### 约瑟夫问题

2009-08-13 19:25

|  |
| --- |
| 设人编号为0..n-1 数到第m个人出列  那么第t轮出列的人编号为f(n,m,t)=(m%n+f(n-1,m,t-1))%n  递归调用  编号1..n的时候 如果只求最后一个 那么  #include <stdio.h> int main() {    int n, m, i, s=0;    printf ("N M = "); scanf("%d%d", &n, &m);    for (i=2; i<=n; i++) s=(s+m)%i;    printf ("The winner is %d\n", s+1); } 或则  #include <iostream> using namespace std; \_\_int64 Josephus(\_\_int64 n,long m,\_\_int64 k) //分别为：人数，出圈步长，起使报数位置, 人编号1..n {     if (m == 1)k = (k == 1 ? n : (k + n - 1) % n);     else     {         for (\_\_int64 i = 1; i <= n; i++)         {             if ((k + m) < i)             {                 \_\_int64 x = (i - k) / (m - 1) - 1;                 if (i + x < n)                 {                     i = i + x;                     k = (k + m \* x);                 }                 else                 {                     k = k + m \* (n - i) ;                     i = n;                 }             }             k = (k + m - 1) % i + 1;         }     }     return k; //返回最后一人的位置 } int m; \_\_int64 n; int main() {  while (scanf("%I64d%d",&n,&m)!=EOF) printf("%I64d\n",Josephus(n,m,1));  return 0; } |
|  |

### 无向图最小割

Stoer-Wagner 算法用来求无向图 G=(V, E)的全局最小割。

算法基于这样一个定理：对于任意s, t V ∈ ，全局最小割或者等于原图的s-t 最小割，或者等于将原图进行 Contract(s,

t)操作所得的图的全局最小割。

算法框架：

1. 设当前找到的最小割MinCut 为+∞

2. 在 G中求出任意 s-t 最小割 c，MinCut = min(MinCut, c)

3. 对 G作 Contract(s, t)操作，得到 G'=(V', E')，若|V'| > 1，则G=G'并转 2，否则MinCut 为原图的全局最

小割

Contract 操作定义：

若不存在边(p, q)，则定义边(p, q)权值w(p, q) = 0

Contract(a, b): 删掉点 a, b 及边(a, b)，加入新节点 c，对于任意 v V ∈ ，w(v, c) = w(c, v) = w(a, v) + w(b,

v)

求 G=(V, E)中任意 s-t 最小割的算法：

定义w(A, x) = ∑w(v[i], x)，v[i] A ∈

定义 Ax 为在x 前加入 A 的所有点的集合（不包括 x）

1. 令集合 A={a}，a为 V中任意点

2. 选取 V - A中的 w(A, x)最大的点 x加入集合 A

3. 若|A|=|V|，结束

令倒数第二个加入 A的点为 s，最后一个加入 A的点为 t，则s-t 最小割为 w(At, t)

#include <iostream>

using namespace std;

const int maxn=510;

int map[maxn][maxn];

int n;

void contract(int x,int y)

{

int i,j;

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

if (i!=x) map[x][i]+=map[y][i],map[i][x]+=map[i][y];

for (i=y+1;i<n;i++) for (j=0;j<n;j++)

{

map[i-1][j]=map[i][j];

map[j][i-1]=map[j][i];

}

n--;

}

int w[maxn],c[maxn];

int sx,tx;

int mincut()

{

int i,j,k,t;

memset(c,0,sizeof(c));

c[0]=1;

for (i=0;i<n;i++) w[i]=map[0][i];

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

{

t=k=-1;

for (j=0;j<n;j++) if (c[j]==0&&w[j]>k)

k=w[t=j];

c[sx=t]=1;

for (j=0;j<n;j++) w[j]+=map[t][j];

}

for (i=0;i<n;i++) if (c[i]==0) return w[tx=i];

}

int main()

{

#ifdef \_DEBUG

freopen("in.in","r",stdin);

#endif

int i,j,k,m;

while (scanf("%d%d",&n,&m)!=EOF)

{

memset(map,0,sizeof(map));

while (m--)

{

scanf("%d%d%d",&i,&j,&k);

map[i][j]+=k;

map[j][i]+=k;

}

int mint=999999999;

while (n>1)

{

k=mincut();

if (k<mint) mint=k;

contract(sx,tx);

}

printf("%d\n",mint);

}

return 0;

}

### 判素数

Source Code

Problem: 1811 User: SpellBreaker

Memory: 700K Time: 0MS

Language: C++ Result: Accepted

Source Code

#include<stdio.h>

#include<stdlib.h>

#include<cmath>

typedef unsigned long long U64;

typedef unsigned int U32;

const U32 MAX\_INDEEP = 10000;

const U32 TABLE\_SIZE = 131071;

const U32 MAX\_FORCE = 500;

U32 sqrt\_table[TABLE\_SIZE] = {0};

inline U64 gcd(U64 a, U64 b)

{

U64 t;

while (b != 0)

{

t = a % b;

a = b;

b = t;

}

return a;

}

U32 try\_ana(U64 N)

{

U32 sqrt\_n = (U32) sqrt((long double) N);

U32 P1 = sqrt\_n, Q2 = 1, Q1 = N - (U64) P1\*P1;

U32 B, P, Q, step = 0;

if (Q1 == 0)return P1;

while (sqrt\_table[Q1 % TABLE\_SIZE] != Q1)

{

B = (sqrt\_n + P1) / Q1;

P = B \* Q1 - P1;

Q = Q2 + B \* (P1 - P);

P1 = P;

Q2 = Q1;

Q1 = Q;

}

U32 sqrt\_Qi = (U32) sqrt((long double) Q1);

B = (sqrt\_n - P1) / sqrt\_Qi;

P1 = B \* sqrt\_Qi + P;

Q2 = sqrt\_Qi;

Q1 = (N - (U64) P1 \* P1) / Q2;

P = P1;

P1 = 0;

while (P != P1 && step < MAX\_INDEEP)

{

P1 = P;

B = (sqrt\_n + P1) / Q1;

P = B \* Q1 - P1;

Q = Q2 + B \* (P1 - P);

Q2 = Q1;

Q1 = Q;

step++;

}

return gcd(N, P);

}

U32 squfof(U64 N)

{

U32 k, t = 0;

for (k = 1; t == 0 || t == 1; k++)

{

t = gcd(try\_ana(k \* N), N);

}

return t;

}

inline U64 mul\_mod(U64 a, U64 b, U64 n)

{

U64 res = 0;

while (b != 0)

{

if (b & 1)

{

res += a;

if (res > n)

res -= n;

}

a = a << 1;

if (a > n)

a -= n;

b >>= 1;

}

return res;

}

U64 pow\_mod(U64 a, U64 b, U64 n)

{

U64 res = 1;

while (b != 0)

{

if (b & 1)

res = mul\_mod(res, a, n);

a = mul\_mod(a, a, n);

b >>= 1;

}

return res;

}

bool miller\_rabin(U64 a, U64 n)

{

U64 r = 0, s = n - 1;

while (!(s & 1))

{

s >>= 1;

r++;

}

U64 x = pow\_mod(a, s, n);

if (x == 1) return true;

while (r--)

{

if (x == n - 1) return true;

x = mul\_mod(x, x, n);

}

return false;

}

U64 min\_factor(U64 n)

{

if (miller\_rabin(37, n)) return n;

U64 p = squfof(n);

U64 p1 = min\_factor(p), p2 = min\_factor(n / p);

return p1 < p2 ? p1 : p2;

}

U64 work(U64 n)

{

if (n % 2 == 0)return 2;

for (int i = 3; i <= MAX\_FORCE; i += 2)

if (n % i == 0)return i;

return min\_factor(n);

}

int main()

{

U64 cas, n, p;

for (U32 i = 0; i < (1 << 16); i++)

sqrt\_table[i \* i % TABLE\_SIZE] = i \* i;

scanf("%I64d", &cas);

while (cas--)

{

scanf("%I64d", &n);

p = work(n);

if (p == n)

printf("Prime\n");

else

printf("%I64d\n", p);

}

return 0;

}

### 判素数

#include <iostream>

#include <time.h>

#include <cmath>

using namespace std;

typedef unsigned \_\_int64 ulld;

ulld chen(ulld x,ulld y,ulld n)//计算x\*y%n

{

if ((x>>32)==0&&(y>>32)==0) return x\*y%n;

ulld a1,b1,a2,b2;

a1=(x>>32);

b1=((x<<32)>>32);

a2=(y>>32);

b2=((y<<32)>>32);

ulld t1,t2,t3,t4;

t1=b1\*b2%n;t2=b1\*a2%n;t3=a1\*b2%n;t4=a1\*a2%n;

int i;

for (i=0;i<32;i++){ t2=(t2<<1)%n;t3=(t3<<1)%n;}

for (i=0;i<64;i++) t4=(t4<<1)%n;

return ((t1+t2)%n+(t3+t4)%n)%n;

}

ulld mood(ulld a,ulld u,ulld n) //计算a^u%n

{

ulld b=1;

while (u)

{

if (u & 1) b =chen(b,a,n);

if (u >>= 1) a=chen(a,a,n);

}

return b;

}

ulld witness(ulld a, ulld n)//计算a^(n-1)%n 其中如果遇到1的非平凡平方根 就直接返回合数

{

ulld i,x,y,t,u;

u=n-1;t=0;

while ((u&1)==0)

{

u>>=1;

t++;

}

x=mood(a,u,n);

for (i=0;i<t;i++)

{

y=chen(x,x,n);

if (y==1&&x!=1&&x!=n-1)

return x;

x=y;

}

return x;

}

ulld miller(ulld n)//miller判素数部分 多测几次

{

int i;

ulld j;

if (n==2) return 1;

if (n<2||n%2==0) return 0;

for (i=0;i<50;i++)

{

j=rand()%(n-2)+1;

ulld k=witness(j,n);

if (k!=1)

return k;

}

return 1;

}

ulld gcd(ulld a,ulld b) //最大公约数

{

ulld c;

if (a<b){c=a;a=b;b=c;}

while (b!=0)

{

c=a;a=b;b=c%b;

}

return a;

}

ulld pollard(ulld n,ulld x)//pollard tot用于卡时防止死掉

{

ulld i,k,y,d;

ulld tot=1<<20;

i=1;y=x;k=2;

while (1)

{

i++;

tot--;

x=(chen(x,x,n)-1+n)%n;

d=gcd((y-x+n)%n,n);

if (d!=1&&d!=n) return d;

if (i==k)

{

y=x;

k=(k<<1);

}

if (tot==0) return 1;

}

}

int main()

{

ulld n,m;

int cass;

srand(time(0));

for (cin>>cass;cass>0;cass--)

{

scanf("%I64d",&n);

if (n==2)

goto isprim;

if ((n&1)==0)

{

cout<<2<<endl;

continue;

}

if (miller(n)==1)

{

isprim: cout<<"Prime"<<endl;

continue;

}

ulld d=pollard(n,2);

n/=d;

ulld mind=d;

if (mind==1) mind=99999999999999;

while (miller(n)!=1)

{

m=rand()%(n-1)+1;

d=pollard(n,m);

if (d>1&&mind>d) mind=d;

n/=d;

}

if (mind>n) mind=n;

printf("%I64d\n",mind);

}

return 0;

}

### 四边形不等式

g[i+1][j]<g[i][j]<g[i][j+1],

对于f[i][j]=Max( f[i][k]+f[k+1][j]+ w[i][j] ) 的区间切两半

g[i-1][j]<g[i][j]<g[i][j+1],

对于f[i][j]=Max( f[i][k]+w[k+1][j] ) 的拿一段出来

