目录

[SG打表: 1](#_Toc463820488)

[树链剖分 3](#_Toc463820489)

[树形dp 7](#_Toc463820490)

[状态压缩dp 9](#_Toc463820491)

[数论公式 （补充） 11](#_Toc463820492)

[素数个数 16](#_Toc463820493)

[最大团 20](#_Toc463820494)

# 

# SG打表:

//f[]：可以取走的石子个数

//sg[]:0~n的SG函数值

//fhash[]:mex{}

int f[N],sg[N],fhash[N];

void getSG(int n)

{

int i,j;

memset(sg,0,sizeof(sg));

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

{

memset(fhash,0,sizeof(fhash));

for(j=1;f[j]<=i;j++)

fhash[sg[i-f[j]]]=1;

for(j=0;j<=n;j++) //求mes{}中未出现的最小的非负整数

{

if(fhash[j]==0)

{

sg[i]=j;

break;

}

}

}

}

dfs：

//注意 S数组要按从小到大排序 SG函数要初始化为-1 对于每个集合只需初始化1遍

//n是集合s的大小 S[i]是定义的特殊取法规则的数组

int s[110],sg[10010],n;

int SG\_dfs(int x)

{

int i;

if(sg[x]!=-1)

return sg[x];

bool vis[110];

memset(vis,0,sizeof(vis));

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

{

if(x>=s[i])

{

SG\_dfs(x-s[i]);

vis[sg[x-s[i]]]=1;

}

}

int e;

for(i=0;;i++)

if(!vis[i])

{

e=i;

break;

}

return sg[x]=e;

}

# 树链剖分

#include <cstdio>

#include <cstring>

#include <vector>

#include <algorithm>

using namespace std;

#define Del(a,b) memset(a,b,sizeof(a))

const int N = 10005;

int dep[N],siz[N],fa[N],id[N],son[N],val[N],top[N]; //top ×î½üµÄÖØÁ´¸¸½Úµã

int num;

vector<int> v[N];

struct tree

{

int x,y,val;

void read(){

scanf("%d%d%d",&x,&y,&val);

}

};

tree e[N];

void dfs1(int u, int f, int d) {

dep[u] = d;

siz[u] = 1;

son[u] = 0;

fa[u] = f;

for (int i = 0; i < v[u].size(); i++) {

int ff = v[u][i];

if (ff == f) continue;

dfs1(ff, u, d + 1);

siz[u] += siz[ff];

if (siz[son[u]] < siz[ff])

son[u] = ff;

}

}

void dfs2(int u, int tp) {

top[u] = tp;

id[u] = ++num;

if (son[u]) dfs2(son[u], tp);

for (int i = 0; i < v[u].size(); i++) {

int ff = v[u][i];

if (ff == fa[u] || ff == son[u]) continue;

dfs2(ff, ff);

}

}

#define lson(x) ((x<<1))

#define rson(x) ((x<<1)+1)

struct Tree

{

int l,r,val;

};

Tree tree[4\*N];

void pushup(int x) {

tree[x].val = max(tree[lson(x)].val, tree[rson(x)].val);

}

void build(int l,int r,int v)

{

tree[v].l=l;

tree[v].r=r;

if(l==r)

{

tree[v].val = val[l];

return ;

}

int mid=(l+r)>>1;

build(l,mid,v\*2);

build(mid+1,r,v\*2+1);

pushup(v);

}

void update(int o,int v,int val) //log(n)

{

if(tree[o].l==tree[o].r)

{

tree[o].val = val;

return ;

}

int mid = (tree[o].l+tree[o].r)/2;

if(v<=mid)

update(o\*2,v,val);

else

update(o\*2+1,v,val);

pushup(o);

}

int query(int x,int l, int r)

{

if (tree[x].l >= l && tree[x].r <= r) {

return tree[x].val;

}

int mid = (tree[x].l + tree[x].r) / 2;

int ans = 0;

if (l <= mid) ans = max(ans, query(lson(x),l,r));

if (r > mid) ans = max(ans, query(rson(x),l,r));

return ans;

}

int Yougth(int u, int v) {

int tp1 = top[u], tp2 = top[v];

int ans = 0;

while (tp1 != tp2) {

//printf("YES\n");

if (dep[tp1] < dep[tp2]) {

swap(tp1, tp2);

swap(u, v);

}

ans = max(query(1,id[tp1], id[u]), ans);

u = fa[tp1];

tp1 = top[u];

}

if (u == v) return ans;

if (dep[u] > dep[v]) swap(u, v);

ans = max(query(1,id[son[u]], id[v]), ans);

return ans;

}

void Clear(int n)

{

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

v[i].clear();

}

int main()

{

//freopen("Input.txt","r",stdin);

int T;

scanf("%d",&T);

while(T--)

{

int n;

scanf("%d",&n);

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

{

e[i].read();

v[e[i].x].push\_back(e[i].y);

v[e[i].y].push\_back(e[i].x);

}

num = 0;

dfs1(1,0,1);

dfs2(1,1);

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

if (dep[e[i].x] < dep[e[i].y]) swap(e[i].x, e[i].y);

val[id[e[i].x]] = e[i].val;

}

build(1,num,1);

char s[200];

while(~scanf("%s",&s) && s[0]!='D')

{

int x,y;

scanf("%d%d",&x,&y);

if(s[0]=='Q')

printf("%d\n",Yougth(x,y));

if (s[0] == 'C')

update(1,id[e[x].x],y);

}

Clear(n);

}

return 0;

}

# 树形dp

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

int first[205];

int nex[205];

int v[205];

int w[205];

int eid;

int dp[205][205];

void init()

{

memset(dp,0,sizeof(dp));

memset(w,0,sizeof(w));

eid=0;

for(int i=0;i<205;i++)

first[i]=-1;

}

void addedge(int m,int n)

{

v[++eid]=n;

nex[eid]=first[m];

first[m]=eid;

}

void solve(int st,int sx)

{

int i;

dp[st][1]=w[st];

for(i=first[st];i!=-1;i=nex[i])

{

int to=v[i];

if(sx>1)

solve(to,sx-1);

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

{

int v=j+1;

for(int k=1;k<=j;k++)

dp[st][v]=max(dp[st][v],dp[st][v-k]+dp[to][k]);

}

}

}

int main()

{

int n,m;

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

{

if(m==0&&n==0)

break;

init();

int i;

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

{

int a,b;

scanf("%d%d",&a,&b);

addedge(a,i);

w[i]=b;

}

solve(0,m+1);

printf("%d\n",dp[0][m+1]);

}

return 0;

}

# 状态压缩dp

#include <iostream>

#include <string>

#include <cstring>

#include <stack>

#include <algorithm>

using namespace std;

const int inf = 1<<30;

struct node

{

string name;

int dead,cost;

} a[50];

struct kode

{

int time,score,pre,now;

} dp[1<<15];

int main()

{

int t,i,j,s,n,end;

cin >> t;

while(t--)

{

memset(dp,0,sizeof(dp));

cin >> n;

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

cin >> a[i].name >> a[i].dead >> a[i].cost;

end = 1<<n;

for(s = 1; s<end; s++)

{

dp[s].score = inf;

for(i = n-1; i>=0; i--)

{

int tem = 1<<i;

if(s & tem)

{

int past = s-tem;

int st = dp[past].time+a[i].cost-a[i].dead;

if(st<0)

st = 0;

if(st+dp[past].score<dp[s].score)

{

dp[s].score = st+dp[past].score;

dp[s].now = i;

dp[s].pre = past;

dp[s].time = dp[past].time+a[i].cost;

}

}

}

}

stack<int> S;

int tem = end-1;

cout << dp[tem].score << endl;

while(tem)

{

S.push(dp[tem].now);

tem = dp[tem].pre;

}

while(!S.empty())

{

cout << a[S.top()].name << endl;

S.pop();

}

}

return 0;

}

# 数论公式 （补充）

1. 卢卡斯定理

表达式

C(n,m)%p=C(n/p,m/p)\*C(n%p,m%p)

适用领域范围

大组合数求模

求解组合数C(n,m)，即从n个相同物品中取出m个的方案数，由于结果可能非常大，对结果模10007即可。

方案1: 暴力求解，C(n,m)=n\*(n-1)\*...\*(n-m+1)/m!，n<=15

方案2: 打表，C(n,m)=C(n-1,m-1)+C(n-1,m)，n<=10,000

方案3: 质因数分解，C(n,m)=n!/(m!\*(n-m)!)，C(n,m)=p1a1-b1-c1p2a2-b2-c2…pkak-bk-ck,n<=10,000,000

方案4: Lucas定理，将m,n化为p进制,有:C(n,m)=C(n0,m0)\*C(n1,m1)...(mod p)，算一个不是很大的C(n,m)%p,p为素数，化为线性同余方程,用扩展的欧几里德定理求解，n在int范围内，修改一下可以满足long long范围内。 方案1

方案1:

int Combination(int n, int m)

{

const int M = 10007;

int ans = 1;

for(int i=n; i>=(n-m+1); --i)

ans \*= i;

while(m)

ans /= m--;

return ans % M;

}

方案2:

方案3：

//用筛法生成素数

const int MAXN = 1000000;

bool arr[MAXN+1] = {false};

vector<int> produce\_prim\_number()

{

vector<int> prim;

prim.push\_back(2);

int i,j;

for(i=3; i\*i<=MAXN; i+=2)

{

if(!arr[i])

{

prim.push\_back(i);

for(j=i\*i; j<=MAXN; j+=i)

arr[j] = true;

}

}

while(i<=MAXN)

{

if(!arr[i])

prim.push\_back(i);

i+=2;

}

return prim;

}

//计算n!中素因子p的指数

int Cal(int x, int p)

{

int ans = 0;

long long rec = p;

while(x>=rec)

{

ans += x/rec;

rec \*= p;

}

return ans;

}

//计算n的k次方对M取模，二分法

int Pow(long long n, int k, int M)

{

long long ans = 1;

while(k)

{

if(k&1)

{

ans = (ans \* n) % M;

}

n = (n \* n) % M;

k >>= 1;

}

return ans;

}

//计算C(n,m)

int Combination(int n, int m)

{

const int M = 10007;

vector<int> prim = produce\_prim\_number();

long long ans = 1;

int num;

for(int i=0; i<prim.size() && prim[i]<=n; ++i)

{

num = Cal(n, prim[i]) - Cal(m, prim[i]) - Cal(n-m, prim[i]);

ans = (ans \* Pow(prim[i], num, M)) % M;

}

return ans;

}

方案4：

#include <stdio.h>

const int M = 10007;

int ff[M+5]; //打表，记录n!，避免重复计算

//求最大公因数

int gcd(int a,int b)

{

if(b==0)

return a;

else

return gcd(b,a%b);

}

//解线性同余方程，扩展欧几里德定理

int x,y;

void Extended\_gcd(int a,int b)

{

if(b==0)

{

x=1;

y=0;

}

else

{

Extended\_gcd(b,a%b);

long t=x;

x=y;

y=t-(a/b)\*y;

}

}

//计算不大的C(n,m)

int C(int a,int b)

{

if(b>a)

return 0;

b=(ff[a-b]\*ff[b])%M;

a=ff[a];

int c=gcd(a,b);

a/=c;

b/=c;

Extended\_gcd(b,M);

x=(x+M)%M;

x=(x\*a)%M;

return x;

}

//Lucas定理

int Combination(int n, int m)

{

int ans=1;

int a,b;

while(m||n)

{

a=n%M;

b=m%M;

n/=M;

m/=M;

ans=(ans\*C(a,b))%M;

}

return ans;

}

int main(void)

{

int i,m,n;

ff[0]=1;

for(i=1;i<=M;i++) //预计算n!

ff[i]=(ff[i-1]\*i)%M;

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

printf("%d\n",func(n,m));

return 0;

}

剩余定理补充：

如果我们想知道x%m等于多少，当且仅当我们知道x%m1,x%m2..

x%mr分别等于多少，其中m1m2...

mr=m，并且mi相互互质，即构成独立剩余系。

即每个因数的mod都知道之后才能知道m的mod

[ C(n,m) %(p1\*p2\*p3\*....pn) ] %pi = C(n,m) % pi

# 素数个数

//Meisell-Lehmer

//G++ 218ms 43252k

#include<cstdio>

#include<cmath>

using namespace std;

#define LL long long

const int N = 5e6 + 2;

bool np[N];

int prime[N], pi[N];

int getprime()

{

int cnt = 0;

np[0] = np[1] = true;

pi[0] = pi[1] = 0;

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

{

if(!np[i]) prime[++cnt] = i;

pi[i] = cnt;

for(int j = 1; j <= cnt && i \* prime[j] < N; ++j)

{

np[i \* prime[j]] = true;

if(i % prime[j] == 0) break;

}

}

return cnt;

}

const int M = 7;

const int PM = 2 \* 3 \* 5 \* 7 \* 11 \* 13 \* 17;

int phi[PM + 1][M + 1], sz[M + 1];

void init()

{

getprime();

sz[0] = 1;

for(int i = 0; i <= PM; ++i) phi[i][0] = i;

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

{

sz[i] = prime[i] \* sz[i - 1];

for(int j = 1; j <= PM; ++j) phi[j][i] = phi[j][i - 1] - phi[j / prime[i]][i - 1];

}

}

int sqrt2(LL x)

{

LL r = (LL)sqrt(x - 0.1);

while(r \* r <= x) ++r;

return int(r - 1);

}

int sqrt3(LL x)

{

LL r = (LL)cbrt(x - 0.1);

while(r \* r \* r <= x) ++r;

return int(r - 1);

}

LL getphi(LL x, int s)

{

if(s == 0) return x;

if(s <= M) return phi[x % sz[s]][s] + (x / sz[s]) \* phi[sz[s]][s];

if(x <= prime[s]\*prime[s]) return pi[x] - s + 1;

if(x <= prime[s]\*prime[s]\*prime[s] && x < N)

{

int s2x = pi[sqrt2(x)];

LL ans = pi[x] - (s2x + s - 2) \* (s2x - s + 1) / 2;

for(int i = s + 1; i <= s2x; ++i) ans += pi[x / prime[i]];

return ans;

}

return getphi(x, s - 1) - getphi(x / prime[s], s - 1);

}

LL getpi(LL x)

{

if(x < N) return pi[x];

LL ans = getphi(x, pi[sqrt3(x)]) + pi[sqrt3(x)] - 1;

for(int i = pi[sqrt3(x)] + 1, ed = pi[sqrt2(x)]; i <= ed; ++i) ans -= getpi(x / prime[i]) - i + 1;

return ans;

}

LL lehmer\_pi(LL x)

{

if(x < N) return pi[x];

int a = (int)lehmer\_pi(sqrt2(sqrt2(x)));

int b = (int)lehmer\_pi(sqrt2(x));

int c = (int)lehmer\_pi(sqrt3(x));

LL sum = getphi(x, a) +(LL)(b + a - 2) \* (b - a + 1) / 2;

for (int i = a + 1; i <= b; i++)

{

LL w = x / prime[i];

sum -= lehmer\_pi(w);

if (i > c) continue;

LL lim = lehmer\_pi(sqrt2(w));

for (int j = i; j <= lim; j++) sum -= lehmer\_pi(w / prime[j]) - (j - 1);

}

return sum;

}

int main()

{

init();

LL n;

while(~scanf("%lld",&n))

{

printf("%lld\n",lehmer\_pi(n));

}

return 0;

}

# 最大团

/\*

最大团 = 补图G的最大独立集数

———>最大独立集数 = 补图G'最大团

\*/

//最大团模板

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <iostream>

using namespace std;

#define N 102

int mx;//最大团数(要初始化为0)

int x[N],tuan[N];

int can[N][N];//can[i]表示在已经确定了经选定的i个点必须在最大团内的前提下还有可能被加进最大团的结点集合

int num[N];//num[i]表示由结点i到结点n构成的最大团的结点数

bool g[N][N];//邻接矩阵(从1开始)

int n,m;

bool dfs(int tot,int cnt){

int i,j,k;

if(tot == 0){

if(cnt > mx){

mx = cnt;

for(i=0;i<mx;i++){

tuan[i] = x[i];

}

return true;

}

return false;

}

for(i=0;i<tot;i++){

if(cnt + (tot-i) <= mx)return false;

if(cnt + num[can[cnt][i]] <= mx)return false;

k = 0;

x[cnt] = can[cnt][i];

for(j=i+1;j<tot;j++){

if(g[can[cnt][i]][can[cnt][j]]){

can[cnt+1][k++] = can[cnt][j];

}

}

if(dfs(k,cnt+1))return false;

}

return false;

}

void MaxTuan(){

int i,j,k;

mx = 1;

for(i=n;i>=1;i--){

k = 0;

x[0] = i;

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

if(g[i][j]){

can[1][k++] = j;

}

}

dfs(k,1);

num[i] = mx;

}

}

int main(){

while(scanf("%d",&n) && n){

int i,j;

memset(g,0,sizeof(g));

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

for(j=1;j<=n;j++){

scanf("%d",&g[i][j]);

}

}

mx = 0;

MaxTuan();

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

}

return 0;

}