

1、2014 年 P1328

用数组将输赢保存起来，第一个下标表示小 A 出的，第二个下标表示小 B 出的
根据出拳次数，就可以判断每一次输赢。

输赢数组：

甲 \ 乙 甲对乙的结果	剪刀	石头	布	蜥蜴人	斯波克
剪刀	平	输	赢	赢	输
石头		平	输	赢	输
布			平	输	赢
蜥蜴人				平	赢
斯波克					平

```
int s[5][5]={0,-1,1,1,-1},
            {1,0,-1,1,-1},
            {-1,1,0,-1,1},
            {-1,-1,1,0,1},
            {1,1,-1,-1,0}};
```

代码：

```
int n,la,lb;
int sa,sb;
int s[5][5]={0,-1,1,1,-1},
            {1,0,-1,1,-1},
            {-1,1,0,-1,1},
            {-1,-1,1,0,1},
            {1,1,-1,-1,0}}; //输赢数组

int a[210],b[210];
int main()
{
    cin >> n >> la >> lb;
    for(int i=0;i<la;i++){
        cin >> a[i];
    }
    for(int i=0;i<lb;i++){
        cin >> b[i];
    }
    for(int i=0;i<n;i++){//循环判断
        if(s[a[i%la]][b[i%lb]]==1){
            sa++;
        }
        else if(s[a[i%la]][b[i%lb]]==-1){
            sb++;
        }
    }
    cout<< sa <<" "<< sb;
    return 0;
}
```

2、2014 年 P2038

思想：模拟、枚举

第一步：存坐标数组

```
for(int i=1; i<=n; i++){
    cin >> x >> y >> k;
    s[x][y]=k;
}
```

第二步：遍历每一个点

```
for(int i=0; i<=128; i++){
    for(int j=0; j<=128; j++){
        solve(i, j);
    }
}
```

第三步：计算最多公共场所的安装地点方案数，以及能覆盖的数量。

```
void solve(int x, int y)//判断该点雷达辐射范围
{
    int sum=0;
    for(int i=x-d; i<=x+d; i++){
        for(int j=y-d; j<=y+d; j++){
            if(i>=0&&j>=0&&i<=128&&j<=128){
                sum+=s[i][j];
            }
        }
    }
    if(sum>ans){//较大，就更新
        num=1;
        ans=sum;
    }
    else if(sum==ans){//相等，就累加方案
        num++;
    }
}
```

完整代码：

```

#include<bits/stdc++.h>
using namespace std;
int num,ans,d,n,s[130][130];
int x,y,k;

void solve(int x, int y)//判断该点雷达辐射范围
{
    int sum=0;
    for(int i=x-d;i<=x+d;i++){
        for(int j=y-d;j<=y+d;j++){
            if(i>=0&&j>=0&&i<=128&&j<=128){
                sum+=s[i][j];
            }
        }
    }
    if(sum>ans){//较大, 就更新
        num=1;
        ans=sum;
    }
    else if(sum==ans){//相等, 就累加方案
        num++;
    }
}

```

```

int main()
{
    cin >> d >> n;
    for(int i=1; i<=n; i++){
        cin >> x >> y >> k;
        s[x][y]=k;
    }
    for(int i=0; i<=128; i++){
        for(int j=0; j<=128; j++){
            solve(i, j);
        }
    }
    cout << num << " " << ans;
    return 0;
}

```

3、2015 年 P2615

思想：模拟、枚举

初始化：

```
cin >> n;  
a[1][n/2+1]=1;  
x=1; //k-1的横坐标  
y=n/2+1; //k-1的纵坐标
```

四种情况：

1. 若 $(K - 1)$ 在第一行但不在最后一列，则将 K 填在最后一行， $(K - 1)$ 所在列的右一列；
2. 若 $(K - 1)$ 在最后一列但不在第一行，则将 K 填在第一列， $(K - 1)$ 所在行的上一行；
3. 若 $(K - 1)$ 在第一行最后一列，则将 K 填在 $(K - 1)$ 的正下方；
4. 若 $(K - 1)$ 既不在第一行，也不在最后一列，如果 $(K - 1)$ 的右上方还未填数，则将 K 填在 $(K - 1)$ 的右上方，否则将 K 填在 $(K - 1)$ 的正下方。

```
if(x==1 && y!=n)  
{  
    nx=n; //k的横坐标  
    ny=y+1; //k的纵坐标  
}  
else if(x!=1 && y==n)  
{  
    nx=x-1;  
    ny=1;  
}  
else if(x==1 && y==n)  
{  
    nx=x+1;  
    ny=y;  
}  
else if(x!=1 && y!=n)  
{  
    if(!a[x-1][y+1])  
    {  
        nx=x-1;  
        ny=y+1;  
    }  
    else{  
        nx=x+1;  
        ny=y;  
    }  
}
```

完整代码：

```
#include<bits/stdc++.h>
using namespace std;
int n,a[40][40],x,y,maxx;
int nx,ny;
int main()
{
    cin >> n;
    a[1][n/2+1]=1;
    x=1; //k-1的横坐标
    y=n/2+1; //k-1的纵坐标
    maxx=n*n;
    for(int k=2;k<=maxx;k++) //将四种情况列举出来
    {
        if(x==1 && y!=n)
        {
            nx=n; //k的横坐标
            ny=y+1; //k的纵坐标
        }
        else if(x!=1 && y==n)
        {
            nx=x-1;
            ny=1;
        }
        else if(x==1 && y==n)
        {
            nx=x+1;
            ny=y;
        }
        else if(x!=1 && y!=n)
        {
            if(!a[x-1][y+1])
            {
                nx=x-1;
                ny=y+1;
            }
            else{
                nx=x+1;
                ny=y;
            }
        }
        a[nx][ny]=k;
        //计算下一个位置，更新坐标
        x=nx;
        y=ny;
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

4、2015 年 P2678

思想：贪心、二分

题解：二分枚举距离，以距离为参数，减少岩石的数量，看是否满足条件

条件：最后一跳要比 mid 大，减少的岩石数量要比可移走的岩石数量 m 小

```
bool Check(int mid){
    int start=0,x=0;//用start表示每次落脚点的坐标，每落一次地更新一次start
    for(int i=1;i<=n;i++){
        if(a[i]-start<mid){
            x++;//x表示去掉的石头数，如果mid大于要跳的距离，就跳过当前这个石头，此时x++
        }
        else{
            start=a[i];//此时落地
        }
    }
    if(l-start<mid){
        return false;//判断最后一跳跳的距离要是小于mid的话那是不可以的
    }
    if(x>m){
        return false;//要是x>m就说明最小距离mid太大
    }
    return true;
}
```

二分模板：

mid 表示最短距离，二分查找最短距离的最大值

如果 mid 满足情况，保存结果，继续求 mid 是否还能更大

```
int Erfen(){
    int left=0,right=l,mid,ans;
    while(right>=left)
    {
        mid=(left+right)/2;//mid表示最小的距离
        if(Check(mid))
        {
            left=mid+1;
            ans=mid;
        }
        else{
            right=mid-1;
        }
    }
    return ans;
}
```


完整代码：

```
#include<bits/stdc++.h>
using namespace std;
int a[50010],l,n,m;

bool Check(int mid){
    int start=0,x=0;//用start表示每次落脚点的坐标，每落一次地更新一次start
    for(int i=1;i<=n;i++){
        if(a[i]-start<mid){
            x++;//x表示去掉的石头数，如果mid大于要跳的距离，就跳过当前这个石头，此时x++
        }
        else{
            start=a[i];//此时落地
        }
    }
    if(l-start<mid){
        return false;//判断最后一跳跳的距离要是小于mid的话那是不可以的
    }
    if(x>m){
        return false;//要是x>m就说明最小距离mid太大
    }
    return true;
}

int Erfen(){
    int left=0,right=l,mid,ans;
    while(right>=left){
        mid=(left+right)/2;//mid表示最小的距离
        if(Check(mid))
        {
            left=mid+1;
            ans=mid;
        }
        else{
            right=mid-1;
        }
    }
    return ans;
}

int main(){
    cin >> l >> n >> m;
    for(int i=1;i<=n;i++){
        cin >> a[i];
    }
    sort(a,a+n+1);
    cout << Erfen();
    return 0;
}
```

5、2016 年 P1563

思想：模拟

题解：

这个谜题中玩具小人的朝向非常关键，因为朝内和朝外的玩具小人的左右方向是相反的：面朝圈内的玩具小人，它的左边是顺时针方向，右边是逆时针方向；而面向圈外的玩具小人，它的左边是逆时针方向，右边是顺时针方向。

存储小人的时候是逆时针，所以逆时针是加，顺时针是减。

代码：

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  struct wanju{
4      int head;
5      string str;
6  }s[100000];
7  int n, m;
8  int now, x, y;
9
10 int main(){
11     cin >> n >> m;
12     for (int i = 0; i < n; i++) {
13         cin >> s[i].head >> s[i].str;
14     }
15     for (int i = 1; i <= m; i++){
16         cin >> x >> y;
17         if (s[now].head == 0 && x == 0) { // 面朝圈内 左数
18             now = (now + n - y) % n;
19             continue;
20         }
21         if (s[now].head == 0 && x == 1) { // 面朝圈内 右数
22             now = (now + y) % n;
23             continue;
24         }
25         if (s[now].head == 1 && x == 0)
26         {
27             now = (now + y) % n;
28             continue;
29         }
30         if (s[now].head == 1 && x == 1) {
31             now = (now + n - y) % n;
32             continue;
33         }
34     }
35     cout << s[now].str;
36     return 0;
37 }
```


6、2016 年 P2822

思想：杨辉三角、前缀和

题解：

组合数求解，可以想到用杨辉三角。

杨辉三角的时候就求余，以防溢出。

```
for(int i=0;i<2001;i++)
{
    c[i][i]=1;
    c[i][0]=1;
}
for(int i=0;i<2001;i++)
{
    for(int j=0;j<=i;j++)
    {
        c[i+1][j+1]=(c[i][j]+c[i][j+1])%k; //先求余，防爆
    }
}
```

二维前缀和公式

$s[i][j]=s[i-1][j]-s[i-1][j-1]+s[i][j-1];$

	1	2	3	4	5
1	1	2	3	4	5
2	6	7	8	9	10
3	11	12	13	14	15

例如 $s[2][4]=s[1][4]-s[1][3]+s[2][3]=9;$

```
for(int i=0;i<2001;i++)
{
    for(int j=0;j<2001;j++)
    {
        s[i][j]=s[i-1][j]-s[i-1][j-1]+s[i][j-1]; //公式
        if(!c[i][j]&& j<=i){
            s[i][j]++;
        }
    }
}
```

完整代码：

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int c[2005][2005];
4  long long s[2005][2005];
5  int t,k,n,m;
6
7  int main(){
8      cin >> t >> k;
9      //先使用杨辉三角打表
10     for(int i=0;i<2001;i++)
11     {
12         c[i][i]=1;
13         c[i][0]=1;
14     }
15     for(int i=0;i<2001;i++)
16     {
17         for(int j=0;j<=i;j++)
18         {
19             c[i+1][j+1]=(c[i][j]+c[i][j+1])%k; //先求余，防爆
20         }
21     }
22     //再前缀和，加上之前的
23     for(int i=0;i<2001;i++)
24     {
25         for(int j=0;j<2001;j++)
26         {
27             s[i][j]=s[i-1][j]-s[i-1][j-1]+s[i][j-1]; //公式
28             if(!c[i][j]&& j<=i){
29                 s[i][j]++;
30             }
31         }
32     }
33     for(int i=1;i<=t;i++)
34     {
35         cin >> n >> m;
36         cout << s[n][m] << endl;
37     }
38     return 0;
39 }
```

初赛：提高组第 10 套模拟卷

2、原码转补码：

1、正数的补码与原码和反码相同

2、负数的补码：原码的基础上，符号位不变，即‘1’，数值位按位取反且加‘1’。（即反码加 1）

补码转原码：

1、正数的补码与原码相同

2、负数（符号位为‘1’）的原码：符号位不变，数值位减 1（得到反码），然后再按位取反。

7、最小生成树：

Boruvka、Prim、Kruskal 时间复杂度 $O(E \log V)$ 。都是贪心法。

9、若要使用 g++ 编译器，开启 -Ofast 优化，且使用 C++ 11 标准，将源文件 prog.cpp 编译为可执行程序 exec (-o)，且保留调试信息 (-g)，则需要使用的编译命令为（D）

A.g++ prog.cpp -Ofast exec -std=c++11 -debug

B.g++ prog.cpp -Ofast exec -std=c++11 -g

C.g++ prog.cpp -o exec -Ofast -std=c++11 -debug

D.g++ prog.cpp -o exec -Ofast -std=c++11 -g

命令行编译 ¶

在命令行下输入 `g++ a.cpp` 就可以编译 `a.cpp` 这个文件了（Windows 系统需提前把编译器所在目录加入到 PATH 中）。

编译过程中可以加入一些编译选项：

- `-o <文件名>`：指定编译器输出可执行文件的文件名。
- `-g`：在编译时添加调试信息（使用 gdb 调试时需要）。
- `-Wall`：显示所有编译警告信息。
- `-O1`，`-O2`，`-O3`：对编译的程序进行优化，数字越大表示采用的优化手段越多（开启优化会影响使用 gdb 调试）。
- `-DDEBUG`：在编译时定义 `DEBUG` 符号（符号可以随意更换，例如 `-DONLINE_JUDGE` 定义了 `ONLINE_JUDGE` 符号）。
- `-UDEBUG`：在编译时取消定义 `DEBUG` 符号。
- `-lm`，`-lgmp`：链接某个库（此处是 `math` 和 `gmp`，具体使用的名字需查阅库文档，但一般与库名相同）。

答案：

一、选择题

1-5 CDCAC

6-10 BBADB

11-15 ADDAB

二、阅读程序

1、√ √ × √ DC

2、√ × × √ AD

3、√ √ × √ BC

三、完善程序

1、AADBC

2、CBDDA