玄学卡常

```
#define isdigit(x) ((x) >= '0' && (x) <= '9')
int read() {
	int res = 0;
	char c = getchar();
	while(!isdigit(c)) c = getchar();
	while(isdigit(c)) res = (res << 1) + (res << 3) + c - 48, c = getchar();
	return res;
}
void printi(int x) {
	if(x / 10) printi(x / 10);
	putchar(x % 10 + '0');
}
```

数学

1. MR && Rho

```
ll prime[10]={2,3,5,7,11,13,17,19,61,24251};
ll pn[100];
ll len;
ll qmul(ll a,ll b,ll mod)
{
	return (a*b-(ll)((ld)a/mod*b)*mod+mod)%mod;
}
ll qpow(ll a,ll b,ll mod)
{
	ll ans=1;
	while(b)
	{
		if(b&1)
			ans=qmul(ans,a,mod)%mod;
		a=qmul(a,a,mod)%mod;
		b>>=1;
	}
	return ans;
}
ll gcd(ll a,ll b)
{
	if(a==0)
		return 1;
	return b?gcd(b,a%b):a;
}
bool MR(ll n)
{
	if(n==0 || n==1 || (n%2==0 && n>2))
```

```
        return false;
    for(int i=0;i<10;i++)
    {
        if(prime[i]==n)
            return true;
    }
    ll d=n-1,s=0;
    while(!(d&1))
    {
        s++;
        d>>=1;
    }
    for(int i=0;i<20;i++)
    {
        ll a;
        if(i>=10)
            a=rand()%(n-2)+2;
        else
            a=prime[i];
        ll x=qpow(a,d,n);
        ll y;
        for(int j=0;j<s;j++)
        {
            y=qmul(x,x,n);
            if(y==1 && x!=1 && x!=n-1)
                return false;
            x=y;
        }
        if(y!=1)
            return false;
    }
    return true;
}
ll rho(ll n,ll c)
{
    ll d,i=1,k=2;
    ll x=rand()%(n-2)+2;
    ll y=x;
    while(1)
    {
        i++;
        x=(qmul(x,x,n)+c)%n;
        d=gcd(abs(y-x),n);
        if(d<n && d>1)
```

```
		{
			return d;
		}
		if(y==x)
			return n;
		if(i==k)
		{
			y=x;
			k<<=1;
		}
	}
}
void fact(ll n,ll c)
{
	if(n==1)
		return;
	if(MR(n))
	{
		pn[++len]=n;
		return;
	}
	ll p=n,k=c;
	while(p>=n)
		p=rho(p,c--);
	fact(p,k);
	fact(n/p,k);
}
srand(time(0));
```

2.筛法

（1）埃氏筛法

```
const int maxn = 100005, N = 100000;
int vis[maxn];
for(int i = 2; i <= N; i++) if(!vis[i])
	for(int j = i + i; j <= N; j += i) vis[j] = 1;
```

（2）欧拉素数筛

```
const int maxn = 100005, N = 100000;
int vis[maxn], phi[maxn], prime[maxn], cnt;
phi[1] = 1;
for(int i = 2; i <= N; i++){
	if(!vis[i]){
		prime[++cnt] = i;
		phi[i] = i - 1;
	}
```

```
        for(int j = 1; j <= cnt; j++){
                int p = prime[j], mul = p * i;
                if(mul > N) break;
                vis[mul] = 1;
                if(i % p == 0){
                        phi[mul] = phi[i] * p;
                        break;
                } else phi[mul] = phi[i] * phi[p];
        }
}
```

3.扩展欧几里得

```
int exgcd(int m,int n,int &x,int &y)
{
        int x1,y1,x0,y0;
        x0=1; y0=0;
        x1=0; y1=1;
        x=0; y=1;
        int r=m%n;
        int q=(m-r)/n;
        while(r)
        {
                x=x0-q*x1; y=y0-q*y1;
                x0=x1; y0=y1;
                x1=x; y1=y;
                m=n; n=r; r=m%n;
                q=(m-r)/n;
        }
        return n;
}
int exgcd(int a,int b,int &x,int &y)
{
        if(b==0)
        {
                x=1;
                y=0;
                return a;
        }
        int r=exgcd(b,a%b,x,y);
        int t=x;
        x=y;
        y=t-a/b*y;
        return r;
}
```

4. 矩阵快速幂

**1.f(n)=a*f(n-1)+b*f(n-2)+c；（a,b,c 是常数）**

$$\begin{pmatrix} a & b & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ C \end{pmatrix} = \begin{pmatrix} f_n \\ f_{n-1} \\ C \end{pmatrix}$$



**2.f(n)=c^n-f(n-1) ；（c 是常数）**

$$\begin{pmatrix} -1 & c \\ 0 & c \end{pmatrix} * \begin{pmatrix} f_{n-1} \\ c^{n-1} \end{pmatrix} = \begin{pmatrix} f_n \\ c^n \end{pmatrix}$$

5. 中国剩余定理

设 $M = m_1 \times m_2 \times \cdots \times m_n = \prod_{i=1}^{n} m_i$ 是整数 $m_1,m_2,\ldots,m_n$ 的乘积，并设 $M_i = M/m_i,\ \forall i \in \{1,2,\cdots,n\}$ 是除了 $m_i$ 以外的 n-1 个整数的乘积。设 $t_i = M_i^{-1}$ 为 $M_i$ 模 $m_i$ 的数论倒数($t_i$ 为 $M_i$ 模 $m_i$ 意义下的逆元)

$M_i t_i \equiv 1 \pmod{m_i},\ \forall i \in \{1,2,\cdots,n\}$.方程组 $(S)$ 的通解形式为

$$x = a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n + kM = kM + \sum_{i=1}^{n} a_i t_i M_i, \quad k \in \mathbb{Z}.$$

在模 $M$ 的意义下，方程组 $(S)$ 只有一个解：$x = \left( \sum_{i=1}^{n} a_i t_i M_i \right) mod M$

```
LL exgcd(LL a,LL b,LL &x,LL &y) {
	if (b==0) {x=1,y=0;return a;}
	LL d=exgcd(b,a%b,y,x);y-=(a/b)*x;
	return d;
}
inline void excrt() {
	LL M=1,A=0;
	for (int i=1;i<=n;i++) {
		LL x,y,d=exgcd(M,m[i],x,y),mm=m[i]/d;
		if ((a[i]-A)%d) {puts("No solution!");return;}
		x=(x%mm+mm)%mm;
		LL k=(LL)((__int128)(a[i]-A)/d*x%mm+mm)%mm;
		LL nM=M*mm;
		A=(LL)((A+(__int128)k*M)%nM);
		M=nM;
	}
	printf("%lld",A);
}
```

6. 拉格朗日插值

对某个多项式函数，已知有给定的 $k + 1$ 个取值点：

$$(x_0, y_0), \ldots, (x_k, y_k)$$

其中$x_j$对应着自变量的位置，而$y_j$对应着函数在这个位置的取值。

假设任意两个不同的 *xj* 都互不相同，那么应用拉格朗日插值公式所得到的**拉格朗日插值多项式**为：

$$L(x) := \sum_{j=0}^{k} y_j \ell_j(x)$$

其中每个$\ell_j(x)$为**拉格朗日基本多项式**（或称**插值基函数**），其表达式为：

$$\ell_j(x) := \prod_{i=0, i\neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}.$$

拉格朗日基本多项式$\ell_j(x)$的特点是在$x_j$上取值为 **1**，在其它的点$x_i,\ i \neq j$上取值为 **0**。

重心拉格朗日插值

定义**重心权**

$$w_j = \frac{1}{\prod_{i=0, i\neq j}^{k} (x_j - x_i)}$$

$$L(x) = \frac{\sum_{j=0}^{k} \frac{w_j}{x - x_j} y_j}{\sum_{j=0}^{k} \frac{w_j}{x - x_j}} \qquad (2)$$

7. 凸包

```
struct point
{
	double x,y;
	point(double a,double b):x(a),y(b){}
	point(){x=0,y=0;}
	point operator-(point a)
	{
		return point(x-a.x,y-a.y);
	}
	bool operator<(const point &a)const
	{
		if(x!=a.x)
			return x<a.x;
		return y<a.y;
	}
};
vector<point> u,l,s,p; //p 为原始点集，s 为凸包
```

```
double area(point a,point b,point c)
{
	return fabs(0.5*(a.x*b.y-a.y*b.x+b.x*c.y-b.y*c.x+a.y*c.x-a.x*c.y));
}
double cha(point a,point b)
{
	return a.x*b.y-a.y*b.x;
}
bool judge(point a,point b,point c)
{
	b=b-a;
	c=c-a;
	return cha(b,c)<-eps;
}
void Andrew()
{
	sort(p.begin(),p.end());
	u.push_back(p[0]);
	u.push_back(p[1]);
	l.push_back(p[p.size()-1]);
	l.push_back(p[p.size()-2]);
	for(int i=2;i<(int)p.size();++i)
	{
		for(int j=u.size();j>=2 && !judge(u[j-2],u[j-1],p[i]);--j)
		{
			u.pop_back();
		}
		u.push_back(p[i]);
	}
	for(int i=p.size()-3;i>=0;--i)
	{
		for(int j=l.size();j>=2 && !judge(l[j-2],l[j-1],p[i]);--j)
		{
			l.pop_back();
		}
		l.push_back(p[i]);
	}
	for(int i=0;i<(int)u.size();++i)
		s.push_back(u[i]);
	for(int i=1;i<(int)l.size()-1;++i)
		s.push_back(l[i]);
}
```
图

1. 无向图最小环个数（重边不影响）

```
#define maxn 108
#define INF 1<<24
int dist[maxn][maxn], g[maxn][maxn];
int n, m;
void init(){
        for ( int i=1; i<=n; ++i )
              for ( int j=1; j<=n; ++j )
                  dist[i][j] = g[i][j] = INF;
        int u, v, w;
        for ( int i=0; i<m; ++i ) {
                  scanf("%d%d%d", &u, &v, &w);
                  if(dist[u][v] > w) {
                          dist[u][v] = dist[v][u] = w;
                          g[u][v] = g[v][u] = w;
                  }
        }
};
void solve(){
        int minn = INF, cnt=0;
        for ( int k=1; k<=n; ++k ){
                // 枚举与 k 相连的两条边，且端点号是小于 k 的
                  for ( int i=1; i<k; ++i ) if(g[i][k]^INF)
                    for( int j=i+1; j<k; ++j ) if(dist[i][j]^INF && g[k][j]^INF)
                    {
                            int tmp = dist[i][j]+g[i][k]+g[k][j];
                            if(tmp < minn ) {// 比最小的还小就更新
                                        minn = tmp;    cnt=1;
                            }else if(tmp == minn) ++cnt; // 和当前最小的环代价相等
                    }

            // 更新最短路
                for(int i=1; i<=n; ++i ) if(dist[i][k]^INF)
                    for(int j=1; j<=n; ++j ) if(dist[k][j]^INF)
                    {
                            int tmp= dist[i][k]+dist[k][j];
                            if(tmp < dist[i][j]) dist[i][j] = tmp;
                    }
        }
        if(cnt > 0) printf("%d %d\n", minn, cnt);
        else puts("-1 -1");
};
int main(){
        int    T; scanf("%d", &T);
```

```
    while( T-- ) {
        scanf("%d%d", &n, &m);
        init();
        solve();
    }
}
```

2. 求简单无向图中环的个数

```
ll dp[1<<maxn][maxn]; // 注意数据
bool g[maxn][maxn];
// dp[s][i] s 中最小的点到其他点路径数；
// dp[s][i] += dp[s^i][j](g[i][j] = true )
int main(){
    int n, m;
    while( cin >> n >> m ) {
        memset(g, 0, sizeof g);
        int state = (1<<n);
        for ( int i=0; i<state; ++i)
            for ( int j=0; j<n; ++j )
                dp[i][j] = 0;

        for ( int i=0, a, b; i<m; ++i ){
            scanf ("%d%d", &a, &b);
            --a, --b;
            g[a][b] = g[b][a] = true;
            dp[(1<<a)|(1<<b)][a] = dp[(1<<a)|(1<<b)][b] = 1;
        }
        ll ans = 0;
        for ( int s=1; s<state; ++s ){
            int i, j, k;
            for ( i=0; i<n && !(s&(1<<i)); ++i );
            for ( j=i+1; j<n; ++j ) if(s&(1<<j) )
            {
                for ( k=i+1; k<n; ++k ) if(s&(1<<k)){
                    if(g[k][j] )
                        dp[s][j] += dp[s^(1<<j)][k];
                }
                if(g[i][j] && (s^(1<<i)^(1<<j))) // 3 个点以上才行
                    ans += dp[s][j];
            }
        }
        // 枚举了环的两侧。
        cout << (ans>>1) << endl;
    }
```

```
};
```

3. 判断是否有奇环

```
bool sign;
void dfs(int x,int time)
{
	if(!sign)
		return;
	for(int i=head[x];i!=-1;i=en[i].nxt)
	{
		if(tim[en[i].to]==-1)
		{
			tim[en[i].to]=time;
			dfs(en[i].to,time+1);
		}
		else if((tim[en[i].to]+time)%2)
		{
		      // cout<<en[i].to<<endl;
			sign=false;
			return ;
		}
	}
}
//调用
bool flag=true;
		for(int i=1;i<=n;i++)
		{
			memset(tim,-1,sizeof(tim));
			sign=true;
			dfs(i,0);
			if(!sign)
			{
				flag=false;
				break;
			}
		}
```

4. 拓扑排序

```
	queue<int>q;
	vector<int>edge[n];
	for(int i=0;i<n;i++)    //n   节点的总数
		if(in[i]==0) q.push(i);    //将入度为 0 的点入队列
	vector<int>ans;     //ans  为拓扑序列
	while(!q.empty())
```

```
{
	int p=q.front(); q.pop(); //  选一个入度为 0 的点，出队列
	ans.push_back(p);
	for(int i=0;i<edge[p].size();i++)
	{
		int y=edge[p][i];
		in[y]--;
		if(in[y]==0)
			q.push(y);
	}
}
if(ans.size()==n)
{
	for(int i=0;i<ans.size();i++)
		printf( "%d ",ans[i] );
	printf("\n");
}
else printf("No Answer!\n");      //    ans  中的长度与 n 不相等，就说明无拓扑序列
```

杂

1. 二分查找+LIS

```
const int maxn =300003, INF = 0x7f7f7f7f;
int low[maxn], a[maxn];
int n, ans;
int binary_search(int *a, int r, int x)
//二分查找，返回 a 数组中第一个>=x 的位置
{
	int l = 1, mid;
	while(l <= r)
	{
		mid = (l+r) >> 1;
		if(a[mid] <= x)
			l = mid + 1;
		else
			r = mid - 1;
	}
	return l;
}
int main()
{
	scanf("%d", &n);
	for(int i=1; i<=n; i++)
	{
		scanf("%d", &a[i]);
```

```
        low[i] = INF;     //由于 low 中存的是最小值，所以 low 初始化为 INF
    }
    low[1] = a[1];
    ans = 1;     //初始时 LIS 长度为 1
    for(int i=2; i<=n; i++)
    {
        if(a[i] > low[ans])      //若 a[i]>=low[ans]，直接把 a[i]接到后面
            low[++ans] = a[i];
        else          //否则，找到 low 中第一个>=a[i]的位置 low[j]，用 a[i]更新 low[j]
            low[binary_search(low, ans, a[i])] = a[i];
    }
    printf("%d\n", ans);     //输出答案
    return 0;
}
```

2. 带权并查集

```
ll p[3000]; //父亲
ll s[3000]; //权值
ll r[3000]; //秩
void init(ll n)
{
    mem(s,0);
    mem(r,0);
    for(ll i=1;i<=n;++i)
        p[i]=i;
}
ll Find(ll x)
{
    if(p[x]==x)
        return x;
    else
    {
        ll temp=p[x];
        p[x]=Find(p[x]);
        s[x]=(s[temp]+s[x])%2;
        return p[x];
    }
}
void Union(ll x,ll y)
{
    ll xroot=Find(x);
    ll yroot=Find(y);
    if(xroot==yroot)
        return;
```

```
	if(r[xroot]>r[yroot])
	{
		p[yroot]=xroot;
		s[yroot]=(s[x]-s[y]+3)%2;
	}
	else if(r[xroot]<r[yroot])
	{
		p[xroot]=yroot;
		s[xroot]=(s[x]+s[y]+3)%2;
	}
	else
	{
		p[yroot]=xroot;
		s[yroot]=(s[x]-s[y]+3)%2;
		++r[xroot];
	}
}
bool sameroot(ll x,ll y)
{
	return Find(x)==Find(y);
}
bool check(ll x,ll y)
{
	ll xroot=Find(x);
	ll yroot=Find(y);
	if(xroot==yroot && s[x]==s[y])
		return false;
	return true;
}
```

3. 背包 1，2，3

```
int dp[10005];
int v;
void ZeroOnePack(int cost,int weight)
{
	for(int i=v;i>=cost;--i)
		dp[i]=max(dp[i],dp[i-cost]+weight);
}
void CompletePack(int cost,int weight)
{
	for(int i=cost;i<=v;++i)
		dp[i]=max(dp[i],dp[i-cost]+weight);
}
void MultiplePack(int cost,int weight,int num)
```

```
{
	if(cost*num>=v)
	{
		CompletePack(cost,weight);
		return;
	}
	int k=1;
	while(k<num)
	{
		ZeroOnePack(k*cost,k*weight);
		num-=k;
		k<<=1;
	}
	ZeroOnePack(num*cost,num*weight);
}
```