数学与简单DP

一、AcWing 1205. 买不到的数目

【题目描述】

小明开了一家糖果店。

他别出心裁：把水果糖包成4颗一包和7颗一包的两种。

糖果不能拆包卖。

小朋友来买糖的时候，他就用这两种包装来组合。

当然有些糖果数目是无法组合出来的，比如要买10颗糖。

你可以用计算机测试一下，在这种包装情况下，最大不能买到的数量是17。

大于17的任何数字都可以用4和7组合出来。

本题的要求就是在已知两个包装的数量时，求最大不能组合出的数字。

【输入格式】

两个正整数n,m，表示每种包装中糖的颗数。

【输出格式】

一个正整数，表示最大不能买到的糖数。

【数据范围】

$2 \leq n,m \leq 1000$

保证数据一定有解。

【输入样例】

```
4 7
```

【输出样例】

```
17
```

【分析】

引理：给定a,b，若$gcd(a,b)>1$，则一定不存在最大不能组合出的数字。

结论：如果a,b均是正整数且互质，那么由$ax+by(x,y\geq 0)$不能凑出的最大数是$(a-1)*(b-1)-1$。

打表找规律代码如下：

```
#include <iostream>
using namespace std;

//给定一个m,判断是否能用a和b凑出来
bool dfs(int m, int a, int b)
{
    if (m == 0) return true;
    if (m >= a && dfs(m - a, a, b)) return true;
    if (m >= b && dfs(m - b, a, b)) return true;
    return false;
}

int main()
{
    int a, b;
    cin >> a >> b;
    int res = 0;
    for (int i = 1; i <= 1000; i++)
        if (!dfs(i, a, b)) res = i;
    cout << res << endl;
    return 0;
}
```

【代码】

```
#include <iostream>
using namespace std;

int main()
{
    int a, b;
    cin >> a >> b;
    cout << (a - 1) * (b - 1) - 1 << endl;
    return 0;
}
```

二、AcWing 1211. 蚂蚁感冒

【题目描述】

长100厘米的细长直杆子上有n只蚂蚁。

它们的头有的朝左，有的朝右。

每只蚂蚁都只能沿着杆子向前爬，速度是1厘米/秒。

当两只蚂蚁碰面时，它们会同时掉头往相反的方向爬行。

这些蚂蚁中，有1只蚂蚁感冒了。

并且在和其它蚂蚁碰面时，会把感冒传染给碰到的蚂蚁。

请你计算，当所有蚂蚁都爬离杆子时，有多少只蚂蚁患上了感冒。

【输入格式】

第一行输入一个整数n，表示蚂蚁的总数。

接着的一行是n个用空格分开的整数X_i，X_i的绝对值表示蚂蚁离开杆子左边端点的距离。

正值表示头朝右，负值表示头朝左，数据中不会出现0值，也不会出现两只蚂蚁占用同一位置。

其中，第一个数据代表的蚂蚁感冒了。

【输出格式】

输出1个整数，表示最后感冒蚂蚁的数目。

【数据范围】

$1 < n < 50$

$0 < |X_i| < 100$

【输入样例1】

```
3
5 -2 8
```

【输出样例1】

```
1
```

【输入样例2】

```
5
-10 8 -20 12 25
```

【输出样例2】

```
3
```

【分析】

设感冒的蚂蚁为0号蚂蚁，$idx[i]$表示X_i，对本题进行分析：

- 首先，两只蚂蚁碰撞后各自向反方向走可以等价看成两只蚂蚁互相穿过对方继续向前走，因此最后一定所有蚂蚁都会走下杆子；
- 无论0号蚂蚁向左走还是向右走，其**左边所有向左走与右边所有向右走**的蚂蚁一定不会被感染；
- 统计出0号蚂蚁左边向右走的蚂蚁数量l以及右边向左走的蚂蚁数量r；
- 如果0号蚂蚁向左走，那么其左边向右走的蚂蚁一定都会被感染，即被感染的数量$+l$，如果其左边向右走的蚂蚁至少有一只被感染，那么其右边向左走的蚂蚁也一定全会被感染，即如果$l \neq 0$，那么被感染的数量$+r$。因此如果$l = 0$，那么答案为1，即只有0号蚂蚁被感染，否则答案为$l + r + 1$，即0号蚂蚁及其左边向右走与右边向左走的蚂蚁都会被感染。

【代码】

```cpp
#include <iostream>
#include <cmath>
using namespace std;

const int N = 60;
int idx[N];
int n, l, r;//l表示1号蚂蚁左边向右走的蚂蚁数量,r表示右边向左走的蚂蚁数量

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> idx[i];
    for (int i = 1; i < n; i++)
        if (abs(idx[i]) < abs(idx[0]) && idx[i] > 0) l++;
        else if (abs(idx[i]) > abs(idx[0]) && idx[i] < 0) r++;
    if ((idx[0] < 0 && !l) || (idx[0] > 0 && !r)) cout << 1 << endl;
    else cout << l + r + 1 << endl;
```

```
18      return 0;
19  }
```

三、AcWing 1216. 饮料换购

【题目描述】

乐羊羊饮料厂正在举办一次促销优惠活动。乐羊羊C型饮料，凭3个瓶盖可以再换一瓶C型饮料，并且可以一直循环下去（但不允许暂借或赊账）。

请你计算一下，如果小明不浪费瓶盖，尽量地参加活动，那么，对于他初始买入的n瓶饮料，最后他一共能喝到多少瓶饮料。

【输入格式】

输入一个整数n，表示初始买入的饮料数量。

【输出格式】

输出一个整数，表示一共能够喝到的饮料数量。

【数据范围】

$0 < n < 10000$

【输入样例】

```
100
```

【输出样例】

```
149
```

【分析】

初始化$res = n$，第一次将n瓶全部喝完后的n个瓶盖能换$n\%3$瓶新的，因此答案$res += n\%3$，将换的这些全部喝完后一共有$n/3 + n\%3$个瓶盖，然后继续换新的，不断循环直到$n < 3$为止。

【代码】

```
#include <iostream>
using namespace std;

```

```cpp
int main()
{
    int n;
    cin >> n;
    int res = n;
    while (n >= 3)
    {
        res += n / 3;
        n = n / 3 + n % 3;
    }
    cout << res << endl;
    return 0;
}
```

四、AcWing 2. 01背包问题

【题目描述】

有N件物品和一个容量是V的背包。每件物品只能使用一次。

第i件物品的体积是v_i，价值是w_i。

求解将哪些物品装入背包，可使这些物品的总体积不超过背包容量，且总价值最大。

输出最大价值。

【输入格式】

第一行两个整数N, V，用空格隔开，分别表示物品数量和背包容积。

接下来有N行，每行两个整数v_i, w_i，用空格隔开，分别表示第i件物品的体积和价值。

【输出格式】

输出一个整数，表示最大价值。

【数据范围】

$0 < N, V \leq 1000$

$0 < v_i, w_i \leq 1000$

【输入样例】

```
4 5
1 2
2 4
3 4
4 5
```

【输出样例】

```
8
```

【分析】

模板题，直接上代码。

【代码】

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

const int N = 1010;
int v[N], w[N], f[N];
int n, m;

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

五、AcWing 1015. 摘花生

【题目描述】

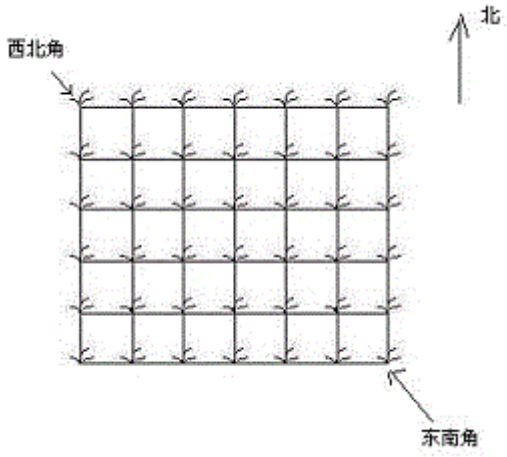
Hello Kitty想摘点花生送给她喜欢的米老鼠。

她来到一片有网格状道路的矩形花生地(如下图)，从西北角进去，东南角出来。

地里每个道路的交叉点上都有种着一株花生苗，上面有若干颗花生，经过一株花生苗就能摘走该它上面所有的花生。

Hello Kitty只能向东或向南走，不能向西或向北走。

问Hello Kitty最多能够摘到多少颗花生。



【输入格式】

第一行是一个整数T，代表一共有多少组数据。

接下来是T组数据。

每组数据的第一行是两个整数，分别代表花生苗的行数R和列数C。

每组数据的接下来R行数据，从北向南依次描述每行花生苗的情况。每行数据有C个整数，按从西向东的顺序描述了该行每株花生苗上的花生数目M。

【输出格式】

对每组输入数据，输出一行，内容为Hello Kitty能摘到得最多的花生颗数。

【数据范围】

$1 \leq T \leq 100$

$1 \leq R, C \leq 100$

$0 \leq M \leq 1000$

【输入样例】

```
2
2 2
1 1
3 4
2 3
2 3 4
1 6 5
```

【输出样例】

```
8
16
```

【分析】

状态表示：

- 集合：$f[i][j]$为从$(1,1)$到达(i,j)的所有方案的集合。
- 属性：最大值

状态转移：

- 从(i,j)的上方$(i-1,j)$转移过来，即$f[i-1][j]$
- 从(i,j)的左方$(i,j-1)$转移过来，即$f[i][j-1]$

所以最终的状态转移方程为：$f[i][j]=max(f[i-1][j],f[i][j-1])+g[i][j]$，$g[i][j]$表示$(i,j)$上的花生数量。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 110;
int g[N][N], f[N][N];
int n, m;

int main()
{
    int T;
    cin >> T;
```

```
    while (T--)
    {
        cin >> n >> m;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= m; j++)
            {
                cin >> g[i][j];
                f[i][j] = max(f[i - 1][j], f[i][j - 1]) + g[i][j];
            }
        cout << f[n][m] << endl;
    }
    return 0;
}
```

六、AcWing 895. 最长上升子序列

【题目描述】

给定一个长度为N的数列，求数值严格单调递增的子序列的长度最长是多少。

【输入格式】

第一行包含整数N。

第二行包含N个整数，表示完整序列。

【输出格式】

输出一个整数，表示最大长度。

【数据范围】

$1 \leq N \leq 1000$

$-10^9 \leq$ 数列中的数 $\leq 10^9$

【输入样例】

```
7
3 1 2 1 8 5 6
```

【输出样例】

```
4
```

【分析】

模板题，直接上代码。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 1010;
int h[N], f[N];
int n, cnt;

int main()
{
    cin >> n;
    for (int i = 0; i < n; i++) cin >> h[i];
    f[cnt++] = h[0];
    for (int i = 1; i < n; i++)
        if (h[i] > f[cnt - 1]) f[cnt++] = h[i];
        else
        {
            int l = 0, r = cnt - 1;
            while (l < r)
            {
                int mid = l + r >> 1;
                if (f[mid] >= h[i]) r = mid;
                else l = mid + 1;
            }
            f[r] = h[i];
        }
    cout << cnt << endl;
    return 0;
}
```

七、AcWing 1212. 地宫取宝

【题目描述】

X国王有一个地宫宝库，是$n \times m$个格子的矩阵，每个格子放一件宝贝，每个宝贝贴着价值标签。

地宫的入口在左上角，出口在右下角。

小明被带到地宫的入口，国王要求他只能向右或向下行走。

走过某个格子时，如果那个格子中的宝贝价值比小明手中任意宝贝价值都大，小明就可以拿起它（当然，也可以不拿）。

当小明走到出口时，如果他手中的宝贝恰好是k件，则这些宝贝就可以送给小明。

请你帮小明算一算，在给定的局面下，他有多少种不同的行动方案能获得这k件宝贝。

【输入格式】

第一行3个整数，n, m, k，含义见题目描述。

接下来n行，每行有m个整数C_i用来描述宝库矩阵每个格子的宝贝价值。

【输出格式】

输出一个整数，表示正好取k个宝贝的行动方案数。

该数字可能很大，输出它对1000000007取模的结果。

【数据范围】

$1 \leq n, m \leq 50$

$1 \leq k \leq 12$

$0 \leq C_i \leq 12$

【输入样例1】

```
2 2 2
1 2
2 1
```

【输出样例1】

```
2
```

【输入样例2】

```
2 3 2
1 2 3
2 1 5
```

【输出样例2】

```
14
```

【分析】

状态表示：$f[i][j][u][v]$表示在点(i,j)时共拿了u个物品，这些物品中价值最大的是v的方案数。

状态计算：首先可以将集合分为从(i,j)上方与左方转移过来两大类，即$f[i-1][j],f[i][j-1]$。对于(i,j)位置上的物品，我们有选和不选两种选择，如果不选，那么前一个状态时就已经拿了u个物品，其中最大价值为v，即对应以下两种转移方程：

1. $f[i][j][u][v]=(f[i][j][u][v]+f[i-1][j][u][v])\%MOD$
2. $f[i][j][u][v]=(f[i][j][u][v]+f[i][j-1][u][v])\%MOD$

如果选(i,j)上的物品，首先需要满足当前状态的$u>0$，即至少选了一个物品，其次是要满足当前状态的$v==w[i][j]$，$w[i][j]$为当前物品的价值，因为选了物品(i,j)后最大价值为v，选的物品的价值一定是严格单调递增的，因此当前物品的价值就是最大价值，即$v==w[i][j]$。

符合选择物品(i,j)的条件后，我们可以枚举上一个状态的v值（用c来表示，$0\le c<v$），对于每个c值，有以下两种转移方程：

1. $f[i][j][u][v]=(f[i][j][u][v]+f[i-1][j][u-1][c])\%MOD$
2. $f[i][j][u][v]=(f[i][j][u][v]+f[i][j-1][u-1][c])\%MOD$

初始化：

1. 不取$(1,1)$的物品的方案数为1，则u为0，v为小于物品最低价值的任意值，即$f[1][1][0][-1]=1$，由于数组下标不能为负，因此我们可以将物品价值都加一，变为$1\sim13$，那么就可以初始化$f[1][1][0][0]=1$
2. 取$(1,1)$的物品的方案数为1，则u为1，v为该物品的价值，即$f[1][1][1][w[1][1]]=1$

最后我们枚举$v(1\le v\le13)$，累加所有情况的方案数即可，即$res=(res+f[n][m][k][v])\%MOD$

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;


const int N = 55, MOD = 1e9 + 7;
```

```cpp
int f[N][N][13][14];//f[i][j][u][v]表示在点(i,j)时共拿了u个物品,这些物品中价值最大的是v
int w[N][N];
int n, m, k;

int main()
{
    cin >> n >> m >> k;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
        {
            cin >> w[i][j];
            w[i][j]++;//将物品价值转换为1~13,0作为初始化边界点
        }
    f[1][1][0][0] = 1;//不取(1,1)的物品方案数为1
    f[1][1][1][w[1][1]] = 1;//取(1,1)的物品方案数为1
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            for (int u = 0; u <= k; u++)
                for (int v = 0; v < 14; v++)
                {
                    int& st = f[i][j][u][v];
                    //不取(i,j)的物品时,分别从上边和左边转移过来的方案
                    st = (st + f[i - 1][j][u][v]) % MOD;
                    st = (st + f[i][j - 1][u][v]) % MOD;
                    //取(i,j)的物品时,只有数量大于0且(i,j)物品的价值等于v时才能取
                    if (u > 0 && v == w[i][j])
                        for (int c = 0; c < v; c++)//前一个状态取的物品价值一定低于当前物品
                        {
                            st = (st + f[i - 1][j][u - 1][c]) % MOD;
                            st = (st + f[i][j - 1][u - 1][c]) % MOD;
                        }
                }
    int res = 0;
    for (int i = 1; i < 14; i++) res = (res + f[n][m][k][i]) % MOD;
    cout << res << endl;
    return 0;
}
```

八、AcWing 1214. 波动数列

【题目描述】

观察这个数列：`1 3 0 2 -1 1 -2 ...`

这个数列中后一项总是比前一项增加2或者减少3，且每一项都为整数。

栋栋对这种数列很好奇，他想知道长度为n和为s而且后一项总是比前一项增加a或者减少b的整数数列可能有多少种呢？

【输入格式】

共一行，包含四个整数n, s, a, b，含义如前面所述。

【输出格式】

共一行，包含一个整数，表示满足条件的方案数。

由于这个数很大，请输出方案数除以100000007的余数。

【数据范围】

$1 \leq n \leq 1000$

$-10^9 \leq s \leq 10^9$

$1 \leq a, b \leq 10^6$

【输入样例】

```
4 10 2 3
```

【输出样例】

```
2
```

【样例解释】

两个满足条件的数列分别是`2 4 1 3`和`7 4 1 -2`。

【分析】

设第一个数为x，则第二个数为$x + d_1$，第三个数为$x + d_1 + d_2 \ldots$。这里的d_i表示a或者$-b$，所以这个数列为：

$$x, x + d_1, x + d_1 + d_2, x + d_1 + d_2 + d_3, \ldots, x + d_1 + d_2 + \cdots + d_{n-1}$$

又因为数列之和为s，所以转化成：

$$n * x + (n-1) * d_1 + (n-2) * d_2 + (n-3) * d_3 + \cdots + d_{n-1} = s$$

$\Rightarrow \frac{s-[(n-1)*d_1+(n-2)*d_2+(n-3)*d_3+\cdots+d_{n-1}]}{n}=x$

因为x是任意整数，所以又转化成：

s与$(n-1)*d_1+(n-2)*d_2+(n-3)*d_3+\cdots+d_{n-1}$模$n$的余数相同，到这里就转化成了组合问题。

状态表示：$f[i][j]$表示要选i个a或者$-b$且模n的余数为j的所有集合的数量。

状态计算：第i个可以选a或者$-b$。

- 第i个数选a，即$d_i=a$：上一个状态为$(n-1)*d_1+(n-2)*d_2+\cdots+(n-i+1)*d_{i-1}$，则上一个状态的$i'=i-1$，$j'=(j-(n-i)*a)\%n$，为防止$j'$为负，可以改为$j'=((j-(n-i)*a)\%n+n)\%n$
- 第i个数选$-b$，即$d_i=-b$：上一个状态为$(n-1)*d_1+(n-2)*d_2+\cdots+(n-i+1)*d_{i-1}$，则上一个状态的$i'=i-1$，$j'=(j+(n-i)*b)\%n$。

综上，状态转移方程为：$f[i][j]=(f[i-1][((j-(n-i)*a)\%n+n)\%n]+f[i-1][(j+(n-i)*b)\%n])$。

【代码】

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 1010, MOD = 1e8 + 7;
int f[N][N];//f[i][j]表示要选i个a或者-b且模n的余数为j的所有集合的数量
int n, s, a, b;

int main()
{
    cin >> n >> s >> a >> b;
    f[0][0] = 1;//选0个a或-b余数只能为0
    for (int i = 1; i < n; i++)
        for (int j = 0; j < n; j++)
            f[i][j] = (f[i - 1][((j - (n - i) * a) % n + n) % n] + f[i - 1][(j + (n - i) * b) % n]) % MOD;
    cout << f[n - 1][(s % n + n) % n] << endl;//注意s可能为负数
    return 0;
}
```