Rambo 模板

1. 数论

1.1 线性筛素数、分解质因数

```
#define N 65536
int sizePrime, prime[N];
bool check[N] = { 0 };
int factSize, factc[128], factv[128];

//线性筛素数
void getPrime(){
	sizePrime = 0;
	for (int i = 2; i < N; i++){
		if (!check[i]) prime[sizePrime++] = i;
		for (int j = 0; j < sizePrime && i*prime[j] < N; j++){
			check[i*prime[j]] = 1;
			if (i%prime[j] == 0) break;
		}
	}
}

//分解质因数
void decompose(int num){
	factSize = 0;
	for (int i = 0; i < sizePrime && prime[i]*prime[i] <= num; i++){
		if (num%prime[i] == 0){
			factv[factSize] = prime[i];
			factc[factSize] = 0;
			while (num%prime[i] == 0){
				factc[factSize]++;
				num /= prime[i];
			}
			factSize++;
		}
	}
	if (num != 1){
		factv[factSize] = num;
		factc[factSize++] = 1;
	}
}
```

1.2 欧几里得算法、求逆元、stein算法求最大公约数

```
typedef long long LL;
LL gcd(LL a, LL b){
	if (b == 0) return a;
	return gcd(b, a%b);
}

//扩展欧几里得
LL extend_euclid(LL a, LL b, LL &x, LL &y){
	if (b == 0){
		x = 1; y = 0;
		return a;
	}
	LL d = extend_euclid(b, a%b, y, x);
	y -= x*(a/b);
```

```
	return d;
}

//求逆元 gcd(a, b) = 1
LL getInverse(LL a, LL b){
	LL x, y;
	extend_euclid(a, b, x, y);
	if (x < 0) x = x + ((-x) / b + 1) * b;
	if (x > 0) x = x - x / b * b;
	return x;
}

//最大公约数stein算法
LL stein(LL a, LL b){
	LL t = 0;
	if (a < b){
		a ^= b; b ^= a; a ^= b;
	}
	while (b){
		if ((a&1) && (b&1)){
			b = (a - b)>>1;
			a -= b;
		}
		else if (a&1) b >>= 1;
		else if (b&1){
			a >>= 1;
			if (a < b){
				a ^= b; b ^= a; a ^= b;
			}
		}
		else{
			a >>= 1; b >>= 1; t++;
		}
	}
	return a <<= t;
}
```

1.3 中国剩余定理

```
// x = rem[i] mod mm[i]
LL mm[N], rem[N];

LL chineseRemainder(LL M, LL pc){
	LL i, ans = 0, x, y;
	for (i = 0; i < pc; i++){
		extend_euclid(M/mm[i], mm[i], x, y);
		ans = (ans + rem[i] * (M/mm[i]) * x) % M;
	}
	return ans;
}
```

1.4 迭代求模线性方程组

```
typedef long long LL;
int sizeF;
LL modNum[N], remNum[N];

LL calc(){
	LL i, j, k, lcm, d, x, y, z, ans = -1;
	for (i = 0; i < sizeF; i++){
```

```
            if (i+1 == sizeF){
                  ans = remNum[i] % modNum[i]; break;
            }
            else{
                  d = extend_euclid(modNum[i], modNum[i+1], x, y);
                  if (d == 0){
                        ans = -1; break;
                  }
                  j = remNum[i+1]-remNum[i];
                  if (j%d != 0){
                        ans = -1; break;
                  }
                  lcm = modNum[i]/d*modNum[i+1];
                  z = modNum[i+1]/d;
                  k = j/d*x;
                  k = (k%z + z) % z;
                  k = (remNum[i]%lcm + k*modNum[i]%lcm) % lcm;
                  if (k < 0) k = k + ((-k)/lcm+1)*lcm;
                  if (k > 0) k = k - k/lcm*lcm;
                  remNum[i+1] = k;
                  modNum[i+1] = lcm;
            }
      }
      return ans;
}
```

1.5 区间筛素数

```
//返回从l到r之间的素数, O(sqrt(n))
void intervalPrime(int l, int r){
      for (int i = 0; i < sizePrime; i++){
            int v = l/prime[i]*prime[i];
            while (v < l) v += prime[i];
            for (int j = v; j <= r; j += prime[i]) visit[j-l] = 1;
      }
      for (int i = l; i <= r; i++) if (!visit[i-l]) printf("%d\n", i);
}
```

1.6 线性求欧拉函数

```
LL phi[N], prime[N], sizePrime;
bool check[N] = { 0 };

void calPhi(){
      sizePrime = 0;
      for (int i = 2; i < N; i++){
            if (!check[i]){
                  prime[sizePrime++] = i;
                  phi[i] = i-1;
            }
            for (int j = 0; j < sizePrime && i*prime[j] < N; j++){
                  check[i*prime[j]] = 1;
                  if (i%prime[j] == 0){
                        phi[i*prime[j]] = phi[i]*prime[j]; break;
                  }
                  else phi[i*prime[j]] = phi[i]*(prime[j]-1);
            }
      }
}
```

1.7 计算a^b mod m

```
//性质：if (b >= phi(m)) b = b % phi(m) + phi(m)
//计算欧拉函数
LL cal_phi(LL x){
	LL ans = x;
	for (int i = 0; i < sizePrime && (LL)prime[i]*prime[i] <= x; i++){
		if (x%prime[i] == 0){
			ans = ans / prime[i] * (prime[i] - 1);
			while (x%prime[i] == 0) x /= prime[i];
		}
	}
	if (x != 1) ans = ans / x * (x - 1);
	return ans;
}
//快速幂取模
LL power(LL x, LL y, LL z){
	if (x == 0 && y == 0) return 1;
	if (x == 0) return 0;
	if (x == 1) return 1;
	LL ans = 1, d = x%z;
	if (x >= z) flag = 1;
	while (y > 0){
		if (y&1) ans = ans*d;
		if (ans >= z){
			flag = 1;
			ans %= z;
		}
		d = d * d;
		if (d >= z){
			flag = 1;
			d %= z;
		}
		y >>= 1;
	}
	return ans;
}

//递归求解
//例子：求a^(a^(a^...)) mod M
LL cal_f(int dep, LL mod){
	if (dep == 0){
		flag = 0;
		return 1;
	}
	LL ans, phi, tmp;
	phi = cal_phi(mod);
	tmp = cal_f(dep-1, phi);
	if (flag) tmp += phi;
	flag = 0;
	ans = power(u, tmp, mod);
	return ans;
}
```

1.8 求二次同余方程

```
/*
    求解x^2 = a mod p
    输入a输出x，解为x和p-x，当x==p-x时，解就一个
    并且返回的是较小的x
```

```
    返回-1表示无解
*/
LL squareRoot(LL a, LL p){
	LL i, j, k, ans;
	if (p == 2) return a%p;
	if (power(a, (p-1)/2, p) == 1){
		if (p%4 == 3) ans = power(a, (p+1)/4, p);
		else {
			for (i = 1; power(i, (p-1)/2, p) == 1; i++);
			j = (p-1)>>1;
			k = 0;
			do {
				j >>= 1;
				k >>= 1;
				if ((power(a, j, p) * power(i, k, p) + 1)%p == 0)
					k += (p-1)/2;
			} while (j%2 == 0);
			ans = (power(a, (j+1)/2, p) * power(i, k>>1, p)) % p;
		}
		if (ans*2 > p) ans = p-ans;
		return ans;
	}
	return -1;
}
```

1.9 Pell方程求最小正整数解

```
/* 最好用java
 * pell: x^2 - n * y^2 = 1
 * 求最小正整数解，当n为完全平方数时无解
 * a0 = sqrt(n) a1 = a0
 * p0 = 0, p1 = 1, q0 = 1, q1 = 0, g0 = 0, h0 = 1
 * gi = ai * hi-1 - gi-1
 * hi = (n - gi * gi)/hi-1
 * ai+1 = (gi + a0)/hi
 * pi = ai*pi-1 + pi-2
 * qi = ai*qi-1 + qi-2
 */
import java.math.*;
import java.util.*;

public class Main {
	public static void main(String[] args){
		Scanner input = new Scanner(System.in);
		long n;
		BigInteger p0, p1, p2;
		BigInteger q0, q1, q2;
		BigInteger a0, a1, a2;
		BigInteger g0, g1, h0, h1;

		while (input.hasNext()){
			n = input.nextLong();
			p0 = BigInteger.ZERO;
			p1 = BigInteger.ONE;
			q0 = BigInteger.ONE;
			q1 = BigInteger.ZERO;
			g0 = BigInteger.ZERO;
			h0 = BigInteger.ONE;

			a0 = BigInteger.valueOf((long)Math.sqrt(n * 1.0));
			a1 = a0;
```

```
                if (a0.multiply(a0).longValue() == n){
                        System.out.println("No solution!");
                        continue;
                }
                while (true){
                        g1 = a1.multiply(h0).subtract(g0);
                        g0 = g1;

                        h1 = BigInteger.valueOf(n).subtract(g1.multiply(g1));
                        h1 = h1.divide(h0);
                        h0 = h1;

                        a2 = g1.add(a0).divide(h1);

                        p2 = a1.multiply(p1).add(p0);
                        p0 = p1;
                        p1 = p2;

                        q2 = a1.multiply(q1).add(q0);
                        q0 = q1;
                        q1 = q2;

                        a1 = a2;

                        BigInteger tmp;
                        tmp = p2.multiply(p2);
                        tmp = tmp.substract(q2.multiply(q2).multiply
(BigInteger.valueOf(n)));
                        if (tmp.equals(BigInteger.ONE)){
                                System.out.println(p2 + " " + q2);
                                break;
                        }
                }
        }
    }
}
```

1.10 不定方程a^2 + b^2 = p

```
//p为质数
int main(){
    LL p, a, b, x, y, u, v, k;
    srand(time(0));
    while (scanf("%lld", &p) != EOF){
        if (p%4 == 3){
            puts("Illegal");
            continue;
        }
        printf("Legal ");
        while (1){
            a = rand() % (p-1) + 1;
            x = power(a, (p-1)/4, p);
            if (x*x%p == p-1) break;
        }
        y = 1;
        while (1){
            if (x > p/2) x = p-x;
            if (y > p/2) y = p-y;
            k = (x*x + y*y)/p;
            if (k == 1) break;
            a = x%k; b = y%k;
```

```
                    if (a > k/2) a -= k;
                    if (b > k/2) b -= k;
                    u = (a*x + b*y)/k;
                    v = (b*x - a*y)/k;
                    if (u < 0) u = -u;
                    if (v < 0) v = -v;
                    x = u; y = v;
             }
             if (x > y) swap(x, y);
             printf("%lld %lld\n", x, y);
      }
}
```

1.11 组合数取模（未预处理版本）

```
#define N 65536
typedef long long LL;
int sizePrime, prime[N];
int facts, factv[128], factc[N];
bool check[N] = { 0 };
LL temp[N];
//快速幂取模
LL power(LL x, LL y, LL m){
      LL ans = 1, d = x%m;
      while (y > 0){
            if (y&1) ans = ans * d % m;
            d = d * d % m;
            y >>= 1;
      }
      return ans;
}

//calculate: n! % p^c
//cnt -- number of p
LL factorialMod(LL l, int pp, int cc, int &cnt){
      int i, j, MOD;
      LL ans;
      for (i = l, cnt = 0; i > 0; i /= pp) cnt += i/pp;
      for (i = 0, MOD = 1; i < cc; i++) MOD *= pp;
      temp[0] = 1;
      for (i = 1; i < MOD; i++){
            if (i%pp == 0){
                  temp[i] = temp[i-1];
                  continue;
            }
            temp[i] = temp[i-1] * i % MOD;
      }
      ans = 1;
      for (i = l; i > 0; i /= pp){
            //if ... MOD > 数组长度
            ans = ans * power(temp[MOD-1], i/MOD, MOD) % MOD;
            ans = ans * temp[i%MOD] % MOD;
      }
      return ans;
}

//calculate: C(n, k) % m --> chinese remainder
LL combinationMod(int nn, int kk, int mm){
      LL rem[128], pi[128], f;
      LL d, x, y, ans = 0;
```

```
	int i, j, sum, ts;
	decompose(mm, factv, factc, facts);
	for (i = 0; i < facts; i++)
		for (pi[i] = j = 1; j <= factc[i]; j++)
			pi[i] *= factv[i];
	for (i = 0; i < facts; i++){
		sum = 0;
		f = factorialMod(nn, factv[i], factc[i], ts);
		sum += ts;
		rem[i] = f;

		f = factorialMod(nn-kk, factv[i], factc[i], ts);
		sum -= ts;
		d = extend_euclid(f, pi[i], x, y);
		if (x < 0) x = x + ((-x) / pi[i] + 1) * pi[i];
		rem[i] = (rem[i] * x % pi[i] + pi[i]) % pi[i];

		f = factorialMod(kk, factv[i], factc[i], ts);
		sum -= ts;
		d = extend_euclid(f, pi[i], x, y);
		if (x < 0) x = x + ((-x) / pi[i] + 1) * pi[i];
		rem[i] = (rem[i] * x % pi[i] + pi[i]) % pi[i];

		if (sum >= factc[i]) rem[i] = 0;
		else rem[i] = rem[i] * power(factv[i], sum, pi[i]) % pi[i];
		//for (j = 0; j < sum; j++) rem[i] = rem[i] * factv[i] % pi[i];
	}

	for (i = 0; i < facts; i++){
		d = extend_euclid(mm/pi[i], pi[i], x, y);
		ans = (ans + mm/pi[i] * x % mm * rem[i] + mm) % mm;
	}
	return ans;
}
```

1.12 组合数取模（预处理）

```
//checked by spoj5093
typedef long long LL;
#define N 32768
int n, k, m;
int sizePrime, prime[N];
bool check[N] = { 0 };
int facts, sg, factv[64], factc[64], pk[64], cnt1[16][N];
LL group[128], dp[N], res[16][N], inv[16][N];
LL deg[N], rev[128], crt[128], mul[64][64];

void decompose(LL num, int val[], int cnt[], int &ss){
	int i, j;
	ss = 0;
	for (i = 0; i < sizePrime && (LL)prime[i]*prime[i] <= num; i++){
		if (num%prime[i] == 0){
			val[ss] = prime[i];
			cnt[ss] = 0;
			while (num%prime[i] == 0){
				cnt[ss]++;
				num /= prime[i];
			}
			ss++;
		}
	}
```

```
	if (num != 1){
		val[ss] = num;
		cnt[ss++] = 1;
	}
}

void dfs(int dep, LL num){
	if (dep == facts){
		group[sg++] = num;
		return ;
	}
	dfs(dep+1, num);
	for (int i = 0; i < factc[dep]; i++){
		num *= factv[dep];
		dfs(dep+1, num);
	}
}

LL combModular(LL u, LL v){
	int i, sum;
	LL ans = 0, rem;
	for (i = 0; i < facts; i++){
		sum = cnt1[i][u] - cnt1[i][v] - cnt1[i][u-v];
		rem = res[i][u] * inv[i][v] % pk[i] * inv[i][u-v] % pk[i];

		if (sum >= factc[i]) rem = 0;
		else rem = rem * mul[i][sum] % pk[i];
		ans =  (ans + rev[i] * rem) % m;
	}
	return ans;
}

void init(){
	int i, j;
	LL d, x, y;
	decompose(k, factv, factc, facts);
	sg = 0;
	dfs(0, 1);
	sort(group, group+sg);
	decompose(m, factv, factc, facts);
	for (i = 0; i < facts; i++){
		pk[i] = mul[i][0] = 1;
		for (j = 1; j <= factc[i]; j++){
			pk[i] *= factv[i];
			mul[i][j] = pk[i];
		}
		d = extend_euclid(m/pk[i], pk[i], x, y);
		if (x < 0) x = x + ((-x)/pk[i]+1)*pk[i];
		if (x > 0) x %= pk[i];
		crt[i] = x;
		rev[i] = m / pk[i] * crt[i];

		res[i][0] = inv[i][0] = 1;
		cnt1[i][0] = 0;
		for (j = 1; j < N && j <= n; j++){
			x = j;
			y = 0;
			while (x%factv[i] == 0){
				y++;
				x /= factv[i];
```

```
                }
                cnt1[i][j] = cnt1[i][j-1] + y;
                res[i][j] = res[i][j-1] * x % pk[i];

                d = extend_euclid(res[i][j], pk[i], x, y);
                if (x < 0) x = x + ((-x)/pk[i]+1)*pk[i];
                if (x > 0) x %= pk[i];
                inv[i][j] = x;
            }
        }
        for (deg[1] = 1, i = 2; i < N; i++) deg[i] = deg[i-1] * (i-1) % m;
}

void DP(){
        int i, j;
        LL ret;
        dp[0] = 1;
        for (i = 1; i <= n; i++){
            dp[i] = 0;
            for (j = 0; j < sg && i-group[j] >= 0; j++){
                ret = combModular(i-1, group[j]-1);
                ret = ret * dp[i-group[j]] % m * deg[group[j]] % m;
                dp[i] = (dp[i] + ret) % m;
            }
        }
        printf("%lld\n", dp[n]);
}

int main(){
        getPrime();
        while (scanf("%d%d%d", &n, &k, &m) != EOF){
            init();
            DP();
        }
}
```

1.13 Miller-Rabin测试、 Pollard-Rho分解

```
//checked by pku1811
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
using namespace std;
typedef long long LL;
LL p; //the smallest prime

LL gcd(LL x, LL y){
        if (y == 0) return x;
        return gcd(y, x%y);
}
//I got wrong here, exceed long long
LL mul(LL x, LL y, LL m){
        LL t,T,a,b,c,d,e,f,g,h,v,ans;
        T = (LL)( sqrt(double(m)+0.5) );
        t=T*T-m;
        a=x/T; b=x%T; c=y/T; d=y%T;
        e=a*c/T; f=a*c%T;
        v=((a*d+b*c)%m+e*t)%m;
        g=v/T; h=v%T;
        ans=(((f+g)*t%m+b*d)%m+h*T)%m;
```

```
	while (ans<0) ans+=m;
	return ans;
}

LL power(LL a, LL u, LL m){
	LL ans = 1, d = a%m;
	while (u > 0){
		if (u&1) ans = mul(ans, d, m);
		d = mul(d, d, m);
		u >>= 1;
	}
	return ans;
}

LL get_random(LL n){
	LL a = rand();
	rand();
	a *= rand()%(n-1);
	a *= rand()%(n-1);
	a = a%(n-1)+1;
	return a;
}

//check pseudo-prime based on a
bool witness(LL a, LL n){
	LL x0, x1, x2, i, u, t;
	u = n-1, t = 0;
	while (u%2 == 0){
		t++;
		u /= 2;
	}
	x1 = x0 = power(a, u, n);
	if ( x1 == 1 ) return true;
	for ( i = 1; i <= t; i++ ){
		x2 = mul( x1, x1, n );
		if ( x2 == 1 && x1 != 1 && x1 != n-1 && x1 != -1 ) return false;
		x1 = x2;
	}
	if ( x1 != 1 ) return false;
	return true;
}

bool miller_rabin( LL n ){
	if ( n == 2 || n == 3 ) return true;
	LL i, a;
	for ( i = 0; i < 10; i++ ){
		a = get_random( n );
		if ( !witness( a, n ) ) break;
	}
	return i == 10;
}

LL f( LL x, LL n ){
	return (mul(x,x,n)+1)%n;
}

LL pollard_rho( LL n ){
	if ( n < 2 ) return 0;
	if ( n%2 == 0 ) return 2;
	LL x, y, i, d;
```

```
	for ( i = 1; i <= 10; i++ ){
		rand( );
		x = rand( )%n;
		y = f( x, n );
		d = gcd( (y-x+n)%n, n );
		while ( d == 1 ){
			x = f( x,n );
			y = f( f(y,n), n );
			d = gcd( (y-x+n)%n, n )%n;
		}
		if ( d ) return d;
	}
	return 0;
}

void smallest_prime( LL n ){
	if ( miller_rabin( n ) ){
		if ( p > n ) p = n;
		return ;
	}
	else {
		LL t = pollard_rho( n );
		if ( t == 0 ) return ;
		smallest_prime( t );
		smallest_prime( n/t );
	}
}

int main( ){
	LL ca, n;
	scanf( "%lld", &ca );
	while ( ca-- ){
		scanf( "%lld", &n );
		if ( miller_rabin( n ) )
			printf( "Prime\n" );
		else {
			p = n;
			smallest_prime( n );
			printf( "%lld\n", p );
		}
	}
}
```

1.14 Mobius反演

```
//checked by zju3435
#include <numeric>
#include <algorithm>
using namespace std;
typedef long long LL;
#define N 1000008
int size = 0, prime[N], mark[N] = { 0 };
int mobius[N], mertens[N];

void getPrime(){
	mobius[0] = 0;
	mobius[1] = 1;
	for (int i = 2; i < N; i++){
		if (!mark[i]){
			prime[size++] = i;
			mobius[i] = -1;
```

```
            }
            for (int j = 0; j < size && prime[j]*i < N; j++){
                  mark[i*prime[j]] = 1;
                  if (i%prime[j] == 0){
                        mobius[i*prime[j]] = 0;
                        break;
                  }
                  else mobius[i*prime[j]] = mobius[i] * mobius[prime[j]];
            }
      }
      partial_sum(mobius, mobius+N, mertens);
}

//return num of coprime pairs in L, W, O(sqrt(N))
LL coprime(LL x, LL y){
      if (x > y) swap(x, y);
      LL ans = 0, u, v;
      for (LL i = x, j; i > 0; i = j){
            u = x/i, v = y/i;
            j = max(x/(u+1), y/(v+1));
            ans += u * v * (mertens[i] - mertens[j]);
      }
      return ans;
}

//return num of coprime triples in L, W, H
LL coprime(LL x, LL y, LL z){
      if (x > y) return coprime(y, x, z);
      if (x > z) return coprime(z, y, x);
      LL ans = 0, u, v, w;
      for (LL i = x, j; i > 0; i = j){
            u = x/i, v = y/i, w = z/i;
            j = max(max(x/(u+1), y/(v+1)), z/(w+1));
            ans += u * v * w * (mertens[i] - mertens[j]);
      }
      return ans;
}

int main(){
      LL x, y, z, ans;
      getPrime();
      while (scanf("%lld%lld%lld", &x, &y, &z) != EOF){
            x--; y--; z--;
            ans = 3;
            ans += coprime(x, y) + coprime(y, z) + coprime(z, x);
            ans += coprime(x, y, z);
            printf("%lld\n", ans);
      }
}
```

2. 博弈

2.1 nim游戏

2.2 multi-sg游戏

Multi-SG 游戏规定,在符合拓扑原则的前提下,一个单一游戏的后继可以为多个单一游戏。

2.3 anti-sg游戏

桌子上有 N 堆石子,游戏者轮流取石子，每次只能从一堆中取出任意数目的石子，但不能不取。取走最后一个为败。
SJ 定理先手必胜：
(1)游戏的 SG 函数不为 0 且游戏中某个单一游戏的 SG 函数大于 1
(2)游戏的 SG 函数为 0 且游戏中没有单一游戏的 SG 函数大于 1

```
//checked by pku3480
#include <cstdio>
#include <cstring>
int main(){
	int ca, n, i, v, sg, mark;
	scanf("%d", &ca);
	while (ca--){
		scanf("%d", &n);
		sg = mark = 0;
		for (i = 0; i < n; i++){
			scanf("%d", &v);
			if (v > 1) mark = 1;
			sg ^= v;
		}
		if ((sg != 0 && mark) || (sg == 0 && !mark)) puts("John");
		else puts("Brother");
	}
}
```

2.4 every-sg游戏

对于没有结束的单一游戏，必须进行一步决策
定义step

```
step(u) = 0          p状态
          max{ step(v) + 1 }sg(u) > 0 sg(v) = 0
           min{ step(v) + 1 }sg(u) = 0
```

先手必胜当且仅当单一游戏中最大step为奇数

2.5 nim积

```
//checked by pku3533
#include <cstdio>
#include <cstring>
using namespace std;
int tab[20][20];

//初始化
void init(){
	int i, j, x, y;
	bool visit[64];
	for (i = 0; i < 16; i++) tab[i][0] = tab[0][i] = 0;
	for (i = 1; i < 16; i++){
		for (j = 1; j < 16; j++){
```

```
                        memset(visit, 0, sizeof(visit));
                        for (x = 0; x < i; x++)
                                for (y = 0; y < j; y++)
                                        visit[tab[x][y]^tab[x][j]^tab[i][y]] = 1;
                        for (x = 0; visit[x]; x++);
                        tab[i][j] = x;
                }
        }
}

int nim_multi_power(int x, int y){
        int M, p, s, t, d1, d2, i;
        if (x < 16) return tab[x][y];
        for (i = 0; ; i++){
                if ((1<<(1<<i)) <= x && x < (1<<(1<<(i+1)))) break;
        }
        M = 1<<(1<<i);
        p = x/M;
        s = y/M; t = y%M;
        d1 = nim_multi_power(p, s);
        d2 = nim_multi_power(p, t);
        return M * (d1 ^ d2) ^ nim_multi_power(M>>1, d1);
}

int nim_multi(int x, int y){
        int c1, c2, c3, M, p, q, s, t, i;
        if (x < y) return nim_multi(y, x);
        if (x < 16) return tab[x][y];
        for (i = 0; ; i++){
                if ((1<<(1<<i)) <= x && x < (1<<(1<<(i+1)))) break;
        }
        M = 1<<(1<<i);
        p = x/M; q = x%M;
        s = y/M; t = y%M;
        c1 = nim_multi(p, s);
        c2 = nim_multi(p, t) ^ nim_multi(q, s);
        c3 = nim_multi(q, t);
        return (c1 ^ c2) * M ^ c3 ^ nim_multi_power(M>>1, c1);
}

int main(){
        init();
        int n, x, y, z, ans, tmp;
        while (scanf("%d", &n) != EOF){
                ans = 0;
                while (n--){
                        scanf("%d%d%d", &x, &y, &z);
                        tmp = nim_multi(x, y);
                        ans ^= nim_multi(tmp, z);
                }
                if (ans) puts("No");
                else puts("Yes");
        }
}
```

3. 字符串

3.1 ac自动机多串匹配

```
//root = 0
struct Node{
	int end, fail, next[KIND];
	void init(){
		end = fail = 0;
		memset(next, 0, sizeof(next));
	}
}node[200000];
int size, mark[1008], pnt[1008];
int que[1000000];
char str[100008];

inline int hash(char ch){
	if (ch < 'a') return ch-'A';
	else return ch-'a'+26;
}

void init(){
	node[0].init();
	size = 1;
	memset(pnt, 0, sizeof(pnt));
}

//插入字典树
void insert(char *s, int id){
	int p = 0, q, i;
	for (i = 0; s[i]; i++){
		q = hash(s[i]);
		if (node[p].next[q] == 0){
			node[p].next[q] = size++;
			p = node[p].next[q];
			node[p].init();
		}
		else p = node[p].next[q];
	}
	if (node[p].end == 0) node[p].end = id;
	else pnt[id] = node[p].end;
}

//建立自动机
void build_ac_automation(){
	int i, p, q, r, head = 0, tail = 1;
	node[0].fail = 0;
	que[head] = 0;
	while (head < tail){
		p = que[head++];
		for (i = 0; i < 52; i++){
			if (node[p].next[i] != 0){
				q = node[p].next[i];
				if (p == 0) node[q].fail = 0;
				else{
					r = node[p].fail;
					while (r != 0){
						if (node[r].next[i] != 0){
							node[q].fail = node[r].next[i];
```

```
                                        break;
                                }
                                r = node[r].fail;
                        }
                        if (node[r].next[i] != 0)
                                node[q].fail = node[r].next[i];
                        else node[q].fail = 0;
                }
                que[tail++] = q;
        }
    }
}

//查询s中匹配了多少个串，mark记录是否匹配
void query(char *s){
    int i, p = 0, q, r;
    memset(mark, 0, sizeof(mark));
    for (i = 0; s[i]; i++){
        q = hash(s[i]);
        while (node[p].next[q] == 0 && p != 0) p = node[p].fail;
        p = node[p].next[q];
        p = (p == 0 ? 0 : p);
        r = p;
        while (r != 0 && node[r].end > 0){
            mark[node[r].end] = 1;
            node[r].end = 0;
            r = node[r].fail;
        }
    }
}

int main(){
    int i, n;
    char patt[2048];
    while (scanf("%s", str) != EOF){
        init();
        scanf("%d", &n);
        for (i = 1; i <= n; i++){
            scanf("%s", patt);
            insert(patt, i);
        }
        build_ac_automation();
        query(str);
        for (i = 1; i <= n; i++){
            if (mark[i] || (pnt[i] != 0 && mark[pnt[i]])) puts("Y");
            else puts("N");
        }
    }
}
```

3.2 字符串最小循环表示

```
//返回最小循环表示的位置
//注意tar数组是原来字符串的2倍
int minimum_expression( ){
    int i = 0, j = 1, k = 0;
    while ( i < size && j < size && k < size ){
        if ( tar[i+k] == tar[j+k] ) k++;
        else {
```

```
                if ( tar[i+k] > tar[j+k] ){
                        i = i+k+1;
                        if ( i == j ) j++;
                }
                else {
                        j = j+k+1;
                        if ( i == j ) j++;
                }
                k = 0;
            }
        }
        i = MIN( i, j );
        return i;
}
```

3.3 KMP算法

```
//计算fail数组
void make_fail(char *str, int *fail){
        int i = 1, j = 0;
        for ( ; i <= len; i++, j++){
                fail[i] = j;
                while (j > 0 && str[i] != str[j]) j = fail[j];
        }
}

//匹配
void kmp(char *pattern, char *text, int *fail, bool *mark){
        int i, j;
        for (i = 1, j = 1; i <= len; i++){
                while (j > 0 && pattern[j] != text[i]) j = fail[j];
                if (i == len){
                        j = fail[j];
                }
                j++;
        }
}
```

3.4 Suffix Array 倍增算法

```
int len, ll, sa1[N], sa2[N], rank1[N], rank2[N], height[N], h[N][20];

//比较函数
inline int cmp( int x, int y ){
        return str[x] < str[y];
}

//求suffix array
void create_suffix_array( ){
        int i, k, *s1 = sa1, *s2 = sa2, *r1 = rank1, *r2 = rank2;
        for ( i = 0; i < len; i++ ) s1[i] = i;
        sort( s1, s1+len, cmp );
        for ( r1[s1[0]] = 0, i = 1; i < len; i++ ){
                if ( str[s1[i]] == str[s1[i-1]] ) r1[s1[i]] = r1[s1[i-1]];
                else r1[s1[i]] = r1[s1[i-1]]+1;
        }
        for ( k = 1; k < len && r1[s1[len-1]] < len-1; k <<= 1 ){
                for ( i = 0; i < len; i++ ) r2[r1[s1[i]]] = i;
                for ( i = len-1; i >= 0; i-- )
                        if ( k <= s1[i] )
                                s2[r2[r1[s1[i]-k]]--] = s1[i]-k;
```

```
            for ( i = len-k; i < len-(k>>1); i++ ) s2[r2[r1[i]]] = i;
            swap( s1, s2 );
            for ( r2[s1[0]] = 0, i = 1; i < len; i++ ){
                  if ( r1[s1[i]] != r1[s1[i-1]]
                    || r1[s1[i]+k] != r1[s1[i-1]+k] )
                        r2[s1[i]] = r2[s1[i-1]]+1;
                  else r2[s1[i]] = r2[s1[i-1]];
            }
            swap( r1, r2 );
      }
      if ( s1 != sa1 ) for ( i = 0; i < len; i++ ) sa1[i] = s1[i];
      if ( r1 != rank1 ) for ( i = 0; i < len; i++ ) rank1[i] = r1[i];
}

//计算高度数组
void cal_height( ){
      int i, j, k;
      for ( i = k = 0; i < len; i++ ){
            if ( rank1[i] == 0 ) rank2[i] = 0;
            else{
                  for ( j = sa1[rank1[i]-1]; str[i+k] == str[j+k]; k++ );
                  rank2[i] = k;
                  if ( k > 0 ) k--;
            }
      }
      for ( i = 0; i < len; i++ ) height[i] = rank2[sa1[i]];
}

//rmq初始
void rmq_init( ){
      int i, j, l;
      for ( i = 0; i < len; i++ ) h[i][0] = height[i];
      for ( j = l = 1; l*2 <= len; j++, l <<= 1 )
            for ( i = 0; i <= len-l*2; i++ )
                  h[i][j] = MIN( h[i][j-1], h[i+l][j-1] );
}

//rmq查询 s+1, t
inline int rmq_query( int left, int right ){
      int j = 0, l = 1;
      while ( l*2 <= right-left+1 ){
            l <<= 1;
            j++;
      }
      return MIN( h[left][j], h[right-l+1][j] );
}
```

3.5 Suffix Array DC3算法

```
const int maxn=210000;
char s[maxn];
int len,k;
int sa[maxn],rank[maxn],h[maxn],height[maxn];
int num[maxn];

inline bool leq(int a1, int a2, int b1, int b2){
      return (a1 < b1 || a1 == b1 && a2 <= b2);
}

inline bool leq(int a1, int a2, int a3, int b1, int b2, int b3){
```

```
	return(a1 < b1 || a1 == b1 && leq(a2, a3, b2, b3));
}

static void radixPass(int* a, int* b, int* r, int n, int K){
	int* c = new int[K + 1];
	for(int i = 0; i <= K; i++) c[i] = 0;
	for(int i = 0; i < n; i++) c[r[a[i]]]++;
	for(int i = 0, sum = 0; i <= K; i++){
		int t = c[i]; c[i] = sum; sum += t;
	}
	for(int i = 0; i < n; i++) b[c[r[a[i]]]++] = a[i];
	delete [] c;
}

void suffixArray(int* T, int* SA, int n, int K){
	int n0 = (n + 2) / 3, n1 = (n + 1) / 3, n2 = n / 3, n02 = n0 + n2;
	int* R = new int[n02 + 3]; R[n02] = R[n02+1] = R[n02 + 2] = 0;
	int* SA12 = new int[n02+3]; SA12[n02] = SA12[n02+1] = SA12[n02+2] = 0;
	int* R0 = new int[n0];
	int* SA0 = new int[n0];
	for(int i = 0, j = 0; i < n + (n0 - n1); i++) if(i % 3 != 0) R[j++] = i;
	radixPass(R , SA12, T + 2, n02, K);
	radixPass(SA12, R , T + 1, n02, K);
	radixPass(R , SA12, T , n02, K);
	int name = 0, c0 = -1, c1 = -1, c2 = -1;
	for(int i = 0; i < n02; i++){
		if(T[SA12[i]] != c0 || T[SA12[i]+1] != c1 || T[SA12[i]+2] != c2){
			name++; c0 = T[SA12[i]]; c1 = T[SA12[i]+1]; c2 = T[SA12[i]+2];
		}
		if(SA12[i] % 3 == 1) { R[SA12[i] / 3] = name; }
		else{ R[SA12[i] / 3 + n0] = name; }
	}
	if(name < n02){
		suffixArray(R, SA12, n02, name);
		for(int i = 0; i < n02; i++) R[SA12[i]] = i + 1;
	}
	else for(int i = 0; i < n02; i++) SA12[R[i] - 1] = i;
	for(int i = 0, j = 0; i < n02; i++) if(SA12[i] < n0) R0[j++] = 3 * SA12[i];
	radixPass(R0, SA0, T, n0, K);
	for(int p = 0, t = n0 - n1, k = 0; k < n; k++){
#define GetI() (SA12[t] < n0 ? SA12[t] * 3 + 1 : (SA12[t] - n0) * 3 + 2)
		int i = GetI();
		int j = SA0[p];
		if(SA12[t] < n0 ?
				leq(T[i], R[SA12[t] + n0], T[j], R[j / 3]) :
				leq(T[i],T[i+1],R[SA12[t]-n0+1], T[j],T[j+1],R[j/3+n0]))
		{
			SA[k] = i; t++;
			if(t == n02)
				for(k++; p < n0; p++, k++) SA[k] = SA0[p];
		}
		else{
			SA[k] = j;
			if(++p == n0)for(k++; t < n02; t++, k++) SA[k] = GetI();
		}
	}
	delete [] R; delete [] SA12; delete [] SA0; delete [] R0;
}
```

4. 经典动态规划

4.1 字符串编辑距离

```
//增删改使得a串和b串相等
//checked by spoj6219
#define MIN(a, b) ((a) < (b) ? (a) : (b))
#define N 2048
int m, n, dp[2][N];
char str1[N], str2[N];

void DP(){
	int i, j, pre, cur, u, v;
	pre = 0, cur = 1;
	m = strlen(str1+1);
	n = strlen(str2+1);
	for (i = 0; i <= n; i++) dp[0][i] = i;
	for (i = 1; i <= m; i++){
		dp[cur][0] = i;
		for (j = 1; j <= n; j++){
			u = dp[pre][j]+1;
			v = dp[cur][j-1]+1;
			if (str1[i] == str2[j])
				dp[cur][j] = MIN(dp[pre][j-1], MIN(u, v));
			else
				dp[cur][j] = MIN(dp[pre][j-1]+1, MIN(u, v));
		}
		pre ^= 1; cur ^= 1;
	}
	printf("%d\n", dp[pre][n]);
}

int main(){
	int ca;
	scanf("%d", &ca);
	while (ca--){
		scanf("%s%s", str1+1, str2+1);
		DP();
	}
}
```

4.2 背包问题

```
#define N 256
#define M 100000
int n, vi[N], wi[N], ci[N];
int s, dp[M];

// 0/1背包
void knapsack01(){
	memset(dp, -1, sizeof(dp));
	dp[0] = 0;
	for (int i = 0; i < n; i++){
		for (int j = s; j-wi[i] >= 0; j--){
			if (dp[j-wi[i]] == -1) continue;
			if (dp[j] == -1 || dp[j] < dp[j-wi[i]] + vi[i])
				dp[j] = dp[j-wi[i]] + vi[i];
		}
	}
}
```

```
// 完全背包
void knapsackComplete(){
	int i, j;
	memset(dp, -1, sizeof(dp));
	dp[0] = 0;
	for (i = 0; i < n; i++){
		for (j = 0; j+wi[i] <= s; j++){
			if (dp[j] == -1) continue;
			if (dp[j+wi[i]] == -1 || dp[j+wi[i]] < dp[j] + vi[i])
				dp[j+wi[i]] = dp[j] + vi[i];
		}
	}
}

// 多重背包 mnlog(n)
void knapsackMulti(){
	int i, j, k, x, y, z;
	memset(dp, -1, sizeof(dp));
	dp[0] = 0;
	for (i = 0; i < n; i++){
		if (ci[i] == 0) continue;
		for (j = 0; ci[i]-(1<<j)+1 > 0; j++);
		j--;
		for (k = 0; k <= j; k++){
			if (k != j) x = 1<<k;
			else x = ci[i]-(1<<k)+1;
			for (y = s; y-x*wi[i] >= 0; y--){
				z = y-x*wi[i];
				if (dp[z] == -1) continue;
				if (dp[y] == -1 || dp[y] < dp[z] + x*vi[i])
					dp[y] = dp[z] + x*vi[i];
			}
		}
	}
}

//二维费用背包
//二维费用的背包问题是指对于每件物品，具有两种不同的费用；选择这件物品必须同时付出这两种代价；
//对于每种代价都有一个可付出的最大值（背包容量）。问怎样选择物品可以得到最大的价值。
//f[i][v][u]=max{f[i-1][v][u],f[i-1][v-a[i]][u-b[i]]+w[i]}

//分组背包
//有N件物品和一个容量为V的背包。第i件物品的费用是c[i]，价值是w[i]。这些物品被划分为若干组，每
//组中的物品互相冲突，最多选一件。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，
//且价值总和最大。
//f[k][v]表示前k组物品花费费用v能取得的最大权值，则有
//f[k][v]=max{f[k-1][v],f[k-1][v-c[i]]+w[i]|物品i属于组k}

//差值背包DP
```

4.3 最长上升公共子序列 n^2

```
int n, m;
int seq1[N], seq2[N];
int dp[N], pre[N];

//longest common increasing subsequence of sequence 1 and 2
void lcis(){
	int i, j, tmp;
	memset(dp, 0, sizeof(dp));
```

```
	memset(pre, 0, sizeof(pre));
	for (i = 1; i <= n; i++){
		for (j = i-1; j > 0; j--)
			if (seq1[i] == seq1[j]) {
				pre[i] = j;
				break;
			}
	}
	for (i = 1; i <= m; i++){
		tmp = 0;
		for (j = 1; j <= n; j++){
			if (seq1[j] < seq2[i] && tmp < dp[j]) tmp = dp[j];
			if (seq1[j] == seq2[i]){
				if (pre[j] != 0 && tmp < dp[pre[j]]) dp[j] = dp[pre[j]];
				else dp[j] = tmp+1;
			}
		}
	}
	for (tmp = 0, i = 1; i <= n; i++) tmp = max(tmp, dp[i]);
	printf("%d\n", tmp);
}
```

4.4 最大M子段和O(n*m)

```
//dp[i,j] = MAX(dp[i-1,j], f[j-1]) + val[i];
//f 数组记录分成i段的最优值
#define N 1000008
#define INF 0x3fffffff
int n, m, seq[N];
int dp[N], f[N];

void DP(){
	int i, j, k;
	for (i = 1; i <= n; i++){
		k = min(i, m);
		for (j = 1; j <= k; j++){
			dp[j] = max(dp[j], f[j-1]) + seq[i];
			f[j-1] = max(f[j-1], dp[j-1]);
		}
		f[j-1] = max(f[j-1], dp[j-1]);
	}
	printf("%d\n", f[m]);
}

int main(){
	int i;
	while (scanf("%d%d", &m, &n) != EOF){
		for (i = 1; i <= n; i++){
			dp[i] = f[i] = -INF;
			scanf("%d", seq+i);
		}
		DP();
	}
}
```

4.5 连通性状态压缩DP

```
//check by fzu1977
//ternary to encode state
//state is a rotation
//0:no brace 1: left brace 2:right brace
```

```
typedef long long LL;
#define N 16
int d[] = { 1, 3, 9, 27, 81, 243, 729, 2187, 6561, 19683, 59049, 177147, 531441,
1594323 };
int digit[16], temp[16], stk[16];
short visit[2][1594323] = { {0}, {0} };
short itr = 1;
LL dp[2][1594323];
int que[2][40000];
int rn, cn, cur, nxt, sq[2];
char mp[N][N];

void ternary(int val, int seq[]){
	for (int i = 0; i <= cn; ++i){
		seq[i] = val%3;
		val /= 3;
	}
}

//return the position matches left brace
int matchLeft(int seq[], int pos){
	int i, j = 0;
	for (i = 0; i <= cn; ++i){
		if (seq[i] == 1) stk[j++] = i;
		else if (seq[i] == 2){
			if (stk[j-1] == pos) return i;
			j--;
		}
	}
	return -1;
}

//return the position matches right brace
int matchRight(int seq[], int pos){
	int i, j = 0;
	for (i = cn; i > 0; --i){
		if (seq[i] == 2) ++j;
		else if (seq[i] == 1) j--;
		if (j < 0) return i;
	}
	return -1;
}

//change the value
inline void set(int s, LL delta){
	if (visit[nxt][s] < itr){
		visit[nxt][s] = itr;
		dp[nxt][s] = 0;
		que[nxt][sq[nxt]++] = s;
	}
	dp[nxt][s] += delta;
}

//9 transmissions
LL DP(){
	int i, j, k, ex, ey, pos, u, v;
	LL ans = 0, delta;
	for (i = 0; i < rn; ++i){
		for (j = 0; j < cn; ++j)
			if (mp[i][j] == '0'){
```

```
                    ex = i; ey = j;
                }
}
dp[0][0] = 1;
cur = 0, nxt = 1;
que[cur][0] = 0;
sq[cur] = 1;
for (i = 0; i < rn; ++i){
      for (j = 0; j < cn; ++j, ++itr){
            sq[nxt] = 0;
            for (k = 0; k < sq[cur]; k++){
                  u = que[cur][k];
                  delta = dp[cur][u];
                  if (j == 0 && u%3 != 0) continue;
                  digit[0] = u%3;
                  digit[1] = u/3%3;
                  if (mp[i][j] == 'X'){
                        if (digit[0] == 0 && digit[1] == 0){
                              v = u/3;
                              set(v, delta);
                        }
                  }
                  else {
                        if (mp[i][j] == '*'){
                              if (digit[0] == 0 && digit[1] == 0){
                                    v = u/3;
                                    set(v, delta);
                              }
                        }
                        if (digit[0] == 0 && digit[1] == 0){
                              v = u/3 + 2 + d[cn];
                              set(v, delta);
                        }
                        else if (digit[0] == 1 && digit[1] == 1){
                              ternary(u, digit);
                              pos = matchLeft(digit, 1);
                              v = (u - 2*d[pos])/3 + d[pos-1];
                              v = v - v%3;
                              set(v, delta);
                        }
                        else if (digit[0] == 2 && digit[1] == 2){
                              ternary(u, digit);
                              pos = matchRight(digit, 1);
                              v = (u - d[pos])/3 + 2*d[pos-1];
                              v = v - v%3;
                              set(v, delta);
                        }
                        else if (digit[0] == 2 && digit[1] == 1){
                              v = u/3 - 1;
                              set(v, delta);
                        }
                        else if (digit[0] == 1 && digit[1] == 2){
                              if (u == 7){
                                    if ((i >= ex && j >= ey) || (i > ex)){
                                          ans += delta;
                                    }
                              }
                        }
                        else if (digit[0] > 0 && digit[1] == 0){
                              v = u/3 + d[cn]*digit[0];
```

```
                                        set(v, delta);
                                        v = u/3 + digit[0];
                                        set(v, delta);
                                }
                                else if (digit[0] == 0 && digit[1] > 0){
                                        v = u/3 - (u/3%3) + d[cn]*digit[1];
                                        set(v, delta);
                                        v = u/3;
                                        set(v, delta);
                                }
                        }
                }
                cur ^= 1; nxt ^= 1;
        }
    }
    return ans;
}

int main(){
    int ca, t, i;
    scanf("%d", &ca);
    for (t = 1; t <= ca; ++t){
        scanf("%d%d", &rn, &cn);
        for (i = 0; i < rn; i++) scanf("%s", mp[i]);
        printf("Case %d: %lld\n", t, DP());
    }
}
```

4.6 最大公共矩形，直方图最大矩形

```
//checked by zju3367 O(n^3)
#include <cstdio>
#include <cstring>
struct anonymous {
    int x, y, l, r, t, b, s;
} tmp, ans;
char a[150][150], b[50][50];
int c[50], l[50], r[50];

int main(){
    bool flag;
    int m1, n1, m2, n2;
    while (scanf("%d%d", &m1, &n1) != EOF){
        memset(a, '#', sizeof(a));
        for (int i = 0; i < m1; ++i){ scanf("%s", a[50 + i] + 50); }
        scanf("%d%d", &m2, &n2);
        for (int i = 0; i < m2; ++i){ scanf("%s", b[i]); }
        flag = false;
        ans.s = 0;
        for (tmp.x = -m2 + 1; tmp.x < m1; ++tmp.x){
            for (tmp.y = -n2 + 1; tmp.y < n1; ++tmp.y){
                memset(c, 0, sizeof(c));
                for (int i = 0; i < m2; ++i){
                    for (int j = 0; j < n2; ++j) {
                        if (a[50+i+tmp.x][50+j+tmp.y] != b[i][j]){
                            c[j] = 0;
                        }
                        else {
                            ++c[j];
                        }
```

```
                                }

                                for (int j = 0; j < n2; ++j) {
                                        int k = j - 1;
                                        while (k >= 0 && c[k] >= c[j]){
                                                k = l[k];
                                        }
                                        l[j] = k;
                                }
                                for (int j = n2 - 1; j >= 0; --j){
                                        int k = j + 1;
                                        while (k < n2 && c[k] >= c[j]){
                                                k = r[k];
                                        }
                                        r[j] = k;
                                }
                                for (int j = 0; j < n2; ++j){
                                        tmp.l = l[j] + 1;
                                        tmp.r = r[j];
                                        tmp.t = i - c[j] + 1;
                                        tmp.b = i + 1;
                                        tmp.s = (tmp.r - tmp.l) * (tmp.b - tmp.t);
                                        if (tmp.s > ans.s){ ans = tmp; }
                                        else if (tmp.s == ans.s){ flag = true; }
                                }
                        }
                }
        }
        if (ans.s == 0) { puts("0 0"); }
        else { printf("%d %d\n%d %d\n%d %d\n", ans.b - ans.t, ans.r - ans.l, ans.x +
ans.t + 1, ans.y + ans.l + 1, ans.t + 1, ans.l + 1); }
    }
    return 0;
}
```

4.7 四边形DP

```
//checked by hdu3369 dp[n] = min{ dp[i] + w[i+1, n] }
#define MAX(a, b) ((a) > (b) ? (a) : (b))
#define MIN(a, b) ((a) < (b) ? (a) : (b))
typedef long long LL;
const LL INF = 1000000000000000000ll;
const int N = 65536;
struct Node {
    int x, y;
    Node(){}
    Node(int _x, int _y): x(_x), y(_y){}
    bool operator<(const Node &v) const {
        if (x != v.x) return x < v.x;
        return y < v.y;
    }
}node[N];
int n, m, top, stk[N], pos[2][N];
LL dp[2][N], yy[N];

void DP(){
    int i, j, k, cur = 0, nxt = 1;
    LL ans, tmp;
    memset(dp, 0, sizeof(dp));
    for (i = 1; i <= n; i++){
```

```
		dp[0][i] = yy[1]*node[i].x;
		pos[0][i] = 0;
	}
	ans = dp[0][n];
	for (i = 2; i <= m; i++){
		for (j = i; j <= n; j++) dp[nxt][j] = INF;
		//I was wrong here
		for (j = MAX(pos[cur][n], i-1); j < n; j++){
			tmp = dp[cur][j] + yy[j+1]*node[n].x;
			if (dp[nxt][n] > tmp){
				dp[nxt][n] = tmp; pos[nxt][n] = j;
			}
		}
		for (j = n-1; j >= i; j--){
			//I was wrong here
			for (k = MAX(pos[cur][j], i-1); k <= pos[nxt][j+1] && k < j; k++){
				tmp = dp[cur][k] + yy[k+1] * node[j].x;
				if (dp[nxt][j] > tmp){
					dp[nxt][j] = tmp; pos[nxt][j] = k;
				}
			}
		}
		ans = MIN(ans, dp[nxt][n]);
		cur ^= 1; nxt ^= 1;
	}
	cout << ans << endl;
}

void init(){
	int i, j;
	top = 0;
	for (i = 1; i <= n; i++){
		while (top > 0){
			if (node[stk[top]].y <= node[i].y)
				top--;
			else break;
		}
		stk[++top] = i;
	}
	for (i = 1, j = 1; i <= top; i++){
		while (j <= stk[i]){
			yy[j] = node[stk[i]].y;
			j++;
		}
	}
}

int main(){
	int i, u, v;
	while (scanf("%d%d", &n, &m) != EOF){
		for (i = 1; i <= n; i++){
			scanf("%d%d", &u, &v);
			node[i] = Node(u, v);
		}
		sort(node+1, node+1+n);
		init();
		DP();
	}
}
```

5. 数据结构

5.1 用树状数组解决区间查询问题

```
#include <cstdio>
#define LOWBIT(x) ((x)&(-(x)))
const int MAXN = 1024;
int B[MAXN], C[MAXN];

void bit_update(int *a, int p, int d) {
	for ( ; p && p < MAXN ; p += LOWBIT(p)) a[p] += d;
}

int bit_query(int *a, int p) {
	int s = 0;
	for ( ; p ; p -= LOWBIT(p)) s += a[p];
	return s;
}

void bit_update2(int *a, int p, int d) {
	for ( ; p ; p -= LOWBIT(p)) a[p] += d;
}

int bit_query2(int *a, int p) {
	int s = 0;
	for ( ; p && p < MAXN ; p += LOWBIT(p)) s += a[p];
	return s;
}

inline void _insert(int p, int d) {
	bit_update(B, p, p*d);
	bit_update2(C, p-1, d);
}

inline int _query(int p) {
	return bit_query(B, p) + bit_query2(C, p) * p;
}

inline void insert_seg(int a, int b, int d) {
	_insert(a-1, -d);
	_insert(b, d);
}

inline int query_seg(int a, int b) {
	return _query(b) - _query(a-1);
}

int main() {
	int com, a, b, c;
```

```
	while (scanf("%d%d%d",&com,&a,&b) != EOF) {
		a += 2; b += 2;			//防止出现负数
		if (com == 0) {			//更新
			scanf("%d",&c);
			insert_seg(a, b, c);
		} else {			//查询
			printf("%d\n",query_seg(a,b));
		}
	}
	return 0;
}
```

5.2 Range minimum query

```
#define N 1<<18
#define MIN(a, b) ((a) < (b) ? (a) : (b))
int size, h[N][18], val[N];

void rmq_init(){
	int i, j, l;
	for (i = 0; i < size; i++) h[i][0] = val[i];
	for (j = l = 1; l*2 <= size; j++, l <<= 1){
		for (i = 0; i <= size-l*2; i++)
			h[i][j] = MIN(h[i][j-1], h[i+l][j-1]);
	}
}

inline int rmq_query(int start, int end){
	int j = 0, l = 1;
	while (2*l <= end-start+1){
		j++;
		l <<= 1;
	}
	return MIN(h[start][j], h[end-l+1][j]);
}
```

5.3 划分树

```
#define N 100008
int n, m;
int seq[N], ind[N], next[N], pos[N];
int cntL[20][N];

//比较函数
inline int cmp(int x, int y){ return seq[x] < seq[y]; }

//建树，cntL记录该区间有多少个数走到左子树
void build(int l, int r, int head, int dep){
	if (l == r){
		cntL[dep][l] = cntL[dep][l-1];
		return ;
	}
	int mid = (l+r)>>1;
	int hl = 0, hr = 0, tl = 0, tr = 0;
	for (int i = head, j = l; i != -1; i = next[i], j++){
		cntL[dep][j] = cntL[dep][j-1];
		if (pos[i] <= mid){
```

```
                        next[tl] = i;
                        tl = i;
                        if (hl == 0) hl = i;
                        cntL[dep][j]++;
                }
                else{
                        next[tr] = i;
                        tr = i;
                        if (hr == 0) hr = i;
                }
        }
        next[tl] = next[tr] = -1;
        build(l, mid, hl, dep+1);
        build(mid+1, r, hr, dep+1);
}

//返回下标
//查询区间[ql, qr]第kth大元素，递归到dep，树的区间为[left, right]
int query(int left, int right, int ql, int qr, int kth, int dep){
        if (left == right) return ind[left];
        int mid = (left+right) >> 1;
        if (cntL[dep][qr] - cntL[dep][ql-1] >= kth){
                return query(left, mid, \
                                left+cntL[dep][ql-1]-cntL[dep][left-1], \
                                left+cntL[dep][qr]-cntL[dep][left-1]-1, \
                                kth, dep+1);
        }
        else{
                return query(mid+1, right,
                        mid+1+ql-left-(cntL[dep][ql-1]-cntL[dep][left-1]), \
                        mid+qr+1-left-(cntL[dep][qr]-cntL[dep][left-1]), \
                kth-(cntL[dep][qr]-cntL[dep][ql-1]), dep+1);
        }
}

int main(){
        int i, u, v, w;
        while (scanf("%d%d", &n, &m) != EOF){
                for (i = 1; i <= n; i++){
                        scanf("%d", seq+i);
                        ind[i] = i;
                }
                //初始化
                sort(ind+1, ind+1+n, cmp);
                for (i = 1; i <= n; i++){
                        pos[ind[i]] = i;
                        next[i] = i+1;
                }
                next[n] = -1;
                build(1, n, 1, 0);
                while (m--){
                        scanf("%d%d%d", &u, &v, &w);
                        i = query(1, n, u, v, w, 0);
                        printf("%d\n", seq[i]);
                }
        }
}
```

5.4 树链剖分模板

```
//主要是建树 checked by toj3701
//基本树链剖分：dfs+线段树 complexity：(log(n))^2
#define N 30008
#define INF 1000000000
#define MAX(a, b) ((a) > (b) ? (a) : (b))
#define LC(x) (x<<1)
#define RC(x) (x<<1|1)
//vn:点的个数
//size:图的边数
//mp:head of link list
//son:子树节点个数
//seq:节点的dfs顺序
//_seq:节点在dfs序中的位置
//level:树链的大小
//_level:节点位于树链的第几层
//start:树链的开始位置
int vn, size, mp[N], visit[N], wei[N];
int pnt[N], son[N];
int sizeS, seq[N], _seq[N];
int sizeL, level[N], _level[N], start[N], sl[N];

//node for the segment tree
struct Node{
	int l, r;
	int sum, val;
}tree[N*32];
//edges of tree
struct Edge{
	int v, next;
	Edge(){}
	Edge(int _v, int _next): v(_v), next(_next){}
}edge[N<<1];

inline void add_edge(int u, int v){
	edge[size] = Edge(v, mp[u]); mp[u] = size++;
	edge[size] = Edge(u, mp[v]); mp[v] = size++;
}

//dfs1: 找出子树大小
void dfs1(int u){
	int i, v;
	visit[u] = son[u] = 1;
	for (i = mp[u]; i != -1; i = edge[i].next){
		v = edge[i].v;
		if (visit[v]) continue;
		pnt[v] = u; dfs1(v); son[u] += son[v];
	}
}

//dfs2: 找出重边
void dfs2(int u, int l){
	int i, v, x = -1, y;
	visit[u] = 1;
	_seq[u] = sizeS; seq[sizeS++] = u;
	level[l]++; _level[u] = l;
	for (i = mp[u]; i != -1; i = edge[i].next){
		if (visit[edge[i].v]) continue;
		if (x < son[edge[i].v]){
```

```
                x = son[edge[i].v]; y = edge[i].v;
            }
        }
        if (x == -1) return ;
        dfs2(y, l);
        for (i = mp[u]; i != -1; i = edge[i].next){
            if (visit[edge[i].v]) continue;
            sizeL++;
            dfs2(edge[i].v, sizeL);
        }
}

//build segtree
//org: the start position of the segtree
void build(int org, int id, int l, int r){
        tree[org+id].l = l; tree[org+id].r = r;
        if (l == r){
            tree[org+id].val = tree[org+id].sum = wei[seq[l]];
            return ;
        }
        int mid = (l+r)>>1;
        build(org, LC(id), l, mid);
        build(org, RC(id), mid+1, r);
        tree[org+id].sum = tree[org+LC(id)].sum + tree[org+RC(id)].sum;
        tree[org+id].val = MAX(tree[org+LC(id)].val, tree[org+RC(id)].val);
}

void change(int org, int id, int p, int v){
        if (p <= tree[org+id].l && tree[org+id].r <= p){
            tree[org+id].val = tree[org+id].sum = v;
            return ;
        }
        int mid = (tree[org+id].l + tree[org+id].r)>>1;
        if (p <= mid) change(org, LC(id), p, v);
        if (p > mid) change(org, RC(id), p, v);
        tree[org+id].sum = tree[org+LC(id)].sum + tree[org+RC(id)].sum;
        tree[org+id].val = MAX(tree[org+LC(id)].val, tree[org+RC(id)].val);
}

int queryMax(int org, int id, int l, int r){
        if (l <= tree[org+id].l && tree[org+id].r <= r)  return tree[org+id].val;
        int mid = (tree[org+id].l + tree[org+id].r)>>1;
        int a = -INF, b = -INF;
        if (l <= mid) a = queryMax(org, LC(id), l, r);
        if (r > mid) b = queryMax(org, RC(id), l, r);
        return MAX(a, b);
}

int querySum(int org, int id, int l, int r){
        if (l <= tree[org+id].l && tree[org+id].r <= r)
            return tree[org+id].sum;
        int mid = (tree[org+id].l + tree[org+id].r)>>1, ret = 0;
        if (l <= mid) ret += querySum(org, LC(id), l, r);
        if (r > mid) ret += querySum(org, RC(id), l, r);
        return ret;
}

void init(){
        int i, j;
        memset(visit, 0, sizeof(visit));
```

```
	memset(level, 0, sizeof(level));
	pnt[1] = 0;
	dfs1(1);
	memset(visit, 0, sizeof(visit));
	sizeL = sizeS = 1;
	dfs2(1, sizeL);
	for (sl[0] = 0, i = 1, j = 0; i <= sizeL; i++){
		start[i] = j*4;
		sl[i] = level[i] + sl[i-1];
		build(start[i], 1, j+1, j+level[i]);
		j += level[i];
	}
}

int main(){
	char cmd[16];
	int i, q, u, v, ans, tmp, x, y, a, b;
	while (scanf("%d", &vn) != EOF){
		size = 0;
		memset(mp, -1, sizeof(mp));
		for (i = 1; i < vn; i++){
			scanf("%d%d", &u, &v);
			add_edge(u, v);
		}
		for (i = 1; i <= vn; i++) scanf("%d", wei+i);
		init();
		scanf("%d", &q);
		while (q--){
			scanf("%s%d%d", cmd, &u, &v);
			if (cmd[1] == 'M'){
				ans = -INF;
				while (_level[u] != _level[v]){
					if (_level[u] < _level[v]) swap(u, v);
					x = sl[_level[u]] - level[_level[u]] + 1;
					y = _seq[u];
					tmp = queryMax(start[_level[u]], 1, x, y);
					ans = MAX(ans, tmp);
					u = pnt[seq[x]];
				}
				if (_seq[u] > _seq[v]) swap(u, v);
				tmp = queryMax(start[_level[u]], 1, _seq[u], _seq[v]);
				ans = MAX(ans, tmp);
				printf("%d\n", ans);
			}
			if (cmd[1] == 'S'){
				ans = 0;
				while (_level[u] != _level[v]){
					if (_level[u] < _level[v]) swap(u, v);
					x = sl[_level[u]] - level[_level[u]] + 1;
					y = _seq[u];
					tmp = querySum(start[_level[u]], 1, x, y);
					ans += tmp;
					u = pnt[seq[x]];
				}
				if (_seq[u] > _seq[v]) swap(u, v);
				tmp = querySum(start[_level[u]], 1, _seq[u], _seq[v]);
				ans += tmp;
				printf("%d\n", ans);
			}
			if (cmd[1] == 'H') change(start[_level[u]], 1, _seq[u], v);
```

```
            }
        }
}
```

5.5 n维矩形切割

```
#define DIM 16
#define MOD 14121413LL
typedef long long LL;
//bot 记录下端点 top 记录上端点
struct Cube{
        int bot[DIM], top[DIM];
        Cube(){ }
};
int n, dim, s1, s2;
Cube seq[1<<7], cube[1<<15], temp[1<<15];

//判断相交
bool isIntersect(Cube &p, Cube &q){
        int i;
        for (i = 0; i < dim; i++)
                if (p.bot[i] >= q.top[i] || p.top[i] <= q.bot[i])
                        return 0;
        return 1;
}

//add into temp
void insD(Cube t){
        for (int i = 0; i < dim; i++) if (t.bot[i] >= t.top[i]) return ;
        temp[s2++] = t;
}

//partition
void divide(Cube p, Cube q){
        Cube t1 = p, t2;
        for (int i = 0; i < dim; i++){
                if (t1.bot[i] <= q.bot[i] && q.bot[i] <= t1.top[i]){
                        t2 = t1;
                        t2.top[i] = q.bot[i];
                        insD(t2);
                        t1.bot[i] = q.bot[i];
                }
                if (t1.bot[i] <= q.top[i] && t1.top[i] >= q.top[i]){
                        t2 = t1;
                        t2.bot[i] = q.top[i];
                        insD(t2);
                        t1.top[i] = q.top[i];
                }
        }
}

//add a new cube
void add(Cube cb){
        int i;
        s2 = 0;
        for (i = 0; i < s1; i++){
                if (isIntersect(cube[i], cb)) divide(cube[i], cb);
                else temp[s2++] = cube[i];
        }
        for (i = s1 = 0; i < s2; i++) cube[s1++] = temp[i];
        cube[s1++] = cb;
```

```
}

void solve(){
	int i, j;
	LL ans = 0, vol;
	s1 = s2 = 0;
	for (i = 0; i < n; i++){
		for (j = 0; j < dim; j++) scanf("%d", &(seq[i].bot[j]));
		for (j = 0; j < dim; j++) scanf("%d", &(seq[i].top[j]));
		add(seq[i]);
	}
	for (i = 0; i < s1; i++){
		vol = 1;
		for (j = 0; j < dim; j++)
			vol = vol * ((LL)cube[i].top[j] - (LL)cube[i].bot[j]) % MOD;
		ans = (ans + vol) % MOD;
	}
	printf("%lld\n", ans);
}
```

5.6 左偏树

```
//checked by 3406
#define N 1000008
int r, n, m, size;
int cnt[100], seq[N/100];
struct LeftTree{
	int key, dist, cnt;
	LeftTree *lc, *rc;
	inline void init(int _key){
		key = _key;
		dist = 0;
		cnt = 1;
		lc = rc = NULL;
	}
	inline void up(){
		cnt = 1;
		if (lc) cnt += lc->cnt;
		if (rc) cnt += rc->cnt;
	}
}node[N], *tree[200], *que[N/10], *play[2];

inline LeftTree *merge(LeftTree *a, LeftTree *b){
	if (a == NULL) return b;
	if (b == NULL) return a;
	if (a->key < b->key) swap(a, b);
	a->rc = merge(a->rc, b);
	a->rc->up();
	if (a->lc == NULL) swap(a->lc, a->rc);
	if (a->rc == NULL){
		a->dist = 0;
		a->up();
		return a;
	}
	if (a->lc->dist < a->rc->dist) swap(a->lc, a->rc);
	a->dist = a->rc->dist+1;
	a->up();
	return a;
}
```

```
LeftTree *build(int s){
	LeftTree *ptr;
	int i, j;
	int head = 0, tail = 0;
	for (i = 0; i < s; i++){
		node[size].init(seq[i]);
		que[tail++] = &node[size];
		size++;
	}
	while (tail-head != 1){
		ptr = merge(que[head], que[head+1]);
		head += 2;
		que[tail++] = ptr;
	}
	return que[head];
}
//删除最小节点
inline void remove(LeftTree *&root){
	if (root == NULL) return ;
	LeftTree *ptr1 = root, *ptr2 = merge(root->lc, root->rc);
	ptr1->lc = ptr1->rc = NULL;
	root = ptr2;
}

inline void exchange(LeftTree *&root, int p){
	if (root == NULL) return ;
	LeftTree *ptr1 = root, *ptr2 = merge(root->lc, root->rc);
	ptr1->lc = ptr1->rc = NULL;
	ptr1->key = p;
	root = merge(ptr2, ptr1);
}

int main(){
	char cmd[10];
	int i, j, u, v, pre, cur;
	int p1 = 0, p2 = 0;
	while (scanf("%d", &r) != EOF){
		while (r--){
			scanf("%d%d", &n, &m);
			for (i = 0; i < m; i++) scanf("%d", cnt+i);
			size = 0;
			for (i = 0; i < m; i++){
				for (j = 0; j < cnt[i]; j++) scanf("%d", seq+j);
				tree[i] = build(cnt[i]);
			}
			play[0] = play[1] = NULL;
			cur = 0, pre = 1;
			for (i = 0; i < n; i++){
				scanf("%s", cmd);
				if (cmd[0] == 'T'){
					scanf("%d", &u);
					u--;
					play[cur] = merge(play[cur], tree[u]);
				}
				else if (cmd[0] == 'C'){
					if (play[cur]->key > play[pre]->key){
						play[cur] = merge(play[cur], play[pre]);
						play[pre] = NULL;
					}
					else if (play[cur]->key < play[pre]->key){
```

```
                                        play[pre] = merge(play[pre], play[cur]);
                                        play[cur] = NULL;
                                }
                        }
                        else if (cmd[0] == 'L'){
                                remove(play[cur]);
                        }
                        else if (cmd[0] == 'A'){
                                scanf("%d", &u);
                                play[cur]->key += u;
                        }
                        else if (cmd[0] == 'E'){
                                scanf("%d", &u);
                                exchange(play[cur], u);
                        }
                        cur ^= 1; pre ^= 1;
                }
                u = 0;
                if (play[0] != NULL) u = play[0]->cnt;
                v = 0;
                if (play[1] != NULL) v = play[1]->cnt;
                printf("%d:%d\n", u, v);
                if (u >= v) p1++;
                else p2++;
        }
        if (p1 > p2) printf("Hahaha...I win!!\n");
        else printf("I will be back!!\n");
    }
}
```

5.7 Splay Tree

```
//checked by 3578
#define N 100008
int n, m, SS, seq[N*3];

struct SplayTree{
    struct Node{
        int key, cnt, rev;
        Node *lc, *rc, *pnt;
    }node[N*3], *root, *size;

    void init(int s){
        Node *p = NULL, *q = NULL;
        root = NULL;
        size = node;
        for (int i = 0; i <= s+1; i++){
            p = newNode(i);
            if (q == NULL) q = p;
            else {
                p->lc = q; q->pnt = p; up(p);
                q = p;
            }
        }
        root = q;
        for (int i = 1; i <= s+1; i += 2) splay(&node[i], root);
    }
    inline Node *newNode( int _key ){
        size->key = _key; size->cnt = 1;
        size->rev = 0;
```

```
	size->lc = size->rc = size->pnt = NULL;
	return size++;
}
inline void down( Node *p ){
	if ( p->rev ){
		if (p->lc) p->lc->rev ^= 1;
		if (p->rc) p->rc->rev ^= 1;
		swap(p->lc, p->rc);
		p->rev = 0;
	}
}
inline void up( Node *p ){
	p->cnt = 1;
	if ( p->lc ) p->cnt += p->lc->cnt;
	if ( p->rc ) p->cnt += p->rc->cnt;
}
inline void zig( Node *p, Node *&rt ){
	int mark = 0;
	Node *q = p->pnt;
	if ( q == rt ) mark = 1;
	q->lc = p->rc;
	if ( p->rc ) p->rc->pnt = q;
	p->rc = q;
	p->pnt = q->pnt;
	q->pnt = p;
	if ( p->pnt ){
		if ( p->pnt->lc == q ) p->pnt->lc = p;
		else p->pnt->rc = p;
	}
	if ( mark ) rt = p;
	up( q ); up( p );
}
inline void zag( Node *p, Node *&rt ){
	int mark = 0;
	Node *q = p->pnt;
	if ( q == rt ) mark = 1;
	q->rc = p->lc;
	if ( p->lc ) p->lc->pnt = q;
	p->lc = q;
	p->pnt = q->pnt;
	q->pnt = p;
	if ( p->pnt ){
		if ( p->pnt->lc == q ) p->pnt->lc = p;
		else p->pnt->rc = p;
	}
	if ( mark ) rt = p;
	up( q ); up( p );
}
inline void splay( Node *p, Node *&rt ){
	Node *q;
	down( p );
	while ( p != rt ){
		if ( p->pnt == rt ){
			if ( p->pnt->lc == p ) zig( p, rt );
			else zag( p, rt );
			break;
		}
		else {
			q = p->pnt->pnt;
			if ( q->lc == p->pnt ){
```

```
                        if ( p->pnt->lc == p ){
                                zig( p->pnt, rt ); zig( p, rt );
                        }
                        else {
                                zag( p, rt ); zig( p, rt );
                        }
                }
                else {
                        if ( p->pnt->lc == p ){
                                zig( p, rt ); zag( p, rt );
                        }
                        else {
                                zag( p->pnt, rt ); zag( p, rt );
                        }
                }
        }
    }
    up( rt );
}
inline Node *selectKth( int x ){
    if ( root == NULL ) return NULL;
    Node *p = root;
    int y, z = x;
    while ( 1 ){
        down( p );
        if ( p->lc == NULL ) y = 0;
        else y = p->lc->cnt;
        if ( y+1 == x ) break;
        if ( x > y ) p = p->rc;
        else p = p->lc;
        if ( x > y ) x -= (y+1);
    }
    return p;
}
inline Node *get_max(Node *rt){
    Node *p = rt;
    down(p);
    while (p->rc){
        p = p->rc; down(p);
    }
    return p;
}
inline Node *get_min(Node *rt){
    Node *p = rt;
    down(p);
    while (p->lc){
        p = p->lc; down(p);
    }
    return p;
}
inline void flip(int l, int r){
    splay(selectKth(r+1), root);
    splay(selectKth(l-1), root->lc);
    if (root->lc && root->lc->rc) root->lc->rc->rev ^= 1;
}
inline void cut(int l, int r, int c){
    Node *p, *q;
    splay(selectKth(r+1), root);
    splay(selectKth(l-1), root->lc);
    p = root->lc->rc;
```

```
        root->lc->rc = NULL;
        p->pnt = NULL;
        up(root->lc);
        up(root);
        splay(selectKth(c), root);
        q = root->rc;
        root->rc = p;
        p->pnt = root;
        up(root);
        splay(get_max(root), root);
        root->rc = q;
        if (q != NULL) q->pnt = root;
        up(root);
}
inline void insert( int _key ){
        Node *p = root, *q = root;
        if ( root == NULL ){
                root = newNode( _key );
                return ;
        }
        while ( 1 ){
                q = p;
                if ( p->key < _key ){
                        if ( p->rc == NULL ){
                                p->rc = newNode( _key );
                                p = p->rc;
                                break;
                        }
                        else p = p->rc;
                }
                else {
                        if ( p->lc == NULL ){
                                p->lc = newNode( _key );
                                p = p->lc;
                                break;
                        }
                        else p = p->lc;
                }
        }
        p->pnt = q;
        splay( p, root );
        root->pnt = NULL;
}
inline void remove( int _key ){
        Node *p, *q;
        if ( root == NULL ) return ;
        splay( find( _key ), root );
        root->pnt = NULL;
        if ( root->lc ){
                q = root->rc;
                splay( get_max( root->lc ), root->lc );
                p = root->lc, q = root->rc;
                p->rc = q;
                if ( q ) q->pnt = p;
                root = p;
        }
        else root = root->rc;
        if ( root != NULL ){
                root->pnt = NULL;
                up( root );
```

```
            }
        }
}tree;

int main(){
        char str[16];
        int l, r, c, ca = 0;
        while (scanf("%d%d", &n, &m) != EOF){
                if (n <= 0 && m <= 0) break;
                tree.init(n);
                int i = 0;
                while ( m-- ){
                        i++;
                        scanf( "%s", str );
                        if ( str[0] == 'F' ){
                                scanf("%d%d", &l, &r);
                                l++, r++;
                                tree.flip(l, r);
                        }
                        else if ( str[0] == 'C' ){
                                scanf("%d%d%d", &l, &r, &c);
                                if (c > n-(r-l+1) ) while ( 1 );
                                l++, r++, c++;
                                tree.cut(l, r, c);
                        }
                }
                SS = 0;
                tree.print(tree.root);
                for (l = 1; l < n; l++) printf("%d ", seq[l]);
                printf("%d\n", seq[n]);
        }
}
```

5.8 Dynamic Tree: cut-link tree

```
//checked by hdu3216 spoj4155
#define MAXN 10008
#define isRoot(t) (lc[fa[t]] != t && rc[fa[t]] != t) //t是否为splay根
int n, m, stk[MAXN], size;
int black[MAXN], white[MAXN], val[MAXN];
int sub[MAXN], lc[MAXN], rc[MAXN], fa[MAXN], mark[MAXN];

inline void down(int x){
        if (mark[x]){
                swap(lc[x], rc[x]);
                if (lc[x]) mark[lc[x]] ^= 1;
                if (rc[x]) mark[rc[x]] ^= 1;
                mark[x] = 0;
        }
}

inline void update(int x){
        black[x] = white[x] = 0;
        if (lc[x]){
                black[x] += black[lc[x]]; white[x] += white[lc[x]];
        }
        if (rc[x]){
                black[x] += black[rc[x]]; white[x] += white[rc[x]];
        }
        if (val[x]) black[x]++;
        else white[x]++;
```

```
}

//zag: left rotation
void left(int x, int y){
	rc[y] = lc[x];
	fa[rc[x]] = lc[x] = y;
	if (y == lc[fa[y]]) lc[fa[y]] = x;
	else if (rc[fa[y]] == y) rc[fa[y]] = x;
	fa[x] = fa[y];
	fa[y] = x;
	update(y); update(x);
}

//zig: right rotation
void right(int x, int y){
	lc[y] = rc[x];
	fa[lc[x]] = rc[x] = y;
	if (y == lc[fa[y]]) lc[fa[y]] = x;
	else if (rc[fa[y]] == y) rc[fa[y]] = x;
	fa[x] = fa[y];
	fa[y] = x;
	update(y); update(x);
}

void splay(int x){
	int f, ff;
	size = 0;
	stk[size++] = x;
	for (int u = x; !isRoot(u); u = fa[u]) stk[size++] = fa[u];
	for ( ; size > 0; size--) down(stk[size-1]);
	while (!isRoot(x)){
		f = fa[x];
		ff = fa[f];
		if (isRoot(f)){
			if (x == lc[f]) right(x, f);
			else left(x, f);
		}
		else {
			if (f == lc[ff]){
				if (x == lc[f]){
					right(f, ff); right(x, f);
				}
				else {
					left(x, f); right(x, ff);
				}
			}
			else {
				if (x == lc[f]){
					right(x, f); left(x, ff);
				}
				else {
					left(f, ff); left(x, f);
				}
			}
		}
	}
	update(x);
}

int expose(int u){
```

```
	int v = 0;
	while (u){
		splay(u);
		rc[u] = v;
		update(v = u);
		u = fa[u];
	}
	for ( ; lc[v]; v = lc[v]);
	return v;
}

void modify(int u, int vv){
	splay(u);
	val[u] = vv;
	update(u);
}

bool join(int u, int v){
	int x = expose(u), y = expose(v);
	if (x == y) return false;
	splay(v);
	rc[v] = 0; mark[v] = 1; fa[v] = u;
	return true;
}
bool remove(int u, int v){
	int x = expose(u); splay(u);
	int y = expose(v); splay(v);
	if (x != y) return false;
	fa[u] = 0;
	if (lc[v] == u) lc[v] = 0;
	if (rc[v] == u) rc[v] = 0;
	return true;
}

void query(int u, int v){
	int x = expose(u), y = expose(v);
	if (x != y){
		puts("-1");
		return ;
	}
	for (x = u, y = 0; x > 0; x = fa[x]){
		if (splay(x), !fa[x]){
			int b = black[rc[x]] + black[y];
			int w = white[rc[x]] + white[y];
			if (val[x]) b++;
			else w++;
			printf("%d %d\n", b, w);
			return ;
		}
		rc[x] = y;
		update(y = x);
	}
}

void init(int s){
	memset(lc, 0, sizeof(int)*s); memset(rc, 0, sizeof(int)*s);
	memset(fa, 0, sizeof(int)*s); memset(mark, 0, sizeof(int)*s);
}

int main(){
```

```
	int i, u, v;
	char str[16];
	while (scanf("%d%d", &n, &m) != EOF){
		if (n == 0 && m == 0) break;
		init(n+1);
		for (i = 1; i <= n; i++){
			scanf("%s", str);
			if (str[0] == 'B') val[i] = 1;
			else val[i] = 0;
		}
		while (m--){
			scanf("%s", str);
			if (str[0] == 'a'){
				scanf("%d%d", &u, &v);
				join(u, v);
			}
			else if (str[0] == 'd'){
				scanf("%d%d", &u, &v);
				remove(u, v);
			}
			else if (str[0] == 's'){
				scanf("%d%s", &u, str);
				modify(u, (str[0] == 'B' ? 1 : 0));
			}
			else if (str[0] == 'q'){
				scanf("%d%d", &u, &v);
				query(u, v);
			}
		}
	}
}
```

5.9 Dancing link for exact cover

```
//checked by pku3740
#include <cstdio>
#include <cstring>
using namespace std;
#define N 1000

struct Node {
	int rn, cn;
	Node *l, *r, *u, *d;
}head, row[N], col[N], node[N*N];

int n, size_n, count[N];

void dance_init(){
	size_n = 0;
	memset(count, 0, sizeof(count));
	head.rn = head.cn = 0;
	head.l = head.r = head.u = head.d = &head;
	for (int i = 0; i < n; i++){
		col[i].cn = i;
		col[i].l = head.l;
		head.l = &col[i];
		col[i].r = &head;
		col[i].l->r = &col[i];
		col[i].r->l = &col[i];
		col[i].u = col[i].d = &col[i];
	}
```

```
	for (int i = 0; i < n; i++){
		row[i].rn = i;
		row[i].u = head.u;
		head.u = &row[i];
		row[i].d = &head;
		row[i].u->d = &row[i];
		row[i].d->u = &row[i];
		row[i].l = row[i].r = &row[i];
	}
}

void removerowhead(){
	for ( int i = 0; i < n; i++ ){
		row[i].l->r = row[i].r;
		row[i].r->l = row[i].l;
	}
}

void link(int x, int y){
	count[y]++;
	Node *tmp = &node[size_n++];
	tmp->rn = x;
	tmp->cn = y;
	tmp->u = &col[y];
	tmp->d = col[y].d;
	col[y].d->u = tmp;
	col[y].d = tmp;
	tmp->l = &row[x];
	tmp->r = row[x].r;
	row[x].r->l = tmp;
	row[x].r = tmp;
}

void remove(int coln){
	col[coln].l->r = col[coln].r;
	col[coln].r->l = col[coln].l;
	for (Node *cur = col[coln].d; cur != &col[coln]; cur = cur->d){
		for (Node *tmp = cur->r; tmp != cur; tmp = tmp->r){
			count[tmp->cn]--;
			tmp->u->d = tmp->d;
			tmp->d->u = tmp->u;
		}
	}
}

void resume(int coln){
	for (Node *cur = col[coln].d; cur != &col[coln]; cur = cur->d){
		for (Node *tmp = cur->l; tmp != cur; tmp = tmp->l){
			tmp->u->d = tmp;
			tmp->d->u = tmp;
			count[tmp->cn]++;
		}
	}
	col[coln].l->r = &col[coln];
	col[coln].r->l = &col[coln];
}

bool solve(int k){
	if (head.r == &head)
		return true;
```

```
	int id, low = N;
	for ( Node *cur = head.r; cur != &head; cur = cur->r ){
		if ( low > count[cur->cn] ){
			low = count[cur->cn];
			id = cur->cn;
		}
	}
	if ( low == 0 ) return false;
	remove( id );
	for (Node *cur = col[id].d; cur != &col[id]; cur = cur->d){
		for (Node *tmp = cur->r; tmp != cur; tmp = tmp->r)
			remove(tmp->cn);
		if (solve(k+1)) return true;
		for (Node *tmp = cur->l; tmp != cur; tmp = tmp->l)
			resume(tmp->cn);
	}
	resume(id);
	return false;
}
```

5.10 Dancing link for multi cover

```
#define N 300
#define EPS 5e-8
int n, m, limit, size_n, cnt[N];

struct Node {
	int rn, cn;
	Node *l, *r, *u, *d;
}head, row[N], col[N], node[N*N];

void dance_init(){
	size_n = 0;
	memset(cnt, 0, sizeof(cnt));
	head.rn = head.cn = 0;
	head.l = head.r = head.u = head.d = &head;
	for (int i = 0; i < n; i++){
		col[i].cn = i;
		col[i].l = head.l;
		head.l = &col[i];
		col[i].r = &head;
		col[i].l->r = &col[i];
		col[i].l->r = &col[i];
		col[i].u = col[i].d = &col[i];
	}
	for ( int i = 0; i < m; i++ ){
		row[i].rn = i;
		row[i].u = head.u;
		head.u = &row[i];
		row[i].d = &head;
		row[i].u->d = &row[i];
		row[i].d->u = &row[i];
		row[i].l = row[i].r = &row[i];
	}
}

void removerowhead(){
	for (int i = 0; i < m; i++){
		row[i].l->r = row[i].r;
		row[i].r->l = row[i].l;
	}
```

```
}

void remove(Node *p){
	for (Node *i = p->d; i != p; i = i->d){
		i->r->l = i->l;
		i->l->r = i->r;
	}
}

void resume(Node *p){
	for (Node *i = p->u; i != p; i = i->u){
		i->r->l = i;
		i->l->r = i;
	}
}

void link( int x, int y ){
	cnt[y]++;
	Node *tmp = &node[size_n++];
	tmp->rn = x;
	tmp->cn = y;
	tmp->u = &col[y];
	tmp->d = col[y].d;
	col[y].d->u = tmp;
	col[y].d = tmp;
	tmp->l = &row[x];
	tmp->r = row[x].r;
	row[x].r->l = tmp;
	row[x].r = tmp;
}

int h(){
	int ret = 0;
	Node *i, *j, *k;
	bool visit[N] = { 0 };
	for (i = head.r; i != &head; i = i->r){
		if (visit[i->cn]) continue;
		visit[i->cn] = 1;
		ret++;
		for ( j = i->d; j != i; j = j->d )
			for ( k = j->r; k != j; k = k->r )
				visit[k->cn] = 1;
	}
	return ret;
}

int dfs(int dep){
	if (dep >= limit || dep+h() >= limit) return 0;
	if (head.r == &head) return 1;
	int low = 1<<10, id;
	Node *i, *j;
	for ( i = head.r; i != &head; i = i->r ){
		if ( cnt[i->cn] < low ){
			low = cnt[i->cn];
			id = i->cn;
		}
	}
	if ( cnt[id] == 0 ) return 0;
	for ( i = col[id].d; i != &col[id]; i = i->d ){
		remove( i );
```

```
            for ( j = i->r; j != i; j = j->r ) remove( j );
            if ( dfs( dep+1 ) ) return 1;
            for ( j = i->l; j != i; j = j->l ) resume( j );
            resume( i );
        }
        return 0;
}
```

5.11 LCA+RMQ

```
//checked by zju3195
#define N 50000
#define INF 1000000000
#define MIN(x,y) ((x)<(y)?(x):(y))
int n, q, mark = 0;
int mp[N], size;
int visit[N], dis[N];
int query[N<<1][3];
int cnt, pos[N], level[N], _level[N];
int seq[N<<1], sq;
int h[N<<1][18];
struct Edge{
        int v, w, next;
}edge[N<<1];

void init(){
        memset(mp, -1, sizeof(mp));
        memset(visit, 0, sizeof(visit));
        size = sq = cnt = 0;
}

inline void add_edge(int u, int v, int w){
        edge[size].v = v; edge[size].w = w;
        edge[size].next = mp[u];
        mp[u] = size;
        size++;
}

void dfs(int u, int d){
        int i, v;
        visit[u] = 1;
        _level[cnt] = u;
        level[u] = cnt++;
        dis[u] = d;
        pos[u] = sq;
        seq[sq++] = level[u];
        for (i = mp[u]; i != -1; i = edge[i].next){
                v = edge[i].v;
                if (visit[v]) continue;
                dfs(v, d+edge[i].w);
                seq[sq++] = level[u];
        }
}

void rmq_init(){
        int i, j, l;
        for (i = 0; i < sq; i++) h[i][0] = seq[i];
        for (j = l = 1; l*2 <= sq; j++, l <<= 1){
                for (i = 0; i <= sq+1-l*2; i++)
                        h[i][j] = MIN(h[i][j-1], h[i+l][j-1]);
        }
```

```
}

inline int rmq_query(int start, int end){
	int j = 0, l = 1;
	while (2*l <= end-start+1){
		j++;
		l <<= 1;
	}
	return MIN(h[start][j], h[end-l+1][j]);
}

inline int calc(int x, int y){
	int ans, p, q, u, v;
	u = pos[x]; v = pos[y];
	if (u > v) swap(u, v);
	p = _level[rmq_query(u, v)];
	ans = dis[x]+dis[y]-2*dis[p];
	return ans;
}

void solve(){
	int i, j, ans, x, y, z;
	dfs(0, 0);
	rmq_init();
	if (mark) puts("");
	mark = 1;
	for (i = 0; i < q; i++){
		x = query[i][0];
		y = query[i][1];
		z = query[i][2];
		ans = calc(x, y) + calc(y, z) + calc(z, x);
		printf("%d\n", ans/2);
	}
}

bool input(){
	if (scanf("%d", &n) == EOF) return false;
	int i, j, u, v, w;
	init();
	for (i = 0; i < n-1; i++){
		scanf("%d%d%d", &u, &v, &w);
		add_edge(u, v, w); add_edge(v, u, w);
	}
	scanf("%d", &q);
	for (i = 0; i < q; i++)
		for (j = 0; j < 3; j++)
			scanf("%d", &query[i][j]);
	return true;
}
```

5.12 树套树

不用模板了，其实就是线段树+set在节点上

5.13 线段树内存模拟

```
//checked by 3358，懒操作，离散化
#define N 65536
int n, m;
struct Node{
```

```
	int l, r, flag;
	int lc, rc, cnt;
	inline void set(int _flag){
		flag = _flag;
		if (_flag == 0) lc = rc = cnt = r-l+1;
		else if (_flag ==1) lc = rc = cnt = 0;
	}
}tree[N*4];
struct Block{
	int l, r;
	Block(){ }
	Block(int _l, int _r): l(_l), r(_r) { }
	bool operator<(const Block &b) const{ return l < b.l; }
	bool operator==(const Block &b) const{ return l == b.l && r == b.r; }
};
vector<Block> mem;
vector<Block>::iterator itr;

void build(int id, int l, int r){
	tree[id].l = l; tree[id].r = r;
	tree[id].set(0);
	if (l == r) return ;
	int mid = (l+r)>>1;
	build(id<<1, l, mid);
	build((id<<1)+1, mid+1, r);
}

inline void down(int id){
	if (tree[id].flag == -1) return;
	if (tree[id].flag == 0){
		tree[id<<1].set(0);
		tree[(id<<1)+1].set(0);
	}
	if (tree[id].flag == 1){
		tree[id<<1].set(1);
		tree[(id<<1)+1].set(1);
	}
}

inline void up(int id){
	if (tree[id<<1].flag == 0 && tree[(id<<1)+1].flag == 0){
		tree[id].set(0);
		return;
	}
	if (tree[id<<1].flag == 1 && tree[(id<<1)+1].flag == 1){
		tree[id].set(1);
		return;
	}
	tree[id].flag = -1;
	tree[id].lc = tree[id<<1].lc;
	if (tree[id<<1].flag == 0) tree[id].lc += tree[(id<<1)+1].lc;
	tree[id].rc = tree[(id<<1)+1].rc;
	if (tree[(id<<1)+1].flag == 0) tree[id].rc += tree[id<<1].rc;
	tree[id].cnt = max(max(tree[id].lc, tree[id].rc), \
			max(tree[id<<1].cnt, tree[(id<<1)+1].cnt));
	tree[id].cnt = max(tree[id].cnt, tree[id<<1].rc+tree[(id<<1)+1].lc);
}

int find(int id, int d){
	if (d > tree[id].cnt) return 0;
```

```
	if (tree[id].flag == 0 && d <= tree[id].cnt) return tree[id].l;
	down(id);
	if (tree[id<<1].cnt >= d) return find(id<<1, d);
	if (tree[id<<1].rc+tree[(id<<1)+1].lc >= d)
		return tree[id<<1].r-tree[id<<1].rc+1;
	if (tree[(id<<1)+1].cnt >= d) return find((id<<1)+1, d);
	return 0;
}

void insert(int id, int ll, int rr){
	if (ll <= tree[id].l && tree[id].r <= rr){
		tree[id].set(1);
		return ;
	}
	int mid = (tree[id].l+tree[id].r)>>1;
	down(id);
	if (ll <= mid) insert(id<<1, ll, rr);
	if (rr > mid) insert((id<<1)+1, ll, rr);
	up(id);
}

void remove(int id, int ll, int rr){
	if (ll <= tree[id].l && tree[id].r <= rr){
		tree[id].set(0);
		return ;
	}
	int mid = (tree[id].l+tree[id].r)>>1;
	down(id);
	if (ll <= mid) remove(id<<1, ll, rr);
	if (rr > mid) remove((id<<1)+1, ll, rr);
	up(id);
}

inline int bsearch(int pos){
	if (mem.size() == 0) return -1;
	int low = 0, up = mem.size()-1, mid;
	while (low < up){
		mid = (low+up)>>1;
		if (mem[mid].l <= pos && pos <= mem[mid].r) return mid;
		if (pos < mem[mid].l) up = mid-1;
		else if (pos > mem[mid].r) low = mid+1;
	}
	if (mem[up].l <= pos && pos <= mem[up].r) return up;
	return -1;
}

int main(){
	Block temp;
	int d, pos;
	char str[16];
	while (scanf("%d%d", &n, &m) != EOF){
		build(1, 1, n);
		mem.clear();
		while (m--){
			scanf("%s", str);
			if (str[0] == 'N'){
				scanf("%d", &d);
				pos = find(1, d);
				if (pos == 0) puts("Reject New");
				else {
```

```
                                temp = Block(pos, pos+d-1);
                                insert(1, pos, pos+d-1);
                                printf("New at %d\n", pos);
                                if (mem.size() == 0)
                                        mem.push_back(temp);
                                else {
                                        itr = lower_bound(mem.begin(), mem.end(), temp);
                                        if (itr == mem.end())
                                                mem.push_back(temp);
                                        else
                                                mem.insert(itr, temp);
                                }
                        }
                }
                else if (str[0] == 'F'){
                        scanf("%d", &d);
                        pos = bsearch(d);
                        if (pos == -1)
                                puts("Reject Free");
                        else {
                                printf("Free from %d to %d\n", mem[pos].l, mem[pos].r);
                                remove(1, mem[pos].l, mem[pos].r);
                                mem.erase(mem.begin()+pos);
                        }
                }
                else if (str[0] == 'G'){
                        scanf("%d", &d);
                        if (d > mem.size())
                                puts("Reject Get");
                        else
                                printf("Get at %d\n", mem[d-1].l);
                }
                else if (str[0] == 'R'){
                        tree[1].set(0);
                        mem.clear();
                        puts("Reset Now");
                }
        }
        puts("");
    }
}
```

6. 图论问题

6.1 找两棵不相交的生成树

```
//checked by hdu3267
#define N 32
struct Edge{
	int u, v, id;
	Edge(){ }
	Edge(int _u, int _v, int _id): u(_u), v(_v), id(_id) {}
	bool operator==(const Edge &e) const{ return id == e.id; }
	bool operator!=(const Edge &e) const{ return !(*this == e); }
}tree[N], edge[N];
int n, m, size, comp;
int set1[N], set2[N], visit[N], mark[N];
vector<int> mp[N];

//先dfs一棵树出来，size记录边数
void dfs(int u){
	int i, v;
	visit[u] = 1;
	for (i = 0; i < mp[u].size(); i+=2){
		v = mp[u][i];
		if (visit[v]) continue;
		tree[size++] = Edge(u, v, mp[u][i+1]);
		dfs(v);
	}
}

//并查集
inline int find_set(int u, int set[]){
	if (set[u] == u) return u;
	return (set[u] = find_set(set[u], set));
}

inline void join_set(int u, int v, int set[]){
	int x = find_set(u, set), y = find_set(v, set);
	if (x != y) set[y] = x;
}

//判断e是否连通
inline int connect1(Edge e[], int set[], int vn, int en){
	int i, j;
	for (i = 0; i < vn; i++) set[i] = i;
	for (i = 0; i < en; i++) join_set(e[i].u, e[i].v, set);
	for (i = 1; i < vn; i++)
		if (set[i] != set[0])
			return 0;
	return 1;
}

//判断剩余边集是否连通
inline int connect2(Edge e[], int set[], int vn, int en){
	int i, j = 1;
	for (i = 0; i < vn; i++) set[i] = i;
	for (i = 0; i < en; i++){
		if (mark[i] == 0) continue;
		join_set(e[i].u, e[i].v, set);
	}
	for (i = 1; i < vn; i++)
```

```
            if (find_set(i, set) == find_set(0, set))
                  j++;
      return j;
}

bool search(int dep){
      int i, j, k, c, cc;
      Edge e1, e2;
      //记录与0点连通个数
      c = connect2(edge, set2, n, m);
      if (c == n) return true;
      for (i = 0; i < size; i++){
            for (j = 0; j < m; j++){
                  if (!mark[j]) continue;
                  e1 = tree[i]; e2 = edge[j];
                  for (k = 0; k < m && edge[k] != e1; k++);
                  tree[i] = e2;
                  mark[j] = 0; mark[k] = 1;
                  if (connect1(tree, set1, n, size)){
                        cc = connect2(edge, set2, n, m);
                        if (cc > c){
                              if (search(dep+1)) return true;
                        }
                  }
                  tree[i] = e1;
                  mark[j] = 1; mark[k] = 0;
            }
      }
      return false;
}

int main(){
      int i, j, u, v;
      while (scanf("%d%d", &n, &m) != EOF){
            if (n == -1 && m == -1) break;
            for (i = 0; i < n; i++) mp[i].clear();
            //加边
            for (i = 0; i < m; i++){
                  scanf("%d%d", &u, &v);
                  mp[u].push_back(v); mp[u].push_back(i);
                  mp[v].push_back(u); mp[v].push_back(i);
                  edge[i] = Edge(u, v, i);
            }
            memset(visit, 0, sizeof(visit));
            size = 0;
            dfs(0);
            if (size < n-1){
                  puts("NO");
                  continue;
            }
            for (i = 0; i < m; i++) mark[i] = 1;
            //边分成两个集合
            for (i = 0; i < size; i++)
                  for (j = 0; j < m; j++)
                        if (tree[i] == edge[j])
                              mark[j] = 0;
            if (search(0)) puts("YES");
            else puts("NO");
      }
}
```

6.2 一般图匹配

```
/*
带花树开花算法
1、贪心初始化
2、找可增广路．交错树．两个外点有边，形成花。
3、缩圈，边两端点在圈B上的，收缩掉。仅一个端点属于B的变成以VB 为一端的边。
*/
#include <stdio.h>
#include <string.h>

#define MAXV 1008
#define MAXE 200008
#define SBN (sizeof(bool) * (n + 1))
#define SIN (sizeof(int) * (n + 1))

struct Edge {
    int v, next;
    Edge(){}
    Edge(int _v, int _next):
        v(_v), next(_next){}
};

struct Graph {
    int n, match[MAXV], que[MAXV], pre[MAXV], base[MAXV];
    bool flag[MAXV], inBlossom[MAXV], inPath[MAXV];
    Edge edge[MAXE];
    int head[MAXV], size;
    inline void initg(int _n) {
        size = 0;
        n = _n;
        memset(head, -1, SIN);
    }
    inline void addEdge(int u, int v) {
        if (u == v) return;
        edge[size] = Edge(v, head[u]);
        head[u] = size++;
        edge[size] = Edge(u, head[v]);
        head[v] = size++;
    }
    int MaxMatch() {
        memset(match, -1, SIN);
        int i, j, ans = 0;
        for (i = 1; i <= n; ++i) {
            if (match[i] != -1) continue;
            for (j = head[i]; j != -1 && match[i] == -1; j = edge[j].next)
                if (match[edge[j].v] == -1) {
                    match[edge[j].v] = i;
                    match[i] = edge[j].v;
                    ans++;
                }
        }
        for (i = 1; i <= n; ++i)
            if (match[i] == -1)
                ans += bfs(i);
        return ans;
    }
    int bfs(int p) {      //寻找可增广路
        int i, j, u, v, b, front, rear;
        memset(pre, -1, SIN);
```

```
    memset(flag, 0, SBN);
    for (i = 1; i <= n; ++i)
        base[i] = i;
    front = rear = 0;
    que[rear++] = p;
    flag[p] = 1;
    while (front != rear) {
        u = que[front++];
        for (i = head[u]; i != -1; i = edge[i].next) {
            v = edge[i].v;
            if (base[u] != base[v] && v != match[u]) {
                if (v == p || (match[v] != -1 && pre[match[v]] != -1)) {
                    b = contract(u, v);
                    for (j = 1; j <= n; ++j) {
                        if (inBlossom[base[j]]) {
                            base[j] = b;
                            if (flag[j] == 0) {
                                flag[j] = 1;
                                que[rear++] = j;
                            }
                        }
                    }
                }
                else if (pre[v] == -1) {
                    pre[v] = u;
                    if (match[v] == -1) {
                        argument(v);
                        return 1;
                    }
                    else {
                        que[rear++] = match[v];
                        flag[match[v]] = 1;
                    }
                }
            }
        }
    }
    return 0;
}
void argument(int u) { //增广
    int v, k;
    while (u != -1) {
        v = pre[u];
        k = match[v];
        match[u] = v;
        match[v] = u;
        u = k;
    }
}
void changeBlossom(int b, int u) { //哪些点属于当前圈
    int v;
    while (base[u] != b) {
        v = match[u];
        inBlossom[base[v]] = inBlossom[base[u]] = true;
        u = pre[v];
        if (base[u] != b)
            pre[u] = v;
    }
}
int contract(int u, int v) { //缩圈
```

```
        memset(inBlossom, 0, SBN);
        int b = findBase(base[u], base[v]);
        changeBlossom(b, u);
        changeBlossom(b, v);
        if (base[u] != b)
            pre[u] = v;
        if (base[v] != b)
            pre[v] = u;
        return b;
    }
    int findBase(int u, int v) { //属于哪个圈
        memset(inPath, 0, SBN);
        while (true) {
            inPath[u] = true;
            if (match[u] == -1)
                break;
            u = base[pre[match[u]]];
        }
        while(!inPath[v])
            v = base[pre[match[v]]];
        return v;
    }
}GP;
```

6.3 弦图判定

```
//checked by 1972
#define N 1024
int n, m;
int mp[N], size;
int mark[N], deg[N], g[N][N], seq[N];

int isChordal(){
       int i, j, p, q;
       memset(mark, 0, sizeof(mark));
       memset(deg, 0, sizeof(deg));
       for (i = 1; i <= n; i++){
              p = q = -1;
              for (j = 1; j <= n; j++)
                     if (!mark[j] && (p == -1 || deg[j] > deg[p]))
                            p = j;
              mark[p] = 1;
              seq[i] = p;
              for (j = i-1; j > 0; j--)
                     if (g[p][seq[j]]){
                            if (q < 0) q = seq[j];
                            else if (!g[q][seq[j]]) return 0;
                     }
              for (j = 1; j <= n; j++)
                     if (!mark[j] && g[p][j])
                            deg[j]++;
       }
       return 1;
}

int main(){
       int u, v;
       while (scanf("%d%d", &n, &m) != EOF){
              if (n == 0 && m == 0) break;
              size = 0;
              memset(mp, -1, sizeof(mp));
```

```
            memset(g, 0, sizeof(g));
            while (m--){
                  scanf("%d%d", &u, &v);
                  g[u][v] = g[v][u] = 1;
            }
            if (isChordal()) puts("Perfect\n");
            else puts("Imperfect\n");
      }
}
```

6.4 无根树的同构

```
//checked by utsc1117
//首先topsort，然后dfs做hash；如果是有根树直接dfs
typedef long long LL;
#define N 1024
const LL MOD = 30007;
const LL MUL = 1911111110;
int vn;
int deg[N], seq[N], visit[N], leave[N], que[N];
LL hash[N], h1[2], h2[2];
vector<int> tree[N];

//比较两个节点的hash值
inline int cmp(int a, int b){ return hash[a] < hash[b]; }

//递归做hash
void dfs(int u, int pnt){
      int i, v, c = 0;
      visit[u] = 1;
      for (i = 0; i < tree[u].size(); i++){
            v = tree[u][i];
            if (v == pnt || visit[v]) continue;
            dfs(v, u);
            c++;
      }
      sort(tree[u].begin(), tree[u].end(), cmp);
      if (c == 0){
            hash[u] = 1; return ;
      }
      else{
            LL h = 1908;
            for (i = 0; i < tree[u].size(); i++){
                  v = tree[u][i];
                  if (v == pnt) continue;
                  h = ((h * MUL) ^ hash[v]) % MOD;
            }
            hash[u] = h;
      }
}
//seq数组记录出队列顺序，leave数组记录时间
void topsort(){
      int i, u, v, s = 0, head = 0, tail = 0;
      if (vn == 1){
            seq[1] = 1;
            return ;
      }
      memset(visit, 0, sizeof(visit));
      for (i = 1; i <= vn; i++) deg[i] = tree[i].size();
      for (i = 1; i <= vn; i++){
```

```
		if (deg[i] == 1){
			visit[i] = 1;
			leave[i] = 0;
			que[tail++] = i;
		}
	}
	while (head < tail){
		u = que[head++];
		seq[++s] = u;
		for (i = tree[u].size()-1; i >= 0; i--){
			v = tree[u][i];
			if (visit[v]) continue;
			deg[v]--;
			if (deg[v] == 1){
				que[tail++] = v;
				visit[v] = 1;
				leave[v] = leave[u]+1;
			}
		}
	}
}
//判断是否同构
bool check(){
	int i, j;
	for (i = 0; i < 2; i++){
		if (h1[i] == -1) continue;
		for (j = 0; j < 2; j++){
			if (h2[j] == -1) continue;
			if (h1[i] == h2[j]) return true;
		}
	}
	return false;
}

int main(){
	int ca, i, u, v;
	scanf("%d", &ca);
	while (ca--){
		scanf("%d", &vn);
		for (i = 1; i <= vn; i++) tree[i].clear();
		for (i = 1; i < vn; i++){
			scanf("%d%d", &u, &v);
			tree[u].push_back(v); tree[v].push_back(u);
		}
		topsort();
		memset(visit, 0, sizeof(visit));
		dfs(seq[vn], -1);
		h1[0] = hash[seq[vn]];
		h1[1] = -1;
		if (vn > 1 && leave[seq[vn]] == leave[seq[vn-1]]){
			memset(visit, 0, sizeof(visit));
			dfs(seq[vn-1], -1);
			h1[1] = hash[seq[vn-1]];
		}
		for (i = 1; i <= vn; i++) tree[i].clear();
		for (i = 1; i < vn; i++){
			scanf("%d%d", &u, &v);
			tree[u].push_back(v); tree[v].push_back(u);
		}
		topsort();
```

```
                memset(visit, 0, sizeof(visit));
                dfs(seq[vn], -1);
                h2[0] = hash[seq[vn]];
                h2[1] = -1;
                if (vn > 1 && leave[seq[vn]] == leave[seq[vn-1]]){
                        memset(visit, 0, sizeof(visit));
                        dfs(seq[vn-1], -1);
                        h2[1] = hash[seq[vn-1]];
                }
                if (check()) puts("same");
                else puts("different");
        }
}
```

6.5 曼哈顿距离最小生成树

```
/*
    曼哈顿距离最小生成树
    题意：给出n<=100000个点坐标，求MST。边权为点之间的曼哈顿距离
    题目给出重要提示：对于一个点0，在45度角度的范围内，最多只有一条连出去的边。
    拓展这个提示，对于一个点0，45°的范围内最多只有1条边，8个方向只有8条
    8n条边用kruskal可以解决
    由于是无向图，只需要找到4个方向即可。通过旋转90°，关于y=x翻转做到这四个方向区域
    关于原点对称的区域没有访问过。
    那怎么快速找到45°范围内距离(xi,yi)最小的点呢?
    由于是45°范围的点，有xj>xi,yj>yi，距离为xj+yj-xi-yi  所以用线段树找x+y最小的点
    而且需要是在45°范围内，还有y>yi
    先按照w=y-x从小到大排序，w相同的y大的优先
    检查每个点，在线段树中找到距离最小的点（没有时为INF） 线段树中找 y 比他大的
    然后再插入
*/
// CII 3662 Another Minimum Spanning Tree

#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
using namespace std;
#define N 100005
#define BIG 100000000
const int INF = 1 << 30;
struct POINT {
        int x, y;
        void get() {
                scanf("%d%d", &x, &y);
        }
        void print() {
                printf("%d %d\n", x, y);
        }
        POINT() {
        }
        POINT(int x, int y) :
                x(x), y(y) {
        }
};

POINT P[N];
int n, m;
int dis(int a, int b) {
        return abs(P[a].x - P[b].x) + abs(P[a].y - P[b].y);
}
```

```
// MST Kruskal
struct EDGE {
	int x, y, w;
	EDGE() {
	}
	EDGE(int x, int y, int w) :
		x(x), y(y), w(w) {
	}
	bool operator <(const EDGE &e) const {
		return w < e.w;
	}
};
EDGE E[N * 10];
int fa[N];
int find(int x) {
	if (fa[x] != x) fa[x] = find(fa[x]);
	return fa[x];
}
long long kruskal() {
	int i, link = 1;
	long long ans = 0;
	sort(E, E + m);
	for (i = 0; i <= n; i++)
		fa[i] = i;
	for (i = 0; i < m && link < n; i++) {
		int x = E[i].x, y = E[i].y;
		x = find(x), y = find(y);
		if (x == y) continue;
		fa[x] = y;
		link++;
		ans += E[i].w;
	}
	return ans;
}
// End of Kruskal
// Sort and query, get all the edges in 45 degree
struct NODE {
	int v, p, l, r;
};
NODE T[N * 5];
int idx[N], yy[N], ny[N];
inline bool cmp(int i, int j) {    //按 w=y-x从小到大排, 然后按y从大到小排
	int v1 = P[i].y - P[i].x;
	int v2 = P[j].y - P[j].x;
	if (v1 == v2) return ny[i] > ny[j];
	return v1 < v2;
}
void build(int p, int l, int r) {
	T[p].l = l, T[p].r = r, T[p].v = INF, T[p].p = -1;
	if (l == r) return;
	int mid = (l + r) >> 1;
	build(p << 1, l, mid);
	build((p << 1) | 1, mid + 1, r);
}
void insert(int p, int i) {
	if (ny[i] < T[p].l || ny[i] > T[p].r) return;
	if (P[i].x + P[i].y < T[p].v) {
		T[p].v = P[i].x + P[i].y;
		T[p].p = i;
	}
```

```
	if (T[p].l == T[p].r) return;
	insert(p << 1, i);
	insert((p << 1) | 1, i);
}
struct ANS {
	int v, p;
	ANS() {
	}
	ANS(int v, int p) :
		v(v), p(p) {
	}
	bool operator <(const ANS &a) const {
		return v < a.v;
	}
};
ANS query(int p, int i) {
	if (T[p].r < ny[i]) return ANS(INF, -1);
	if (ny[i] <= T[p].l) return ANS(T[p].v, T[p].p);
	return min(query(p << 1, i), query((p << 1) | 1, i));
}

void make() {
	int i, tot;
	for (i = 0; i < n; i++)
		idx[i] = i, yy[i] = P[i].y;
	sort(yy, yy + n);
	tot = unique(yy, yy + n) - yy;
	for (i = 0; i < n; i++)
		ny[i] = lower_bound(yy, yy + tot, P[i].y) - yy;
	sort(idx, idx + n, cmp);
	build(1, 0, tot + 1);
	for (i = 0; i < n; i++) {
		ANS a = query(1, idx[i]);
		if (a.p != -1) E[m++] = EDGE(idx[i], a.p, dis(idx[i], a.p));
		insert(1, idx[i]);
	}
}
void solve() {
	int i, j;
	for (i = 0; i < n; i++)
		P[i].get();
	m = 0;
	for (i = 0; i < 4; i++) {	//旋转
		if (i > 0) {
			for (j = 0; j < n; j++) {
				int x = -P[j].y, y = P[j].x;
				if (i == 2) swap(x, y);
				P[j] = POINT(x, y);
			}
		}
		make();
	}
}
int main() {
	int T = 1;
	while (scanf("%d", &n) && n) {
		solve();
		printf("Case %d: Total Weight = %lld\n", T++, kruskal());
	}
}
```

6.6 网络流无向图的连通度

```
//拆点，固定源点枚举汇点，checked by 1692
#include <cstdio>
#include <cstring>
#define N 200
#define MIN(x,y) ((x)<(y)?(x):(y))
#define INF 1000000
int n, m;
struct Edge{
	int v, cap, flow, next;
	inline void set(int _v, int _cap, int _flow, int _next){
		v = _v; cap = _cap; flow = _flow; next = _next;
	}
}edge[N*N];
int vn, size, src, dst, mp[N], dist[N], que[N*N];

void init(){
	size = 0;
	memset( mp, -1, sizeof(mp) );
}

inline void add_edge(int u, int v, int c1, int c2){
	edge[size].set(v, c1, 0, mp[u]); mp[u] = size++;
	edge[size].set(u, c2, 0, mp[v]); mp[v] = size++;
}

bool dinic_bfs(){
	int head = 0, tail = 1, u, v;
	memset(dist, -1, sizeof(dist));
	dist[src] = 0;
	que[head] = src;
	while (head < tail){
		u = que[head++];
		for (v = mp[u]; v != -1; v = edge[v].next){
			if (edge[v].cap > edge[v].flow && dist[edge[v].v] < 0){
				dist[edge[v].v] = dist[u]+1;
				que[tail++] = edge[v].v;
			}
		}
	}
	return dist[dst] > 0;
}

int dinic_dfs(int u, int f){
	if (u == dst) return f;
	int i, v, ret = 0, tmp;
	for (i = mp[u]; i != -1; i = edge[i].next){
		v = edge[i].v;
		if (edge[i].cap > edge[i].flow && dist[v] == dist[u]+1){
			tmp = dinic_dfs(v, MIN(f, edge[i].cap-edge[i].flow));
			ret += tmp;
			f -= tmp;
			edge[i].flow += tmp;
			edge[i^1].flow -= tmp;
		}
		if (f == 0) break;
	}
	return ret;
```

```
}

int dinic(){
	int ans = 0, tmp;
	while (dinic_bfs()){
		tmp = dinic_dfs(src, INF);
		if (tmp == 0) break;
		ans += tmp;
	}
	return ans;
}

int main(){
	int i, j, u, v;
	while (scanf("%d%d", &n, &m) != EOF){
		if (n == 0 || n == 1){
			printf("%d\n", n);
			continue;
		}
		init();
		for (i = 1; i <= n; i++) add_edge(i, i+n, 1, 0);
		while (m--){
			while (getchar() != '(');
			scanf("%d", &u); getchar();
			scanf("%d", &v); getchar();
			if (u == v) continue;
			u++, v++;
			add_edge(u+n, v, INF, 0);
			add_edge(v+n, u, INF, 0);
		}
		int ans = 2*INF, tmp;
		for (i = 1; i <= n; i++){
			if (i == 1) continue;
			src = 1+n;
			dst = i;
			for (u = 1; u <= 2*n; u++)
				for (v = mp[u]; v != -1; v = edge[v].next)
					edge[v].flow = 0;
			tmp = dinic();
			if (ans > tmp) ans = tmp;
		}
		if (ans >= n) printf("%d\n", n);
		else printf("%d\n", ans);
	}
}
```

6.7 树的分治

```
//checked by pku1741 pku1987
const int MAX_N = 10000;
bool flag[MAX_N];
int k, n, ret, v[MAX_N];
queue<pair<int, int> > q;

struct edge{int v, w; edge *next; } *e[MAX_N], data[MAX_N*2-2], *it;
void insert(int u, int v, int w){
	*it = (edge){v, w, e[u]}; e[u] = it++;
	*it = (edge){u, w, e[v]}; e[v] = it++;
}
```

```
int count(int *first, int *last){
	int ret = 0;
	sort(first, last--);
	while (first < last)
		if (*first+*last <= k) ret += last-first++;
		else --last;
	return ret;
}

int best_size, center;
int centerOfGravity(int root, int pred){
	int max_sub = 0, size = 1;
	for (edge *it = e[root]; it; it = it->next)
		if (it->v != pred && flag[it->v]){
			int t = centerOfGravity(it->v, root);
			size += t;
			if (t > max_sub) max_sub = t;
		}
	if (q.front().second-q.front().first-max_sub > max_sub)
		max_sub = q.front().second-q.front().first-max_sub;
	if (max_sub < best_size)
		best_size = max_sub, center = root;
	return size;
}

int dists[MAX_N], len;
void find(int root, int pred, int dist){
	v[len] = root;
	dists[len++] = dist;
	int last = len;
	for (edge *it = e[root]; it; it = it->next)
		if (it->v != pred && flag[it->v]){
			find(it->v, root, dist+it->w);
			if (pred == -1){
				q.push(make_pair(last, len));
				ret -= count(dists+last, dists+len);
				last = len;
			}
		}
}

int main(){
	int x;
	char dir[4];
	while (scanf("%d%d", &n, &k) != EOF){
		if (n == 0 && k == 0) break;
		it = data;
		memset(e, 0, sizeof(e[0])*n);
		for (int i = 1; i < n; i++){
			int u, v, w;
			scanf("%d%d%d", &u, &v, &w);
			--u; --v;
			insert(u, v, w);
		}
		ret = 0;
		for (int i = 0; i < n; ++i) v[i] = i;
		for (q.push(make_pair(0, n)); !q.empty(); q.pop()){
			if (q.front().first == q.front().second-1) continue;
			for (int i = q.front().first; i < q.front().second; ++i)
				flag[v[i]] = true;
```

```
                        best_size = numeric_limits<int>::max();
                        centerOfGravity(v[q.front().first], -1);
                        len = q.front().first;
                        find(center, -1, 0);
                        ret += count(dists+q.front().first, dists+q.front().second);
                        for (int i = q.front().first; i < q.front().second; ++i)
                                flag[v[i]] = false;
                }
                printf("%d\n", ret);
        }
}
```

6.8 割点割边

```
//checked by zju2588 有重边的情况
#define N 10008
struct Edge{
        int v, id, next;
        Edge(){}
        Edge(int _v, int _id, int _next): v(_v), id(_id), next(_next){}
}edge[200008];
int n, m, b, size, mp[N];
int dep[N], low[N], visit[N], mark[100008], bridge[100008];

inline void add_edge(int u, int v, int id){
        edge[size] = Edge(v, id, mp[u]);
        mp[u] = size++;
}

void dfs(int u, int d, int p){
        int i, v, son = 0;
        visit[u] = 1;
        dep[u] = low[u] = d;
        for (i = mp[u]; i != -1; i = edge[i].next){
                if (mark[edge[i].id]) continue;
                mark[edge[i].id] = 1;
                v = edge[i].v;
                if (visit[v]){
                        low[u] = min(low[u], dep[v]);
                        continue;
                }
                dfs(v, d+1, u);
                low[u] = min(low[u], low[v]);
                son++;
                //cut-vertex -> if ((u == rt && son > 1) || (u != rt && low[v] >= dep[u]))
                //bridge
                if (low[v] > dep[u]){
                        b++;
                        bridge[edge[i].id] = 1;
                }
        }
}

int main(){
        int t, ca, i, u, v;
        scanf("%d", &t);
        for (ca = 0; ca < t; ++ca){
                scanf("%d%d", &n, &m);
                size = 0;
                memset(mp, -1, sizeof(mp));
```

```
            for (i = 1; i <= m; ++i){
                  scanf("%d%d", &u, &v);
                  add_edge(u, v, i);
                  add_edge(v, u, i);
            }
            memset(dep, 0, sizeof(dep));
            memset(low, 0, sizeof(low));
            memset(visit, 0, sizeof(visit));
            memset(mark, 0, sizeof(int)*(m+1));
            memset(bridge, 0, sizeof(int)*(m+1));
            b = 0;
            dfs(1, 1, 0);
            if (ca) puts("");
            printf("%d\n", b);
            for (u = 1, v = 0; u <= m; ++u){
                  if (bridge[u]){
                        if (v) putchar(' ');
                        v = 1;
                        printf("%d", u);
                  }
            }
            if (b != 0) puts("");
      }
}
```

6.9 2-SAT建图

经典2-sat验证题。
假设a为1,则!a为0。建图转换:
a and b = 1 转换为 ：(!a -> a),(!b->b)，因为a，b必选，!a -> a这样会导致a一定被选，为什么？想想构造一组解的时候，经过求强连通分量和缩点后，对新图进行拓扑排序，然后按拓扑倒序对点进行染色。现在明白了吧！因为存在!a->a，所以呢染色的顺序一定是a ,!a，这时如果a没被染色，那么a就一定被染红色，则!a一定被染蓝色。如果a与!a之间不存在边的关系，则表示a,!a任选一个，因为其中一个被染红色，则另外一个一定被染蓝色，具体你的程序选择哪个，就看你的拓扑部分是怎么写的（点的访问顺序）。
a and b = 0 转换为: (a->!b),(b->!a)
a or b = 1   转换为: (!a->b),(!b->a)
a or b = 0   转换为: (a->!a),(b->!b)
a xor b = 1   转换为: (a->!b),(!b->a),(b->!a),(!a->b)
a xor b = 0   转换为: (a->b),(b->a),(!a->!b),(!b->!a)
建完图后，剩下的就是套模板了。

6.10 最小直径生成树

```
//checked by spoj735, Dynamic programming
#define MAXN 1024
int n, m, ans;
vector<int> g[MAXN];
int dist[MAXN][MAXN], flg[MAXN];
int qu[MAXN], qs, qe;

void bfs(int s){
    int i, j, x, y, cnt;
    memset(flg, 0, sizeof(flg));
    flg[s] = 1;
    qs = qe = 0;
    qu[qe++] = s;
    cnt = 0;
    dist[s][s] = 0;
    while (qs < qe){
```

```
        x = qu[qs++];
        for (i = 0; i < g[x].size(); i++){
            y = g[x][i];
            if (flg[y] == 0){
                flg[y] = 1;
                qu[qe++] = y;
                dist[s][y] = dist[s][x] + 1;
            }
        }
    }
}

int main(){
    int i, j, x, y, k;
    int csnum, id;
    scanf ("%d", &csnum);
    while (csnum--){
        scanf ("%d", &n);
        for (i = 1; i <= n; i++) g[i].clear();
        for (i = 1; i <= n; i++){
            scanf ("%d %d", &id, &k);
            for (j = 0; j < k; j++){
                scanf ("%d", &x);
                g[id].push_back(x);
            }
       }
       for (i = 1; i <= n; i++) bfs(i);
       int mx;
       ans = n;
       for (i = 1; i <= n; i++){
            mx = 0;
            for (j = 1; j <= n; j++) if (dist[i][j] > mx) mx = dist[i][j];
            if (2 * mx < ans) ans = 2 * mx;
       }
       int l;
       for (i = 1; i <= n; i++){
           for (k = 0; k < g[i].size(); k++){
               j = g[i][k];
               mx = 0;
               for (l = 1; l <= n; l++){
                    mx = max(mx, min(dist[i][l], dist[j][l]));
               }
               ans = min(ans, 2*mx+1);
           }
       }
       printf ("%d\n", ans);
   }
}
```

7. 组合数学、数值计算

7.1 常用公式

多重组合：n个球放入k个盒子
C(n+k-1, n)，C(n+k-1, k-1)

集合划分：将大小为n的集合划分为k个子集，第二类斯特林数
S(n, k) = S(n-1, k-1) + k * S(n-1, k)
S(n, k) = sigma(0...k){ (-1)^j * C(k, j) * (k - j)^n } / k!

bell数：将大小为n的集合划分为若干个子集
B(n) = sigma(1...n)S(n, k)
B(n+1) = sigma(0...n)B(k) * C(n, k)

整数拆分：将n拆成k个正整数，sigma(Ak) = n，a1 >= a2 >= ... >= ak

```
//dp[i][j] = dp[i-1][j-1] + dp[i-j][j];
LL dp[N][N] = { 0 };
int main(){
	int i, j, t;
	LL ans;
	for (i = 1; i < N; i++){
		dp[i][1] = dp[i][i] = 1;
		for (j = 2; j < i; j++)
			dp[i][j] = dp[i-1][j-1] + dp[i-j][j];
	}
	while (scanf("%d", &t) != EOF){
		for (ans = 0, i = 1; i <= t; i++) ans += dp[t][i];
		printf("%lld\n", ans);
	}
}
```

第一类斯特林数：n排列有k个cycle
c(n, k) = c(n-1, k-1) + (n-1) * c(n-1, k)
n! = sigma(c(n, k))
sigma(c(n, k) * x^k) = (x+n-1) * (x+n-2) * ... * (x+1) * x

permutation of a given type
p:(a1, a2, ... , an) ai cycles for length i
n! / (1^a1 * 2^a2 * ... n^an * a1! * a2! * ... * an!)

欧拉数：n的排列含有k个升程
A(n, k) = k * A(n-1, k) + (n-k+1) * A(n-1, k-1)
A(n, k) = Sum (-1)^j*(k-j)^n*C(n+1, j), j=0..k
A(n, k) = A(n, n-1-k)

second-order eulerian number:
<<n, k>> = (k+1) * <<n-1, k>> + (2*n-1-k) * <<n-1, k-1>>

逆序数：n的排列逆序数为k
b(n, k) = b(n-1, k) + b(n-1, k-1) + ... + b(n-1, m)
m = max(0, k-n+1)

cayley公式：有标号的无根树个数 n^(n-2)
Prufer编码

Given positive integers d1, . . . , dn summing to 2n − 2.
(n-2)! / π((di-1)!)

矩阵树定理：
无圈图G, a[i,j]为vi, vj边的条数，
Q为矩阵，i=j时，Q[i,j] = d[i], i!=j, Q[i,j] = -a[i,j]
生成树个数为Q的任意一个n-1阶代数余子式

the number of unlabeled rooted trees with n nodes
a(n+1) = (1/n) * sum_{k=1..n} (sum_{d|k} d*a(d)) * a(n-k+1)

the number of unlabeled unrooted trees with n nodes
Generating Function:
A(x) = 1 + T(x)-T^2(x)/2+T(x^2)/2
T(x)为上面数列的母函数

catalan数：
c[n] = C(2n, n)/(n+1)
c[n] = (4n-2)/(n+1)*c[n-1];
c[n] = sigma(c[i] * c[n-i]) i >= 1
相关问题 ：
不穿过对角线的路径数、完全加括号方案数、满二叉树数目、多边形三角剖分方案数
合法括号序列方案数、排队找零钱问题、出栈顺序问题

7.2 求连通图个数

```
/*pku 1737
  1.随便从n个点里面拿出来2个点,放到2边,当成是2个子连通图
  2.然后从剩下的n-2个点选k-1个,ans=c(n-2,k-1),放到左边,那么它和我们刚才取出来的某个点组合
  成了一个大小为k的连通图它的种类数是dp[k],表示大小为k的时候的连通种类数
  显然右边有(n-k)个,所以ans*=dp[k]*dp[n-k];
  3.由于你必须保证2个子图最少有一条边连起来,所以ans*=2^k - 1,既对于左边的k个点,要么和右边
  的有连边,要么没边,所以一共是2^k,必须减掉都不连边的情况.
*/
import java.util.*;
import java.math.*;

public class Main {
       public static BigInteger []g = new BigInteger[60];
       public static BigInteger []f = new BigInteger[60];
       public static BigInteger [][]co = new BigInteger[60][60];

       static void init(){
              int i, j;
              co[0][0] = BigInteger.ONE;
              for (i = 1; i <= 50; i++){
                     co[i][0] = co[i][i] = BigInteger.ONE;
                     for (j = 1; j < i; j++)
                            co[i][j] = co[i-1][j].add(co[i-1][j-1]);
              }
       }

       public static void main( String args[] ){
              init();
              int i, j;
              Scanner cin = new Scanner(System.in);
              g[1] = f[1] = BigInteger.ONE;
              for (i = 2; i <= 50; i++)
                     g[i] = BigInteger.valueOf(2).pow(co[i][2].intValue());
              for (i = 2; i <= 50; i++){
                     f[i] = g[i];
                     for (j = 1; j < i; j++){
              f[i] = f[i].subtract(f[j].multiply(g[i-j]).multiply(co[i-1][j-1]));
```

```
                }
            }
            while (cin.hasNext()){
                i = cin.nextInt();
                if (i == 0) break;
                System.out.println(f[i]);
            }
        }
}
```

7.3 置换

正方形棋盘的置换：
1. 旋转0，90，180，270三种
   0度，置换为n*n
   90度，n*n/4(n为偶)，(n*n-1)/4+1(n为奇)
   180度，n*n/2(n为偶)，(n*n-1)/2+1(n为奇)
   270度，n*n/4(n为偶)，(n*n-1)/4+1(n为奇)
2. 沿对角线反射，沿边中线反射
n为偶时：
   沿边中线反射n*n/2
   沿对角线反射(n*n-n)/2+n
n为奇时：
   沿边中线反射(n*n-n)/2+n
   沿对角线反射(n*n-n)/2+n

```
//checked by 3532
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <map>
#include <algorithm>
using namespace std;
#define N 65536
typedef long long LL;
const LL MOD = 1000000007;
int A, C;
int size, prime[N], check[N] = {0};
int seq[32], cnt[32], fcnt[32], sq, sp;
LL val[N], phi[N], col_sqr, ring, len, total;
map<LL, LL> phi_set;

void get_prime(){
    int i, j;
    size = 0;
    for (i = 2; i < N; i++){
        if (!check[i]) prime[size++] = i;
        for (j = 0; j < size && i*prime[j] < N; j++){
            check[i*prime[j]] = 1;
            if (i%prime[j]==0) break;
        }
    }
}

void decompose(LL u, LL v, int seq[], int cnt[], int &sq){
    int i;
    sq = 0;
    for (i = 0; i < size &&
        ((LL)prime[i]*prime[i] <= v || (LL)prime[i]*prime[i] <= u); i++){
        if (v%prime[i] == 0 || u%prime[i] == 0){
```

```
                        seq[sq] = prime[i];
                        cnt[sq] = 0;
                        while (v%prime[i] == 0){
                                cnt[sq]++;
                                v /= prime[i];
                        }
                        while (u%prime[i] == 0){
                                cnt[sq]++;
                                u /= prime[i];
                        }
                        sq++;
                }
        }
        if (v != 1){
                seq[sq] = v;
                cnt[sq++] = 1;
        }
        if (u != 1){
                seq[sq] = u;
                cnt[sq++] = 1;
        }
}

LL power(LL x, LL y){
        LL ans = 1, d = x%MOD;
        while (y > 0){
                if (y&1) ans = ans*d%MOD;
                d = d*d%MOD;
                y >>= 1;
        }
        return ans;
}

LL extend_euclid(LL a, LL b, LL &x, LL &y){
        if (b == 0){
                x = 1; y = 0;
                return a;
        }
        LL d = extend_euclid(b, a%b, y, x);
        y -= x*(a/b);
        return d;
}

LL calSquare(LL b){
        LL ans = 0, x, y;
        if (b == 1) return C;
        if (b%2){
                ans = (ans + power((LL)C, b * b)) % MOD;
                ans = (ans + 2 * power((LL)C, (b * b - 1) / 4 + 1) % MOD) % MOD;
                ans = (ans + power((LL)C, (b * b -1)/2 + 1)) % MOD;
        }
        else{
                ans = (ans + power((LL)C, b * b)) % MOD;
                ans = (ans + 2 * power((LL)C, b * b / 4) % MOD) % MOD;
                ans = (ans + power((LL)C, b * b / 2)) % MOD;
        }
        ans = ans * (LL)250000002 % MOD;
        return ans;
}
```

```
void calPhi(){
	int i, j, k, t;
	map<LL, LL>::iterator itr1, itr2;
	sp = 1, val[0] = 1;
	phi_set.clear();
	phi_set.insert(pair<LL, LL>(1ll, 1ll));
	for (i = 0; i < sq; i++){
		for (j = 0; j < cnt[i]; j++){
			t = sp;
			for (k = 0; k < t; k++){
				LL x = val[k]*seq[i];
				itr1 = phi_set.find(val[k]);
				itr2 = phi_set.find(x);
				if (itr2 == phi_set.end()){
					val[sp++] = x;
					if (itr1->first % seq[i] == 0)
				phi_set.insert(pair<LL, LL>(x, (itr1->second)*seq[i]));
					else
			phi_set.insert(pair<LL, LL>(x, (itr1->second)*(seq[i]-1)));
					}
			}
		}
	}
	for (i = 0, itr1 = phi_set.begin(); itr1 != phi_set.end(); i++, itr1++){
		val[i] = itr1->first;
		phi[i] = itr1->second;
	}
	sp = i;
}

void dfs(int dep, LL fact){
	if (dep == sq){
		LL sub = (LL)(A-1)*(A+1)/(LL)fact;
		LL b = (LL)sqrt(sub*1.0);
		if (b*b == sub){
			col_sqr = calSquare(b);
			len = fact;
			ring = 0;
			LL x, y;
			for (int i = 0; i < sp && val[i] <= fact; i++){
				if (fact%val[i] == 0){
			ring += (power(col_sqr, fact/val[i]) * (phi[i] % MOD))%MOD;
					ring %= MOD;
				}
			}
			extend_euclid(len, MOD, x, y);
			if (x < 0) x = x + ((-x) / MOD + 1) * MOD;
			if (x > 0) x = x - x/MOD*MOD;
			ring = ring * x % MOD;
			total = (total + ring*C%MOD) % MOD;
		}
		return ;
	}
	dfs(dep+1, fact);
	for (int i = 1; i <= cnt[dep]; i++){
		fact *= seq[dep];
		dfs(dep+1, fact);
	}
}
```

```
int main(){
	int ca, t = 0;
	get_prime();
	scanf("%d", &ca);
	while (ca--){
		t++;
		scanf("%d%d", &A, &C);
		if (A == 1){
			printf("Case %d: %d\n", t, C);
			continue;
		}
		decompose(A-1, A+1, seq, cnt, sq);
		total = 0;
		calPhi();
		dfs(0, 1);
		printf("Case %d: %lld\n", t, total);
	}
}
```

7.4 行列式取模

```
//checked by spoj2832 uestc1217
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
#define N 208
typedef long long LL;
LL n, p, a[N][N];

void det(){
	int i, j, k;
	LL ans = 1, t;
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			a[i][j] = (a[i][j]%p + p) % p;
	for (i = 0; i < n; i++){
		for (j = i+1; j < n; j++){
			while (a[j][i] != 0){
				t = a[i][i] / a[j][i];
				for (k = i; k < n; k++){
					a[i][k] = (a[i][k] - a[j][k] * t) % p;
					a[i][k] = (a[i][k] + p) % p;
				}
				for (k = 0; k < n; k++) swap(a[i][k], a[j][k]);
				ans = -ans;
			}
		}
		if (a[i][i] == 0){
			ans = 0;
			break;
		}
		ans = ans * a[i][i] % p;
		ans = (ans + p) % p;
	}
	printf("%lld\n", ans);
}

int main(){
	int i, j;
```

```
	while (scanf("%lld%lld", &n, &p) != EOF){
		for (i = 0; i < n; i++)
			for (j = 0; j < n; j++)
				scanf("%lld", &a[i][j]);
		det();
	}
}
```

7.5 高斯消元法xor equation

```
int n, a[N][N], b[N], val[N];
void gauss( ){
    int i, j, k, row, ans = 0;
    for ( i=row=0; i<n; i++ ){
        for ( j=row; j<n&&!a[j][i]; j++ );
        if ( j>=20 ) continue;
        if ( j>row ){
            for ( k = 0; k <= 20; k++ )
                swap( a[row][k], a[j][k] );
        }
        for ( j = row+1; j < 20; j++ ){
            if ( a[j][i] == 0 ) continue;
            for ( k = 0; k <= 20; k++ ) a[j][k] ^= a[row][k];
        }
        row++;
    }
}

void check(){
    int i, j, k;
    for (i=n-1; i>=0; i--){
        k = a[i][n];
        for (j = i+1; j < n; j++) k ^= a[i][j]*val[j];
        val[i] = k;
    }
}
```

7.6 线性规划单纯形(我的模板)

```
//checked by uva10498，缺少线性规划的初始化步骤
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;
#define N 128
#define INF 1000000000
const double EPS = 1e-6;
int n, m;
int mark_n[N], mark_b[N];
double a[N][N], b[N], c[N];//a*x <= b
double _a[N][N], _b[N], _c[N];//中间变量
double V, var[N];
//for debugging
void output(){
	int i, j;
	puts("----------------------");
	for (i = 0; i < n+m; i++) printf("%.2lf ", c[i]); puts("");
	for (i = 0; i < n+m; i++){
		for (j = 0; j < n+m; j++) printf("%.2lf ", a[i][j]);
```

```
            printf(": %.2lf\n", b[i]);
        }
        printf("AA %d %d\n", n, m);
        puts("Non-basic");
        for (i = 0; i < n+m; i++) printf("%d ", mark_n[i]); puts("");
        puts("Basic");
        for (i = 0; i < n+m; i++) printf("%d ", mark_b[i]); puts("");
        puts("---------------------");
}

bool input(){
        if (scanf("%d%d", &n, &m) == EOF) return false;
        int i, j;
        memset(a, 0, sizeof(a));
        memset(b, 0, sizeof(b));
        memset(c, 0, sizeof(c));
        for (i = 0; i < n; i++){
            scanf("%lf", c+i);
            c[i] *= m;
        }
        for (i = 0; i < m; i++){
            for (j = 0; j < n; j++) scanf("%lf", a[i+n]+j);
            scanf("%lf", &b[i+n]);
        }
        memset(mark_n, 0, sizeof(mark_n));
        memset(mark_b, 0, sizeof(mark_b));
        for (i = 0; i < n; i++) mark_n[i] = 1;
        for (i = n; i < n+m; i++) mark_b[i] = 1;
        return true;
}

void init_simplex(){
        int i, j;
        for (i = 0; i < n; i++) var[i] = 0;
        for (i = n; i < n+m; i++) var[i] = b[i];
        V = 0;
}
//算法导论上的伪代码
void pivot(int leave, int enter){
        int i, j;
        memset(_a, 0, sizeof(_a));
        memset(_b, 0, sizeof(_b));
        memset(_c, 0, sizeof(_c));
        //part 1
        _b[enter] = b[leave]/a[leave][enter];
        for (i = 0; i < n+m; i++){
            if (mark_n[i] == 1 && i != enter)
                _a[enter][i] = a[leave][i]/a[leave][enter];
        }
        _a[enter][leave] = 1/a[leave][enter];
        for (i = 0; i < n+m; i++){
            if (mark_b[i] == 0 || i == leave) continue;
            _b[i] = b[i] - a[i][enter]*_b[enter];
            for (j = 0; j < n+m; j++)
                if (mark_n[j] == 1 && j != enter)
                    _a[i][j] = a[i][j] - a[i][enter]*_a[enter][j];
            _a[i][leave] = -a[i][enter]*_a[enter][leave];
        }
        V = V+c[enter]*_b[enter];
        for (i = 0; i < n+m; i++){
```

```
            if (!mark_n[i] || i == enter) continue;
            _c[i] = c[i] - c[enter]*_a[enter][i];
        }
        _c[leave] = -c[enter]*_a[enter][leave];
        mark_n[enter] = 0; mark_b[leave] = 0;
        mark_b[enter] = 1; mark_n[leave] = 1;
        memcpy(a, _a, sizeof(_a));
        memcpy(b, _b, sizeof(_b));
        memcpy(c, _c, sizeof(_c));
}

void simplex(){
        int i, j, k;
        int enter, leave;
        double delta;
        while (1){
            for (i = 0; i < n+m; i++){
                if (mark_n[i] == 0) continue;
                if (c[i] > EPS) break;
            }
            if (i == n+m) break;
            enter = i;
            delta = INF;
            for (j = 0; j < n+m; j++){
                if (mark_b[j] == 0) continue;
                if (a[j][enter] > 0){
                    if (delta > b[j]/a[j][enter]){
                        delta = b[j]/a[j][enter];
                        leave = j;
                    }
                }
            }
            if (delta == INF){
                puts("Wrong Answer");
                break;
            }
            pivot(leave, enter);
        }
        for (i = 0; i < n+m; i++){
            if (mark_b[i]) var[i] = b[i];
            else var[i] = 0;
        }
        printf("Nasa can spend %d taka.\n", (int)(ceil(V)));
}

int main(){
        while (input()){
            init_simplex();
            simplex();
        }
}
```

7.7 Romberg积分

```
//checked by hdu3310
#include<iostream>
#include<stdio.h>
#include<string.h>
#include<cmath>
#define eps 1e-5
```

```
using namespace std;
#define MAX_N 1000
double r1,r2;
double f(double x){
	return sqrt(r1*r1-x*x)*sqrt(r2*r2-x*x);
}

double Romberg (double a, double b, double (*f)(double x)){
	int i, j, temp2, mini;
	double h, R[2][MAX_N], temp4;
	for (i=0; i<MAX_N; i++) {
		R[0][i] = 0.0;
		R[1][i] = 0.0;
	}
	h = b-a;
	mini = (int)(log(h*10.0)/log(2.0));
	R[0][0] = ((*f)(a)+(*f)(b))*h*0.50;
	i = 1;
	temp2 = 1;
	while (i<MAX_N){
		i++;
		R[1][0] = 0.0;
		for (j=1; j<=temp2; j++)
			R[1][0] += (*f)(a+h*((double)j-0.50));
		R[1][0] = (R[0][0] + h*R[1][0])*0.50;
		temp4 = 4.0;
		for (j=1; j<i; j++) {
			R[1][j] = R[1][j-1] + (R[1][j-1]-R[0][j-1])/(temp4-1.0);
			temp4 *= 4.0;
		}
		if ((fabs(R[1][i-1]-R[0][i-2])<eps)&&(i>mini))
			return R[1][i-1];
		h *= 0.50;
		temp2 *= 2;
		for (j=0; j<i; j++) R[0][j] = R[1][j];
	}
	return R[1][MAX_N-1];
}

int main(){
	int t;
	for(scanf("%d",&t);t;t--){
		scanf("%lf%lf",&r1,&r2);
		if(r1>r2)swap(r1,r2);
		printf("%.2lf\n",8*Romberg(0,r1,f));
	}
	return 0;
}
```

8. 附录：vim配置

```
set encoding=utf8
set modelines=0
syntax enable
set showcmd
filetype indent on
set autoindent
set cindent
set showmatch
set matchtime=5
set ruler
set number
set backspace=2
set tabstop=4
set shiftwidth=4
set ai!

map <F5> :call CompileRunGcc()<CR>
func! CompileRunGcc()
       exec "w"
       exec "!gcc -Wall %"
       exec "! ./a.out"
endfunc

map <F6> :call CompileRunGpp()<CR>
func! CompileRunGpp()
       exec "w"
       exec "!g++ -Wall %"
       exec "! ./a.out"
endfunc
```