结果填空：18岁生日

题解

计算出 $2015-2032$ 一共有多少个闰年。

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
bool ok(int y) {
    return y % 400 == 0 || y % 100 != 0 && y % 4 == 0;
}
int main() {
    int n;
    cin >> n;
    int sum = 18 * 365;
    for (int i = n + 1; i <= n + 18; i++) {
        if (ok(i)) {
            sum++;
        }
    }
    cout << sum << endl;
    return 0;
}
```

结果填空：数字拆分

题解

暴力搜索，或者完全背包。

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
int f[100];
int main() {
    int n;
    cin >> n;
    f[0] = 1;
    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j++) {
            f[j] += f[j - i];
        }
    }
    cout << f[n] << endl;
    return 0;
}
```

结果填空：一笔画

题解

暴力搜索。

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 3;
const int M = 5;
int vis[10][10];
int ans = 0;
int dx[] = {-1, 1, 0, 0};
int dy[] = {0, 0, -1, 1};
void dfs(int x, int y, int cnt) {
    if (x == 2 && y == 0 && cnt == 14) {
        ans++;
        return;
    }
    vis[x][y] = 1;
    for (int i = 0; i < 4; i++) {
        int tx = x + dx[i];
        int ty = y + dy[i];
        if (tx >= 0 && tx < N && ty >= 0 && ty < M && vis[tx][ty] == 0) {
            dfs(tx, ty, cnt + 1);
        }
    }
    vis[x][y] = 0;
}
int main() {
    dfs(0, 0, 0);
    cout << ans << endl;
    return 0;
}
```

结果填空：排列

题解

状压 dp。

$f[i,j]$ 表示前 i 行，第 i 行状态为 j 的所有合法方案。

首先预处理出所有合法的行状态，然后枚举每行合法的状态，和上一行的状态进行状态转移，如果可以：

$f[i,j]+=f[i-1,k]$。

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;
long long f[20][1 << 20];
int status[1 << 20], eid = 0;
int main() {
    int n;
    cin >> n;
    long long sum = 0;
    f[0][0] = 1;
    for (int i = 0; i < (1 << n); i++) {
        if (!(i & (i >> 1))) {
            status[eid++] = i;
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < eid; j++) {
            for (int k = 0; k < eid; k++) {
                if (!(status[j] & status[k])) {
                    f[i][j] += f[i - 1][k];
                    f[i][j] %= mod;
                }
            }
            if (i == n) {
                sum += f[i][j];
                sum %= mod;
            }
        }
    }
    cout << sum << endl;
    return 0;
}
```

代码填空：排序

题解

桶排序，也叫计数排序。

```cpp
while (a[i]-- != 0)
```

程序设计：好友

50%

暴力枚举。

100%

排序后，二分查找，比 p_i 大 d 的数字有多少个，就是和第 i 户渔民相互认识的个数。

这里只统计了一边，所以最后的和就是答案。

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 9;
typedef long long ll;
int a[N];
int main() {
    int n, d;
    scanf("%d%d", &n, &d);
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    sort(a, a + n);
    ll sum = 0;
    for (int i = 0; i < n; i++) {
        sum += upper_bound(a, a + n, a[i] + d) - a - i - 1;
    }
    printf("%lld\n", sum);
    return 0;
}
```

程序设计：幻方矩阵

30%

暴力模拟就可以了。

60%

使用 vector 模拟就可以了。

100%

开两个数组。

一个模拟行的交换，一个模拟列的交换就可以了，每次交换只交换里面的数字就可以了。

标程

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 9;
vector<int> f[N];
int x[N], y[N];
int main() {
    int n, m;
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++) {
        f[i].push_back(0);
        for (int j = 1; j <= m; j++) {
            int a;
            scanf("%d", &a);
            f[i].push_back(a);
            y[j] = j;
        }
        x[i] = i;
    }
    int op;
    scanf("%d", &op);
    for (int i = 0; i < op; i++) {
        int a, b, c;
        scanf("%d%d%d", &a, &b, &c);
        if (a) {
            swap(y[b], y[c]);
        } else {
            swap(x[b], x[c]);
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            printf("%d%c", f[x[i]][y[j]], j != m ? ' ' : '\n');
        }
    }
    return 0;
}
```

程序设计：瞬间移动

题解

递推。

$f[i,j]$ 为前 i 行 j 列 能获取到的最大分数值。

$$f[i,j] = max(max(f[i-1,j], f[i,j-1]) + grade[i,j], f[i-1,j], f[i,j-1])$$

在处理的时候一定要注意边界条件问题。

当然这道题目也可以使用记忆化搜索的方法解决。

标程

```cpp
#include <iostream>
using namespace std;
const int N = 2e3 + 9;
int f[N][N];

int main() {
    int n, m, x;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            cin >> x;
            if (i == 1 && j == 1 || i == n && j == m) {
                f[i][j] = max(f[i - 1][j], f[i][j - 1]) + x;
            } else if (i == 1) {
                f[i][j] = f[i][j - 1] + max(x, 0);
            } else if (j == 1) {
                f[i][j] = f[i - 1][j] + max(x, 0);
            } else {
                f[i][j] = max(f[i - 1][j], f[i][j - 1]);
                if (x > 0) {
                    f[i][j] += x;
                }
            }
        }
    }
    cout << f[n][m] << endl;
    return 0;
}
```

程序设计：快速过河

题解

本题是一道经典的原题，在蓝桥杯每日一练中。因为蓝桥杯今年可能会在每日一练中出原题，所以在这里放了这道题目。

30%

暴力枚举。

100%

贪心（动态规划可以很简单的计算出最少用的时间）

动态规划：

我们先将所有人按花费时间递增进行排序，假设前 i 个人过河花费的最少时间为 $opt[i]$，那么考虑前 $i-1$ 个人已经过河的情况，即河这边还有 1 个人，河那边有 $i-1$ 个人，并且这时候手电筒肯定在对岸，所以 $opt[i]=opt[i-1]+a[1]+a[i]$ (让花费时间最少的人把手电筒送过来，然后和第 i 个人一起过河) 。

如果河这边还有两个人，一个是第 i 号，另外一个无关，河那边有 $i-2$ 个人，并且手电筒肯定在对岸，所以 $opt[i]=opt[i-2]+a[1]+a[i]+2\times a[2]$（让花费时间最少的人把电筒送过来，然后第 i 个人和另外一个人一起过河，由于花费时间最少的人在这边，所以下一次送手电筒过来的一定是花费次少的，送过来后花费最少的和花费次少的一起过河，解决问题），所以

$$opt[i]=min(opt[i-1]+a[1]+a[i],opt[i-2]+a[1]+a[i]+2\times a[2])$$

贪心：

一个人：时间为 a_0。

两个人：时间为 a_1。

三个人：时间为 $a_0+a_1+a_2$。

多个人：比较下面两种方案中的最优值。

$a[0]*2+a[n]+a[n-1]\quad ?\quad a[0]+2*a[1]+a[n]$

具体：

```
cout << a[0] << ' ' << a[1] << endl;
cout << a[0] << endl;
cout << a[n - 1] << ' ' << a[n] << endl;
cout << a[1] << endl;
```

```
cout << a[0] << ' ' << a[n] << endl;
cout << a[0] << endl;
```

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e3 + 9;
int a[N], f[N];
int main() {
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    sort(a, a + n);
    f[0] = a[0];
    f[1] = a[1];
    for (int i = 2; i < n; ++i) {
        f[i] = min(f[i - 1] + a[0] + a[i], f[i - 2] + a[0] + 2 * a[1] + a[i]);
    }
    cout << f[n - 1] << endl;
    n--;
    int s = 0;
    while (1) {
        if (n == 0) {
            s += a[0];
            cout << 1 << ' ' << a[0] << endl;
            break;
        } else if (n == 1) {
            s += a[1];
            cout << 2 << ' ' << a[0] << ' ' << a[1] << endl;
            break;
        } else if (n == 2) {
            s += a[0] + a[1] + a[2];
            cout << 2 << ' ' << a[0] << ' ' << a[1] << endl;
            cout << 1 << ' ' << a[0] << endl;
            cout << 2 << ' ' << a[0] << ' ' << a[2] << endl;
            break;
        } else {
            if (a[0] * 2 + a[n] + a[n - 1] > a[0] + 2 * a[1] + a[n]) {
                s += a[0] + 2 * a[1] + a[n];
                cout << 2 << ' ' << a[0] << ' ' << a[1] << endl;
                cout << 1 << ' ' << a[0] << endl;
                cout << 2 << ' ' << a[n - 1] << ' ' << a[n] << endl;
                cout << 1 << ' ' << a[1] << endl;
                n -= 2;
            } else {
                s += a[0] + a[n];
                cout << 2 << ' ' << a[0] << ' ' << a[n] << endl;
                cout << 1 << ' ' << a[0] << endl;
                n -= 1;
            }
        }
    }
```

```
    return 0;
}
```

程序设计：第 k 大数字

题解

30 %

暴力枚举，然后排序输出就可以了。

70 %

1：二分答案，然后暴力判断。

2：找规律，然后暴力枚举

100 %

1：二分答案，然后二分判断

2：找规律，然后二分

标程

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;
const int INF = 0x3f3f3f3f;
const LL mod = 1e9 + 7;
const int N = 100005;

int n, m;
LL k;
bool check(int x) {
    int t1 = max(1, x - m);
    int t2 = min(n, x - 1);
    LL sum = (LL)m*(t1 - 1) + (LL)(x - t1 + x - t2) * (t2 - t1 + 1) / 2;
    return sum >= k;
}
int main() {
    cin >> n >> m >> k;
    int l = 2, r = n + m;
    k = (LL)n * m - k + 1;
    while (l < r) {
        int mid = l + r >> 1;
        if (check(mid)) {
            r = mid;
        } else {
            l = mid + 1;
        }
    }
    printf("%d\n", l);
    return 0;
}
```