**2018年SIW第九次热身赛题解报告(QAQ)**

# D、 Circle 计数

题目：

### Problem Description

在一个圆上，有2\*K个不同的结点，我们以这些点为端点，连K条线段，使得每个结点都恰好用一次。在满足这些线段将圆分成最少部分的前提下，请计算有多少种连线的方法。

### Input

仅一行，一个整数K(1<=K<=30)。

### Output

两个用空格隔开的数，后者为最少将圆分成几块，前者为在此前提下连线的方案数。

### Sample Input

2

### Sample Output

2 3

分析：

显然最少将圆分成的块数是k+1，保证任意两条线段都不相交就可以构造出来。

我们给圆上的这些点按顺序编个号：P\_1,P\_2,P\_3···,P\_(2\*k)。我们选定P\_1作为某条线段的一个端点，那么显然另外一个端点的编号一定不能是奇数。因为如果选择奇数端点，那么这条线段两侧的端点数都是奇数，随便怎么连都会相交。

不妨设f(x)为圆上有x个点时的方案数，当然任意时刻x只可能为偶数。当我们端点P\_1选择端点P\_i时（i为小于等于x的偶数），线段P\_1-P\_i显然会把圆分为两块分别有i-2和x-i个点的方案数，根据乘法原理这个选择方案对答案的贡献为f(i-2)\*f(x-i)，于是得到递推公式f(x)=sum{f(i-2)\*f(x-i) | 2<=i<=x且i%2==0}。

你可以更改一下f(x)的定义为圆上有x对点时的方案数，同样的方法推出来的公式就是Catalan数的递推公式。

代码：

C++语言:

#include <cstdio>

#include <memory.h>

int n;

long long f[31];

long long g(int x)

{

if(f[x]) return f[x];

for(int i=0;i<x;i++)

f[x]+=g(i)\*g(x-i-1);

return f[x];

}

int main()

{

memset(f,0,sizeof(f));

f[0]=1;

scanf("%d",&n);

printf("%lld %d",g(n),n+1);

return 0;

}

# F、USSTSIW Band DP

题目：

### Problem Description

你刚刚继承了流行的USSTSIW Band录制的尚未发表的N(1 <= N <= 100)首歌的版权。你打算从中精选一些歌曲，发行M(1 <= M <= 100)张CD。每一张CD最多可以容纳T(1 <= T <= 100)分钟的音乐，一首歌不能分装在两张CD中。

不巧你是一位古典音乐迷，不懂如何判定这些歌的艺术价值。于是你决定根据以下标准进行选择：

1.歌曲必须按照创作的时间顺序在CD盘上出现。

2.选中的歌曲数目尽可能地多。。

### Input

第一行： 三个整数：N, T, M.

第二行： N个整数，分别表示每首歌的长度，按创作时间顺序排列。

### Output

一个整数，表示可以装进M张CD盘的乐曲的最大数目。

### Sample Input

4 5 2

4 3 4 2

### Sample Output

3

分析：

设f[i][j][k]表示前i首歌,前j个光盘,当前光盘用了k分钟时能放进CD的乐曲的最大数目。a[i]表示第i首歌的长度。

那么当前状态可能会被三种状态转移而来  
1.这首歌放进了这张光盘:f[i-1][j][k-a[i]]+1  
2.这首歌放下一张光盘:f[i-1][j-1][t]+1  
3.不放:f[i-1][j][k]  
于是有状态转移方程

我们发现f[i]只会从f[i-1]转移，所以可以滚动数组降维。

时间复杂度，空间复杂度。

代码：

C++语言:

#include <cstdio>

#include <memory.h>

int f[101][101],a[101],n,m,t;

inline int max(int a,int b){return a>b?a:b;}

int main()

{

memset(f,0,sizeof(f));

scanf("%d%d%d",&n,&t,&m);

for(int i=1;i<=n;i++) scanf("%d",&a[i]);

for(int i=1;i<=n;i++)

for(int j=m;j>=1;j--)

for(int k=t;k>=0;k--)

if(k>=a[i])

{

f[j][k]=max(f[j][k],f[j][k-a[i]]+1);

f[j][k]=max(f[j][k],f[j-1][t]+1);

}

printf("%d",f[m][t]);

return 0;

}

##### B、素数环

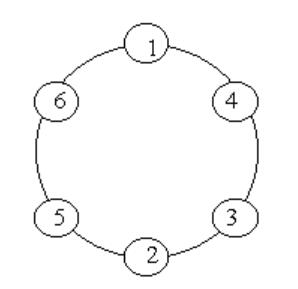
### Problem Description

一个大小为N(N<=17)的质数环是由1到N共N个自然数组成的一个数环，数环上每两个相邻的数字之和为质数。如下图是一个大小为6的质数环。为了方便描述，规定数环上的第一个数字总是1。如下图可用1 4 3 2 5 6来描述。若两个质数环，数字排列顺序相同则视为本质相同。现在要求你求出所有本质不同的数环。

。

### Input

只有一个数N，表示需求的质数环的大小。如：



。

### Output

每一行描述一个数环，如果有多组解，按照字典序从小到大输出。如：

### Sample Input

6

### Sample Output

1 4 3 2 5 6

1 6 5 2 3 4

分析：

Dfs就行了

注意奇数的时候，因为奇数n个数中奇数会比偶数多一个，所以一定会有两个奇数相邻，两个奇数相邻的话是偶数就不是素数了，所以此时会无解，不过测试数据也没有奇数- -

代码：

#include<bits/stdc++.h>

using namespace std;

int a[1001];

bool v[1001];

bool prime[1001];

bool isp(int a)

{

if(a==2)return true;

if(a%2==0)return false;

for(int i=3;i<a;i=i+2)

if(a%i==0)return false;

return true;

}

void dfs(int count)

{

if(count==n+1)

{

if(prime[1+a[n]]==true)

{

for(int i=1;i<=n;++i)

{

if(i!=n)

printf("%d ",a[i] );

else printf("%d",a[i] );

}

printf("\n");

}

return;

}

for(int i=2;i<=n;++i)

{

if(!v[i]&&prime[a[count-1]+i]==true)

{

v[i]=true;

a[count]=i;

dfs(count+1);

v[i]=false;

}

}

}

int main()

{

while(cin>>n)

{

per(i,2,200)

if(isp(i)){prime[i]=true;}

if(n%2==1){printf("no");continue;}

a[1]=1;

v[1]=true;

dfs(2);

}

}

##### C、Travelling

### Problem Description

现在，USSTSIW的一个名叫wzq的小朋友想从USST走到人民广场去买炸鸡...假设把路线图简化为一张二维X-Y坐标系,wzq行走规则如下：可以向下、或者向右行走。同时在坐标系的任一点有一个障碍点（如下图的C点），该障碍点所在的点和所有跳跃一步可达的点称为障碍点的控制点。例如下图C点可以控制9个点（图中的P1，P2 … P8 和 C）。wzq不能通过障碍点的控制点。

假设USST的坐标为S（0，0）、人民广场为D（n,m）(n,m 为不超过 20 的整数，并由键盘输入)，同样障碍点的位置坐标是需要给出的（约定: C不等于S，同时C不等于D）。现在要求你计算出waq从S点能够到达D点的路径的条数。

1<=n,m<=20

### Input

### 输入D点坐标（n,m）以及障碍点C点坐标(x,y)，无需判错！

### Output

输出一个整数（路径的条数）

### Sample Input

6 6 3 2

### Sample Output

17

分析：

简单dp

f[x][y]表示走到坐标x，y的方法总数，因为只能走下边和右边，所以f[x][y]=f[x-1][y]+f[x][y-1]

有障碍的点不能走到，也就不用去算这个点

代码：

#include<bits/stdc++.h>

#define sd(a) scanf("%d",&a)

using namespace std;

long long ans=0;

int mp[22][22];

int n,m;

int x,y;

long long f[25][25];

int main()

{

sd(n);sd(m);sd(x);sd(y);

mp[x][y]=1;

if(x>=2&&y<m)mp[x-2][y+1]=1;

if(x>=2&&y>=1)mp[x-2][y-1]=1;

if(x>=1&&y<=m-2)mp[x-1][y+2]=1;

if(x>=1&&y>=2)mp[x-1][y-2]=1;

if(x<=n-2&&y<m)mp[x+2][y+1]=1;

if(x<=n-2&&y>=1)mp[x+2][y-1]=1;

if(x<n&&y<m-1)mp[x+1][y+2]=1;

if(x<n&&y>1)mp[x+1][y-2]=1;

f[0][0]=1;

for(int i=0;i<=n;++i)

for(int j=0;j<=m;++j)

{

if(mp[i][j])continue;

if(i)f[i][j]+=f[i-1][j];

if(j)f[i][j]+=f[i][j-1];

}

printf("%lld\n",f[n][m]);

}