Nov.12th 2022 Sat.

T1 Desire

预测：$60pts$ 实际：$100pts$ $delta$：$+40pts$

[题目描述]

命莲寺和神灵庙之间宗教战争已经持续已久，丰聪耳神子决定使用名为二叉树的武器来战胜命莲寺。 神子会给定长度为n的数组，要求构造一棵n个叶子的有根二叉树，树上每个节点到左儿子的边长为1，到右儿子的边长为2。你还要将a_i赋值给每个叶子，一个a_i只能赋值给一个叶子。 一个叶子的深度为其到根的简单路径的长度，搭建这棵二叉树所花费的时间为所有叶子的权值与深度之积的和。神子希望能尽快地搭建这棵二叉树，所以希望你帮忙求出最少的搭建时间。

[orz]

贪心能A,离大谱……

贪心策略：先把a数组从大往小排序，然后往树上的叶子结点一个一个一个（？）插点，优先往权值与深度之积最小的那个点上插，如果有积相同的就优先插到权值更小的那个点（因为权值小的深度$+1$后的积会更小）。

因为贪心A了所以没去看正解，草生……

下附代码qwq

```
#include<bits/stdc++.h>
#include<algorithm>
#pragma GCC optimize(2)
using namespace std;
int n,dep;
int a[2750];
struct Node{
    int num,dep;
}f[10751];
template<typename T> inline void re(T &x)
{
    x=0;
    int f=1;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=(x<<1)+(x<<3)+(ch^48);
    x*=f;
    return;
}
bool cmp(int x,int y)
{
    return x>y;
}
bool cmp2(int a,int b)
{
    long long num=f[a].num*f[a].dep-f[b].num*f[b].dep;
    if(num) return num<0;
```

```
    return f[a].num<f[b].num;
}
int leaf[10751],head,tail,tot=1;
long long dfs(int x)
{

f[++tot].num=f[leaf[head]].num,f[tot].dep=f[leaf[head]].dep+1,leaf[++tail]=tot;
    f[++tot].num=a[x],f[tot].dep=f[leaf[head]].dep+2,leaf[++tail]=tot;
    head++;
    sort(leaf+head,leaf+tail+1,cmp2);
    if(x<n) dfs(x+1);
}
int main()
{
//  freopen("desire.in","r",stdin);
//  freopen("desire.out","w",stdout);
    re(n);
    for(int i=1;i<=n;i++) re(a[i]);
    sort(a+1,a+n+1,cmp);
    f[1].num=a[1],f[1].dep=0,leaf[1]=1,head=1,tail=1;
    dfs(2);
    long long ans=0;
    for(int i=head;i<=tail;i++)
     ans+=f[leaf[i]].num*f[leaf[i]].dep;
    printf("%lld",ans);
    return 0;
}
```

T2 Dealing

预测：$10pts$ 实际：$10pts\ delta：0pt$

[题目描述]

异变结束了，鬼人正邪和少名针妙丸被巫女，魔法使和女仆长狠狠收拾了一顿，现在她们只想给辉针城装一把密码锁。 她们从河童处购买的密码锁只能生成26个小写英文字母，她们想要生成一个长度为n的字符串作为密码。当密码中存在相同的子串，密码会变得相对容易背下来。所以针妙丸要求生成的字符串有m个限制，具体来说，第i条限制被描述为len_i,x_i,y_i，表示从x_i开始的长度为len_i的子串和表示从y_i开始的长度为len_i的子串是相同的（下标从1开始）。 鬼人正邪很好奇可以生成多少个不同的满足条件的密码。因为她是天邪鬼，所以她只想要知道答案对10^9+7取模。

[orz]

暴力做法是并查集连边，最后判scc个数cnt，最后输出26^{cnt} 取模10^9+7。但我没连边和判scc,只在并查集上上下其手，然后就挂了qwq。

优化方法：

这题的计数是诈你的，题目中的限制可以转换为：$[x,x+len-1]$ 和 $[y,y+len-1]$ 之间对应的字符相等（$s_x=s_y,s_{x+1}=s_{y+1},\ldots$），一个 $O(\sum len_i)$ 的暴力做法很显然，用并查集暴力去连边，最后统计连通分量个数 cnt，输出 26^{cnt} 就行。

考虑优化连边过程，有若干种方法。

方法 1：考虑倍增优化，令 $fa(k,i)$ 表示，区间 $[i,i+2^k-1]$ 集体被依次连到了哪个区间的左端点上，具体地，修改的时候将 len 拆分二进制，令目前是二进制的第 k 位，将 $fa(k,x)$ 和 $fa(k,y)$ 连起来，然后将 x,y 加上 2^k；查询的时候从大往小枚举 k，对于每个 $fa(k,i)$ 下传到 $fa(k-1,i)$ 上，具体地，将 $fa(k-1,i)$ 和 $fa(k-1,fa(k,i))$ 连起来，将 $fa(k-1,i+2^{k-1})$ 和 $fa(k-1,fa(k,i)+2^{k-1})$ 连起来，直接在 $fa(0,*)$ 上统计答案就行。

方法 2：观察到总共只会有 $O(n)$ 次有效连边，如果我们能较快的找到一个有效连边的位置，问题就得以解决。假设我们目前要找到最小的一个 d，使得 $find(x+d)$ 和 $find(y+d)$ 不同，这个可以转化为，x,y 位置上的最长公共前缀问题，我们维护序列 $find(i)$ 的哈希值，可以用 BIT 去维护，每次二分去找，时间复杂度是 $O(n\log^2 n)$，卡卡常能过。

$(What?)$

下附**正解（大嘘）代码**qwq

```
#include<bits/stdc++.h>
using namespace std;
int fa[100001];
int get(int x)
{
    if(fa[x]==x) return x;
    return fa[x]=get(fa[x]);
}
void merge(int y,int x)
{
    fa[get(x)]=get(y);
}
template<typename T> inline void re(T &x)
{
    x=0;
    int f=1;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=(x<<1)+(x<<3)+(ch^48);
    x*=f;
    return;
}
const int p=1000000007;
int power(int a,int b)
{
    int ans=1;
    for(;b;b>>=1)
    {
        if(b&1) ans=(long long)ans*a%p;
        a=(long long)a*a%p;
    }
    return ans;
}
int main()
{
```

```
//  freopen("dealing.in","r",stdin);
//  freopen("dealing.out","w",stdout);
    int n,m,len,x,y;
    re(n),re(m);
    for(int i=1;i<=n;i++) fa[i]=i;
    for(int i=1;i<=m;i++)
    {
        re(len),re(x),re(y);
        if(x>y) swap(x,y);
        for(int k=x+1;k<=x+len-1;k++) if(get(x)!=get(k)) merge(x,k);
        for(int k=y;k<=y+len-1;k++)
         if(get(x)!=get(k)) merge(x,k);
    }
    int cnt=0;
    for(int i=1;i<=n;i++) if(fa[i]==i) cnt++;
    printf("%d",power(26,cnt));
    return 0;
}
```

T3 Lunatic

预测：$0pt$ 实际：$0pt$ $delta$：$0pts$

【题目描述】

纯狐和赫卡提亚进攻月之都的计划正在进行中，她们派遣克劳恩皮丝去摧毁月之都的防空系统。 月之都的防空系统被描述成n条坐落在数轴上的线段，每条线段左端点是l_i，右端点是r_i。 克劳恩皮丝要将这些线段分成k个非空的组，一组线段被摧毁，会给月之都带来这组的线段的线段交的长度的代价，总的代价为每一组的代价的和，为了尽可能干扰月之都，克劳恩皮丝想要这个代价最大。 在发动攻击前，她想知道这个代价是多少。

【orz】

样例没懂，直接没写捏。（恼

观察到，如果线段 i 包含线段 j，那么线段 i 可以通过和 j 放一起而无代价的“跳过”。

我们预处理出所有满足包含另一个线段的线段，将其记做线段集合 A，将其余线段记做线段集合 B。

假设我们在集合 B 中划分出 x 组，那么最优的情况是，在 A 中选出长度最大的 $k-x$ 条线段单独成组，其余跳过。

将 B 中的线段按左端点升序排序，由于之前的处理，右端点也是升序。

令 $dp(i,j)$ 表示，我们已经用前 j 条线段分成 i 组，则有转移：
$dp(i,j)=\max_{k<j}\{dp(i-1,k)+\max\{0,r_{k+1}-l_i\}\}$，将 $-l_i$ 提出来，也就是说，我们要找到一个最大的 k 使得 $dp(i-1,k)+r_{k+1}$ 最大，这个可以用前缀和优化，可以得到一个 $O(n^2)$ 的做法。

我们考虑一下 dp 的决策点，假设 $i_1,i_2,\ldots,i_m$ 分别是每一组的开头，则答案应该为：
$\max\{0,r_{i_1}-l_{i_2-1}\}+\max\{0,r_{i_2}-l_{i_3-1}\}+\max\{0,r_{i_3}-l_{i_4-1}\}\ldots$

这题没改完，没有代码。qwq

T4 Season

没改，没改得斯（目移

对不起做不到（孙笑川.jpg)

Nov.14th 2022 Mon.

T1 Substr

预测：$60pt$ 实际：$70pt$ $delta$：$+10pts$

【题目描述】

给定 n 个字符串 $S1,S2,\cdots,Sn$ ，要求找到一个最短的字符串 T，使得这 n 个 字符串都是 T 的子串。

【orz】

纯暴力，预处理了一下每个串能匹配的最多字符数量和字典序，分别第一关键字第二关键字排序一手，然后合并一下，居然能拿这么多分，离大谱

正解要写幻海梦蝶自动机，不会。（恼）

需要用自动机，其中多记录一个$state$表示点的状态，然后把所有串插进幻海梦蝶自动机（CWQ点了个踩），串i末尾的$state$就是2^{i-1}.

然后从根开始按字典序BFS,走到一个点继承它的$state$直到包含所有串9星.

贺来的题解就不粘了吧，目移

T2 Flower

预测：$40pts$ 实际：$40pts$ $delta$：$0pt$

【题目描述】

小 C在家种了 n 盆花, 每盆花有一个艳丽度 a_i. 在接下来的 m 天中, 每天早晨他会从一段编号连续的花中选择一盆摆放在客厅, 并在晚上放回. 同时每天有特定的光照强度 k_i, 如果这一天里摆放在客厅的花艳丽度为 x, 则他能获得的喜悦度为 $x \ mod \ k_i$. 他希望知道, 每一天他能获得的最大喜悦度是多少.

【orz】

本来写的线段树，但我的线段树太逊了，大样例跑了$33.89s$，反手试了试打暴力，只有$7.23s$，草生

正解卡利斯如是说（？）：

考虑当 k 确定的时候如何求答案, 显然对于所有形如 $[ak, (a+1)k)$ 的值域区间, 最大值一定是最优的.

进一步观察发现, 这样的区间总个数只有 $k \ln k$ 个. 考虑分块, 那么我们可以在 $O(n+k \ln k)$ 的时间复杂度内处理出一个块对于任意 k 的答案. 询问时复杂度是 $O(mS)$ 的, 取 $S = \sqrt{k \ln k}$ 可以达到最优复杂度 $O(n\sqrt{k \ln k})$.

分块还真没想到，因为我一直觉得分块比线段树逊，偏见害死人orz

```
#include<bits/stdc++.h>
#pragma GCC optimize(2)
```

```
#define ri register int
using namespace std;
int a[100001];
int P;
template<typename T> inline void re(T &x)
{
    x=0;
    int f=1;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=(x<<3)+(x<<1)+(ch^48);
    x*=f;
    return;
}
template<typename T> void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10^'0');
    return;
}
int main()
{
//  freopen("flower.in","r",stdin);
//  freopen("flower.out","w",stdout);
    int n,m,l,r,ans;
    re(n),re(m);
    for(ri i=1;i<=n;i++) re(a[i]);
    for(ri i=1;i<=m;i++)
    {
        ans=0;
        re(l),re(r),re(P);
        for(ri i=l;i<=r;i++) ans=max(ans,a[i]%P);
        wr(ans);
        putchar('\n');
    }
    return 0;
}
```

芝士暴力

```
#include <bits/stdc++.h>

using std::pair;
using std::vector;
using std::string;

typedef long long ll;
typedef pair<int, int> pii;

#define fst first
#define snd second
#define pb(a) push_back(a)
#define mp(a, b) std::make_pair(a, b)
#define debug(...) fprintf(stderr, __VA_ARGS__)
```

```cpp
template <typename T> bool chkmax(T& a, T b) {
    return a < b ? a = b, 1 : 0;
}
template <typename T> bool chkmin(T& a, T b) {
    return a > b ? a = b, 1 : 0;
}

const int oo = 0x3f3f3f3f;

string procStatus() {
    std::ifstream t("/proc/self/status");
    return string(std::istreambuf_iterator<char>(t),
std::istreambuf_iterator<char>());
}

template<typename T> inline T read(T& x)
{
    int f=1;
    x=0;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch == '-') f=-1;
    for(;isdigit(ch);ch=getchar()) x = x*10+ch-48;
    return x*=f;
}
const int B = 1000;
const int N = 100000;
int n, q;
int a[N + 5];
int lst[N + 5];
int ans[105][N + 5];
int main()
{
    freopen("flower.in", "r", stdin);
    freopen("flower.out", "w", stdout);
    read(n);
    read(q);
    for(int i = 0; i < n; ++i) read(a[i]);
    int blks = (n-1) / B + 1;
    for(int i = 0; i < blks; ++i) {
        memset(lst, 0, sizeof lst);
        for(int j = i * B; j < (i+1) * B && j < n; ++j) lst[a[j]] = a[j];
        for(int j = 1; j <= N; ++j) chkmax(lst[j], lst[j-1]);
        for(int j = 1; j <= N; ++j) {
            for(int k = 0; k <= N; k += j) {
                chkmax(ans[i][j], lst[std::min(k + j - 1, N)] - k);
            }
        }
    }
    while(q--)
    {
        static int l,r,k;
        read(l),read(r),read(k);
        --l,--r;
        int x=(l/B)+1,y=(r/B),res=0;
```

```
        if(x==y+1)
        {
            for(int i=l;i<=r;++i)
             chkmax(res,a[i]%k);
        } else {
            for(int i = x; i < y; ++i) chkmax(res, ans[i][k]);
            for(int i = l; i < x*B; ++i) chkmax(res, a[i] % k);
            for(int i = r; i >=y*B; --i) chkmax(res, a[i] % k);
        }

        printf("%d\n", res);
    }

    return 0;
}
```

芝士题解

T3 Refract

预测：$20pts$ 实际：$10pts$ $delta$：$-10pt$

【题目描述】

小 Y 十分喜爱光学相关的问题，一天他正在研究折射.
他在平面上放置了 n 个折射装置，希望利用这些装置画出美丽的折线. 折线将从某个装置出发，并且在经过一处装置时可以转向，若经过的装置坐标依次为 (x1, y1), (x2, y2), . . . (xk, yk)，则必须满足:
∀j ∈ (1, k], y[j]< y[j−1]
∀j ∈ (2, k], x[j−2] < x[j]< x[j−1] or x[j−1] < x[j] < x[j−2]
现在他希望你能告诉他，一共有多少种不同的光线能被他画出来，两种光线不同当且仅当经过的折射装置的集合不同. 你只需要告诉他答案对 10^9 + 7 取模后的结果.

【orz】

瞎jr打了个dp，骗到分了，没想到吧.jpg

芝士题解：

若将所有点按照 y_i 的顺序转移，有上界和下界两个限制，难以优化.

考虑按照 x_i 排序转移，并记录 $f_{i,0/1}$ 表示以第 i 个点为顶端接下来向左或向右的折线方案数. 从左到右加点，考虑前 i 个点构成的包含 i 点的折线. 由于新点横坐标最大，所以只可能在折线的第一位或第二位:

1. $\forall y_j < y_i, f_{i,0} \leftarrow f_{j,1}$
2. $\forall y_j > y_i, f_{j,1} \leftarrow f_{k,0} \mid x_k > x_j \text{ and } y_k < y_i$

第二种情况可以前缀和优化，复杂度 $O(n^2)$.

所以其实还是dp，就是转移方程有点毛病。

下放代码：

```
#include <bits/stdc++.h>
using std::pair;
using std::vector;
using std::string;
typedef long long ll;
typedef pair<int, int> pii;
#define fst first
#define snd second
#define pb(a) push_back(a)
#define mp(a, b) std::make_pair(a, b)
#define debug(...) fprintf(stderr, __VA_ARGS__)
template<typename T> bool chkmax(T& a, T b)
{
    return a<b?a=b,1:0;
}
template<typename T> bool chkmin(T& a, T b)
{
    return a>b?a=b,1:0;
}
const int oo=0x3f3f3f3f;
string procStatus() {
    std::ifstream t("/proc/self/status");
    return string(std::istreambuf_iterator<char>(t),
std::istreambuf_iterator<char>());
}
template <typename T> T read(T& x) {
    int f=1;x=0;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=x*10+ch-48;
    return x*=f;
}
const int N=6000;
const int mo=1e9+7;
pii p[N+5];
int dp[N+5][2], n;
int main()
{
    freopen("refract.in","r",stdin);
    freopen("refract.out","w",stdout);
    read(n);
    for(int i=1;i<=n;i++)
    {
        read(p[i].fst), read(p[i].snd);
    }
    sort(p+1, p+n+1);
    for(int i=1;i<=n;i++)
    {
        dp[i][0]=dp[i][1]=1;
        for(int j=i-1;j>=1;j--)
        {
            if(p[j].snd>p[i].snd)
            {
```

```
                (dp[j][1]+=dp[i][0])%=mo;
            }
            else
            {
                (dp[i][0]+=dp[j][1])%=mo;
            }
        }
    }
    int ans=mo-n;
    for(int i=1;i<=n;i++)
     ans=((ans+dp[i][0])%mo+dp[i][1])%mo;
    printf("%d\n",ans);
    return 0;
}
```

T4 Paint

又没写又没改。以后这个环节就删了吧.$mp3$

Nov.15th 2022 Tue.

T1 Reverse

预测：$45pts$ 实际：$82pts$ $delta$：$+37pts$

【题目描述】

小G有一个长度为n的01串T，其中只有T_S =1，其余位置都是0。现在小G可以进行若干次以下操作：

- 选择一个长度为K的连续子串（K是给定的常数），翻转这个子串。

  对于每个i, i ∈ [1, n]，小G想知道最少要进行多少次操作使得Ti = 1. 特别的，有m个“禁止位置”，你需要保证在操作过程中1始终不在任何一个禁止位置上。

【orz】

一眼BFS，但是剪枝太菜力，没卡过，但是改题的时候加了个卡常就过力.....

BFS搜的是当前点下一次翻转后1所在位置，注意k的范围不要越界就可。

```
#pragma GCC optimize(1)
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC target("avx")
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("-falign-jumps")
```

```
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#include<bits/stdc++.h>
using namespace std;
template<typename T> inline void re(T &x)
{
    x=0;
    int f=1;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=(x<<1)+(x<<3)+(ch^48);
    x*=f;
    return;
}
template <typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10^'0');
    return;
}
int team[200001],ans[200001],tot,step[200001];
int n,k,m,s,head,tail;
bool pd[200001];
int bfs()
```

```
{
    tail=1;
    int now=1;
    team[++head]=s;
    while(head<=tail&&(!ans[1]||!ans[2]||!ans[n-1]||!ans[n]))
    {
        now=ans[team[head]];
        pd[team[head]]=1;
        for(register int i=1;i<=tot;i++)
        {
            if(!(team[head]+step[i]/2+k/2<=n && team[head]+ceil(step[i]*1.0/2)-
k/2>0)&&!(team[head]-ceil(step[i]*1.0/2)+k/2<=n && team[head]-step[i]/2-k/2>0))
             break;
            if(team[head]+step[i]/2+k/2<=n && team[head]+ceil(step[i]*1.0/2)-
k/2>0     &&!pd[team[head]+step[i]])

ans[team[head]+step[i]]=now+1,team[++tail]=team[head]+step[i],pd[team[head]+step
[i]]=1;
            if(team[head]-ceil(step[i]*1.0/2)+k/2<=n && team[head]-step[i]/2-
k/2>0    &&!pd[team[head]-step[i]])
             ans[team[head]-step[i]]=now+1,team[++tail]=team[head]-
step[i],pd[team[head]-step[i]]=1;
        }
        head++;
    }
}
int main()
{
    freopen("reverse.in","r",stdin);
    freopen("reverse.out","w",stdout);
    int x;
    re(n),re(k),re(m),re(s);
    for(register int i=1;i<=m;i++) re(x),pd[x]=1;
    if(k%2==0)
    {
        for(register int i=1;i<=k;i+=2)
         step[++tot]=i;
    }
    else
    {
        for(register int i=2;i<=k;i+=2)
         step[++tot]=i;
    }
    bfs();
    for(register int i=1;i<=n;i++)
    {
        if(!ans[i]&&i!=s)
         putchar('-'),putchar('1'),putchar(' ');
        else
         wr(ans[i]),putchar(' ');
    }
    return 0;
}
```

T2 Silhouette

预测：$5pts$ 实际：$5pts$ $delta$：$0pt$

【题目描述】

有一个n × n的网格，在每个格子上堆叠了一些边长为1的立方体。 现在给出这个三维几何体的正视图和左视图，求有多少种与之符合的堆叠立方体的方 案。两种方案被认为是不同的，当且仅当某个格子上立方体的数量不同。 输出答案对10^9 + 7取模的结果。

【orz】

没改，但是我们要水总结的字数（握拳）

T3 seat

预测：$0pt$ 实际：$0pt$ $delta$：$0pt$

【题目描述】

有n + 2个座位等距地排成一排，从左到右编号为0至n + 1。 最开始时0号以及n + 1号座位上已经坐了一个小G，接下来会有n个小G依次找一个空座位 坐下。由于小G们坐得太近就容易互相搏弈，每个小G会找一个当前离最近的小G距离最远 的座位坐下。如果有多个备选的座位，这个小G会等概率选择其中一个。 给出n，求第i个坐下的小G坐在j号座位的概率, 对P 取模。具体来说，如果答案化为最简 分数可以表示a/b,你需要输出a × b^{-1}，其中b^{-1} = b^{P-2} (mod P).

【orz】

没改，但是我们要水总结的字数（握拳）

顺便附上一些考场迷惑行为：

```
    puts("NeverGonnaGiveYouUp");
```

T4 Ancient

预测:$0pts$ 实际:$10pts$ $delta$：$+10pts$

【题目描述】

猪王国的文明源远流长，博大精深。 iPig 在大肥猪学校图书馆中查阅资料，得知远古时期猪文文字总个数为 N。当然，一种语 言如果字数很多，字典也相应会很大。当时的猪王国国王考虑到如果修一本字典，规模 有可能远远超过康熙字典，花费的猪力、物力将难以估量。故考虑再三没有进行这一项 劳猪伤财之举。当然，猪王国的文字后来随着历史变迁逐渐进行了简化，去掉了一些不 常用的字。 iPig 打算研究古时某个朝代的猪文文字。根据相关文献记载，那个朝代流传的猪文文字恰好为远古时期的 k 分之一，其中 k 是 N 的一个正约数（可以是 1 和 N）。不过具体是哪 k 分之一，以及 k 是多少，由于历史过于久远，已经无从考证了。 iPig 觉得只要符合文献，每一种能整除 N 的 k 都是有可能的。他打算考虑到所有可能的 k。显然当 k 等于某个定值时，该朝的猪文文字个数为 N/k 。然而从 N 个文字中保留下 N/k 个的情况也是相当多的。iPig 预计，如果所有可能的 k 的所有情况数加起来为 P 的 话，那么他研究古代文字的代价将会是 G 的 P 次方。 现在他想知道猪王国研究古代文字的代价是多少。由于 iPig 觉得这个数字可能是天文数 字，所以你只需要告诉他答案除以 999911659 的余数就可以了。

【orz】

蓝皮书原题，但是不会数论，题目都也没看懂，想了半天样例的指数也不知道怎么凑出来的，，，

题目大意：求$G^{\sum d|n\ C_n^d}\ mod\ 999911659$

思路与其他题解相像，考虑到999911659是质数，那么就用欧拉定理的推论得：

$$G^{\sum d|n\ C_n^d}\ mod\ 999911659 = G^{\sum d|n\ C_n^d\ mod\ 999911658}\ mod\ 999911659$$

那么关键计算$\sum d|n\ C_n^d\ mod\ 999911658$.直接$Lucas$绝对挂，那么尝试把模数缩小再合并

将999911658因数分解，可得$999911658 = 2 \times 3 \times 4679 \times 35617$.那么把模数缩小，枚举$n$的因数$d$，然后运用$Lucas$定理把$C_n^d$算出来，分别计算出$\sum d|n\ C_n^d$对$2, 3, 4679, 35617$四个质数取模的结果，记为$a_1, a_2, a_3, a_4$.

最后，用中国剩余定理求解一下方程组：

$$\begin{cases} x \equiv a_1 (\mathrm{mod}\ 2) \\ x \equiv a_2 (\mathrm{mod}\ 3) \\ x \equiv a_3 (\mathrm{mod}\ 4679) \\ x \equiv a_4 (\mathrm{mod}\ 35617) \end{cases}$$

然后就得到了最小的非负整数解x，之后用快速幂求一下G^x就得到答案

抄？？？qwq

```
#include<bits/stdc++.h>
#define int long long
using namespace std;
template<typename T> inline void re(T &x)
{
    x=0;
    int f=1;
    char ch=getchar();
    for(;!isdigit(ch);ch=getchar()) if(ch=='-') f=-1;
    for(;isdigit(ch);ch=getchar()) x=(x<<1)+(x<<3)+(ch^48);
    x*=f;
    return;
}
template<typename T>void wr(T x)
{
    if(x<0) putchar('-'),x=-x;
    if(x>9) wr(x/10);
    putchar(x%10^'0');
    return;
}
```

```
int n,g;//输入内容
const int maxn=36000;//约数最多35000多个
int d[maxn],tot;//d存因数
int power(int a,int b,int p)
{
    int ans=1;
    for(;b;b>>=1)
    {
        if(b&1) ans=(long long)ans*a%p;
        a=(long long)a*a%p;
    }
    return ans;
}
int p[5]={0,2,3,4679,35617},a[5];
int pre[maxn],inv[maxn];//阶乘与阶乘逆元（用于求组合数）
void get(int x)//第x个因数(p[x])
{
    memset(pre,0,sizeof(pre));
    memset(inv,0,sizeof(inv));
    pre[0]=inv[0]=1;
    for(int i=1;i<p[x];i++) pre[i]=pre[i-1]*i%p[x];
    inv[p[x]-1]=power(pre[p[x]-1],p[x]-2,p[x]);
    for(int i=p[x]-2;i;i--) inv[i]=inv[i+1]*(i+1)%p[x];
}
int C(int n,int m,int x)//计算C(n,m)%p[x]
{
    if(m>n) return 0;
    return pre[n]*inv[m]%p[x]*inv[n-m]%p[x];
}
int lucas(int n,int m,int x)//Lucas 计算C(n,m)%p[x]
{
    if(m==0) return 1;
    return lucas(n/p[x],m/p[x],x)*C(n%p[x],m%p[x],x)%p[x];
}
void calc(int x)//计算出a数组
{
    get(x);
    for(int i=1;i<=tot;i++)
     a[x]=(a[x]+lucas(n,d[i],x))%p[x];
}
int CRT()//求出指数
{
    int ans=0;
    for(int i=1;i<=4;i++)
    {
        int mul=999911658/p[i],t=power(mul,p[i]-2,p[i]);
        ans=(ans+(long long)a[i]*mul%999911658*t%999911658)%999911658;
    }
    return ans;
}
signed main()
{
    freopen("ancient.in","r",stdin);
    freopen("ancient.out","w",stdout);
    scanf("%lld%lld",&n,&g);
```

```
    if(g%999911658==0)
    {
        wr(0);
        return 0;
    }
    for(int i=1;i*i<=n;i++)
    {
        d[++tot]=i;
        if(i*i!=n) d[++tot]=n/i;
    }
    for(int i=1;i<=4;i++)
     calc(i);//计算出每个因数（9999的）的a数组
    wr(power(g,CRT(),999911658));
    return 0;
}
```

爱莉神教宣传环节！！！！！！

他们曾如此骄傲地活过，贯彻始终，以生命奏响了文明的颂歌。

这是被称作英桀的人们的故事，是十三位逐火者未尽的旅途。

但来访者，你们的道路仍将延续，不是吗？

那就听凭心意前进吧！

沿着脚下的足迹，去见证这段逐火的征程。

最后，跨越逝者们的终墓，去创造，我们所未能迎接的——

未来吧！

TruE

演唱者：黄龄

Say my name when a tree susurrates

Once again telling a story lost in time

The way it starts and the way it ends

Never again making up stories in dismay

With several starts but just one end

Ah how I long to embrace

The future breaking out of shade from the past

Still ablaze

Save your tears for the day so far away

To irrigate the wilderness that's still asleep

In the world awaiting to be lit

To spread over the riverbed so dry and dead

Let ships that ran around relaunch their sails

Ah how I long to embrace

The future breaking out of shade from the past

Still ablaze

Seeds bear new life when flowers dare to fade

Petals linger about Awaiting one last dance

Shaking off all the dust from the past

New stories have yet to start