CSP-S 2022 初赛模拟

1. 本卷总分一百分，考试时间 120 分钟。
2. 出题人很菜，奉命出题，秒了别骂我。

一、单项选择题（共 15 题，每题 2 分，共计 30 分；每题有且仅有一个正确选项）

1. 十进制数 89 转成二进制原码的结果是：（C）

   A. $(11111001)_2$

   B. $(11111011)_2$

   C. $(01011001)_2$

   D. $(01000001)_2$

   解析：略。

2. 在 Linux 系统终端中，用于比较文件的命令是：（D）

   A. `cat`

   B. `compare`

   C. `fc`

   D. `diff`

   解析：A 是查看文件，B 没有这个命令，C 是 windows 下的文件比较，D 正确。

3. 在 Linux 系统终端中，远程连接其他电脑的命令是：（B）

   A. `ping 192.60.8.17 -p 22`

   B. `ssh noilinux@192.60.8.17 -p 22`

   C. `mstsc /v:192.60.8.17`

   D. `sudo rm -rf /*`

   解析：A 是 ping 端口，B 是 ssh，C 是 windows 远程桌面，D 是删除所有文件。

4. 对一个 n 个顶点，m 条边的带正权有向简单图使用 Dijkstra 算法计算单源最短路时，如果使用一个堆，各操作复杂度如下，则整个 Dijkstra 算法的时间复杂度为：（C）

操作	复杂度
查询堆内最小值	$\Theta(\log n)$
合并两个堆	$\Theta(\sqrt{n})$
将堆内一个元素变小	$\Theta(1)$
弹出堆内最小值	$\Theta(\log n)$

A. $\Theta(n+m\log n)$

B. $\Theta((n+m)\log n)$

C. $\Theta(m+n\log n)$

D. $\Theta(m\sqrt{n}+\log n)$

解析：dijkstra 只要用到加查删，和合并没关系，都 log 的，复杂度就 m + nlogn

5. 现有一个地址区间为 $0 \sim n-1$ （n 为质数）的哈希表，哈希函数为 $h(x) = x^{-1} \bmod n$，若发生冲突，则会放弃存储。现在要从小到大依次存储 $1 \sim n-1$ 的所有整数，请问冲突个数的级别为（A）

   A. 0（无冲突）

   B. $O(1)$（非零）

   C. $O(\sqrt{n})$

   D. $O(\log n)$

   解析： 简单推一下，假设 x 和 y 冲突了，就有 xp = yp = 1 (mod n)，但是由于 n 是质数且 x 不等于 y，有 y >= x + np > n，所以只存储 0 到 n - 1 不会冲突。

6. 下列算法中，没有运用分治思想的一项是（D）

   A. 归并排序算法

   B. 求二叉树的前序遍历

   C. 快速排序算法

   D. 求二叉树的层次遍历

   解析： 略。

7. 设 $x = (100101)_2$，下列表达式值为 `true` 的一项是： （A）

   A. `x & -x & 1`

   B. `x >> 1 & 1`

   C. `(x - (x & -x)) & 1 << 3`

   D. `x << 1 & x`

   解析： 略。

8. 有 4 个结点和 4 条边的有标号简单无向图的数量是： （ A）

   A. 15

   B. 16

   C. 6

   D. 4

   解析： 略。

9. 栈 S 的进栈序列为 `1 2 3 4`，有多少种不同的出栈序列： （D）

   A. 4

   B. 16

   C. 12

   D. 14

   解析： 卡特兰数。

10. 若某算法的时间计算表示为递推关系
    $T(n) = 8T(\frac{n}{2}) + 2n$
    $T(1) = 1$
    则该算法的时间复杂度是（C）

A. $\Theta(n)$

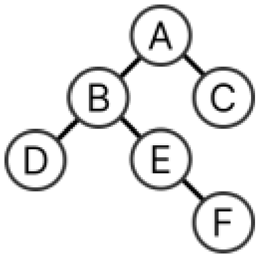
B. $\Theta(n^2)$

C. $\Theta(n^3)$

D. $\Theta(n \log n)$

解析：主定理。

11. 下图是一棵二叉树，它的后序遍历是（B）。



A. `BDEFCA`

B. `DFEBCA`

C. `DBEFCA`

D. `BCDEFA`

解析：略。

12. 有 8 个苹果从左到右排成一排，从中挑选至少一个苹果，并且不能同时挑选相邻的两个苹果的方案数为（D）。

A. 24

B. 36

C. 48

D. 54

解析：枚举一下选几个。

13. 可可爱爱数学题 I：

高斯还是个小 P 孩的时候就求出

$$\sum_{i=1}^{n} i = \frac{n \times (n+1)}{2}$$

LT 还是个小 P 孩的时候求出

$$\sum_{i=1}^{n-1} \frac{1}{i(i+1)} = 1 - \frac{1}{n}$$

现在，你还是个小 P 孩的时候，你要求出（$m > 1, n > 0$ 且 m, n 均为正整数）：

$$\sum_{i=1}^{n} \frac{1}{\prod_{j=i}^{i+m-1} j} = \ ?$$

你的答案是（B）

A. $\dfrac{n^{\underline{1-m}} - 1}{1-m}$

B. $\dfrac{n^{\underline{1-m}} - 0^{\underline{1-m}}}{1-m}$

C. $\dfrac{(n+m-1)^{\underline{n}} - 0^{\underline{1-m}}}{(m-1)(n+m-1)!}$

D. $\dfrac{(n+m-1)^{\underline{n}}}{(m-1)(n+m-1)!}$

解析： 发现是个简单求和，有限微积分一下，用那个 $\delta(x^{\underline{m}}) = mx^{\underline{m-1}}$。

14. 若某算法的时间计算表示为递推关系
    $T(n) = 2T(\frac{n}{2}) + n\log n$
    $T(1) = 1$
    则该算法的时间复杂度是（D）

A. $\Theta(n)$

B. $\Theta(n^2)$

D. $\Theta(n\log n)$

C. $\Theta(n\log^2 n)$

解析：主定理的 case2，历年都没考到好像。

15. 中国计算机协会成立于（B）年。

A. 1961

B. 1962

C. 1971

D. 1972

解析：这你让我怎么写，略。

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填 √，错误填 ×；除特殊说明外，判断题 2 分，选择题 3 分，共计 40 分）

```cpp
#include <bits/stdc++.h>
using namespace std;

const int N = 3000010;

int n, a[N], f[N];
stack<int> s;

int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
    for (int i = n; i >= 1; i--) {
        while (!s.empty() && a[s.top()] <= a[i]) s.pop();
        f[i] = s.empty() ? 0 : s.top();
        s.push(i);
    }
    for (int i = 1; i <= n; i++) printf("%d ", f[i]);
    return 0;
}
```

假设输入的所有数的绝对值都不超过 1000，完成下面的判断题和单选题：

判断题

16. 把第 7 行的 `stack` 改为 `vector`，不会影响程序运行的结果。（F）

解析：喜提编译错误。

17. 代码的时间复杂度是 $\Theta(n)$。（T）

解析：人畜无害单调栈。

18. 交换第 13 行 `&&` 两侧的表达式，不会影响程序运行的结果。（F）

解析：喜提段错误。

19. 当序列单调递减时，`f` 数组单调不减。(T)

解析：人畜无害单调栈。

单选题

20. f_i 的含义是（B）。

A. i 后第一个大于 a_i 的数

B. i 后第一个大于 a_i 的数的下标

C. i 前大于 a_i 数的下标

D. i 前第一个大于 a_i 的数

解析：人畜无害单调栈。

21. （2 分）当输入为 `5 1 4 2 3 5` 时，输出为（A）。

A. `2 5 4 5 0`

B. `4 5 3 5 0`

C. `0 0 2 2 0`

D. `0 0 4 4 0`

解析：会 20 就会 21。

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const int N = 200010;
const int p = 1000000007;
const int inv2 = 500000004;

struct node {
    ll a, b;
    node(ll a = 0, ll b = 0) : a(a), b(b) {}

    node operator + (const node& x) const {
        node res;
        res.a = (a + x.a) % p;
        res.b = (b + x.b) % p;
        return res;
    }
    node operator - (const node& x) const {
        node res;
        res.a = (a - x.a + p) % p;
```

```
        res.b = (b - x.b + p) % p;
        return res;
    }
    node operator * (const node& x) const {
        node res;
        res.a = (a * x.a + 5 * b * x.b) % p;
        res.b = (a * x.b + x.a * b) % p;
        return res;
    }
};

inline node qpow(node a, ll t) {
    node res = node(1, 0);
    while (t) {
        if (t & 1) res = res * a;
        a = a * a;
        t >>= 1;
    }
    return res;
}

ll n, ans;
node x, y, res;

int main() {
    cin >> n;
    x = node(inv2, inv2);
    y = node(inv2, p - inv2);
    x = qpow(x, n);
    y = qpow(y, n);
    res = x - y;
    cout << res.b << endl;
    return 0;
}
```

输入的数据为在 $[1, 2^{63})$ 中的正整数。

判断题

22. 该程序没有编译错误。（T）

解析：放了这道题是为了让选手好好看看代码，理解一下在干嘛，不然看不懂有趣数学题。

23. 该程序的时间复杂度是 $\Theta(n)$。（F）

解析：快速幂复杂度分析都不会了？

24. 把 `const node &x` 都替换为 `const node x`，程序仍能正常运行。（T）

解析：啊那确实，没动过 x。

单选题

25. `node(3, 5) * node(8)` 的结果是（D）。

A. `node(24, 0)`

B. `node(11, 13)`

C. `node(3, 40)`

D. `node(24, 40)`

解析：这其实是个 fibonacci 数列的通项公式解法，因为 5 对于 1e9 + 7 没有二次剩余不能使用公式中的根号五，所以学过群论的人都知道要扩域，然后就扩扩就完了。

26. （2 分）当输入为 `10` 时，程序的输出是（C）。

A. `0`

B. `5050`

C. `55`

D. `117`

解析：f(10) = 55。

27. `res.a` 的值是（A）。

A. `0`

A. `1`

B. `n`

C. `(2 * n) % p`

解析：根据公式要输出 res / sqrt(5)，而且这是数列的第 n 项，是个整数，所以 res / sqrt(5) 是个整数，所以 res（res.a + res.b * sqrt(5)）形如 kqrt(5)，那 res.a 就是 0。

```cpp
#include <bits/stdc++.h>
using namespace std;

map<string, string> def;
map<string, bool> vis;

inline bool isID(char ch) {
    if ((ch >= 65 && ch <= 90) || (ch >= 97 && ch <= 122) || (ch >= 48 && ch <= 57) || ch == '_') return true;
    else return false;
}

inline tuple<string, bool, int> reads(string src, int pos) {
    if (isID(src[pos])) {
        string str = "";
        int len = src.length();
        while (pos < len && isID(src[pos])) {
            str += src.substr(pos, 1);
            pos++;
        }
        return make_tuple(str, true, pos);
    } else {
        string str = "";
        str += src.substr(pos, 1);
        return make_tuple(str, false, pos + 1);
    }
}

inline pair<string, int> readDef(string src, int pos) {
    int len = src.length();
```

```cpp
    while (pos < len && isspace(src[pos])) pos++;
    string str = "";
    while (pos < len && !isspace(src[pos])) str += src[pos++];
    return make_pair(str, pos);
}

void solve(string str) {
    if (def.count(str) != 0 && !vis[str]) {
        vis[str] = true;
        string src = str;
        str = def[str];
        tuple<string, bool, int> getReads = make_tuple("", true, 0);
        int len = str.length();
        vector<string> vec;
        while (get<2>(getReads) < len) {
            getReads = reads(str, get<2>(getReads));
            vec.push_back(get<0>(getReads));
        }
        for (string each : vec) solve(each);
        vis[src] = false;
    } else cout << str;
}

int main() {
    int n;

    cin >> n;
    cin.ignore();

    for (int i = 1; i <= n; i++) {
        string str;
        getline(cin, str);
        tuple<string, bool, int> getReads = make_tuple("", true, 0);
        int len = str.length();

        while (get<2>(getReads) < len) {
            getReads = reads(str, get<2>(getReads));

            if (get<1>(getReads)) {
                vis.clear();
                for (auto each : def) vis[each.first] = false;
                solve(get<0>(getReads));
            } else if (get<0>(getReads) == "#") {
                getReads = reads(str, get<2>(getReads));
                if (get<0>(getReads) == "define") {
                    pair<string, int> from = readDef(str, get<2>(getReads));
                    string to = str.substr(from.second + 1);
                    def[from.first] = to;
                    getReads = make_tuple(to, true, len);
                } else if (get<0>(getReads) == "undef") {
                    string from = str.substr(get<2>(getReads) + 1);
                    def.erase(from);
                    getReads = make_tuple(from, true, len);
                } else {
                    cout << "#";
                    getReads = make_tuple("#", false, get<2>(getReads) + 1);
                }
            } else cout << get<0>(getReads);
```

```
        }
        cout << "\n";
    }
    return 0;
}
```

输入的第一行包含一个正整数 n，接下来输入 n 行字符串。

判断题

28. `isID('q')` 表达式的值是 `true`。（T）

解析：联合省选预处理器。

29. 在输入合法的情况下，程序的输入行数和输出行数一样。（F）

解析：输入多一行 n。

30. 输出可能只包含不可见字符（换行，空格与 `EOF`）。（T）

解析：那确实。

单选题

31. 若输入如下数据，程序的输出为（B）。（忽略换行）

```
2
#define a a a
a
```

A. `a`

B. `a a`

C. `a a a`

D. `a a a a a ...`（陷入死循环）

解析：那确实。

32. （2 分）若输入如下数据，程序的输出为（D）。

```
5
#define a b
#ifdef a
    #define p q
#endif
a p
```

A.

```



a p
```

B.

```



b q
```

C.

```
   #ifdef b

   #endif

   b q
```

D.

```
   #b

   #

   b q
```

解析：可以发现他没处理 `#ifdef`，这是非预期输入不能根据题目要求来脑内想象，手动模拟一下得到 D。

33.（2 分）该程序（B）。

A. 完全按照 GNU C++11 标准处理了 `#define` 与 `#undef` 两个预处理命令（在这两个命令上行为和预处理器一样）

B. 部分按照 GNU C++11 标准处理了 `#define` 与 `#undef` 两个预处理命令（在这两个命令上行为和预处理器不一样）

C. 完全按照 GNU C++11 标准处理了 C++ 的宏定义预处理命令（在所有宏定义命令上行为和预处理器一样）

D. 在 `#define` 与 `#undef` 两个预处理命令行为上和 GNU-C++11 不一样，具体表现为会陷入死循环

解析：那确实，上面说了，而且对于代码 `#define a b` `cout << "a";` 被处理后会输出 b，寄了。

三、完善程序（单选题，每小题 3 分，共计 30 分）

（动态最大子段和） n 个数，q 次操作，每次操作有三个数 opt，x，y。

如果 $opt=1$，则把第 x 个数修改为 y。

如果 $opt=2$，则输出区间 $[x,y]$ 的最大子段和。（此时保证 $[x,y]\neq\varnothing$）

提示：

考虑没有修改的情况，可以使用平凡的 dp 算法在线性时间解决，设 f_i 表示以 a_i 结尾的最大子段和，g_i 表示前 i 项的，有如下转移：

$f_i=\max\{f_{i-1}+a_i,a_i\}$

$g_i=\max\{g_{i-1},f_i\}$

当然这题没有这么简单，你还有修改没有解决。

我们有矩阵乘法：

$$C_{i,j} = \sum_k A_{i,k} \times B_{k,j}$$

若将其改成

$$C_{i,j} = \max_k \{A_{i,k} + B_{k,j}\}$$

修改后的广义矩阵乘法仍然具有结合律。（核心在于 $\max$ 运算与加法一样对加法具有分配律，此处不证）

所以可以对每个元素构造一个矩阵，用矩阵乘法来表示递推。

所以可以使用线段树维护 n 个矩阵。区间求积，单点修改矩阵，试补全以下程序。

```
#include <bits/stdc++.h>
using namespace std;

template <typename T> inline void read(T& x) {
    int f = 0, c = getchar(); x = 0;
    while (!isdigit(c)) f |= c == '-', c = getchar();
    while (isdigit(c)) x = x * 10 + c - 48, c = getchar();
    if (f) x = -x;
}
template <typename T, typename... Args>
inline void read(T& x, Args&... args) {
    read(x); read(args...);
}
template <typename T> void write(T x) {
    if (x < 0) x = -x, putchar('-');
    if (x > 9) write(x / 10);
    putchar(x % 10 + 48);
}
template <typename T> void writeln(T x) { write(x); puts(""); }
template <typename T> inline bool chkmin(T& x, const T& y) { return y < x ? (x = y, true) : false; }
template <typename T> inline bool chkmax(T& x, const T& y) { return x < y ? (x = y, true) : false; }

const int maxn = 5e4 + 207;
const int inf = INT_MAX >> 2;

struct Matrix {
    int data[3][3];
};

Matrix mat[maxn << 2];
int a[maxn];
int n, m;

inline Matrix mul(const Matrix& A, const Matrix& B) {
    Matrix C;
    for (int i = 0; i <= 2; ++i)
        for (int j = 0; j <= 2; ++j)
            C.data[i][j] = ①; // 1
    for (int k = 0; k <= 2; ++k)
        for (int i = 0; i <= 2; ++i)
            for (int j = 0; j <= 2; ++j)
                chkmax(C.data[i][j], A.data[i][k] + B.data[k][j]);
    return C;
```

```
}

inline void update(int o) {
    ② // 2
}
void build(int o, int l, int r) {
    if (l == r) {
        mat[o].data[0][0] = mat[o].data[0][2] = mat[o].data[1][0] = mat[o].data[1][2] = a[l];
        mat[o].data[0][1] = mat[o].data[2][0] = mat[o].data[2][1] = -inf;
        mat[o].data[1][1] = mat[o].data[2][2] = 0;
        return;
    }
    int mid = (l + r) >> 1;
    build(o << 1, l, mid);
    build(o << 1 | 1, mid + 1, r);
    update(o);
}
void modify(int o, int l, int r, int p, int v) {
    if (l == r) {
        ③ // 3
        return;
    }
    int mid = (l + r) >> 1;
    if (p <= mid) modify(o << 1, l, mid, p, v);
    else modify(o << 1 | 1, mid + 1, r, p, v);
    update(o);
}
Matrix query(int o, int lb, int rb, int l, int r) {
    if (④) return mat[o]; // 4
    int mid = (lb + rb) >> 1;
    if (l <= mid && r > mid)
        return mul(query(o << 1, lb, mid, l, r), query(o << 1 | 1, mid + 1, rb, l, r));
    else {
        if (l <= mid) return query(o << 1, lb, mid, l, r);
        else return query(o << 1 | 1, mid + 1, rb, l, r);
    }
}

int main() {
    read(n);
    for (int i = 1; i <= n; ++i) read(a[i]);
    build(1, 1, n);
    read(m);
    while (m--) {
        int q; read(q);
        if (q) {
            int l, r; read(l, r);
            if (l > r) swap(l, r);
            Matrix ret = query(1, 1, n, l, r);
            writeln(⑤); // 5
        } else {
            int x, y;
            read(x, y);
            a[x] = y;
            modify(1, 1, n, x, y);
        }
```

```
    }
    return 0;
}
```

34. ① 处应填（D）。

A. `0`

B. `1`

C. `inf`

D. `-inf`

解析：在这个值域里，-inf 是 max 运算的零元（对于 x 属于值域，有 max(x, -inf) = x = max(-inf, x)）。

35. ② 处应填（D）。

A. `mat[o] = mat[o << 1] + mat[o << 1 | 1];`

B. `mat[o] = mat[o << 1] * mat[o << 1 | 1];`

C. `mat[o] = add(mat[o << 1], mat[o << 1 | 1]);`

D. `mat[o] = mul(mat[o << 1], mat[o << 1 | 1]);`

解析：瞄准不会编译错误的那个，送大分。

36. ③ 处应填（C）。

A. `mat[o].data[0][2] = mat[o].data[1][2] = v;`

B. `mat[o].data[0][0] = mat[o].data[0][2] = mat[o].data[1][2] = v;`

C. `mat[o].data[0][0] = mat[o].data[0][2] = mat[o].data[1][0] = mat[o].data[1][2] = v;`

D. `mat[o].data[0][0] = mat[o].data[0][2] = mat[o].data[1][0] = mat[o].data[1][1] = mat[o].data[2][2] = v;`

解析：推一下矩阵就知道该填什么，而且上面有写。

37. ④ 处应填（A）。

A. `l <= lb && r >= rb`

B. `lb <= l && rb <= r`

C. `lb <= l && rb >= r`

D. `l <= lb && r <= rb`

解析：一眼丁真，鉴定为：纯纯的线段树。

38. ⑤ 处应填（B）。

A. `ret.data[1][0]`

B. `max(ret.data[1][0], ret.data[1][2])`

C. `ret.data[1][2]`

D. `max(ret.data[1][0], max(ret.data[1][2], ret.data[1][3]))`

解析：这个在做题时也是难点，答案最终存哪里，矩阵有四个地方是 v，但有的地方存的是 g，有的地方是 f，只不过对于这个单点来说 g(i) = f(i) = v 罢了，手动模拟一下矩阵乘法就知道哪里是答案了。

（可可爱爱数学题 II） 在 $n \times n$ 的国际象棋棋盘上放 n 个车，要求满足两个条件：

- 所有的空格子都能被至少一个车攻击到。
- 恰好有 k 对车可以互相攻击到。

（两辆车在同一行或者同一列并且中间没有隔其他车就可以互相创）

答案对 998244353 取模。

提示：

由于此题太简单且为了节约资源，贯彻环保理念，这里不放提示（好吧提示在代码注释里）。

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

const int N = 500010;
const int p = 998244353;

ll qpow(ll x, ll y) {
    y %= p;
    ll now = x % p, ans = 1;
    while (y) {
        if (y & 1) {
            ans = ans * now % p;
        }
        now = now * now % p;
        y >>= 1;
    }
    return ans;
}

ll fac[N + 10], inv[N + 10];
void init() {
    fac[0] = 1;
    for (int i = 1; i <= N; i++) {
        fac[i] = fac[i - 1] * i % p;
    }
    inv[N] = qpow(fac[N], p - 2);
    for (int i = N - 1; i >= 0; i--) {
        inv[i] = ①; // 1
    }
}
ll C(ll x, ll y) {
    if (x < y || x < 0) {
        return 0;
    } else {
        return fac[x] * inv[y] % p * inv[x - y] % p;
    }
}
ll A(ll x, ll y) {
    if (x < y) return 0;
    else return ②; // 2
}

```

```cpp
/*
（提示：你自己想啊看什么看？2300 都做不出来？这个位置我本来放了一道 3500 哦？）
*/

ll n, k;

ll s(ll y, ll x) {
    ll res = 0;
    for (int i = 0; i <= x; i++) {
        ll temp = ③; // 3
        if ((x - i) % 2) {
            temp = ④;
        }
        res = ((res + temp) % p + p) % p;
    }
    return res;
}

int main() {
    init();
    cin >> n >> k;
    if (k == 0) {
        cout << fac[n];
    } else if (n >= k) {
        cout << ⑤ << endl; // 5
    } else {
        cout << 0 << endl;
    }
}
```

39. ① 处应填（A）。

A. `inv[i + 1] * (i + 1) % p`

B. `inv[i + 1] * i % p`

C. `qpow(i, p - 2)`

D. `qpow(i, p - 2) % p`

40. ② 处应填（D）。

A. `fac[x] * fac[y] % p`

B. `fac[x] * fac[x - y] % p`

C. `fac[x] * inv[y] % p`

D. `fac[x] * inv[x - y] % p`

41. ③ 处应填（C）。

A. `qpow(i, y) * inv[i] % p * fac[x - i] % p`

B. `qpow(i, y) * fac[i] % p * inv[x - i] % p`

C. `qpow(i, y) * inv[i] % p * inv[x - i] % p`

D. `qpow(i, y) * fac[i] % p * fac[x - i] % p`

42. ④ 处应填（B）。

A. `temp * 2 % p`

B. `temp * (-1)`

C. `temp * temp % p`

D. `temp * inv[x - i] % p`

43. ⑤ 处应填（B）。

A. `2 * A(n, n - k) % p * fac[n - k] % p * S(n, n - k) % p`

B. `2 * C(n, n - k) % p * fac[n - k] % p * S(n, n - k) % p`

C. `A(n, n - k) % p * fac[n - k] % p * S(n, n - k) % p`

D. `C(n, n - k) % p * fac[n - k] % p * S(n, n - k) % p`

解析：放 o2 考场注释：

```
行列中 肯定有一个每一 行 / 列 都有且仅有一个数字
钦定这个是行 然后答案 * 2
考虑列
记 cnt[i] 表示第 i 列车的数量
有 k = sum max(0, cnt[i] - 1), n = sum cnt[i]
然后发现 n + k = sum max(1, cnt[i])
然后减一减就得到了 k = sum [cnt[i] = 0]
k 就是 cnt[i] = 0 的个数

对样例进行一个拟的模
#1
3 2
这个时候，满足条件的 cnt 只有 {0, 0, 3}
然后有 3 种排列
然后还可以转 90 度又是一个方案
哈哈答案就是 6
然后我懂了
#2
3 3
三个都是 0，但是和要为 3
放屁
#3
4 0
这个比较特殊，都是 1，所以行列会算重
然后当 k 等于 0 的时候答案应该就是阶乘

样例模拟完了，当 k != 0 的时候
有 n - k 个不为零
然后他们加起来为 n
C(n - k + 1, n - 1) * A()
啊不对 这不是斯特林数？
```

润润润