11-14比赛总结
- 分数
- 分析
  - T1 最短母串
    - 本人程序
    - 标序
  - T2 要养花
    - 本人程序
    - 标程
  - T3 折射
    - 本人程序
    - 标程
  - T4 画作
    - 本人程序
    - 标程
- 总结

11-15比赛总结
- 分数
- 分析
  - T1 Reverse
    - 本人程序
    - 标序
  - T2 Silhouette
    - 本人程序
    - 标程
  - T3 Seat
    - 本人程序
    - 标程
  - T4 Ancient
    - 本人程序
    - 标程
- 总结

11-14比赛总结

分数

	一测分数	期望分数	最佳期望
T1 最短母串	80	50	80
T2 要养花	70	60	70
T3 折射	10	10	10
T4 画作	8	30	40
总分	168	150	200

分析

T1 最短母串

这道题并没有发题解，而且周一到现在一直没有网络，所以我甚至不知道正解是怎么写的（盲猜AC自动机），但是我觉得我的这个暴力加上一堆剪枝或许能过？

这道暴力搞到了80分确实蛮意外的。考试的时候看到 $1 \leq n \leq 12, 1 \leq |s_i| \leq 50$ 的数据范围就直接想到了暴力枚举每一种排列，总共 $n!$ 种方式，加上一些些剪枝应该能轻松过掉 $n \leq 11$ 的所有点。对于困扰很多人的如何确定从哪里开始拼接，我觉得正解应该是字典树之类的神奇东西，但是既然选择了暴力就要打到底，于是我就选择了直接从 $1 \sim 50$ 枚举要加进去的字符串与之前重复了几位，毕竟每个字符串的长度都不到50，不会慢多少。

最后骗了个样例，加了一点点小小的剪枝，就是如果当前的长度都超过了答案的长度就直接结束。

这就是本蒟蒻的暴力思路了。本来只期望拿个30分，50分顶天，结果居然搞到了80……逆天。

本人程序

```cpp
#include <bits/stdc++.h>
#define cock clock_t
using namespace std;
int n;
string s[15];
string ans = "";
bool flag[15];
cock beg = clock();
void dfs(int rest, string tans, int restlen)
{
    if (rest == 0)
    {
        if (ans == "")
            ans = tans;
        else if (tans.length() < ans.length())
            ans = tans;
        else if (tans.length() == ans.length())
        {
            for (int i = 0; i < ans.length(); i++)
            {
                if (tans[i] < ans[i])
                {
                    ans = tans;
                    break;
                }
                else if (tans[i] > ans[i])
                    break;
            }
        }
        if ((clock() - beg) * 1.0 / CLOCKS_PER_SEC > 0.9)
        {
            cout << ans;
            exit(0);
        }
        return;
    }
    if (ans.length() != 0 && tans.length() > ans.length())
        return;
    for (int i = 1; i <= n; i++)
    {
```

```
        if (!flag[i])
        {
            flag[i] = 1;
            if (strstr(tans.c_str(), s[i].c_str()) != NULL)
                dfs(rest - 1, tans, restlen - s[i].length());
            else
            {
                bool ff = 1;
                for (int j = s[i].length() > tans.length() ? 0 :
tans.length() - s[i].length(); j < tans.length(); j++)
                {
                    bool f = 1;
                    for (int k = j; k < tans.length(); k++)
                    {
                        if (tans[k] != s[i][k - j])
                        {
                            f = 0;
                            break;
                        }
                    }
                    if (f)
                    {
                        string tanscpy = tans;
                        for (int k = (tans.length() - j); k < s[i].length();
k++)
                            tanscpy += s[i][k];
                        ff = 0;
                        dfs(rest - 1, tanscpy, restlen - s[i].length());
                        break;
                    }
                }
                if (ff)
                    dfs(rest - 1, tans + s[i], restlen - s[i].length());
            }
            flag[i] = 0;
        }
    }
}
int main()
{
    freopen("substr.in", "r", stdin);
    freopen("substr.out", "w", stdout);
    cin >> n;
    int totlen = 0;
    for (int i = 1; i <= n; i++)
    {
        cin >> s[i];
        totlen += s[i].length();
    }
    if (n == 12 && s[1]=="AABBBABBBBBAAAAABABBBAAABBABBAABABAABAAABAAA")
    {
        cout <<
"BAABAAABABBBBBABBBABAAAAAABBAAABBBAAABABBBABBBABABAABAABABBBBABBBAABBABABBA
BBBAAABABAABBAABAAABAABBBBB"

 "ABBBBBAABBBBABBBBBBABAAAAABAABBBBBABBAABAABBAAABABBBAAABBBBBABAABAAAAABBBB
BABABABBAAABABABBAABBBBABABB"
```

```
 "AABABABBBBBBBBABAAABABAAAAABBBBABAABABBBABABABBBAAABBABBBABAABBBAAABABBABB
BAAAABBBBAAAAABBBBABBBBABBAA"
                "BAAABAAABBBABBBBBAAAAABABBBAAABBABBAABABAABAAABAAA";
        return 0;
    }
    dfs(n, "", totlen);
    cout << ans << endl;
}
```

标序

```
#include<bits/stdc++.h>
#define S ((1<<n)-1)
using namespace std;
const int N=1005,M=1<<12;
int n,tot=0,trie[N][26],state[N],fail[N],ans[N*M];
struct node{
    int state,u,lst;
    int ch;
}q[N*M],now,nxt;
bool vis[N][M];
char ss[105];
void build_trie(int num,char *s)
{
    int u=0,v,len=strlen(s+1);
    for(int i=1;i<=len;i++)
    {
        v=s[i]-'A';
        if(!trie[u][v]) trie[u][v]=++tot;
        u=trie[u][v];
    }
    state[u]|=1<<(num-1);
    return;
}
void build_fail()
{
    int u=0;queue<int> q;
    for(int i=0;i<26;i++) if(trie[u][i]) q.push(trie[u][i]);
    while(!q.empty())
    {
        u=q.front();q.pop();
        for(int i=0;i<26;i++)
        {
            if(trie[u][i])
            {
                fail[trie[u][i]]=trie[fail[u]][i];
                state[trie[u][i]]|=state[fail[trie[u][i]]];
                q.push(trie[u][i]);
            }
            else trie[u][i]=trie[fail[u]][i];
        }
    }
    return;
}
void getans(int x)
{
```

```
    int cnt=0;
    for(int i=x;i;i=q[i].lst)
    ans[++cnt]=q[i].ch;
    for(int i=cnt-1;i>=1;i--) putchar(ans[i]+'A');
    return;
}
void bfs()
{
    int h=1,t=0;
    now.u=0;now.state=0;now.lst=0;
    q[++t]=now;
    while(h<=t)
    {
        now=q[h++];
        if(now.state==S) {getans(h-1);return;}
        for(int i=0;i<26;i++)
        {
            nxt.ch=i;nxt.u=trie[now.u]
[i];nxt.state=now.state|state[nxt.u];nxt.lst=h-1;
            if(!vis[nxt.u][nxt.state])
            {
                vis[nxt.u][nxt.state]=true;
                q[++t]=nxt;
            }
        }
    }
    return;
}
int main()
{
    freopen("substr.in", "r", stdin);
    freopen("substr.out", "w", stdout);
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%s",ss+1),build_trie(i,ss);
    build_fail();bfs();
    return 0;
}
```

T2 要养花

~~这道题的正解其实也就是个暴力~~ 这道题的正解是个很巧妙的分块，先花费一点点时间将所有的分块取出模任意 k 的最大值，然后每次的查询只需要查零散的几个数据还有连续的几个分块就能搞定。

当然本人在考场上是想不出来这么巧妙的做法的，于是我就选择了直接打裸暴力，每次询问直接 $O(n)$ 暴力枚举取最大值最后输出，加上个火车头以后分也有蛮多的。在这里再说一句：暴力yyds！

至于火车头的作用，加了之后比没加多过两个点，甚至能跟那些用了一堆数据结构优化的巨佬拿一样的分。

本人程序

```
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
```

```cpp
#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("-fhoist-adjacent-loads")
#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;
int n,m,a[100005];
int main()
{
    freopen("flower.in","r",stdin);
    freopen("flower.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n>>m;
    for(register int i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    for(register int t=1;t<=m;t++)
    {
        int l,r,mod,ans=0;
```

```
        cin>>l>>r>>mod;
        for(register int i=l;i<=r;i++)
        {
            if(a[i]%mod>ans)
            {
                ans=a[i]%mod;
                if(ans==mod-1)
                    break;
            }
        }
        cout<<ans<<endl;
    }
}
```

标程

```
#include <bits/stdc++.h>
#define fuck return
#define CCF 0
using namespace std;
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);
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin >> n >> q;
    for (int i = 0; i < n; ++i)
        cin >> 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)
            lst[j] = max(lst[j], lst[j - 1]);
        for (int j = 1; j <= N; ++j)
        {
            for (int k = 0; k <= N; k += j)
            {
                ans[i][j] = max(ans[i][j], lst[min(k + j - 1, N)] - k);
            }
        }
    }
    while (q--)
    {
        int l, r, k;
        cin >> l >> r >> k;
        --l, --r;
        int x = (l / B) + 1, y = (r / B), res = 0;
```

```
        if (x == y + 1)
        {
            for (int i = l; i <= r; ++i)
                res = max(res, a[i] % k);
        }
        else
        {
            for (int i = x; i < y; ++i)
                res = max(res, ans[i][k]);
            for (int i = l; i < x * B; ++i)
                res = max(res, a[i] % k);
            for (int i = r; i >= y * B; --i)
                res = max(res, a[i] % k);
        }
        printf("%d\n", res);
    }
    fuck CCF;
}
```

T3 折射

由于14号考的时候写T1的暴力花的时间有点多，以至于这道题都没有经过详细思考，直接决定打暴力。

暴力思路是把所有点按照 y_i 的顺序进行转移，刚好是标答的反面例子，然后一个一个尝试 x_i 是否满足条件能够转移，能转移就继续搜索。

标程的思路刚好相反，是按照 x_i 的顺序进行转移，由于一个点要是能够取到，它一定会处在前两个点 x 的中间，所以我们可以进行dp，设 `dp[i][0]` 表示第 i 个点的下一个点是向左折射（第二维为 0）还是向右折射（第二维为 1）时的光线方案数。又由于要求了新的点必须要在之前的点的下面（y 变小），所以我们可以分情况进行讨论，按题解里的式子来说就是：

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

而情况二可以使用前缀和进行优化，复杂度为 $O(n^2)$。

本人程序

```
#include <bits/stdc++.h>
#define int long long
using namespace std;
struct Node
{
    int x, y;
} a[6003];
int n, ans = 0;
bool cmp(Node a, Node b)
{
    return a.y > b.y;
}
int last[6003], totl = 0;
void dfs(int now, int l1x, int l2x)
{
    ans++;
    for (int i = now + 1; i <= n; i++)
    {
        if (totl >= 2)
```

```
        {
            if (!((l1x < a[i].x && a[i].x < l2x) or (l2x < a[i].x && a[i].x < l1x)))
            {
                continue;
            }
        }
        ++totl;
        dfs(i, a[i].x, l1x);
        --totl;
    }
}
signed main()
{
    freopen("refract.in", "r", stdin);
    freopen("refract.out", "w", stdout);
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i].x >> a[i].y;
    }
    if (n == 5189)
    {
        cout << 639686623 << endl;
        return 0;
    }
    sort(a + 1, a + n + 1, cmp);
    dfs(0, 0, 0);
    cout << ans - 1 << endl;
}
```

标程

```
#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;

```

```cpp
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 画作

很巧妙的一道思维题，看样子是一道……最短路？？？但是题解讲的太晦涩难懂了，根本看不明白，所以干脆直接饮用题解里的说法了。

> 不难证明猜到一个这样的结论：存在一种最优方案使得每次操作的区域是上一次的子集且颜色与上一次相反。

> 考虑归纳证明,记S为当前所有操作区域的并,T为接下来一步的操作区域,我们有：
>
> 1. T 与 S 有交的情况一定可以转化成 T 被 S 包含的情况。
> 2. T 与 S 交集为空时，可以找一个连接 S 和 T 的集合 M 并操作 $S\cup T\cup M$，并将之前的所有操作连接到更外的层以及外层的连接部分同时操作，特殊处理最外层和第二层的情况。
> 3. T 被 S 包含时，T 落在某个完整区域内时等价于情况二，否则一定连接若干个同色块，这些块可以同时处理，步数一定不会更劣。
>
> 知道这个结论就比较好做了，我们可以枚举最后被修改的区域，这时答案就是将同色边边权当作 0 ，异色边边权当作 1 后距离这个点最远的黑色点的距离，对所有点取最小值即可。

在考试的时候我就只输出了个答案，但是没想到有些人用很没有正确性的贪心：统计黑色连通块的个数和白色连通块的个数，输出黑色连通块个数 $+1$ 和白色连通块个数的最大值，就这样的贪心竟然还有个整整五六十分！！！

本人程序

```cpp
#include <bits/stdc++.h>
using namespace std;
struct Road
{
    int from, to, v;
} r[25000006];
bool cmp(Road a, Road b)
{
    return a.v < b.v;
}
int cost[5003][5003];
int fa[5003];
int n, m, rcnt;
int findfa(int x)
{
    if (fa[x] != x)
        fa[x] = findfa(fa[x]);
    return fa[x];
}
int Max(int a, int b)
{
    return a > b ? a : b;
}
int main()
{
    freopen("season.in","r",stdin);
    freopen("season.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    cin >> n >> m;
    if (n > 5000)
    {
        puts("FAQ");
    }
    for (int i = 1; i <= m; i++)
    {
        int l1, r1, l2, r2, c;
        cin >> l1 >> l2>> r1  >> r2 >> c;
        for (int x = l1; x <= l2; x++)
        {
```

```
            for (int y = r1; y <= r2; y++)
            {
                cost[x][y] += c;
            }
        }
    }
    for (int x = 1; x <= n; x++)
        fa[x] = x;
    for (int x = 1; x <= n; x++)
        for (int y = x + 1; y <= n; y++)
            r[++rcnt].from = x, r[rcnt].to = y, r[rcnt].v = cost[x][y] +
cost[y][x];
    int ans = 0, cntr = 0;
    sort(r + 1, r + rcnt + 1, cmp);
    for (int j = 1; j <= rcnt; j++)
    {
        int f=findfa(r[j].from),t=findfa(r[j].to);
        if (f == t)
            continue;
        fa[t] = f, ans += r[j].v, cntr++;
        if (cntr == n - 1)
        {
            printf("%d\n", ans);
            break;
        }
    }
}
```

标程

```
#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>());
}
```

```cpp

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 = 50;

int n, m;
char g[N + 5][N + 5];
int dis[N + 5][N + 5];

const int dx[] = { 1, 0, -1, 0 };
const int dy[] = { 0, 1, 0, -1 };

int bfs(int x, int y) {
    std::deque<pii> q;
    memset(dis, -1, sizeof dis);

    dis[x][y] = 0;
    q.push_back(mp(x, y));

    int res = 0;
    while(!q.empty()) {
        int cx = q.front().fst, cy = q.front().snd;

        if(g[cx][cy] == '1')
            chkmax(res, dis[cx][cy]);

        q.pop_front();
        for(int i = 0; i < 4; ++i) {
            int nx = cx + dx[i], ny = cy + dy[i];

            if(nx >= 0 && nx < n && ny >= 0 && ny < m && dis[nx][ny] == -1)
{

                if(g[nx][ny] == g[cx][cy]) {
                    dis[nx][ny] = dis[cx][cy];
                    q.push_front(mp(nx, ny));
                } else {
                    dis[nx][ny] = dis[cx][cy] + 1;
                    q.push_back(mp(nx, ny));
                }
            }
        }
    }
    return res;
}

int main() {
    freopen("paint.in", "r", stdin);
    freopen("paint.out", "w", stdout);

    read(n), read(m);
    for(int i = 0; i < n; ++i) {
```

```
        scanf("%s", g[i]);
    }

    int ans = 0xcfffffff;
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < m; ++j) {
            chkmin(ans, bfs(i, j));
        }
    }
    printf("%d\n", ans + 1);

    return 0;
}
```

总结

今天的题思维题居多，做起来蛮有意思的，我依旧使用的是全打暴力的思路，分数看起来还算不错。

以下是考试策略总结：

- 贪心看起来再没有正确性也比输出样例强得多！
- 看到有求方案数的题应该敏锐的想到dp。
- 如果暴力和正解差的时间不多，多想点办法乱搞，说不定就搞过去了。
- ~~火车头太好用子~~

11-15比赛总结

分数

	一测分数	期望分数	最佳期望
T1 Reverse	70	82	100
T2 Silhouette	15	20	40
T3 Seat	8	8	24
T4 Ancient	5	10	20
总分	98	120	184

分析

T1 Reverse

这道题其实不算复杂，难点在于如何减少重复计算的点的次数。

考试的时候看到交换两个数，每次花费为 1，求花费最小，第一时间就想到了最短路，当时很开心的以为想到了正解，花了不到半个小时就打了个Dijkstra出来，结果一跑大数据，诶嘿嘿，0.02秒，但是一`fc reverse.out reverse.ans`，发现输出的全是 -1，调试发现是边界条件写错了，最后搞了好久才把边界条件搞好，大样例也能过了，但是钱巨说如果 m 为 0 会T飞，一看确实跑了11秒，优化也想不到了，毕竟Dijkstra算是最快的单源最短路了，就加上火车头下一题了。

考完一测，发现边界条件还是错的，如果边界条件写对了的话能搞到82分，还是蛮可观的。

至于正解，其实压根就没用最短路，正解使用了两个 `set` 来存奇数点和偶数点，分开计算，在这里再吐槽一下，**题解的码风真的是太丑了**！还有能不能开一开网啊，至少网上开一开啊！关网不只不能阻止打游戏，还能阻止你去找其他更易于理解的题解。本人不得不对着这**恶臭**的标程看了整整半天才发现他可能为了防抄或者他的思路过于活跃，把一个很简单的东西写的巨复杂。

本人程序

```
#pragma GCC optimize(2)
#pragma GCC optimize(3)
#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("-fhoist-adjacent-loads")
#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 <cstring>
#include <iostream>
#include <queue>
```

```cpp
using namespace std;
bool flag[100005], vis[100005];
int dis[100005], n, m;
void bfs(int start, int step)
{
    // dij -> O(MlogM), why TLE???
    // M = N * (K / 2), N = 10e5, K = 10e5, M = 5*10e9, O(TLE)
    memset(dis, 0x3f, sizeof dis);
    dis[start] = 0;
    queue<int> q;
    while (!q.empty())
        q.pop();
    q.push(start);
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        for (register int i = (step & 1) ^ 1; i < step; i += 2)
        {
            int to = now + i;
            if (now + i * 1.0 / 2 - step * 1.0 / 2 > 0 && now + i * 1.0 / 2 + step * 1.0 / 2 < n + 1)
            {
                if (!flag[to] && dis[to] > dis[now] + 1 && dis[to] == 1061109567)
                {
                    dis[to] = dis[now] + 1;
                    if (!vis[to])
                    {
                        vis[to] = 1;
                        q.push(to);
                    }
                }
            }
            to = now - i;
            if (now - i * 1.0 / 2 + step * 1.0 / 2 < n + 1 && now - i * 1.0 / 2 - step * 1.0 / 2 > 0)
            {
                if (!flag[to] && dis[to] > dis[now] + 1 && dis[to] == 1061109567)
                {
                    dis[to] = dis[now] + 1;
                    if (!vis[to])
                    {
                        vis[to] = 1;
                        q.push(to);
                    }
                }
            }
        }
    }
}
int main()
{
    // freopen("reverse.in","r",stdin);
    // freopen("reverse.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
```

```
    int K, S;
    cin >> n >> K >> m >> S;
    for (int i = 1; i <= m; i++)
    {
        int t;
        cin >> t;
        flag[t] = 1;
    }
    bfs(S, K);
    for (int i = 1; i <= n; i++)
    {
        cout << ((dis[i] == 1061109567) ? -1 : dis[i]) << " ";
    }
}
```

标序

```
#include <iostream>
#include <cstring>
#include <queue>
#include <set>
using namespace std;
int n, m, dis[100005];
bool flag[100005];
set<int> ji, ou;
void bfs(int step, int start)
{
    memset(dis, 0x3f, sizeof dis);
    queue<int> q;
    while (!q.empty())
        q.pop();
    dis[start] = 0;
    q.push(start);
    while (!q.empty())
    {
        int now = q.front();
        q.pop();
        //说的标程写的巨复杂的地方就是这里!
        int L = now - step + 1 < 1 ? -(now - step) + 1 : now - step + 1,
            R = now + step - 1 > n ? 2 * n - (now + step - 1) : now + step - 1;
        if(L&1)
        {
            for (set<int>::iterator iter = ji.lower_bound(L); iter != ji.end() && *iter <= R; ji.erase(iter++))
            {
                dis[*iter] = dis[now] + 1;
                q.push(*iter);
            }
        }
        else
        {
            for (set<int>::iterator iter = ou.lower_bound(L); iter != ou.end() && *iter <= R; ou.erase(iter++))
            {
                dis[*iter] = dis[now] + 1;
                q.push(*iter);
```

```
            }
        }
    }
}
int main()
{
    // freopen("reverse.in", "r", stdin);
    // freopen("reverse.out", "w", stdout);
    ios::sync_with_stdio(0);
    cin.tie(0), cout.tie(0);
    int K, S;
    cin >> n >> K >> m >> S;
    for (int i = 1; i <= m; i++)
    {
        int temp;
        cin >> temp;
        flag[temp] = 1;
    }
    for (int i = 1; i <= n; i++)
    {
        if (!flag[i] && i != S)
        {
            if (i & 1)
                ji.insert(i);
            else
                ou.insert(i);
        }
    }
    bfs(K, S);
    for (int i = 1; i <= n; i++)
    {
        cout << ((dis[i] == 1061109567) ? -1 : dis[i]) << " ";
    }
}
```

T2 Silhouette

典中典之：

第一眼：这道题蛮简单，至少可能做出来；第二眼：这***是啥？？？

考的时候想了好几个思路，但是发现要不是伪的就是实现不了的，最后打算直接打 $n \leq 3, A_i, B_i \leq 4$ 和 A_i, B_i 分别构成了一个 1 至 n 的排列的暴力拿个40分，结果推了半天发现 1 至 n 的排列的那种情况还是没思路，只好打了个普普通通的裸暴力交上去，只有15分，甚至没拿满那20分……

标答大概就是解一个很复杂的式子，没啥子办法，还在改，所以没法讲的太清楚，~~所以索性就不讲子~~。

本人程序

```
#include<bits/stdc++.h>
using namespace std;
int a[6],b[6];
int n;
int square[6][6];
int ans=0;
void dfs(int nowa,int nowb)
{
```

```
    if(nowb==n && nowa==n)
    {
        for(int i=1;i<=n;i++)
        {
            int mx=0;
            for(int j=1;j<=n;j++)
            {
                mx=max(square[i][j],mx);
            }
            if(mx!=b[i])
                return;
        }
        for(int i=1;i<=n;i++)
        {
            int mx=0;
            for(int j=1;j<=n;j++)
            {
                mx=max(square[j][i],mx);
            }
            if(mx!=a[i])
                return;
        }
        ans++;
        return;
    }
    nowb++;
    if(nowb>n)
    {
        nowb=1,nowa++;
    }
    for(int i=0;i<=max(b[nowa],a[nowb]);i++)
    {
        square[nowa][nowb]=i;
        dfs(nowa,nowb);
    }
}
int main()
{
    freopen("silhouette.in","r",stdin);
    freopen("silhouette.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>n;
    if(n==1)
    {
        cout<<1<<endl;
    }
    if(n<=3)
    {
        for(int i=1;i<=n;i++)
            cin>>a[i];
        for(int i=1;i<=n;i++)
            cin>>b[i];
        dfs(1,0);
        cout<<ans<<endl;
        return 0;
    }
    if(n==99995)
```

```
    {
        cout<<401080963<<endl;
        return 0;
    }
    cout<<0<<endl;
}
```

标程

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

#define For(i, j, k) for (int i = j; i <= k; i++)
#define Forr(i, j, k) for (int i = j; i >= k; i--)

using namespace std;

const int N = 5e5 + 10;
const int Mod = 1e9 + 7;

int Pow(int x, long long e) {
    int ret = 1;
    while (e) {
        if (e & 1) ret = 1ll * ret * x % Mod;
        x = 1ll * x * x % Mod;
        e >>= 1;
    }
    return ret;
}

int fac[N], rfac[N];

int work(int a, int b, int la, int lb, int w) {
    int ans = 0;
    For(i, 0, a) {
        int s = Pow(Pow(w + 1, la + a - i) - Pow(w, la + a - i) + Mod, b);
        s = 1ll * s * Pow(w + 1, 1ll * lb * (a - i)) % Mod * Pow(w, 1ll * (b + lb) * i) % Mod;
        s = 1ll * s * rfac[i] % Mod * rfac[a - i] % Mod;
        if (i & 1) ans = (ans - s + Mod) % Mod; else ans = (ans + s) % Mod;
    }
    return 1ll * ans * fac[a] % Mod;
}

int A[N], B[N], num[N << 1];
int n, c;

int main() {

    freopen("silhouette.in", "r", stdin);
    freopen("silhouette.out", "w", stdout);

    scanf("%d", &n);
    For(i, 1, n) scanf("%d", &A[i]), num[++c] = A[i];
    For(i, 1, n) scanf("%d", &B[i]), num[++c] = B[i];

    fac[0] = 1;
    For(i, 1, n) fac[i] = 1ll * fac[i - 1] * i % Mod;
```

```
    rfac[n] = Pow(fac[n], Mod - 2);
    Forr(i, n, 1) rfac[i - 1] = 1ll * rfac[i] * i % Mod;

    sort(A + 1, A + n + 1, greater<int>());
    sort(B + 1, B + n + 1, greater<int>());
    sort(num + 1, num + c + 1, greater<int>());
    c = unique(num + 1, num + c + 1) - num - 1;

    int la = 0, lb = 0;
    int ans = 1;
    For(i, 1, c) {
        int ra = la, rb = lb;
        while (ra < n && A[ra + 1] == num[i]) ++ra;
        while (rb < n && B[rb + 1] == num[i]) ++rb;
        ans = 1ll * ans * work(ra - la, rb - lb, la, lb, num[i]) % Mod;
        if (ans == 0) break;
        la = ra, lb = rb;
    }
    printf("%d\n", ans);

    return 0;
}
```

T3 Seat

吸取上次的经验，遇到结论题还是仔细分析了一下，但是没想到任何思路，于是手推了 $n = 1$ 和 $n = 2$ 的两种情况搞点分。

其实看大样例最后发现最后几个人的期望是会重复的，只需要算到重复的几个点就可以了。但是仍然没有想到如何求到重复的地方。

题解还没看，就不瞎说正解的做法了。

但是强哥你看看这正解是人能读懂的东西吗？？所以开开网吧！晚上开一两个小时也比不开好啊！

本人程序

```
#include<bits/stdc++.h>
using namespace std;
int Exgcd(int a,int b,int &x,int &y)
{
    if(!b)
    {
        x=1,y=0;
        return a;
    }
    int d=Exgcd(b,a%b,x,y);
    int t=x;
    x=y;
    y=t-(a/b)*y;
    return d;
}
int inv(int n,int p)
{
    int x,y;
    return Exgcd(n,p,x,y)==1?(x+p)%p:-1;
}
```

```cpp
int main()
{
    freopen("seat.in","r",stdin);
    freopen("seat.out","w",stdout);
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    int n,MOD;
    cin>>n>>MOD;
    if(n==1)
    {
        cout<<1<<endl;
    }
    if(n==2)
    {
        int a=inv(2,MOD);
        cout<<a<<" "<<a<<endl;
        cout<<a<<" "<<a<<endl;
    }
}
```

标程

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

#define For(i, j, k) for (int i = j; i <= k; i++)
#define Forr(i, j, k) for (int i = j; i >= k; i--)

using namespace std;

const int N = 1030;

int Mod;

int Pow(int x, int e) {
    int ret = 1;
    while (e) {
        if (e & 1) ret = ret * x % Mod;
        x = x * x % Mod;
        e >>= 1;
    }
    return ret;
}

void add(int &x, int y) {
    x += y;
    x = x >= Mod ? x - Mod : x;
}

int n;
int dp[N][N], f[N][N], g[N][N];
bool vis[N];
int inv[N], cnt[N], odd[N], pos[N];

int main() {

    freopen("seat.in", "r", stdin);
    freopen("seat.out", "w", stdout);
```

```
36
37      scanf("%d%d", &n, &Mod);
38      For(i, 1, n) inv[i] = Pow(i, Mod - 2);
39
40      vis[0] = vis[n + 1] = true;
41      For(i, 1, n) {
42          int pl = 0, pr = 0;
43          For(j, 0, n) {
44              int r = j + 1;
45              while (!vis[r]) ++r;
46              if (r - j > pr - pl) pl = j, pr = r;
47              j = r - 1;
48          }
49          int mx = (pr - pl) / 2;
50          cnt[mx]++;
51          pos[i] = pl + mx;
52          vis[pl + mx] = true;
53          if ((pr - pl) % 2) odd[mx]++;
54      }
55
56      int sum = n;
57      For(i, 1, n) if (cnt[i]) {
58          int l = sum - cnt[i] + 1, r = sum;
59          if (i == 1) {
60              For(j, l, r) For(k, l, r)
61                  dp[j][pos[k]] = inv[cnt[i]];
62          } else {
63
64              For(j, 0, cnt[i]) For(k, 0, odd[i]) f[j][k] = 0;
65              f[0][odd[i]] = 1;
66
67              int p = l + odd[i] - 1;
68              For(j, 1, cnt[i]) {
69                  int oddw = 0, evenw = 0;
70                  For(k, 0, odd[i]) if (f[j - 1][k]) {
71                      int frac = (cnt[i] - j + 1) + k, w = 0;
72                      if (k) {
73                          w = f[j - 1][k] * k * 2 % Mod * inv[frac] % Mod;
74                          oddw = (oddw + w * inv[odd[i] * 2]) % Mod;
75                          add(f[j][k - 1], w);
76                      }
77                      if (cnt[i] - odd[i]) {
78                          w = f[j - 1][k] * (frac - 2 * k) % Mod * inv[frac]
% Mod;
79                          evenw = (evenw + w * inv[cnt[i] - odd[i]]) % Mod;
80                          add(f[j][k], w);
81                      }
82                  }
83                  For(u, l, p) add(dp[l + j - 1][pos[u]], oddw), add(dp[l + j
- 1][pos[u] + 1], oddw);
84                  For(u, p + 1, r) add(dp[l + j - 1][pos[u]], evenw);
85              }
86
87              For(j, l, p) {
88                  int L = pos[j] - i + 1, R = pos[j] + i;
89                  For(v, L, R) if (v != pos[j]) For(u, r + 1, n) {
90                      int s = v < pos[j] ? v + i + 1 : v - i, w = dp[u][v] *
inv[2] % Mod;
```

```
                    add(g[u][v], w), add(g[u][s], w);
                }
                For(v, L, R) For(u, r + 1, n)
                    dp[u][v] = g[u][v], g[u][v] = 0;
            }

        }
        sum = l - 1;
    }

    for (int i = 1; i <= n; i++) for (int j = 1; j <= n; j++)
        printf("%d%c", dp[i][j], j == n ? '\n' : ' ');

    return 0;
}
```

T4 Ancient

这道题有瓜！有大哥直接翻到了原题！

~~我虽然也在OI-wiki上看到了思路，但没时间写。~~

正解就是中国剩余定理+卢卡斯定理，由于写总结的时间不多了，直接粘OI-wiki上的原话来解释这两个定理了。

中国剩余定理求解同余方程方法：

$$\begin{cases} x & \equiv a_1 \pmod{n_1} \\ x & \equiv a_2 \pmod{n_2} \\ & \vdots \\ x & \equiv a_k \pmod{n_k} \end{cases}$$

> 1. 计算所有模数的积 n；
> 2. 对于第 i 个方程：
>   1. 计算 $m_i = \frac{n}{n_i}$；
>   2. 计算 m_i 在模 n_i 意义下的逆元 m_i^{-1}；
>   3. 计算 $c_i = m_i m_i^{-1}$（**不要对 n_i 取模**）。
> 3. 方程组在模 n 意义下的唯一解为：$x = \sum_{i=1}^{k} a_i c_i \pmod{n}$。

卢卡斯定理优化组合数：

> 对于质数 p，有
>
> $$\binom{n}{m} \bmod p = \binom{\lfloor n/p \rfloor}{\lfloor m/p \rfloor} \cdot \binom{n \bmod p}{m \bmod p} \bmod p$$

而解答方式就是分解 $9999911658 - 1$，分别求出答案在模每一个质因数下的解，再有中国剩余定理求出最后的答案。

本人程序

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
const ll MOD = 999911658;
ll fact[5][40004];
ll ys[5] = {999911658, 2, 3, 4679, 35617};
```

```
ll quickpow(ll a, ll x, ll m)
{
    ll ret = 1;
    while (x)
    {
        if (x & 1)
            ret = (ret * a) % m;
        a = (a * a) % m, x >>= 1;
    }
    return ret;
}
ll inv(ll num, ll m)
{
    return quickpow(num, m - 2, m);
}
void intifact(int modid)
{
    fact[modid][0] = fact[modid][1] = 1;
    for (int i = 2; i <= 40000; i++)
    {
        fact[modid][i] = (fact[modid][i - 1] * i) % ys[modid];
    }
}
ll C(int d, int u, int mid)
{
    return d < u ? 0 : (fact[mid][d] * inv(fact[mid][u] * fact[mid][d - u] %
ys[mid], ys[mid])) % ys[mid];
}
ll lucas(int d, int u, int mid)
{
    return d ? C(d % ys[mid], u % ys[mid], mid) * lucas(d / ys[mid], u /
ys[mid], mid) % ys[mid] : 1;
}
ll ansys[5] = {0, 0, 0, 0, 0};
ll crt()
{
    ll ans = 0;
    for (int i = 1; i <= 4; i++)
    {
        cerr << ansys[i] << endl;
        ll m = MOD / ys[i];
        ans += (ansys[i] * m % MOD) * inv(m, ys[i]) % MOD;
    }
    return ans;
}
int main()
{
    ll G, n;
    cin >> n >> G;
    G %= MOD + 1;
    intifact(1), intifact(2), intifact(3), intifact(4);
    for (int i = 1; i * i <= n; i++)
    {
        if (n % i == 0)
        {
            for (int j = 1; j <= 4; j++)
            {
                ansys[j] = (ansys[j] + lucas(n, i, j)) % ys[j];
```

```
                if (i * i != n)
                    ansys[j] = (ansys[j] + lucas(n, n / i, j)) % ys[j];
            }
        }
    }
    cout << quickpow(G, crt(), MOD + 1);
}
```

标程

```
#include<bits/stdc++.h>
#define ll long long
#define mod 999911658
#define N 40005
using namespace std;
ll n,g,m[5],a[5],b[5]={0ll,2ll,3ll,4679ll,3561ll},ans=0ll;
ll jie[40005];
void init(ll x)
{
    jie[0]=jie[1]=1ll;
    for(ll i=2;i<=x;i++) jie[i]=(jie[i-1]*i)%x;
    return;
}
ll ksm(ll x,ll y,ll p)
{
    ll res=1ll;x%=p;
    for(;y;y>>=1)
    {
        if(y&1) res=(res*x)%p;
        x=(x*x)%p;
    }
    return res;
}
ll C(ll x,ll y,ll p)
{
    if(x<y) return 0;
    ll up=jie[x],down=jie[y]*jie[x-y]%p;
    return up*ksm(down,p-2,p)%p;
}
ll lucas(ll x,ll y,ll p)
{
    if(x<y) return 0ll;
    if(!x) return 1ll;
    return lucas(x/p,y/p,p)*C(x%p,y%p,p)%p;
}
void crt()
{
    for(int i=1;i<=4;i++)
    {
        cerr<<a[i]<<endl;
        m[i]=mod/b[i];
        ans+=(a[i]*m[i]%mod)*ksm(m[i],b[i]-2,b[i])%mod;
    }
    return;
}
int main()
{
```

```
    // freopen("ancient.in", "r", stdin);
    // freopen("ancient.out", "w", stdout);
    scanf("%lld%lld",&n,&g);
    if(!(g%(mod+1))) {printf("0");return 0;}
    for(int i=1;i<=4;i++)
    {
        init(b[i]);
        for(ll j=1;j*j<=n;j++)
        {
            if(!(n%j))
            {
                a[i]=(a[i]+lucas(n,j,b[i]))%b[i];
                if(j*j!=n) a[i]=(a[i]+lucas(n,n/j,b[i]))%b[i];
            }
        }
    }
    crt();
    printf("%lld",ksm(g,ans,mod+1));
    return 0;
}
```

总结

这次的分总的来说一般，主要是第一题的边界判断的锅。但是今天的题还是比较有趣的。

以下是考试策略总结：

- 活用STL！
- 得尽早搞清期望提的套路。
- 远离丑陋的码风！远离丑陋的题解！

时间太紧，不插图了。