            in[a] = 1;
        }
        else if (op == 1) {
            if (vis[a]) continue;
            else in[a] = 1;
        }
        else if (op == 2) {
            in[a] = 0;
            if (owner == a) {
                printf("Group Chat Disbanded\n");
                return 0;
            }
        }
        else if (op == 3) {
            scanf("%d", &b);
            if (vis[b]) Kick(a);
            else {
                add(a, b);
                add(b, a);
                in[b] = 1;
            }
        }
        else Kick(a);
    }
    if (!flag) printf("Nice Group Chat\n");
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 1000005
#define maxm 2000005
int n, op, a, b, owner, cnt, head[maxn];
bool in[maxn], vis[maxn], flag;
struct Edge{int u, v, next;} edge[maxm];
inline void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
```

```cpp
inline void Kick(int x) {
    queue<int> q;
    q.push(x);
    vis[x] = 1;
    int ans = 0;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        if (owner == u) {
            cout << "Incredible Group Chat" << endl;
            exit(0);
        }
        if (in[u]) {
            ans++;
            in[u] = 0;
        }
        for (int i = head[u]; i; i = edge[i].next) {
            int v = edge[i].v;
            if (vis[v]) continue;
            q.push(v);
            vis[v] = 1;
        }
    }
    cout << ans << endl;
    flag = 1;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin >> n;
    while (n--) {
        cin >> op >> a;
        if (op == 0) {
            owner = a;
            in[a] = 1;
        }
        else if (op == 1) {
            if (vis[a]) continue;
            in[a] = 1;
        }
        else if (op == 2) {
            in[a] = 0;
            if (owner == a) {
                cout << "Group Chat Disbanded" << endl;
                return 0;
            }
        }
        else if (op == 3) {
            cin >> b;
            if (vis[b]) Kick(a);
            else {
                add(a, b);
                add(b, a);
                in[b] = 1;
            }
        }
```

```
72          else Kick(a);
73      }
74      if (!flag) cout << "Nice Group Chat" << endl;
75      return 0;
76  }
```

C 统计人数

时间限制： 1000ms
空间限制： 256MB
难度： ★★★☆☆ ☆☆☆☆☆

标签： 数学 数论 质数

题目描述

GJX 在统计报名人数的时候发现了一个有趣的现象：以 29 为例，如果分别使用它对前 3 个质数 2、3、5 取模，可以分别得到 1、2、4；反之，如果我们已知一个数字 x 模 2 余 1、模 3 余 2、模 5 余 4，这些条件能够确定的最小正整数 x 就是 29。

GJX 不禁猜想，如果已知一个整数 x 分别对前 k 个质数取模的结果，当 k 充分大时，我们使用这些条件就可以确定 x 的值。

现在输入 x，请你找出尽可能小的 k，使得仅仅已知 x 对前 k 个质数取模的结果时，可以确定的最小正整数是 x。

输入描述

一行，一个正整数 x $(2 \le x \le 10^{18})$

输出描述

一行，一个整数 k，代表题面中所求的 k

题解

1. $O(1)$: Wrong Answer 不同的质数两两互质，根据某数 x 对两个不同的质数 a, b 取模的结果，一定可以在 $[1, ab]$ 中找到唯一一个满足条件的数 y。若想使得 $x = y$，则需要 $x \le ab$，对 k 个不同的质数取模同理。从 2 开始暴力枚举质数或直接打表，同时计算前 i 个质数的乘积，找到满足要求的 i 即为 k

2. $O(1)$: Accepted 在 1. 的基础上，特判 $i = 16$ 的情况，此时乘积会超过 $long\ long\ int$ 而变为负数

$unsigned\ long\ long$ 选手，$\_\_int128$ 选手，$BigInteger$ 选手，$Python$ 选手无需考虑特判

参考代码

C

```c
#include <stdio.h>
#include <stdbool.h>
using namespace std;
int ans;
long long x, mul = 1;
bool flag;
int main()
{
    scanf("%lld", &x);
    for (int i = 2; mul < x; i++) {
        flag = 1;
```

```
        for (int  j = 2; j * j <= i; j++) {
            if (!(i % j)) {
                flag = 0;
                break;
            }
        }
        if (flag) {
            ans++;
            if (ans == 16) break;
            mul *= i;
        }
    }
    printf("%d\n", ans);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
int ans;
long long x;
__int128 mul = 1;
bool flag;
int main()
{
    cin >> x;
    for (int i = 2; mul < x; i++) {
        flag = 1;
        for (int  j = 2; j * j <= i; j++) {
            if (!(i % j)) {
                flag = 0;
                break;
            }
        }
        if (flag) {
            ans++;
            mul *= i;
        }
    }
    cout << ans << endl;
    return 0;
}
```

Java

```
import java.util.Scanner;
import java.math.BigInteger;
public class Main {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        BigInteger x = stdin.nextBigInteger(), mul = BigInteger.valueOf(1);
        int ans = 0;
        boolean flag;
        for (int i = 2; mul.compareTo(x) < 0; i++) {
            flag = true;
```

```
            for (int j = 2; j * j <= i; j++) {
                if (i % j == 0) {
                    flag = false;
                    break;
                }
            }
            if (flag == true) {
                ans++;
                mul = mul.multiply(BigInteger.valueOf(i));
            }
        }
        System.out.println(ans);
    }
}
```

Python3

```
from functools import reduce
prime = list(filter(lambda n : min(map(lambda i : n == 2 or n % i, range(2, max(n, 3)))), range(1, 100)))
x = int(input())
print(min(filter(lambda i : reduce(lambda n, m : n * m, prime[0:i + 1]) >= x, range(1, 20))))
```

D 昔日的军训时光

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ☆☆☆☆☆

标签： 前缀和 动态规划 环形动态规划

题目描述

比赛前夕，同学们纷纷围在 GJX 身边，求他讲一讲自己刚刚来到学校时的故事，GJX 想了想，讲起了自己军训时的趣事。

军训刚刚开始的时候，同学们被分成了三个连。有一天，三个连的同学们围成了一个圈在做游戏，突然有一个同学看到教官走了过来，大家立刻慌了神。

按照军训的要求，同一个连的同学必须站在一起（同一个连的同学在圈中的位置要连续），可是随着教官的靠近，大范围的移动很容易被立刻发现！GJX 提议每次选择两个同学交换位置，只要交换位置的次数足够少，他们就有可能在教官发现之前调整到合适的位置，现在请你帮助 GJX 解决这个问题。

输入描述

第一行，一个整数 n $(1 \leq n \leq 2 \times 10^5)$，表示同学的数量；

第二行，一个长度为 n 的字符串，仅包含 1、2、3，代表围成一圈的同学中，从某一位同学开始，沿顺时针方向依次对应的每一位同学所在连队的编号

输出描述

一行，一个整数，表示最少的交换次数

题解

1. $O(n)$: Accepted 破环成链，并计算各数字出现次数的前缀和。枚举环的断开点，再枚举三个连期望的排列顺序。根据期望排列，分别计算排在左边、中间、右边的连中被其他连的同学占用的位置数，例如用 $t[a][b]$ 表示第 a 连中被第 b 连的同学占用的位置数。那么，可以交换 $t[a][b], t[b][a]$ 中 $\min(t[a][b],\ t[b][a])$ 组同学，从而使他们回到正确位置，剩余 $|t[a][b] - t[b][a]|$ 位同学，均至少需要交换 2 次才能回到正确位置

此情况下
$ans = \min(t[1][2],\ t[2][1]) + \min(t[1][3],\ t[3][1]) + \min(t[2][3],\ t[3][2]) + 2|t[1][2] - t[2][1]|$

参考代码

C

```c
#include <stdio.h>
#include <stdbool.h>
#define maxn 200005
#define INF 1234567890
int n, ans = INF, sum[maxn << 1][4], num[4], st[4], t[4][4];
bool vis[4];
char ch, s[maxn << 1];
int min(int x, int y) {return x < y ? x : y;}
```

```
int abs(int x) {return x > 0 ? x : -x;}
void Dfs(int l, int step) {
    if (step == 4) {
        for (int i = 1, a = 0; i <= 3; a += num[st[i]], i++) {
            for (int j = 1; j <= 3; j++) {
                t[i][j] = sum[l + a + num[st[i]] - 1][st[j]] - sum[l + a - 1]
[st[j]];
            }
        }
        ans = min(ans, min(t[1][2], t[2][1]) + min(t[1][3], t[3][1]) + min(t[2][3],
t[3][2]) + 2 * abs(t[1][2] - t[2][1]));
        return;
    }
    for (int i = 1; i <= 3; ++i) {
        if (vis[i]) continue;
        vis[i] = 1;
        st[step] = i;
        Dfs(l, step + 1);
        vis[i] = 0;
    }
}
int main()
{
    scanf("%d", &n);
    while ((ch = getchar()) != '\n') continue;
    for (int i = 1; i <= n; i++) {
        scanf("%c", s + i);
        s[n + i] = s[i];
    }
    for (int i = 1; i <= 2 * n; i++) {
        for (int j = 1; j <= 3; j++)
            sum[i][j] = sum[i - 1][j];
        sum[i][s[i] ^ 48]++;
        num[s[i] ^ 48]++;
    }
    for (int i = 1; i <= 3; i++)
        num[i] /= 2;
    for (int i = 1; i <= n; i++)
        Dfs(i, 1);
    printf("%d\n", ans);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 200005
#define INF 1234567890
int n, ans = INF, sum[maxn << 1][4], num[4], st[4], t[4][4];
bool vis[4];
string s;
inline void Dfs(int l, int step) {
    if (step == 4) {
        for (int i = 1, a = 0; i <= 3; a += num[st[i]], i++) {
            for (int j = 1; j <= 3; j++) {
```

```
                t[i][j] = sum[l + a + num[st[i]] - 1][st[j]] - sum[l + a - 1]
[st[j]];
            }
        }
        ans = min(ans, min(t[1][2], t[2][1]) + min(t[1][3], t[3][1]) + min(t[2][3],
t[3][2]) + 2 * abs(t[1][2] - t[2][1]));
        return;
    }
    for (int i = 1; i <= 3; ++i) {
        if (vis[i]) continue;
        vis[i] = 1;
        st[step] = i;
        Dfs(l, step + 1);
        vis[i] = 0;
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> s;
    s = " " + s + s;
    for (int i = 1; i <= 2 * n; i++) {
        for (int j = 1; j <= 3; j++)
            sum[i][j] = sum[i - 1][j];
        sum[i][s[i] ^ 48]++;
        num[s[i] ^ 48]++;
    }
    for (int i = 1; i <= 3; i++)
        num[i] /= 2;
    for (int i = 1; i <= n; i++)
        Dfs(i, 1);
    cout << ans << endl;
    return 0;
}
```

E 寻找钥匙

时间限制： 1000ms
空间限制： 256MB
难度： ★★☆☆☆ ☆☆☆☆☆

标签： 数学

题目描述

竞赛在即，服务器却突然失去了响应，心急如焚的 GJX 急忙赶往网络中心。到了门口，GJX 突然想起，自己在前不久刚刚把钥匙弄丢了，因此 GJX 不得不求助楼层管理员寻求服务器所在房间的钥匙。

网络中心有很多房间，每个房间有两把钥匙，每把钥匙上都写有一个编号。楼层管理员拉开存放钥匙的抽屉，让 GJX 自己寻找钥匙，粗心的 GJX 甚至忘记了服务器所在房间的编号！

所幸的是，除了服务器所在的房间外，其余所有房间的两把钥匙都在抽屉里—而 GJX 想要找的钥匙在抽屉里只有一把，因为另一把被他丢掉了……

现在输入抽屉中所有钥匙的编号，请你为 GJX 找到他想要的钥匙（只出现了一次的编号）。

输入描述

第一行，一个整数 $n\ (1 \leq n < 5 \times 10^6)$，表示钥匙的数量，数据保证 n 为奇数；

第二行，n 个正整数（保证在 int 范围内），对应每把钥匙的编号，有一个编号只出现一次，其余编号出现两次

输出描述

一行，一个整数，输出 GJX 所想找的钥匙的编号

题解

1. $O(n \log_2 n)$：Time Limit Exceeded 将所有钥匙按编号排序，找到单独编号的一把

2. $O(n)$：Runtime Error Memory Limit Exceeded Compile Error 使用一个数组记录各编号出现的次数，找到只出现一次的编号

3. $O(n \log_2 n)$：Time Limit Exceeded 使用 map 记录各编号出现的次数，找到只出现一次的编号，常数大

4. $O(n)$：Time Limit Exceeded 使用 $unordered\_map$ 记录各编号出现的次数，找到只出现一次的编号，常数大

5. $O(n)$：Accepted 由于 $a \oplus a = 0, 0 \oplus a = a$，所有数的异或和结果就是只出现一次的编号

参考代码

C

```c
#include <stdio.h>
int n, x, ans;
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &x);
        ans ^= x;
    }
    printf("%d\n", ans);
    return 0;
}
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, x, ans;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for(int i = 1; i <= n; i++) {
        cin >> x;
        ans ^= x;
    }
    cout << ans << endl;
    return 0;
}
```

Java

```java
// 900+ ms, May be TLE sometimes.
import java.io.BufferedInputStream;
import java.io.IOException;
public class Main {
    public static final BufferedInputStream stdin;
    static {
        stdin = new BufferedInputStream(System.in, 0x100000);
    }
    public static int quickRead() throws IOException {
        int res = 0, sign = 1, in;
        while ((in = stdin.read()) != -1 && in > 0x20) {
            if (in == '+' || in == '-') {
                sign *= in == '-' ? -1 : 1;
                continue;
            }
            res = res * 10 + in - '0';
        }
        return res * sign;
    }
    public static void main(String[] args) throws IOException {
        int n = quickRead(), ans = 0;
        while (n-- != 0) {
            ans ^= quickRead();
```

```
        }
        System.out.println(ans);
    }
}
```

Python3

```python
# TLE
n = int(input())
ans = 0
for i in range(1, n + 1):
    x = int(input())
    ans ^= x
print(ans)
```

F GJX的兄弟们

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★☆☆☆☆☆

标签： 树形结构 并查集 图论 深度优先搜索DFS 广度优先搜索BFS

题目描述

在 GJX 复杂的交际圈中，每个人（包括 GJX 自己）有可能有父亲或儿子。每个人最多有一个父亲，但可以有多个儿子。在这样一个交际圈中，“辈分”就这样产生了。

对于每个人，他的父亲的辈分是他的辈分 $+1$，他的儿子的辈分是他的辈分 -1，在这个交际圈中，所有辈分相同的人互相称彼此为“兄弟”。

现在，请帮助 GJX 算一算，他有多少个兄弟。

输入描述

第一行，一个整数 $n\ (1 \leq n \leq 2 \times 10^5)$，表示 GJX 的交际圈中一共有多少人；

接下来的 $n-1$ 行，每行两个用空格分开的整数 $a, b\ (1 \leq a, b \leq n)$，代表编号为 b 的人是编号为 a 的人的儿子。

最后一行，一个整数 $c\ (1 \leq c \leq n)$，代表 GJX 的编号

输出描述

一行，一个整数，代表 GJX 的兄弟的数量（不包括 GJX 自己）

题解

1. $O(n)$: Accepted 使用并查集将所有结点合并成一棵树，同时维护每个结点到根节点的距离

2. $O(n)$: Accepted 选择合适的结构建树，从根节点开始，进行 DFS 或 BFS 遍历并记录每个结点的深度

参考代码

C

```c
#include <stdio.h>
#define maxn 200005
int n, ans, cnt, head[maxn], fa[maxn], dis[maxn];
struct Edge{int u, v, next;} edge[maxn];
void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
void Dfs(int u) {
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].v;
        dis[v] = dis[u] + 1;
```

```
        Dfs(v);
    }
}
int main()
{
    scanf("%d", &n);
    int u, v;
    for (int i = 1; i < n; i++) {
        scanf("%d %d", &u, &v);
        add(u, v);
        fa[v] = u;
    }
    for (int i = 1; i <= n; i++)
        if (!fa[i]) Dfs(i);
    scanf("%d", &u);
    for (int i = 1; i <= n; i++)
        if (dis[i] == dis[u]) ans++;
    printf("%d\n", ans - 1);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 200005
int n, ans, cnt, head[maxn], fa[maxn], dis[maxn];
struct Edge{int u, v, next;} edge[maxn];
inline void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
inline void Dfs(int u) {
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].v;
        dis[v] = dis[u] + 1;
        Dfs(v);
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    int u, v;
    for (int i = 1; i < n; i++) {
        cin >> u >> v;
        add(u, v);
        fa[v] = u;
    }
    for (int i = 1; i <= n; i++)
        if (!fa[i]) Dfs(i);
    cin >> u;
    for (int i = 1; i <= n; i++)
        if (dis[i] == dis[u]) ans++;
```

```
    cout << ans - 1 << endl;
    return 0;
}
```

G 终极较量

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ★☆☆☆☆

标签： 排序 图论 深度优先搜索DFS 广度优先搜索BFS

题目描述

比赛很快就开始了，所有参赛的选手将在这里展开一场全面的终极较量。

每一位选手都有三个属性 A、B、C，每个属性都是一个正整数，如果一个选手 X 有一个属性的值大于另一个选手 Y 对应的属性值，则认为 X 可以击败 Y；如果 X 能够击败 Y，Y 能够击败 Z，则认为 X 也能击败 Z；在一个选手能够击败的人中，每个人统计且仅统计一次，请你帮 GJX 算一算，他能够击败多少角色。

特别的，选手自己不能击败自己，即自己不会出现在自己击败的计数中。

输入描述

第一行，一个整数 $n\ (1 \le n \le 2 \times 10^5)$，代表该游戏中的所有玩家数；

第二行，3 个正整数（不超过 int），分别代表 GJX 的三个属性值；

接下来的 $n-1$ 行，每行 3 个数，代表除了 GJX 以外的选手的三个属性的值；

数据保证不会出现两位选手的所有属性值相同

输出描述

一行，一个整数，代表 GJX 最多可以击败多少选手一（不包括 GJX 自己）

题解

1. $O(n \log_2 n)$：Accepted 分别按三个属性值的大小对所有的选手进行排序，选择合适的结构建图，排序后相邻的选手之间建立一条有向边，从属性值高的选手指向属性值低的选手。从 GJX 开始，进行 DFS 或 BFS，遍历其能击败的选手并记录数量。

参考代码

C

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define maxn 200005
#define maxm 600005
int n, ans, cnt, head[maxn], q[maxn];
bool vis[maxn];
typedef struct {int id, a, b, c;} Player;
Player player[maxn];
typedef struct {int u, v, next;} Edge;
Edge edge[maxm];
int cmp1(const void * x, const void * y) {
    Player a = *(Player *) x;
```

```
    Player b = *(Player *) y;
    return a.a < b.a;
}
int cmp2(const void * x, const void * y) {
    Player a = *(Player *) x;
    Player b = *(Player *) y;
    return a.b < b.b;
}
int cmp3(const void * x, const void * y) {
    Player a = *(Player *) x;
    Player b = *(Player *) y;
    return a.c < b.c;
}
void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf("%d %d %d", &player[i].a, &player[i].b, &player[i].c);
        player[i].id = i;
    }
    qsort(player + 1, n, sizeof(Player), cmp1);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    qsort(player + 1, n, sizeof(Player), cmp2);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    qsort(player + 1, n, sizeof(Player), cmp3);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    int l = 1, r = 0;
    q[++r] = 1;
    vis[1] = 1;
    while (l <= r) {
        int u = q[l++];
        ans++;
        for (int i = head[u]; i; i = edge[i].next) {
            int v = edge[i].v;
            if (!vis[v]) {
                q[++r]=v;
                vis[v] = 1;
            }
        }
    }
    printf("%d\n", ans - 1);
    return 0;
}
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
#define maxn 200005
#define maxm 600005
int n, ans, cnt, head[maxn];
bool vis[maxn];
struct Player{int id, a, b, c;} player[maxn];
struct Edge{int u, v, next;} edge[maxm];
inline bool cmp1(Player x, Player y) {return x.a > y.a;}
inline bool cmp2(Player x, Player y) {return x.b > y.b;}
inline bool cmp3(Player x, Player y) {return x.c > y.c;}
inline void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> player[i].a >> player[i].b >> player[i].c;
        player[i].id = i;
    }
    sort(player + 1, player + n + 1, cmp1);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    sort(player + 1, player + n + 1, cmp2);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    sort(player + 1, player + n + 1, cmp3);
    for (int i = 2; i <= n; i++)
        add(player[i-1].id, player[i].id);
    queue<int> q;
    q.push(1);
    vis[1] = 1;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        ans++;
        for (int i = head[u]; i; i = edge[i].next) {
            int v = edge[i].v;
            if (!vis[v]) {
                q.push(v);
                vis[v] = 1;
            }
        }
    }
    cout << ans - 1 << endl;
    return 0;
}
```

H GJX的数学游戏

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ★★★☆☆

标签： 数学 数论 高性能

题目描述

我们把从 1 到 n 的 n 个数字的排成一列形成的序列叫做一个“n 阶排列”，在此基础之上，我们可以定义排列与排列的大小关系：

权值：我们首先规定一个排列中的每一个数字 a_i 都拥有一个对应的权值 $p(a_i)$，权值将会在输入中给出；

大小：一个排列 A 小于排列 B，当且仅当排列 A、B 的前若干项（可以为 0 项）相同，并且在 A、B 的从左往右数第一个不同项（假定为第 i 项）上，成立 $p(a_i) < p(b_i)$。

有了排列之间大小关系的定义，现在我们知道，对于 $n!$ 个不同的 n 阶排列，我们一定可以把它们按照从小到大的顺序排成一列从而得到一个由排列构成的序列 S，记序列 S 的第 i 项为 S_i。

定义函数 $f(x)$ 为序列 S 中的第 x 项（即排列 S_i）的每一个数字的正负交错和（$f(x) = \sum_{i=1}^{n} (-1)^{i+1} (S_x)_i$），现在给定 m 个数：b_1、b_2、… b_m，求 $\sum_{i=1}^{m} f(b_i)$。

输入描述

第一行，一个整数 $n\ (1 \le n \le 10^5)$，代表题目中的 n；

第二行，n 个用空格分开的正整数 a_1、a_2、… a_n，是一个 n 阶排列，代表一个 n 阶排列中，数字 i 的权值是 a_i；

第三行，一个整数 $m\ (1 \le m \le 10^5)$，代表题目中的 m；

第四行，m 个用空格分开的正整数 b_1、b_2、… $b_m\ (1 \le b_i \le \min(n!,\ 10^{18}))$，代表题目中的 m 个数

输出描述

一行，一个整数，输出题目中所求的答案

题解

1. $O(n!\ m)$：Time Limit Exceeded 暴力枚举计算

2. $O(n + 20!\ m)$：Time Limit Exceeded 由于 $1 \le b_i \le \min(n!,\ 10^{18})$，而 $19! < 10^{18} < 20!$。因此，当 $n \ge 20$ 时，排列 S_{b_i} 与 S_1 的区别只可能在最后 20 个数，也就是说，题目中所询问的所有排列的前 $n-20$ 个数是完全相同的，均为 $\{1, 2, 3, \ldots, n-20\}$，很容易求出前 $n-20$ 个数的正负交错和。注意，根据题目要求，集合中的数 i 代表权值第 i 大的数。对于最后 20 个数，单独进行处理，相当于处理一个 $n = 20$ 的新排列。对于每个 b_i，暴力枚举相应的排列，再进行计算

3. $O(n + 20^2 m)$：Accepted 在 2. 的基础上，对于每个 b_i，通过“变进制数-阶乘数系”将其转化为 $b_i = \sum_{j=0}^{19} k_j\ j!$，其中 $0 \le k_j \le j$

为了方便理解，我们以 $n=5$ 为例，如 $5=2\times 2!+1\times 1!+0\times 0!$ 即 $5=(00210)$。之后通过“变进制数-阶乘数系”的性质，找到对应的排列：从首位开始，第 j 位上的数便是剩余数中第 k_j+1 小的数，例如 $5=(00210)$ 可以得到排列 $\{1,2,5,4,3\}$，以及 $119=(43210)\Rightarrow\{5,4,3,2,1\}$，注意 $1=(00010)\Rightarrow\{1,2,3,5,4\}$ 而 $0=(00000)\Rightarrow\{1,2,3,4,5\}$

在得到相应排列之后，暴力计算答案。特别地，当 $n<20$ 时，直接使用以上方法计算即可

参考代码

C

```c
#include <stdio.h>
#include <stdbool.h>
#define maxn 100005
int n, m, a[maxn], k[25];
long long sum,ans;
bool vis[25];
int min(int x, int y) {return x < y ? x : y;}
int max(int x, int y) {return x > y ? x : y;}
void Solve(long long x) {
    int mn = min(n, 20), mx = max(0, n - 20);
    for (int i = mn; i >= 1; i--) {
        k[i] = x % (mn + 1 - i);
        x /= (mn + 1 - i);
        vis[i] = 0;
    }
    for (int i = 1; i <= mn; i++)
        for (int j = 1, t = 0; j <= mn; j++) {
            if (vis[j]) continue;
            if (t == k[i]) {
                vis[j] = 1;
                ans += (mx + i) & 1 ? a[j + mx] : -a[j + mx];
                break;
            }
            t++;
        }
}
int main()
{
    scanf("%d", &n);
    long long x;
    for(int i = 1; i <= n; i++) {
        scanf("%lld", &x);
        a[x] = i;
    }
    if (n >= 20)
        for (int i = 1; i <= n - 20; i++)
            ans += i & 1 ? a[i] : -a[i];
    scanf("%d", &m);
    ans *= m;
    while(m--) {
        scanf("%lld", &x);
        Solve(x - 1);
    }
    printf("%lld\n", ans);
    return 0;
}
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
#define maxn 100005
int n, m, a[maxn], k[25];
long long sum,ans;
bool vis[25];
inline void Solve(long long x) {
    int mn = min(n, 20), mx = max(0, n - 20);
    for (int i = mn; i >= 1; i--) {
        k[i] = x % (mn + 1 - i);
        x /= (mn + 1 - i);
        vis[i] = 0;
    }
    for (int i = 1; i <= mn; i++)
        for (int j = 1, t = 0; j <= mn; j++) {
            if (vis[j]) continue;
            if (t == k[i]) {
                vis[j] = 1;
                ans += (mx + i) & 1 ? a[j + mx] : -a[j + mx];
                break;
            }
            t++;
        }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    long long x;
    for(int i = 1; i <= n; i++) {
        cin >> x;
        a[x] = i;
    }
    if (n >= 20)
        for (int i = 1; i <= n - 20; i++)
            ans += i & 1 ? a[i] : -a[i];
    cin >> m;
    ans *= m;
    while(m--) {
        cin >> x;
        Solve(x - 1);
    }
    cout << ans << endl;
    return 0;
}
```

I 最短路径

时间限制： 1000ms
空间限制： 256MB
难度： ★★★☆☆ ☆☆☆☆☆

标签： 图论 哈密顿回路

题目描述

给定一个 $n \times m$ 的矩阵状点阵，相邻两点（可以是横向相邻、纵向相邻、斜向相邻）的距离只可能是 1 或 $\sqrt{2}$，从一个点出发，每次只允许前进到相邻的一个点。

现在给定 n、m，求从左上角的点出发，至少走过每一个点并最终回到起点的最短路径。

输入描述

一行，两个整数 n、m $(1 \leq n,\ m \leq 2 \times 10^5)$

输出描述

一行，一个实数，保留两位小数，代表题目中所求的最短路径

题解

$O(nm)$：Runtime Error Memory Limit Exceeded Compile Error Time Limit Exceeded 搜索整张图

$O(1)$：Accepted 构造一条哈密顿回路，通过观察或者暴力可以发现，当 $n, m > 1$ 时，若 n, m 中至少有一个是偶数，则答案为 nm，否则答案为 $nm + \sqrt{2} - 1$；特别地，若 n, m 中至少有一个为 1 时，答案为 $\max\{2(n-1),\ 2(m-1)\}$

参考代码

C

```c
#include <stdio.h>
int max(int a, int b) {return a > b? a : b;}
int n, m;
double ans;
int main()
{
    scanf("%d %d", &n, &m);
    if (n == 1 || m == 1) ans = 2.0 * max(n - 1, m - 1);
    else if (n * m & 1) ans = 1.0 * n * m + sqrt(2) - 1;
    else ans = 1.0 * n * m;
    printf("%.2lf\n", ans);
    return 0;
}
```

C++

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, m;
double ans;
int main()
{
    cin >> n >> m;
    if (n == 1 || m == 1) ans = 2.0 * max(n - 1, m - 1);
    else if (n * m & 1) ans = 1.0 * n * m + sqrt(2) - 1;
    else ans = 1.0 * n * m;
    cout << fixed << setprecision(2) << ans << endl;
    return 0;
}
```

Java

```java
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        int n = stdin.nextInt(), m = stdin.nextInt();
        double ans;
        if (n == 1 || m == 1) {
            ans = 2.0 * Math.max(n - 1, m - 1);
        } else if ((n * m & 1) != 0) {
            ans = 1.0 * n * m + Math.sqrt(2) - 1;
        } else {
            ans = 1.0 * n * m;
        }
        System.out.printf("%.2f\n", ans);
    }
}
```

Python3

```python
import math
n, m = [int(i) for i in input().split(" ")]
ans = float()
if n == 1 or m == 1:
    ans = 2 * max(n - 1, m - 1)
elif n * m & 1:
    ans = n * m + math.sqrt(2) - 1
else:
    ans = n * m
print("%.2f" %ans)
```

J 调皮的GJX

时间限制： 1000ms
空间限制： 256MB
难度： ★★★☆☆ ☆☆☆☆☆

标签： 模拟 暴力 字符串

题目描述

一位同学想要报名参加兰州大学第四届程序设计大赛，但是不知道是否来得及，于是他给比赛的工作人员 GJX 发送了邮件，但是 GJX 很调皮，把回信写成了一行仅由 0、1 和 $-$ 组成的二进制字符串，并附上了一份“密码表”。

如果回信中包含 `WELCOME`，则意味着这位同学报名成功了，现在请你帮这位同学看看 GJX 的回信，看看他是否报名成功。

输入描述

第一行，一个字符串，代表回信的全部内容。回信的格式是用`-`分隔成的若干个长度为 5、只由 0、1 组成的二进制字符串，长度不会超过 5000；

接下来的 26 行，每行一个大写字母和一个长度为 5、只由 0、1 组成的二进制字符串，中间用空格分隔，表示与字母对应的二进制串

输出描述

一行，一个字符串，如果使用“密码表”将回信还原后（去掉 `-`）包含 `WELCOME` 则输出 `YES`，否则输出 `NO`

题解

$O(length)$：Accepted 记录字母与编码的对应关系，暴力将原字符串中的编码转化为字母，查找是否存在 `WELCOME`；或直接暴力查找原字符串中是否存在 `W-E-L-C-O-M-E` 所对应的编码

参考代码

C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define maxn 5005
int x, len;
char s[maxn], t[maxn], ans[maxn], c, dic[maxn << 2];
int main()
{
    scanf("%s", s);
    while ((c = getchar()) != '\n') continue;
    for (int i = 1; i <= 26; i++) {
        scanf("%c %d", &c, &x);
        dic[x] = c;
        while ((c = getchar()) != '\n') continue;
    }
    for (int i = 0; s[i] != '\0'; i += 6) {
```

```
        strncpy(t, s + i, 5);
        ans[len++] = dic[atoi(t)];
    }
    if (strstr(ans, "WELCOME")) printf("YES\n");
    else printf("NO\n");
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
int k;
char c;
string s, t, x;
map<char, string> dic;
int main()
{
    cin >> s;
    for (int i = 1; i <= 26; i++) {
        cin >> c >> x;
        dic[c] = x;
    }
    t += dic['W'] + '-' + dic['E'] + '-' + dic['L'] + '-' + dic['C'] + '-' + dic['O'] + '-' + dic['M'] + '-' + dic['E'];
    if ((k = s.find(t)) != string::npos) cout << "YES" << endl;
    else cout << "NO" << endl;
    return 0;
}
```

Java

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner stdin = new Scanner(System.in);
        String message[] = stdin.nextLine().split("-");
        Map<String, String> table = new HashMap<String, String>();
        String maps[];
        for (int i = 0; i < 26; ++i) {
            maps = stdin.nextLine().split(" ");
            table.put(maps[1], maps[0]);
        }
        StringBuilder builder = new StringBuilder();
        for (String iterator : message) {
            builder.append(table.get(iterator));
        }
        System.out.println(builder.toString().contains("WELCOME") ? "YES" : "NO");
    }
}
```

Python3

```python
s = list(input().split("-"))
dic = {}
for i in range(26):
    c, x = list(input().split())
    dic[x] = c

for i in range(len(s)):
    s[i] = dic[s[i]]

if "WELCOME" in "".join(s):
    print("YES")
else:
    print("NO")
```

K 回文整数求和

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ☆☆☆☆☆

标签： 构造 前缀和 二分查找 高性能

题目描述

如果一个字符串是轴对称的，即这个字符串的正数第 i 个字符等于倒数第 i 个字符，我们就称这个字符串是“回文串”。

GJX 很好奇都有哪些整数的无前导零（除 0 外，最高位不是 0 ）的十进制表示是一个回文串，现在请你帮他解决这个问题。

输入描述

第一行，一个整数 T $(1 \leq T \leq 10^5)$，表示一共有 T 组样例；

接下来的 T 行，每行两个整数a, b $(1 \leq a \leq b \leq 10^9)$，代表一个闭区间 $[a,\ b]$

输出描述

T 行，每行一个整数，对于每个给定的闭区间，输出闭区间内的所有无前导零的十进制表示是回文串的整数的和

题解

1. $O(T(b-a))$: Time Limit Exceeded 对于每组询问，暴力枚举区间的所有数，判断其是否为回文数

2. $O(10^9+T)$: Runtime Error Memory Limit Exceeded Compile Error Time Limit Exceeded 暴力预处理出所有回文数，并计算前缀和，对于每组询问，通过前缀和求解

3. $O(500000+100000T)$: Time Limit Exceeded 通过构造法预处理出 $[1, 10^9]$ 内的回文数，并计算前缀和。对于每组询问，枚举区间左右端点，通过前缀和求解

4. $O(500000+T(\log_2 a+\log_2 b))$: Accepted 在 3. 的基础上，对于每组询问，二分查找区间左右端点，通过前缀和求解

参考代码

C

```c
#include <stdio.h>
#define maxn 200005
int T, a, b, num, p[maxn];
long long sum[maxn];
void Dfs(int pos, int len, int now) {
    if (pos > (len + 1) / 2) {
        int t = now;
        if (len & 1) t /= 10;
        while(t) {
            now = (now << 3) + (now << 1) + (t % 10);
            t /= 10;
```

```
        }
        p[++num] = now;
        return;
    }
    if (pos != 1) Dfs(pos + 1, len, (now << 3) + (now << 1));
    for (int i = 1; i <= 9; i++)
        Dfs(pos + 1, len, (now << 3) + (now << 1) + i);
}
int main()
{
    p[++num] = 0;
    for (int len = 1; len <= 9; len++)
        Dfs(1, len, 0);
    p[++num] = 1000000001;
    for (int i = 1; i <= num; i++)
        sum[i] = sum[i-1] + p[i];
    scanf("%d", &T);
    while (T--) {
        scanf("%d %d", &a, &b);
        int l = 1, r = num, l0 = 1, r0 = num;
        while (l < r0) {
            int mid = (l + r0) >> 1;
            if (p[mid] < a) l = mid + 1;
            else r0 = mid;
        }
        while (l0 < r) {
            int mid = (l0 + r) >> 1;
            if (p[mid] <= b) l0 = mid + 1;
            else r = mid;
        }
        r--;
        if(l > r)printf("%lld\n", 0ll);
        else printf("%lld\n", sum[r] - sum[l-1]);
    }
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 200005
int T, a, b, num, p[maxn];
long long sum[maxn];
inline void Dfs(int pos, int len, int now) {
    if (pos > (len + 1) / 2) {
        int t = now;
        if (len & 1) t /= 10;
        while (t) {
            now = (now << 3) + (now << 1) + (t % 10);
            t /= 10;
        }
        p[++num] = now;
        return;
    }
    if (pos != 1) Dfs(pos + 1, len, (now << 3) + (now << 1));
    for (int i = 1; i <= 9; i++)
```

```cpp
        Dfs(pos + 1, len, (now << 3) + (now << 1) + i);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    p[++num] = 0;
    for (int len = 1; len <= 9; len++)
        Dfs(1, len, 0);
    p[++num] = 1000000001;
    for (int i = 1; i <= num; i++)
        sum[i] = sum[i-1] + p[i];
    cin >> T;
    while (T--) {
        cin >> a >> b;
        int l = lower_bound(p + 1, p + num + 1, a) - p;
        int r = upper_bound(p + 1, p + num + 1, b) - p - 1;
        if (l > r) cout << 0 << endl;
        else cout << sum[r] - sum[l-1] << endl;
    }
    return 0;
}
```

L 子序列

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★☆ ☆☆☆☆☆

标签： 前缀和 哈希HASH

题目描述

给定一个整数序列 A，记序列 A 的每一项分别为：A_1、A_2、… A_n，求序列 A 中满足 $\sum_{i=l}^{r} A_i = |r-l+1|\ (l \le r)$ 的连续子序列的个数。

输入描述

第一行，一个整数 $n\ (1 \le n \le 10^5)$，代表序列 A 的长度；

第二行，n 个整数 A_1、A_2、… $A_n\ (-10^5 \le A_i \le 10^5)$

输出描述

一行，一个整数，代表满足条件的子序列的个数

题解

1. $O(n^2)$：Time Limit Exceeded 计算前缀和，枚举区间左右端点，通过前缀和计算

2. $O(n)$：Runtime Error Memory Limit Exceeded Compile Error 所有元素减 1，计算前缀和，用数组记录各值出现次数，前缀和相同的任意两者可作为一种答案

3. $O(n \log_2 n)$：Accepted 在 2. 的基础上，将数组换为 map 记录各值出现次数，常数大

4. $O(n)$：Accepted 在 2. 的基础上，将数组换为 $unordered\_map$ 记录各值出现次数，常数大

参考代码

C

```c
// It is extremely difficult to find a suitable HASH function.
// Also you can use Treap to solve it.
#include <malloc.h>
#include <stdio.h>
#include <string.h>
#define ll long long
#ifndef NULL
#define NULL ((void*)0)
#endif
struct entry {
    ll hashcode, key, value;
    struct entry* next;
};
struct entry* getEntry(ll _hashcode, ll _key, ll _value, struct entry* _next) {
    struct entry* _entry = (struct entry*)malloc(sizeof(struct entry));
    if (_entry == NULL) {
        return NULL;
```

```
    }
    _entry->hashcode = _hashcode;
    _entry->key = _key;
    _entry->value = _value;
    _entry->next = _next;
    return _entry;
}
ll hash(ll _hash) {
    _hash ^= (_hash >> 20) ^ (_hash >> 12);
    return _hash ^ (_hash >> 7) ^ (_hash >> 4);
}
int mapindex(ll _hash, int _length) {
    return (int)_hash & (_length - 1);
}
struct hashmap {
    int map_size, map_capacity;
    struct entry* (*table);
};
struct hashmap* getHashMap(int _capacity) {
    struct hashmap* _hashmap = (struct hashmap*)malloc(sizeof(struct hashmap));
    if (_hashmap == NULL) {
        return NULL;
    }
    _hashmap->map_capacity = _capacity;
    _hashmap->map_size = 0;
    _hashmap->table = (struct entry**)malloc(sizeof(struct entry*) * _capacity);
    if (_hashmap->table == NULL) {
        free(_hashmap);
        return NULL;
    }
    memset(_hashmap->table, 0, sizeof(struct entry*) * _capacity);
    return _hashmap;
}
int get(struct hashmap* _hashmap, ll _key) {
    ll _hash = hash(_key);
    for (struct entry* it_entry = _hashmap->table[mapindex(_hash, _hashmap->map_capacity)];
        it_entry != NULL;
        it_entry = it_entry->next) {
        if (it_entry->hashcode == _hash && it_entry->key == _key) {
            return (int)(it_entry->value);
        }
    }
    return -1;
}
void addEntry(struct hashmap* _hashmap, ll _hash, ll _key, ll _value, int _index)
{
    struct entry* _entry = _hashmap->table[_index];
    _hashmap->table[_index] = getEntry(_hash, _key, _value, _entry);
}
void put(struct hashmap* _hashmap, ll _key, ll _value) {
    ll _hash = hash(_key);
    for (struct entry* it_entry = _hashmap->table[mapindex(_hash, _hashmap->map_capacity)];
        it_entry != NULL;
        it_entry = it_entry->next) {
        if (it_entry->hashcode == _hash && it_entry->key == _key) {
            it_entry->value = _value;
```

```
            return;
        }
    }
    addEntry(_hashmap, _hash, _key, _value, mapindex(_hash, _hashmap->map_capacity));
}
int containsKey(struct hashmap* _hashmap, ll _key) {
    return get(_hashmap, _key) >= 0;
}
#define CAPACTTY 40000000
#define maxn 100010
int n;
int aa[maxn];
ll sum[maxn];
struct hashmap* map;
void solve()
{
    ll ans = 0;
    put(map, 0, 1);
    for(int i = 1; i <= n; ++i)
    {
      if(containsKey(map, sum[i]-i)) {
        ans += get(map, sum[i] - i);
        put(map, sum[i] - i, get(map, sum[i] - i) + 1);
      }
      else
        put(map, sum[i] - i, 1);
    }
    printf("%lld", ans);
}
int main()
{
    map = getHashMap(CAPACTTY);
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
            scanf("%d", &aa[i]);
        sum[i] = sum[i - 1] + aa[i];
    }
    solve();
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 100005
int n, a[maxn];
long long ans, sum[maxn];
unordered_map<long long, long long> m;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
```

```
        sum[i] = sum[i - 1] + a[i];
    }
    m[0] = 1;
    for (int i = 1; i <= n; i++) {
        ans += m[sum[i] - i];
        m[sum[i] - i]++;
    }
    cout << ans << endl;
    return 0;
}
```

M 方阵的GJX数

时间限制： 1000ms
空间限制： 256MB
难度： ★★★☆☆ ☆☆☆☆☆

标签： 数学 数论 质数

题目描述

对于一个给定的 n 阶方阵，定义“GJX 数”为：方阵中所有行号和列号（从 1 数）中有且仅有一个是质数的元素之积，给定一个方阵，求方阵的 GJX 数。

输入描述

第一行，一个整数 $n\ (3 \leq n \leq 2000)$，代表方阵的阶数；

接下来的 n 行，每行 n 个用空格分开的正整数，大小在 int 范围内

输出描述

一行，一个整数，输出方阵的 GJX 数；特别的，考虑到结果可能很大，要求结果对 1000000007 取模之后再输出

题解

$O(n^2\sqrt{n})$：Time Limit Exceeded 枚举每个位置，判断行号和列号是否为质数，按要求计算

$O(n^2)$：Accepted 预处理出 $[1,n]$ 内的所有质数并记录，枚举每个位置，判断行号和列号是否为质数，按要求计算

参考代码

C

```c
#include <stdio.h>
#include <stdbool.h>
#define maxn 2005
#define p 1000000007
int n, cnt, prime[maxn];
long long ans = 1;
bool vis[maxn];
void Get_Prime() {
    vis[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (!vis[i]) prime[++cnt] = i;
        for (int j = 1; j <= cnt && i * prime[j] < maxn; j++) {
            vis[i * prime[j]] = 1;
            if (!(i % prime[j])) break;
        }
    }
}
int main()
{
    Get_Prime();
```

```
    scanf("%d", &n);
    int x;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++) {
            scanf("%d", &x);
            if (vis[i] + vis[j] == 1) ans = ans * x % p;
        }
    printf("%lld\n", ans);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 2005
#define p 1000000007
int n, cnt, prime[maxn];
long long ans = 1;
bool vis[maxn];
inline void Get_Prime() {
    vis[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (!vis[i]) prime[++cnt] = i;
        for (int j = 1; j <= cnt && i * prime[j] < maxn; j++) {
            vis[i * prime[j]] = 1;
            if (!(i % prime[j])) break;
        }
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    Get_Prime();
    cin >> n;
    int x;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++) {
            cin >> x;
            if (vis[i] + vis[j] == 1) ans = ans * x % p;
        }
    cout << ans << endl;
    return 0;
}
```

Java

```
// 900+ ms, May be TLE sometimes.
import java.io.BufferedInputStream;
import java.io.IOException;
public class Main {
    public static final BufferedInputStream stdin;
    static {
        stdin = new BufferedInputStream(System.in, 0x100000);
    }
```

```java
    public static int quickRead() throws IOException {

        int res = 0, sign = 1, in;
        while ((in = stdin.read()) != -1 && in > 0x20) {
            if (in == '+' || in == '-') {
                sign *= in == '-' ? -1 : 1;
                continue;
            }
            res = res * 10 + in - '0';
        }
        return res * sign;
    }
    public static void main(String[] args) throws IOException {
        int n = quickRead(), cnt = 0, prime[] = new int[2005], vis[] = new
int[2005], p = 1000000007;
        long ans = 1;
        for (int i = 1; i < 2005; i++)
            vis[i] = 0;
        vis[1] = 1;
        for (int i = 2; i < 2005; i++) {
            if (vis[i] == 0) prime[++cnt] = i;
            for (int j = 1; j <= cnt && i * prime[j] < 2005; j++) {
                vis[i * prime[j]] = 1;
                if (i % prime[j] == 0) break;
            }
        }
        int x;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++) {
                x = quickRead();
                if (vis[i] + vis[j] == 1) ans = ans * x % p;
            }
        System.out.println(ans);
    }
}
```

Python3

```python
# TLE
vis = [0 for i in range(2006)]
prime = []
vis[1] = 1
for i in range(2, 2005):
    if vis[i] == 0:
        prime.append(i)
    for j in prime:
        if i * j >= 2005:
            break
        vis[i * j] = 1
        if i % j == 0:
            break
n = int(input())
ans = 1
p = 1000000007
for i in range(1,n + 1):
    x = list(input().split(" "))
    for j in range(0,n):
```

```
        if vis[i] + vis[j+1] == 1:
            ans = ans * int(x[j]) % p
print(ans)
```

N 旅游计划

时间限制： 1000ms
空间限制： 256MB
难度： ★★★★★ ★★☆☆☆

标签： RMQ 笛卡尔树 ST表 线段树 树状数组 动态规划

题目描述

经历了漫长的筹备，兰州大学第四届程序设计大赛终于顺利开始了，GJX 计划在比赛结束之后出去旅游，放松一下自己疲惫的身心。

景区一共有 n 个景点，每个景点都要收取一定的门票费。

由于精力有限，GJX 每天至多可以参观 k 个景点。考虑到 GJX 在这段时间的努力，GJX 的爸爸们同意每天为 GJX 报销当天去过的景点中门票价格最高的费用，现在请你帮 GJX 安排旅游计划，使得在花费最少天数游览完所有景点的情况下，使得 GJX 的开支最小。

输入描述

第一行，一个整数 $n\ (1 \le n \le 10000)$，表示旅游景点的个数；

第二行，一个整数 $k\ (1 \le k \le 100)$，表示每天最多参观 k 个景点；

第三行，n 个用空格分开的正整数（不超过 100000），代表 n 个景点的门票费，GJX 的参观顺序与费用的输入顺序一致

输出描述

一行，一个整数，输出 GJX 在保证花费天数最少的情况下，参观完所有景点的最小开支

题解

1. $O(nk^2)$：Time Limit Exceeded 用 $dp[i][j]$ 表示前 i 天参观了 $ik-j$ 个景点时能够获得的最大报销。那么有 $dp[i][j]=\max\{dp[i-1][t]+\max(cost[p])\}$，其中 $0 \le i \le \left\lceil \frac{n}{k} \right\rceil$，$0 \le j \le k-(n \bmod k)$，$0 \le t \le j$，$(i-1)k-t+1 \le p \le ik-j$。最后用总花费减去最大报销

2. $O(n\log_2 n+nk\log_2 n)$：Accepted 在 1. 的基础上，建立并使用树状数组查询区间最大值

3. $O(n+nk\log_2 n)$：Accepted 在 1. 的基础上，建立并使用线段树查询区间最大值

4. $O(n\log_2 n+nk)$：Accepted 在 1. 的基础上，建立并使用 ST 表查询区间最大值

5. $O(n+nk)$：Accepted 在 1. 的基础上，建立并使用笛卡尔树查询区间最大值

参考代码

C

```c
#include <stdio.h>
#include <math.h>
#define maxn 10005
int n, k, sum, days, rem, cost[maxn], st[maxn][25], dp[maxn][105], log_2[maxn];
int max(int x, int y) {return x > y ? x : y;}
int query(int l, int r) {
```

```
    int x = log_2[r - l + 1];
    return max(st[l][x], st[r - (1 << x) + 1][x]);
}
int main()
{
    scanf("%d %d", &n, &k);
    for (int i = 1, now = 1, t = -1; i <= n; log_2[i] = t, i++) {
        scanf("%d", cost + i);
        sum += st[i][0] = cost[i];
        if (i >= now) now <<= 1, t++;
    }
    for (int j = 1; (1 << j) <= n; j++)
        for(int i = 1; i + (1 << j) - 1 <= n; i++)
            st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
    days = ceil(1.0 * n / k), rem = (k - (n % k)) % k;
    for (int i = 1; i <= days; i++)
        for (int j = 0; j <= rem; j++)
        {
            dp[i][j] = 0;
            for (int t = 0; t <= j; t++)
                dp[i][j] = max(dp[i][j], dp[i - 1][t] + query((i - 1) * k - t + 1,
i * k - j));
        }
    printf("%d\n", sum - dp[days][rem]);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 10005
int n, k, sum, days, rem, cost[maxn], st[maxn][25], dp[maxn][105], log_2[maxn];
inline int query(int l, int r) {
    int x = log_2[r - l + 1];
    return max(st[l][x], st[r - (1 << x) + 1][x]);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> k;
    for (int i = 1, now = 1, t = -1; i <= n; log_2[i] = t, i++) {
        cin >> cost[i];
        sum += st[i][0] = cost[i];
        if (i >= now) now <<= 1, t++;
    }
    for (int j = 1; (1 << j) <= n; j++)
        for(int i = 1; i + (1 << j) - 1 <= n; i++)
            st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
    days = ceil(1.0 * n / k), rem = (k - (n % k)) % k;
    for (int i = 1; i <= days; i++)
        for (int j = 0; j <= rem; j++)
        {
            dp[i][j] = 0;
            for (int t = 0; t <= j; t++)
```

```
                dp[i][j] = max(dp[i][j], dp[i - 1][t] + query((i - 1) * k - t + 1,
i * k - j));
        }
    cout << sum - dp[days][rem] << endl;
    return 0;
}
```

0 疫情阻击战

时间限制： 2000ms
空间限制： 512MB
难度： ★★★★★ ★★★☆☆

标签： 树形结构 动态规划 树形动态规划 深度优先搜索DFS 广度优先搜索BFS

题目描述

GJX 制定好旅游计划后，突然意识到疫情还远远没有结束，责任心满满的他决定放下自己的旅游计划，投身到抗击疫情的战斗中。

GJX 提出了一种抗击疫情的方案：把交通网络改造成树形图，把每个城市看做是一个点，不同形势的地区对应不同的点：

1. 疫情点：疫情点存在疫情，有向外传播的风险；
2. 阻隔点：能够阻隔疫情传播的点，设置阻隔点需要花费一定的资金；
3. 要塞点：抗疫基地，最终不能被感染的点；

对于要塞点 A 和任意疫情点 B 之间有连通的路径上，要求必须存在阻隔点（可以是 A 或 B ）；特别的，如果一个点既是疫情点，又是要塞点，那么必须把它设置成阻隔点，求满足条件的最小代价。

输入描述

第一行，三个整数 $n\ (1 \le n \le 10^6)$、$m\ (1 \le m \le n)$、$p\ (1 \le p \le n)$，n 代表树的大小，m 代表疫情点数目，p 代表要塞点数目；

接下来的 $n-1$ 行，每行两个用空格分开的整数 $x, y\ (1 \le x, y \le n)$，代表点 x 与点 y 之间存在一条边；

接下来一行 n 个用空格分开的非负整数，分别代表在每个点设置阻隔点的代价；

接下来一行 m 个用空格分开的整数，为疫情点编号；

最后一行 p 个用空格分开的整数，为要塞点编号

输出描述

一行，一个整数，输出为符合要求而设置阻隔点的最小代价

题解

1. $O(n\sqrt{n})$：Time Limit Exceeded 通过拆点建立新图，求该图的最小割，其等于网络最大流

2. $O(n)$：Accepted 建立一棵树，从根节点开始进行 DFS 或 BFS。用 $dp[i][j][k]$ 表示结点 i 的类型为 j ，是否设为阻隔点的最小花费。例如 $j=0$ 代表结点既不是疫情点也不是要塞点，$j=1$ 表示疫情点，$j=2$ 表示要塞点，$j=3$ 表示两者都是，$k=0$ 表示不设为阻隔点，$k=1$ 表示设为阻隔点，设当前结点为 u，子结点为 v 则有以下可能的情况：

- u 设阻隔点，那么 v 什么情况都可以，$dp[u][j_1][1] = \min(dp[v][j_2][k]) + cost[u]$
- u 不设阻隔点，且 u 是疫情点，那么 v 不能是要塞点，除非 v 设阻隔点，$dp[u][1][0] = \min(dp[v][j][1],\ dp[v][0][0],\ dp[v][1][0])$

- u 不设阻隔点，且 u 是要塞点，那么 v 不能是疫情点，除非 v 设阻隔点，$dp[u][2][0] = \min(dp[v][j][1],\ dp[v][0][0],\ dp[v][2][0])$
- u 不设阻隔点，u 两者都不是，那么 v 不能是要塞点或疫情点，除非 v 设阻隔点，$dp[u][0][0] = \min(dp[v][j][1],\ dp[v][0][0])$

  $ans = \min(dp[1][j][k])$，常数大

参考代码

C

```c
#include <stdio.h>
#include <stdbool.h>
#define maxn 1000005
#define maxm 2000005
#define INF 1234567890000000
int n, m, s, cost[maxn], cnt, head[maxn];
long long ans = INF, dp[maxn][4][2];
bool a[maxn], b[maxn], vis[maxn];
typedef struct {int u, v, next;} Edge;
Edge edge[maxm];
long long min(long long x, long long y) {return x < y ? x : y;}
void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
void Dfs(int u) {
    vis[u] = 1;
    for (int j = 0; j <= 3; j++)
        for (int k = 0; k <= 1; k++)
            dp[u][j][k] = INF;
    if(a[u] && b[u]) dp[u][3][1] = cost[u];
    else if (a[u] || b[u]) {
        dp[u][a[u] + 2 * b[u]][1] = cost[u];
        dp[u][a[u] + 2 * b[u]][0] = 0;
    }
    else {
        dp[u][0][1] = dp[u][1][1] = dp[u][2][1] = cost[u];
        dp[u][0][0] = dp[u][1][0] = dp[u][2][0] = 0;
    }
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].v;
        if (vis[v]) continue;
        Dfs(v);
        long long x = INF;
        for (int j = 0; j <= 3; j++)
            for (int k = 0; k <= 1; k++)
                x = min(x, dp[v][j][k]);
        for (int j = 0; j <= 3; j++)
            if(dp[u][j][1] != INF) dp[u][j][1] += x;
        x = INF;
        for (int j = 0; j <= 3; j++)
            x = min(x, dp[v][j][1]);
        x = min(x, dp[v][0][0]);
        if(dp[u][0][0] != INF)dp[u][0][0] += x;
```

```
        if(dp[u][1][0] != INF)dp[u][1][0] += min(x, dp[v][1][0]);
        if(dp[u][2][0] != INF)dp[u][2][0] += min(x, dp[v][2][0]);
    }
}
int main()
{
    scanf("%d %d %d", &n, &m, &s);
    int u, v;
    for (int i = 1; i < n; i++) {
        scanf("%d %d", &u, &v);
        add(u, v);
        add(v, u);
    }
    for (int i = 1; i <= n; i++)
        scanf("%d", cost + i);
    for (int i = 1; i <= m; i++) {
        scanf("%d", &u);
        a[u] = 1;
    }
    for (int i = 1; i <= s; i++) {
        scanf("%d", &u);
        b[u] = 1;
    }
    Dfs(1);
    for (int j = 0; j <= 3; j++)
        for (int k = 0; k <= 1; k++)
            ans = min(ans, dp[1][j][k]);
    printf("%lld\n", ans);
    return 0;
}
```

C++

```
#include <bits/stdc++.h>
using namespace std;
#define maxn 1000005
#define maxm 2000005
#define INF 1234567890000000
int n, m, s, cost[maxn], cnt, head[maxn];
long long ans = INF, dp[maxn][4][2];
bool a[maxn], b[maxn], vis[maxn];
struct Edge{int u, v, next;} edge[maxm];
inline void add(int u, int v) {
    edge[++cnt].u = u;
    edge[cnt].v = v;
    edge[cnt].next = head[u];
    head[u] = cnt;
}
inline void Dfs(int u) {
    vis[u] = 1;
    for (int j = 0; j <= 3; j++)
        for (int k = 0; k <= 1; k++)
            dp[u][j][k] = INF;
    if(a[u] && b[u]) dp[u][3][1] = cost[u];
    else if (a[u] || b[u]) {
        dp[u][a[u] + 2 * b[u]][1] = cost[u];
        dp[u][a[u] + 2 * b[u]][0] = 0;
```

```
    }
    else {
        dp[u][0][1] = dp[u][1][1] = dp[u][2][1] = cost[u];
        dp[u][0][0] = dp[u][1][0] = dp[u][2][0] = 0;
    }
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].v;
        if (vis[v]) continue;
        Dfs(v);
        long long x = INF;
        for (int j = 0; j <= 3; j++)
            for (int k = 0; k <= 1; k++)
                x = min(x, dp[v][j][k]);
        for (int j = 0; j <= 3; j++)
            if(dp[u][j][1] != INF) dp[u][j][1] += x;
        x = INF;
        for (int j = 0; j <= 3; j++)
            x = min(x, dp[v][j][1]);
        x = min(x, dp[v][0][0]);
        if(dp[u][0][0] != INF)dp[u][0][0] += x;
        if(dp[u][1][0] != INF)dp[u][1][0] += min(x, dp[v][1][0]);
        if(dp[u][2][0] != INF)dp[u][2][0] += min(x, dp[v][2][0]);
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> m >> s;
    int u, v;
    for (int i = 1; i < n; i++) {
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    for (int i = 1; i <= n; i++)
        cin >> cost[i];
    for (int i = 1; i <= m; i++) {
        cin >> u;
        a[u] = 1;
    }
    for (int i = 1; i <= s; i++) {
        cin >> u;
        b[u] = 1;
    }
    Dfs(1);
    for (int j = 0; j <= 3; j++)
        for (int k = 0; k <= 1; k++)
            ans = min(ans, dp[1][j][k]);
    cout << ans << endl;
    return 0;
}
```