```
        int n;   一个整数
```

1 数学

1.1 高精度

```
const int e[9] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000};
const int bs = 8;
const int maxlen = 20;

struct bign {
        int a[maxlen]; int n;

        void suppress() { while (n > 1 && !a[n-1]) -- n; }

        bign() { }
        bign(char* s) {
                int m = strlen(s);

                n = 0;
                for (int i = m-1; i >= 0; i -= bs) {
                        int t = 0;
                        for (int j = 0; j < bs && (i-j) >= 0; ++ j) {
                                t = t + (s[i-j]-'0') * e[j];
                        }
                        a[n ++] = t;
                }
                suppress();
        }
        bign(LL x) { n = 0; for (; x; x /= e[bs]) a[n ++] = x % e[bs]; }
        bign(const bign& A) { n = A.n; for (int i = 0; i < n; ++ i) a[i] = A.a[i]; }
        bool operator ==(const bign& A) const {
                if (n != A.n) return 0;
                for (int i = n-1; i >= 0; -- i) if (a[i] != A.a[i]) return 0;
                return 1;
        }
        bool operator <(bign& A) const {
                if (n != A.n) return n < A.n;
                for (int i = n-1; i >= 0; -- i) if (a[i] != A.a[i]) return a[i] < A.a[i];
                return 0;
        }

        void print() {
                printf("%d", a[n-1]); for (int i = n-2; i >= 0; -- i) printf("%0*d", bs, a[i]);
        }
};

bign C;

bign operator +(const bign& A, const bign &B) {
        C.n = std::max(A.n, B.n) + 1;

        int t = 0;
        for (int i = 0; i < C.n; ++ i) {
                t = t + ((i < A.n ? A.a[i] : 0) + (i < B.n ? B.a[i] : 0));
                if (t >= e[bs]) {
                        C.a[i] = t - e[bs];
                        t = 1;
                } else {
                        C.a[i] = t;
                        t = 0;
                }
        }
        C.suppress();
        return C;
}
bign operator -(const bign &A, const bign &B) {
        C.n = std::max(A.n, B.n);

        int t = 0;
        for (int i = 0; i < C.n; ++ i) {
                t = t + ((i < A.n ? A.a[i] : 0) - (i < B.n ? B.a[i] : 0));
                if (t < 0) {
                        C.a[i] = t + e[bs];
                        t = -1;
                } else {
                        C.a[i] = t;
                        t = 0;
                }
        }
        C.suppress();
        return C;
}
bign operator *(const bign &A, const bign &B) {
        C.n = A.n + B.n;
        memset(C.a, 0, sizeof(int)*C.n);

        LL t = 0;
        for (int i = 0; i < A.n; ++ i) {
                for (int j = 0; j <= B.n; ++ j) {
                        t = t + C.a[i+j] + (j < B.n ? ((LL)A.a[i] * B.a[j]) : 0);
                        C.a[i+j] = t % e[bs];
                        t = t / e[bs];
                }
        }

        C.suppress();
        return C;
}
bign operator /(const bign& A, LL b) {
        C.n = A.n;

        LL t = 0;
        for (int i = C.n-1; i >= 0; -- i) {
                t = (t%b)*e[bs] + A.a[i];
                C.a[i] = t/b;
        }

        C.suppress();
        return C;
}

bign operator +=(bign& A, const bign& B) { return A = A + B; }
```

1.2 模数

```
#include<cstdio>
typedef long long LL;
int MOD;
struct modn{
        int n,MOD;
        modn(){}
        modn(LL x=0,int MOD):MOD(MOD){n=(x>=0?(x<MOD?x:x%MOD):(x%MOD+MOD)%MOD);}
        // Attention: MOD is not participate in the comparison.
        bool operator <(const modn& B){return n<B.n;}
        bool operator ==(const modn& B){return n==B.n;}
        bool operator !=(const modn& B){return n!=B.n;}
        modn operator +(const modn& B){return modn(n+B.n>=MOD?n+B.n-MOD:n+B.n,MOD);}
        modn operator -(const modn& B){return modn(n-B.n<0?n-B.n+MOD:n-B.n,MOD);}
        modn operator *(const modn& B){return modn((LL)n*B.n,MOD);}
        modn& operator +=(const modn& B){return *this=*this+B;}
        modn& operator -=(const modn& B){return *this=*this-B;}
        modn& operator *=(const modn& B){return *this=*this*B;}
        modn& operator -() {n=MOD-n;return *this;}
        void print()const{printf("%d\n",n);}
};
modn pow(const modn& A,LL x){
        modn s=modn(1,A.MOD),t=A;for(;x;x>>=1){if(x&1)s*=t;t*=t;}return s;
}
modn zero; // zero should be initialed.
bool iszero(const modn& A){return A.n==0;}

/*
modn fact_mod[MOD];
modn fact(LL n) {
        return n == 0 ? 1 : (((n/mod)&1) ? -1 : 1) * fact(n/mod) * fact_mod[n%mod];
}

modn mod_fact(LL n, int& e) {
        if (n == 0) { e = 0; return modn(1); }
        modn res = mod_fact(n/mod, e);
        e += n/mod;
        return (((n/mod)&1) ? -res : res) * fact_mod[n%mod];
}
modn C_mod(LL n, LL k, int& e) {
        if (n < 0 || k < 0 || n < k) return 0;
        int e1, e2, e3;
        modn n1 = mod_fact(n, e1), k1 = mod_fact(k, e2), k2 = mod_fact(n-k, e3);
        e = e1 - (e2 + e3);
        return n1 / (k1*k2);
}
*/
```

1.3 分数

```
LL gcd(LL a, LL b) { LL c; while (b) { c = a; a = b; b = c%b; } return a; }

struct frac {
        LL a, b;
        frac() { }
        frac(LL a) : a(a), b(1) { }
        frac(LL a, LL b) : a(a), b(b) { }

        frac normalize() const { LL d = gcd(a, b); return b/d < 0 ? frac(-a/d, -b/d) : frac(a/d, b/d); }

        void print() {
                if (b == 1) printf("%I64d", a);
                else if (a < 0) printf("-\\frac{%I64d}{%I64d}", -a, b);
                else printf("\\frac{%I64d}{%I64d}", a, b);
        }
};
frac operator +(const frac& A, const frac& B) { return frac(A.a * B.b + A.b * B.a, A.b * B.b).normalize(); }
frac operator -(const frac& A, const frac& B) { return frac(A.a * B.b - A.b * B.a, A.b * B.b).normalize(); }
frac operator *(const frac& A, const frac& B) { return frac(A.a * B.a, A.b * B.b).normalize(); }
frac operator /(const frac& A, const frac& B) { return frac(A.a * B.b, A.b * B.a).normalize(); }
frac operator <(const frac& A, const frac& B) { return A.a * B.b - A.b * B.a < 0; }
bool operator ==(const frac& A, const frac& B) { return A.a == B.a && A.b == B.b; }
bool operator !=(const frac& A, const frac& B) { return !(A.a == B.a && A.b == B.b); }
```

1.4 矩阵

```
#include<cstdio>

const int msz = 3;

template<class T>
struct Mat {
        int m, n;
        T a[msz][msz];

        Mat(){}
        Mat(int m,int n):m(m),n(n){
            for(int i=0;i<m;++i)for(int j=0;j<n;++j)a[i][j]=0;
        }
        Mat(int m,int n,T* A):m(m),n(n){
            for(int i=0;i<m;++i)for(int j=0;j<n;++j)a[i][j]=A[i*n+j];
        }
        T* operator [](int i) { return a[i]; }
        const T* operator [](int i) const { return a[i]; }

        // assert(m==B.m&&n==B.n);
        Mat operator +(const Mat& B){
            static Mat C;new(&C)Mat(m,n);
            for(int i=0;i<C.m;++i)for(int j=0;j<C.n;++j)C[i][j]=a[i][j]+B[i][j];
            return C;
        }
        // zero(T) is needed
        // assert(n==B.m);
        Mat operator *(const Mat& B){
            static Mat C;new(&C)Mat(m,B.n);
            for(int i=0;i<m;++i)
                for(int j=0;j<n;++j)if(zero(a[i][j]))
                    for(int k=0;k<B.n;++k)if(zero(B[j][k]))
                        C[i][k]+=a[i][j]*B[j][k];
            return C;
        }
        void print() {
            for (int i = 0; i < m; ++ i,putchar('\n'))
                for (int j = 0; j < n; ++ j,putchar('␣'))
                    a[i][j].print();
        }
```

```
};

template<class T>
Mat<T> power(Mat<T> a, int n) {
        assert(a.m==a.n);
        Mat<T> s(a.n,a.n),t=a;
        for(int i=0;i<s.n;++i)s[i][i]=1;
        for(;n;n>>=1){if(n&1)s=s*t;t=t*t;}
        return s;
}

// Output: Det(A)
template<class T>
T Determinant(Mat<T> A){
        assert(A.m==A.n);
        int n=A.n;
        T res=1;
        for(int i=0;i<n;++i){
                for(int j=i+1;j<n;++j){
                        T *p=A[i], *q=A[j];
                        while(q[i]!=0){
                                int t=p[i]/q[i];
                                for(int k=i;k<n;++k)
                                        p[k]=p[k]-q[k]*t;
                                swap(p,q);
                        }
                        if(p!=A[i]){
                                for(int k=i;k<n;++k)
                                        swap(p[k],q[k]);
                                swap(p,q);
                                res=-res;
                        }
                }
                if(A[i][i]==0) return 0;
                res=res*A[i][i];
        }
        return res;
}
```

1.5 异或找基

```
LL b[100], bN;
void solve() {
        bN = 0;
        for (int i = 1; i <= tot; ++ i) {
                for (int j = 0; j < bN; ++ j) {
                        if ((num[i]^b[j]) < num[i])
                                num[i] = num[i] ^ b[j];
                }
                if (num[i])
                        b[bN ++] = num[i];
        }

        sort(b, b+bN, greater<LL>());

        LL res = 0;
        for (int i = 0; i < bN; ++ i)
                if ((res ^ b[i]) > res)
```

```
                        res ^= b[i];
        printf("%I64d\n", res);
}
```

1.6 k 阶等差数列

```
        B[0][0] = 0; B[0][1] = 1;
        for (int d = 1; d < maxn; ++ d) {
                for (int j = 0; j <= d; ++ j) {
                        B[d][j] = 0;
                        for (int i = max(0, j-1); i <= d-1; ++ i)
                                B[d][j] += - C[d+1][i] * B[i][j];
                        B[d][j] = B[d][j] / (d+1);
                }
                B[d][d+1] = frac(1) / (d+1);
        }
```

1.7 推导

$(d+1)S_d(n) = n^{d+1} - \sum_{i=0}^{d-1}\binom{d+1}{i}S_i(n)$
由此，令
$S_d(n) = \sum_{i=0}^{d+1} B_d(i)n^i$
带入，得：
$(d+1)S_d(n) = n^{d+1} - \sum_{j=0}^{d}(\sum_{i=\max(j-1,0)}^{d-1}\binom{d+1}{i}B_i(j))n^j$

1.8 球坐标转三维坐标

设球坐标为 (r, θ, ϕ) (r 表示与原点距离，θ 表示纬度（北），ϕ 表示经度（东））
那么三维坐标为 $(r\sin\theta\cos\phi, r\sin\theta\sin\phi, r\cos\theta)$

若三维坐标为 (x, y, z)
球坐标为 $(\sqrt{x^2+y^2+z^2}, \arccos(\frac{z}{\sqrt{x^2+y^2+z^2}}), \arctan(\frac{y}{x}))$

2 数论

2.1 线性筛

```
int n;

int prime[maxp],pN;
int phi[maxn],mu[maxn];
// mf:minfactor, mfk:the num of mf, mfx:pow(mf,mfk)
int mf[maxn],mfk[maxn],mfx[maxn];
// divN:the num of divisors, divsN:the num of divisors set
int divN[maxn],divsN[maxn];
// f:any function that can be summarized by gen, sf:sum of f
// such as "multiplicative function" or some function that can be derived by (i|p) and/or !(i|p)
LL f[maxn],sf[maxn];
bool isp[maxn];
void gen(int n) {
        pN = 0;
        memset(isp, 1, sizeof isp);

        isp[0]=isp[1]=0;
        phi[1]=1, mu[1]=1;
        mf[1]=1, mfk[1]=0, mfx[1]=1;
        divN[1]=divsN[1]=1;
        f[1]=1;
        for(int i=2,p;i<n;++i){
                if (isp[i]) {
```

```
            prime[pN++]=p=i;
            phi[p]=p-1, mu[p]=-1;
            mf[p]=p, mfk[p]=1, mfx[p]=p;
            divN[p]=divsN[p]=2;
            f[p]=2LL*p-1;
        }

        for(int j=0,x;j<pN && (x=i*(p=prime[j]))<n;++j){
            isp[x] = 0;

            if(!(i%p)){
                phi[x]=phi[i]*p, mu[x]=0;
                mf[x]=p, mfk[x]=mfk[i]+1, mfx[x]=mfx[i]*p;
                divN[x]=divN[i]+divN[p], divsN[x]=divsN[i];
                if(mfx[x]==x) // x=p^k
                    f[x]=x+mfk[x]*(x/p*(p-1));
                else
                    f[x]=f[x/mfx[x]]*f[mfx[x]];
                break;
            } else {
                phi[x]=phi[i]*(p-1), mu[x]=-mu[i];
                mf[x]=p, mfk[x]=1, mfx[x]=p;
                divN[x]=divN[i]+divN[p], divsN[x]=divsN[i]+1;
                f[x]=f[i]*f[p];
            }
        }
    }
}
int get_pri(int *pri, int* pri_num, int n) {
	int pri_n = 0;
	for (int i = 0, p; (p = prime[i]) <= n/p; ++ i) if (n%p == 0) {
		pri[pri_n] = p; pri_num[pri_n] = 0;
		while (n%p == 0) { n /= p; ++ pri_num[pri_n]; }
		++ pri_n;
	}
	if (n > 1) { pri[pri_n] = n; pri_num[pri_n] = 1; ++ pri_n; }
	return pri_n;
}
void dfs_divisor(int d, int div) {
	if (d == pri_n) {
		/* do something here */
		return;
	}
	for (int i = 0; i <= pri_num[d]; ++ i) {
		dfs_divisor(d+1, div);
		div = div * pri[d];
	}
}
dfs_divisor(0, 1);

int phi(int n) {
	int res = n;
	for (int i = 0, j; (j = prime[i]) <= n/j; ++ i) if (n%j == 0) {
		res = res / j * (j-1);
		while (n%j == 0) n /= j;
	}
	if (n > 1) res = res / n * (n-1);
	return res;
```

```
}

int get_primitive_root(int MOD, int phi) {
        vector<int> factors;
        int n = phi;
        for(int i = 2; i <= n/i; i++) {
                if(n % i != 0) continue;
                factors.push_back(i);
                while(n % i == 0) n /= i;
        }
        if(n > 1) factors.push_back(n);

        for (int m = 2; ; ++ m) {
                bool ok = true;
                for(int i = 0; i < factors.size(); i++)
                        if(pow_mod(m, phi/factors[i], MOD) == 1) { ok = false; break; }
                if(ok) return m;
        }
}

inv_mod[1] = 1; for (int i = 2; i < mod; ++ i) inv_mod[i] = (mod-mod/i) * inv_mod[mod%i];
fact_mod[0] = 1; for (int i = 1; i < maxn; ++ i) fact_mod[i] = fact_mod[i-1] * i;
ifact_mod[0] = 1; for (int i = 1; i < maxn; ++ i) ifact_mod[i] = ifact_mod[i-1] * inv_mod[i];

gen(n+1);
```

2.2 素数和

```
LL N(int n, int _p) {
        if (_p < 0) return n-1;
        int p = prime[_p];
        if (p*p > n) return N(n, _p-1);
        return N(n, _p-1) - (N(n/p, _p-1) - N(p-1, _p-1));
}
LL S(int n, int _p) {
        if (_p < 0) return (LL)n*(n+1)/2 - 1;
        int p = prime[_p];
        if (p*p > n) return S(n, _p-1);
        return S(n, _p-1) - (LL)p * (S(n/p, _p-1) - S(p-1, _p-1));
}
LL fact[maxn];
LL P(int n, int _p) {
        if (_p < 0) return fact[n];
        int p = prime[_p];
        if (p*p > n) return P(n, _p-1);
        return P(n, _p-1) / power(p, N(n/p, _p-1) - N(p-1, _p-1)) / (P(n/p, _p-1) / P(p-1, _p-1));
}

int n, p;

void solve() {
        scanf("%d", &n);
        p = std::upper_bound(prime, prime+prime_N, (int)(sqrt(n)+1)) - prime - 1;

        fact[0] = 1; for (int i = 1; i <= 15; ++ i) fact[i] = fact[i-1] * i;
        printf("%I64d\n", N(n, p));
}
```

2.3 NTT

```
/*
find_NTT_prime

        gen(1000000001);

        for (int i = 1; i < prime_N; ++ i) {
                int p = prime[i]; -- p;
                int s = 0;
                for (; !(p&1); p >>= 1) ++ s;

                p = prime[i];
                int PR = get_primitive_root(p, p-1);
                if (s >= 20) printf("%d %d %d %d\n", p, s, PR, pow_mod(PR, p-2, p));
        }

167772161 25 3
377487361 23 7
469762049 26 3
595591169 23 3
645922817 23 3
754974721 24 11
880803841 23 26
897581057 23 3
998244353 23 3
*/

typedef long long LL;

```

```
const int mod = 998244353;
const int m2k = 23;
const int PR = 3;
const int inv_PR = 332748118;

modn PRw[30], inv_PRw[30];

typedef long long LL;
typedef std::vector<modn> Vec;

int n;

void NTT(Vec& A, bool inv) {
        int n = A.size();

        for (int i = 0, j = 0; i < n; ++ i) {
                if (j > i) std::swap(A[i], A[j]);
                int k = n;
                while (j & (k >>= 1)) j &= ~k;
                j |= k;
        }

        for (int step = 1, j = 1; step < n; step <<= 1, ++ j) {
                modn wn = inv ? PRw[j] : inv_PRw[j];
                modn wnk = 1;
                for (int k = 0; k < step; ++ k) {
                        for (int Ek = k; Ek < n; Ek += (step<<1)) {
                                int Ok = Ek + step;
                                modn t = wnk * A[Ok];
                                A[Ok] = A[Ek] - t;
                                A[Ek] = A[Ek] + t;
                        }
                        wnk = wnk * wn;
                }
        }
        modn inv_n = pow_mod(n, mod-2);
        if (inv)
                for (int i = 0; i < n; ++ i) A[i] = A[i] * inv_n;
}

Vec t1, t2;
Vec operator *(const Vec& A, const Vec& B) {
        int a = A.size(), b = B.size(), S = 2;
        for (; S <= (a+b-1); S <<= 1);
        t1.resize(S); for (int i = 0; i < S; ++ i) t1[i] = (i < a ? A[i] : 0); NTT(t1, 0);
        t2.resize(S); for (int i = 0; i < S; ++ i) t2[i] = (i < b ? B[i] : 0); NTT(t2, 0);
        for (int i = 0; i < S; ++ i) t1[i] = t1[i] * t2[i]; NTT(t1, 1); t1.resize(a+b-1);
        return t1;
}

Vec a, b;
void solve() {
        PRw[m2k] = pow_mod(PR, (mod-1)/(1<<m2k));
        for (int i = m2k-1; i >= 0; -- i) PRw[i] = PRw[i+1] * PRw[i+1];
        inv_PRw[m2k] = pow_mod(inv_PR, (mod-1)/(1<<m2k));
        for (int i = m2k-1; i >= 0; -- i) inv_PRw[i] = inv_PRw[i+1] * inv_PRw[i+1];

        a.resize(2); a[0] = 1; a[1] = 2;
        b.resize(2); b[0] = 2; b[1] = 0;
        a = a * b;
        for (int i = 0; i < a.size(); ++ i) a[i].print();  putchar('\n');
}
```

2.4 一些结论

可逆矩阵循环节 从组合的角度可以算出来 Zp 下的 n 阶可逆阵形成的群的阶为：$(p^n-1)(p^n-p)(p^n-p^2)(p^n-p^3)....(p^n-p^{n-1})$。
对于 Zp 下可逆矩阵 A 的幂，可以看作是 A 做为群中的元素在某一个轨道上移动。那么它的 order 一定是这个群的阶的因素。所有群的阶一定是一个合法的循环节，当然它并不是最小的

3 字符串

3.1 Hash

```
#include <algorithm>
#include <cassert>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <deque>
#include <iostream>
#include <map>
#include <queue>
#include <vector>

typedef unsigned long long LL;
typedef std::pair<int, int> P;

```

```
const int hash = 123;
const int INF = 0x3f3f3f3f;
const int maxn = 500010;

struct StringHash{
        int MOD;
        char s[maxn]; int n; // the string is from 1 !!!!!!
        LL H[maxn],iH[maxn],xk[maxn];
        LL Hash(int l,int r){ // the Hash of [l,r)
                return ((H[l]−(H[r]*xk[r−l]%MOD))%MOD+MOD)%MOD;
        }
        LL iHash(int l,int r){ // the inverseHash of (r,l]
                return ((iH[l]−(iH[r]*xk[l−r]%MOD))%MOD+MOD)%MOD;
        }
        void init(int MOD,int n,char*s){
                this−>MOD=MOD; strcpy(this−>s+1,s); this−>n=n;
                H[n+1]=0;for(int i=n;i>0;−−i)H[i]=(H[i+1]*hash+this−>s[i])%MOD;
                iH[0]=0;for(int i=1;i<=n;++i)iH[i]=(iH[i−1]*hash+this−>s[i])%MOD;
                xk[0]=1;for(int i=1;i<=n;++i)xk[i]=(xk[i−1]*hash)%MOD;
//              printf("%I64d %I64d\n",Hash(3,5),iHash(2,0));
        }
        bool equal(int xl,int xr,int yl,int yr){
        // Determin [xl,xr)substring and [yl,yr)substring is equal
        // (the reverse string is accepted.)
                ++xl,++xr,++yl,++yr;
                if(abs(xr−xl)!=abs(yr−yl))return 0;
                return (xl<=xr ? Hash(xl,xr) : iHash(xl,xr))==(yl<=yr ? Hash(yl,yr) : iHash(yl,yr));
        }
};
```

```

int n;
char s[maxn];
StringHash H7,H9;
bool equal(int xl,int xr,int yl,int yr){
        return H7.equal(xl,xr,yl,yr) && H9.equal(xl,xr,yl,yr);
}

// 0:even, 1:odd;
bool check(int type, int L) {
        if (type == 0) {
                for (int i = L-1; i <= n-(L+1); ++ i)
                        if (equal(i,i-L,i+1,i+1+L))
                                return 1;
                return 0;
        } else {
                for (int i = L-1; i <= n-L; ++ i)
                        if (equal(i,i-L,i,i+L))
                                return 1;
                return 0;
        }
}

void solve() {
        scanf("%s", s); n = strlen(s);
        H7.init(1000000007,n,s);
        H9.init(1000000009,n,s);
        for (int t = 0; t < 2; ++ t) {
                int l = 0, r = n/2+1;
                while (l + 1 < r) {
                        int m = (l + r) >> 1;
                        if (check(t, m)) l = m; else r = m;
                }
                printf("%d␣%d\n", t, l*2-t);
        }
}

int main() {
        freopen("hash.in", "r", stdin);
//      freopen(".out", "w", stdout);
        solve();
        for(;;);

        return 0;
}
```

3.2 Manacher

```
int n, m;

char str[maxn], s[maxn << 1];
int f[maxn << 1];

void init() {
        scanf("%s", str); n = strlen(str);
        for (int i = 0; i < n; ++ i) {
                s[i<<1] = '*';
                s[i<<1|1] = str[i];
        }
        m = n<<1; s[m] = '*'; m ++;

        int id = 1, mx = 0; f[0] = 1;
        for (int i = 1; i < m; ++ i) {
                int p;
                if (mx > i)
                        p = std::min(f[id*2-i], mx-i);
                else
                        p = 1;

                while (i-p >= 0 && s[i-p] == s[i+p]) ++ p;
                f[i] = p;
                if (i+f[i]-1 > mx) {
                        id = i;
                        mx = i+f[i]-1;
                }
        }

        for (int i = 0; i < m; ++ i)
                printf("%d␣", f[i]);
        putchar('\n');
}
```

3.3 AC

```
const int maxsigma = 26;
const int maxnode = 500010;
const int maxs = 60;
const int maxS = 100010;

int idx(char x) { return x - 'a'; }

struct Trie {
        Trie *ch[maxsigma], *pre, *lst;
        int v;
        Trie() { memset(ch, 0, sizeof ch); lst = 0; v = 0; }

        int calc() { int x = (lst ? lst->calc() : 0) + v; v = 0; return x;}
} trie[maxnode], *rot, *trieR;

void insert(char *s) {
        int n = strlen(s);
        Trie *p = rot;
        for (int i = 0; i < n; ++ i) {
                int x = idx(s[i]);
                if (!p->ch[x]) {
                        p->ch[x] = trieR ++;
                        *p->ch[x] = Trie();
                }
                p = p->ch[x];
        }
        ++ p->v;
}

Trie* Q[maxnode]; int l, r;
void getFail() {
        rot->pre = rot;

        l = r = 0;
        for (int x = 0; x < maxsigma; ++ x) if (rot->ch[x]) {
                rot->ch[x]->pre = rot;
                Q[r ++] = rot->ch[x];
        } else
                rot->ch[x] = rot;

        for (; l != r; ++ l) {
                Trie *u = Q[l];
                for (int x = 0; x < maxsigma; ++ x) if (u->ch[x]) {
                        Trie *v = u->ch[x], *p = u->pre->ch[x];
                        v->pre = p;
                        v->lst = p->v ? p : p->lst;
                        Q[r ++] = v;
                } else
                        u->ch[x] = u->pre->ch[x];
        }
}

int n, m;
char s[maxs], S[maxS];

void init() {
        rot = trieR = trie; trieR ++;
        *rot = Trie();
        for (int i = 0; i < m; ++ i) {
                scanf("%s", s); insert(s);
        }
        getFail();
}

void solve() {
        scanf("%d", &m);
        init();
        scanf("%s", S); n = strlen(S);

        int res = 0;
        Trie *p = rot;
        for (int i = 0; i < n; ++ i)
                res += (p = p->ch[idx(S[i])])->calc();
        printf("%d\n", res);
}

int main() { }
```

3.4 SA

```
#include<algorithm>
#include<cstdio>
#include<cstring>
using namespace std;

typedef long long LL;

const int MAXN=100010;
const int MAXLG=18;

```

```
int lg2[MAXN];
int icmp(int x){ return x==0 ? 0 : (x<0 ? -1 : 1); }
const int MAXM=256;
int n;
struct SA{
        int n;
        char s[MAXN];
        int sa[MAXN],rk[MAXN],ht[MAXLG][MAXN];
        LL subs[MAXN]; // subs[n□□ǰ]:□□□□ʳ□□Ч□□□□n¸sa°°¬»¥²»µ®

        //==================================
        /*
        struct cmp{
                int* s;
                cmp(int* s):s(s){}
                bool operator()(const int& a,const int& b){
                        return s[a]<s[b];
                }
        };
        */
        //==================================

        void init(int n,char *a){
                this->n=n; for(int i=0;i<n;++i)s[i]=a[i];

                static int t[MAXN], c[MAXN];
                s[n++]=0;
                int *x=sa, *y=rk, *z=t;
                //==================================
                for(int i=0;i<MAXM;++i) c[i]=0;
                for(int i=0;i<n;++i) c[y[i]=s[i]]++;
                for(int i=1;i<MAXM;++i) c[i]+=c[i−1];
                for(int i=0;i<n;++i) x[−−c[y[i]]]=i;
                //==================================
                /*
                for(int i=0;i<n;++i) x[i]=i;
                sort(x,x+n,cmp(s));
                c[y[x[0]]=0]=0; int p=1;
                for(int i=1;i<n;++i){
                        if(!(s[x[i]]==s[x[i−1]]))c[p++]=i;
                        y[x[i]]=p−1;
                }
                */
                //==================================
                for(int k=1;k<n;k<<=1){
                        swap(x,z);
                        for(int i=0;i<n;++i){
                                int j=z[i]−k;
                                if(j<0) j+=n;
                                x[c[y[j]]++]=j;
                        }
                        swap(y,z);
                        c[y[x[0]]=0]=0; int p=1;
                        for(int i=1;i<n;++i){
                                if(!(z[x[i−1]]==z[x[i]]&&z[x[i−1]+k]==z[x[i]+k])) c[p++]=i; // □□h 鴗 □□□□ 磬 □□Ïǰh□□y□□□□□□□□□h
                                y[x[i]]=p−1;
                        }
                        if(p==n)break;
                }
                if(x!=sa){ memcpy(c,sa,sizeof(int)*n); memcpy(sa,x,sizeof(int)*n); if(y==sa)y=c; }
                if(y!=rk) memcpy(rk,y,sizeof(int)*n);

                ht[0][0]=0;
                for(int i=0,j=rk[0],k=0;i<n−1;++i,++k)
                        while(~k&&s[i]!=s[sa[j−1]+k]) ht[0][j]=k−−,j=rk[sa[j]+1];
                for(int j=1;j<MAXLG;++j)
                        for(int i=1;i+(1<<j)<=n;++i)
                                ht[j][i]=min(ht[j−1][i],ht[j−1][i+(1<<j−1)]);

                subs[0]=0; for(int i=1;i<=n;++i) subs[i]=subs[i−1]+(n−sa[i]−ht[0][i]);
        }
        int LCP(int x,int y){ // □□□□□x°yµLCP
                if(x==y)return n−x;
                x=rk[x]+1,y=rk[y];
                if(x>y)swap(x,y);
                int k=lg2[y−x+1];
                return min(ht[k][x],ht[k][y−(1<<k)+1]);
        }
        void kth(LL k,int &sl,int &sr){ // □□¸□□□□□Ӵ□e□□»®k¸(□□k´0¼)□□□□ч‚□£¬•µ»[sl,sr) Input: k Output sl,sr
                int p=upper_bound(subs,subs+n+1,k)−subs;
                if(p>n) sl=sr=−1;
                else sl=sa[p],sr=sa[p]+ht[0][p]+(k−subs[p−1])+1;
        }
        int scmp(int xl,int xr,int yl,int yr){ // □ж□[xl,xr□Ӵ)® □□ [yl,yr□Ӵ)®(−1,0,1)
                int t=LCP(xl,yl);
                if(t>=min(xr−xl,yr−yl)) return icmp((xr−xl)−(yr−yl));
                return icmp(s[xl+t]−s[yl+t]);
```

```
    }
    bool check(LL lam,int K){
        int sl,sr; kth(lam,sl,sr);
        int k=0;
        for(int r=n,l;r;r=l+1,++k){
            for(l=r-1;~l;--l)if(scmp(sl,sr,l,r)<0) break;
            if(l==r-1)return 0;
        }
        return k<=K;
    }

    void solve(int K){
        LL L=-1,R=subs[n]-1;
        while(L+1<R){
            LL M=(L+R)>>1;
            if(check(M,K))R=M;else L=M;
        }
        check(R,K);
        int sl,sr; kth(R,sl,sr);
        for(int i=sl;i<sr;++i)putchar(s[i]); putchar('\n');
    }
    void print(){
        puts(s);for(int i=0;i<n;++i)printf("%d",i%10);putchar('\n');puts("====================");
    }
} sa;

char s[MAXN];
void solve(){
    int K;scanf("%d",&K);
```

```
        scanf("%s",s); n=strlen(s);
        sa.init(n,s);
        sa.solve(K);
}

int main(){
//      freopen("in.txt","r",stdin);
        lg2[0]=-1; for(int i=1;i<MAXN;++i) lg2[i]=lg2[i>>1]+1;
        solve();
//      for(;;);
        return 0;
}
```

subs[n]: 求前 n 个 sa 包含互不相同的子串总个数

LCP: 求后缀 x 和后缀 y 的 LCP

kth: 求顺序互不相同子串中第 k 个 (k 从 0 计)，返回左右端点 [sl,sr) Input: k Output sl,sr

scmp: 判断 [xl,xr) 子串与 [yl,yr) 子串 (-1,0,1)

3.5 SAM

```
#include<algorithm>
#include<cstdio>
#include<cstring>
using namespace std;

const int maxn=100010;

char s[maxn];
int n;

```

```
// mx: □□□□´®  v: □□□□□´®³  fst: ´®  lst: ĩγ□□□□Ч„□´®´
struct Sam {
        Sam *pre, *ch[26];
        int mx, v, fst, lst;
} sam[maxn << 1], *samR, *r[maxn << 1], *rot, *now;
int num;

void insert(int x) {
        Sam *p = now, *np = samR ++;
        memset(np->ch, 0, sizeof np->ch);
        np->mx = p->mx + 1; np->v = 0; np->fst = n; np->lst = 0;

        for (; p && !p->ch[x]; p = p->pre) p->ch[x] = np;

        if (!p) np->pre = rot;
        else {
                Sam *q = p->ch[x];
                if (q->mx == p->mx + 1) np->pre = q;
                else {
                        Sam *nq = samR ++;
                        *nq = *q;
                        nq->mx = p->mx + 1;
                        np->pre = q->pre = nq;
                        for (; p && p->ch[x] == q; p = p->pre) p->ch[x] = nq;
                }
        }

        now = np;
}
```

```

char ss[maxn];
void dfs(int d,Sam* p){
        if(d)puts(ss);
        for(int x=0;x<26;++x)if(p->ch[x]){
                ss[d]=x+'a';
                dfs(d+1,p->ch[x]);
                ss[d]=0;
        }
}

int c[maxn];
void solve(){
        scanf("%s", s); n = strlen(s);

        rot = now = samR = sam; samR ++;
        memset(rot->ch, 0, sizeof rot->ch);
        rot->mx = rot->v = 0; rot->fst = n; rot->lst = 0;

        for (int i = 0; i < n; ++ i) insert(s[i]-'a');
        num = samR - sam;

        for (int i = 0; i <= n; ++ i) c[i] = 0;
        for (int i = 0; i < num; ++ i) ++ c[sam[i].mx];
        for (int i = 1; i <= n; ++ i) c[i] += c[i-1];
        for (int i = num-1; i >= 0; -- i) r[-- c[sam[i].mx]] = &sam[i];

        Sam *p = rot;
        for (int i = 0; i < n; ++ i) {
```

```
				p = p->ch[s[i]-'a'];
				++ p->v; p->fst = p->lst = i;
		}
		for (int i = num-1; i > 0; -- i) {
				r[i]->pre->v += r[i]->v;
				r[i]->pre->fst = min(r[i]->pre->fst, r[i]->fst);
				r[i]->pre->lst = max(r[i]->pre->lst, r[i]->lst);
		}

		dfs(0,rot);
}
int main(){
		freopen("in.txt","r",stdin);
		freopen("out.txt","w",stdout);
		int kase;scanf("%d",&kase);
		while(kase--)solve();
		return 0;
}
```

4 图论

4.1 最大流 Dinic

```
typedef pair<int,int> P;
struct Dinic {
		int n, tot, s, t;
		int st[MAXN], st0[MAXN];
		int lk[MAXM << 1], b[MAXM << 1], f[MAXM << 1]; bool del[MAXM << 1];
		int Q[MAXN]; int l, r;
```

```
        int d[MAXN];
        map<P,int> idx;
        int su[MAXN];

        void init(int n) {
                this->n = n;
                memset(st, 0, sizeof st); tot = 1;
                memset(su, 0, sizeof su);
                idx.clear();
        }

        void addedge(int u, int v, int w) {
//              printf("%d %d %d\n",u,v,w);
                if (!idx.count(P(u,v))) {
                        lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; del[tot] = 0; st[u] = tot;
                        lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; del[tot] = 0; st[v] = tot;
                        idx[P(u,v)] = tot-1; idx[P(v,u)] = tot;
                } else {
                        f[idx[P(u,v)]] += w;
                }
                su[u] += w; su[v] += w;
        }
        bool BFS() {
                memset(d, 0, sizeof d);
                l = r = 0;
                d[ Q[r ++] = s ] = 1;
                for (; l != r; ++ l) {
                        int u = Q[l];
                        for (int i = st[u]; i; i = lk[i]) if (!del[i]) {
                                int v = b[i];
                                if (f[i] && !d[v]) {
                                        d[v] = d[u] + 1;
                                        Q[r ++] = v;
                                }
                        }
                }
                return d[t];
        }
        int DFS(int u, int a) {
                if (u == t || a == 0) return a;
                int flow = 0, df;
                for (int& i = st0[u]; i; i = lk[i]) if (!del[i]) {
                        int v = b[i];
                        if (d[v] == d[u] + 1 && (df = DFS(v, std::min(a, f[i])))) {
                                f[i] -= df; f[i^1] += df;
                                a -= df; flow += df;

                                if (a == 0) break;
                        }
                }
                return flow;
        }
        void solve(int s, int t, int& flow) {
                this->s = s; this->t = t;
                while (BFS()) {
                        memcpy(st0, st, sizeof st0);
                        flow += DFS(s, INF);
                }
        }
        bool aug(int s, int t) {
                this->s = s; this->t = t;
                if (BFS()) {
                        memcpy(st0, st, sizeof st0);
                        DFS(s, 1);
                        return 1;
                }
                return 0;
        }
        int flow(int u, int v) {
                return f[idx[make_pair(u,v)]^1];
        }
        void dt(int e) {
                del[e] ^= 1; del[e^1] ^= 1;
        }
        void dt(int u, int v) {
                dt(idx[make_pair(u,v)]);
        }
        int cap(int u) {
                return su[u];
        }
        int isCut(int u, int v) {
                if(!idx[make_pair(u,v)]) return 1;
                return d[u] && !d[v] && f[idx[make_pair(u,v)]^1];
        }
        bool inS(int u){
                return d[u];
        }
        void clear() {
                for (int i = 2; i <= tot; i += 2) {
                        f[i] += f[i^1]; f[i^1] = 0;
                }
        }
} solver;
```

4.2 最大流 ISAP

```
struct ISAP {
        int n, tot, s, t;
        int st[MAXN], st0[MAXN];
        int lk[MAXM << 1], b[MAXM << 1], f[MAXM << 1];
        int Q[MAXN]; int l, r;
        int d[MAXN], p[MAXN];
        int num[MAXN];

        void init(int n) { // all indices should < n
                this->n = n;
                memset(st, 0, sizeof st); tot = 1;
        }
        void addedge(int u, int v, int w) {
                lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; st[u] = tot;
                lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; st[v] = tot;
        }
        bool BFS() {
                for (int u = 0; u < n; ++ u) d[u] = n;
                l = r = 0;
                d[ Q[r ++] = t ] = 0;
        for (; l != r; ++ l) {
            int u = Q[l];
            for (int i = st[u]; i; i = lk[i]) {
                int v = b[i];
                if (f[i^1] && d[v] >= n) {
                    d[v] = d[u] + 1;
                    Q[r ++] = v;
                }
            }
        }
        return ~d[s];
    }

    int augment() {
        int a = INF;
        for (int u = t; u != s; ) {
            int i = p[u];
            a = min(a, f[i]);
            u = b[i^1];
        }
        for (int u = t; u != s; ) {
            int i = p[u];
            f[i] -= a; f[i^1] += a;
            u = b[i^1];
        }
        return a;
    }

    void solve(int s, int t, int& flow) {
```

```
                this->s = s; this->t = t;
                BFS();
                memset(num, 0, sizeof num);
                for (int i = 0; i < n; ++ i) ++ num[d[i]];
                memcpy(st0, st, sizeof st0);
                for (int u = s; d[s] < n; ) {
                        if (u == t) {
                                flow += augment();
                                u = s;
                        }
                        int ok = 0;
                        for (int& i = st0[u]; i; i = lk[i]) {
                                int v = b[i];
                                if (f[i] && d[u] == d[v] + 1) {
                                        ok = 1;
                                        p[v] = i;
                                        u = v;
                                        break;
                                }
                        }
                        if (!ok) {
                                int mn = n-1;
                                for (int i = st0[u] = st[u]; i; i = lk[i]) {
                                        int v = b[i];
                                        if (f[i] && d[v] < mn) mn = d[v];
                                }
                                if ((-- num[d[u]]) == 0) break;
                                ++ num[ d[u] = mn+1 ];
                                if (u != s)
```

```
                                                u = b[p[u]^1];
                                }
                        }
                }
} solver;
```

4.3 最小费用最大流

```
struct MCMF {
        int n, tot, s, t;
        int st[MAXN], lk[MAXM << 1], b[MAXM << 1], c[MAXM << 1], f[MAXM << 1];
        int Q[MAXN]; int l, r;
        int inq[MAXN];
        int d[MAXN], p[MAXN], a[MAXN];

        void init(int n) {
                this->n = n;
                memset(st, 0, sizeof st); tot = 1;
        }
        void addedge(int u, int v, int w, int x) {
                lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; c[tot] = x; st[u] = tot;
                lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; c[tot] = -x; st[v] = tot;
//              printf("%d %d %d %d\n", u, v, w, x);
        }
        bool SPFA(int& flow, int& cost) {
                l = r = 0;
                memset(d, 0x3f, sizeof d);
                memset(inq, 0, sizeof inq);
                d[ Q[r ++] = s ] = 0; inq[s] = 1; a[s] = INF;
        for (; l != r; ) {
            int u = Q[l]; l = (l+1 == MAXN ? 0 : l+1); inq[u] = 0;
            for (int i = st[u]; i; i = lk[i]) {
                int v = b[i];
                if (f[i] && d[v] > d[u] + c[i]) {
                    d[v] = d[u] + c[i];
                    p[v] = i;
                    a[v] = min(a[u], f[i]);
                    if (!inq[v]) {
                        if (l == r || d[v] < d[Q[l]]) {
                            l = (l-1 == -1 ? MAXN-1 : l-1);
                            Q[l] = v;
                        } else {
                            Q[r] = v;
                            r = (r+1 == MAXN ? 0 : r+1);
                        }
                        inq[v] = 1;
                    }
                }
            }
        }

        if (d[t] == INF) return 0;
        flow += a[t];
        cost += a[t] * d[t];

        for (int u = t; u != s; ) {
            int i = p[u];
            f[i] -= a[t]; f[i^1] += a[t];
```

```
                        u = b[i^1];
                }
                return 1;
        }
        void solve(int s, int t, int& flow, int& cost) {
                this->s = s; this->t = t;
                while (SPFA(flow, cost));
        }
} solver;
```

4.4 KM

```
#define FOR(i, a, b) for (int i = (a); i <= (b); ++ i)

int n;
int W[maxn][maxn];
int Lx[maxn], Ly[maxn], slack[maxn];
int left[maxn];
bool S[maxn], T[maxn];

bool match(int i) {
        S[i] = 1;
        FOR(j, 1, n) if (!T[j]) {
                int tmp = Lx[i]+Ly[j] - W[i][j];
                if (tmp == 0) {
                        T[j] = 1;
                        if (!left[j] || match(left[j])) {
                                left[j] = i;
                                return 1;
                        }
                }
                else if (tmp < slack[j]) slack[j] = tmp;
        }
        return 0;
}

void update() {
        int a = INF;
        FOR(i, 1, n) if (!T[i] && slack[i] < a)  a = slack[i];
        FOR(i, 1, n) { if (S[i]) Lx[i] -= a; if (T[i]) Ly[i] += a; }
}

void KM() {
        FOR(i, 1, n) {
                left[i] = 0; Lx[i] = Ly[i] = 0;
                FOR(j, 1, n) Lx[i] = max(Lx[i], W[i][j]);
        }
        FOR(i, 1, n) for (;;) {
                FOR(j, 1, n) { S[j] = T[j] = 0; slack[j] = INF; }
                if (match(i)) break; else update();
        }
}

void init() {
        scanf("%d", &n);
        for (int i = 1; i <= n; ++ i)
                for (int j = 1; j <= n; ++ j)
                        scanf("%d", &W[i][j]);
```

```
}

void solve() {
        KM();
        int ans = 0;
        for (int i = 1; i <= n; ++ i)
                ans += W[i][left[i]];
        printf("%d\n", ans);
}
```

4.5 2-SAT

```
int n;

struct TwoSAT {
        int n;
        vector<int> G[maxn<<1];
        bool mark[maxn];
        int a[maxn<<1], aN;

        void init(int n) : n(n) {
                for (int i = 0; i < n*2; ++ i) G[i].clear();
                memset(mark, 0, sizeof mark);
        }
        void addedge(int u, int uval, int v, int vval) {
                u = u * 2 + uval; v = v * 2 + vval;
                G[u].push_back(v);
        }
        bool dfs(int x) {
                if (mark[x^1]) return 0;
                if (mark[x]) return 1;
                mark[x] = 1;
                a[aN ++] = x;
                for (int i = 0; i < G[x].size(); ++ i) if (!dfs(G[x][i])) return 0;
                return 1;
        }
        bool solve() {
                for (int i = 0; i < n*2; i += 2) if (!mark[i] && !mark[i^1]) {
                        aN = 0;
                        if (!dfs(i)) {
                                for (int i = 0; i < aN; ++ i) mark[a[i]] = 0;
                                if (!dfs(i^1)) return 0;
                        }
                }
                return 1;
        }
} TS;
```

4.6 割点、割边

```
int n, m;
int st[maxn], lk[maxm << 1], b[maxm << 1];
int tot;
void addedge(int u, int v) {
        lk[++ tot] = st[u]; b[tot] = v; st[u] = tot;
}

void init() {
        scanf("%d%d", &n, &m);

        memset(st, 0, sizeof st); tot = 1;
        for (int i = 1; i <= m; ++ i) {
                int u, v; scanf("%d%d", &u, &v);
                addedge(u, v); addedge(v, u);
        }
}

int isnode[maxn];
int dfn[maxn], low[maxn];
int dfs_clock;
void dfs1(int u, int fa) {
        int ch = 0;
        dfn[u] = low[u] = ++ dfs_clock;
        for (int i = st[u]; i; i = lk[i]) if ((i^1) != fa) {
                int v = b[i];
                if (!dfn[v]) {
                        ++ ch;
                        dfs1(v, i);
                        if (low[v] >= dfn[u])
                                isnode[u] = 1;
                        low[u] = std::min(low[u], low[v]);
                } else {
                        low[u] = std::min(low[u], dfn[v]);
                }
        }
        if (dfn[u] == 1 && ch <= 1) isnode[u] = 0;
}
```

```
void cut_node() {
        memset(dfn, 0, sizeof dfn); dfs_clock = 0;
        dfs1(1, 0);

        int s = 0; for (int i = 1; i <= n; ++ i) if (isnode[i]) ++ s;

        if (s == 0) puts("Null");
        else {
                int t = 0;
                for (int i = 1; i <= n; ++ i) if (isnode[i]) {
                        ++ t; printf("%d%c", i, t == s ? '\n' : '␣');
                }
        }
}

std::vector<P> edges;

void dfs2(int u, int fa) {
        dfn[u] = low[u] = ++ dfs_clock;
        for (int i = st[u]; i; i = lk[i]) if ((i^1) != fa){
                int v = b[i];
                if (!dfn[v]) {
                        dfs2(v, i);
                        if (low[v] == dfn[v]) {
                                if (u <= v)
                                        edges.push_back(P(u, v));
                                else
                                        edges.push_back(P(v, u));
                        }
                        low[u] = std::min(low[u], low[v]);
                } else {
                        low[u] = std::min(low[u], dfn[v]);
                }
        }
}

void cut_edge() {
        memset(dfn, 0, sizeof dfn); dfs_clock = 0;
        dfs2(1, 0);

        sort(edges.begin(), edges.end());
        for (int i = 0; i < (int)edges.size(); ++ i)
                printf("%d␣%d\n", edges[i].first, edges[i].second);
}
```

5 计算几何

5.1 模板

```
#include<algorithm>
#include<cmath>
#include<vector>
using namespace std;

const double PI = acos(-1.0);
const double eps = 1e-5;
const double INF = 1e5;

```

```
int dcmp(double x) { if (fabs(x) < eps) return 0; else return x < 0 ? -1 : 1; }

// Point Vector
struct Point {
        double x, y;
        Point() { }
        Point(double x, double y) : x(x), y(y) { }
        bool operator < (const Point &A) const {
                return dcmp(x-A.x) < 0 || (dcmp(x-A.x) == 0 && dcmp(y-A.y) < 0);
        }
        void read(){
                double x,y; scanf("%lf%lf",&x,&y);
                this->x=x, this->y=y;
        }
};
typedef Point Angle;
typedef Point Vector;
Point operator + (const Point &A, const Point &B)       { return Point(A.x+B.x, A.y+B.y); }
Point operator - (const Point &A, const Point &B)       { return Point(A.x-B.x, A.y-B.y); }
Point operator * (const Point &A, double b)                     { return Point(A.x*b, A.y*b); }
Point operator / (const Point &A, double b)                     { return Point(A.x/b, A.y/b); }
bool operator == (const Point &A, const Point &B)       { return dcmp(A.x-B.x) == 0 && dcmp(A.y-B.y) == 0; }
double Dot(const Point &A, const Point &B)                      { return A.x*B.x + A.y*B.y; }
double Cross(const Point &A, const Point &B)            { return A.x*B.y - A.y*B.x; }

double Length(const Point &A) { return sqrt(Dot(A, A)); }
double Area2(const Point &A, const Point &B, const Point &C) { return Cross(B-A, C-A); }
Point Rotate(const Point &A, double ang) {
        return Point(cos(ang)*A.x - sin(ang)*A.y, sin(ang)*A.x + cos(ang)*A.y);
```

```
}
Point Normal(const Point &A) {
        double L = Length(A); return Point(-A.y/L, A.x/L);
}

// Line Segment
struct Line {
        Point A, B, v;
        double ang;

        Line() { }
        Line(Point A, Point B): A(A), B(B) { v = B-A; ang = atan2(v.y, v.x); }
        // ax + by + c > 0
        Line(double a, double b, double c) {
                v = Point(b, -a);
                if (fabs(a) > fabs(b)) A = Point(-c/a, 0); else A = Point(0, -c/b);
                B = A + v;
                ang = atan2(v.y, v.x);
        }
        bool operator < (const Line &A) const { return ang < A.ang; }
        Point point(double t) const { return A + v*t; }
        double pos(Point p) const { return Dot(p-A,v)/Dot(v,v); }
};

// t[0] is the t of L1
int LineLineIntersection(const Line& L1, const Line& L2, Point* p, double* t=0) {
        if(dcmp(Cross(L1.v,L2.v))==0)
                return dcmp(Cross(L1.v,L2.A-L1.A))==0 ? -1 : 0;
        p[0]=L1.point(Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v));
        if(t) t[0]=Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v);
        return 1;
}

double DistanceToLine(const Point &P, const Line &L) {
        return fabs(Cross(L.v, P-L.A)) / Length(L.v);
}
Point GetLineProjection(const Point &P, const Line &L) {
        return L.point(Dot(P-L.A, L.v) / Dot(L.v, L.v));
}
int Position(const Point &P, const Line &L) { return dcmp(Cross(L.v, P-L.A)); }

double DistanceToSegment(const Point &P, const Line &L) {
        if (dcmp(Dot(L.v, P-L.A)) < 0) return Length(P-L.A);
        else if (dcmp(Dot(L.v, P-L.B) > 0)) return Length(P-L.B);
        else return DistanceToLine(P, L);
}
bool SegmentProperIntersection(const Line &L1, const Line &L2) {
        int     c1 = Position(L2.A, L1), c2 = Position(L2.B, L1),
                c3 = Position(L1.A, L2), c4 = Position(L1.B, L2);
        return c1*c2 < 0 && c3*c4 < 0;
}

int SegmentSegmentIntersection(const Line& L1, const Line& L2, Point* p, double* t=0) {
        if(dcmp(Cross(L1.v,L2.v))==0)
                return dcmp(Cross(L1.v,L2.A-L1.A))==0 ? -1 : 1;
        p[0]=L1.point(Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v));
        if(t) t[0]=Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v);
        return 1;
```

```
}

bool OnSegment(const Point &P, const Line &L) {
        return  dcmp(Cross(L.A-P, L.B-P)) == 0 &&
                        dcmp(Dot(L.A-P, L.B-P)) <= 0;
}
bool OnSegment2(const Point &P, const Line &L) {
        return dcmp(L.A.x - P.x) * dcmp(L.B.x - P.x) <= 0 && dcmp(L.A.y - P.y) * dcmp(L.B.y - P.y) <= 0;
}

// Polygon
typedef vector<Point> Polygon;
typedef std::vector<Line> Lines;
void print(const Polygon& A) {
        for (int i = 0; i < A.size(); ++ i) printf("%.2lf␣%.2lf\n", A[i].x, A[i].y);
}

double PolygonArea(const Polygon &p) {
        int n = p.size();
        double area = 0;
        for (int i = 1; i < n-1; ++ i)
                area += Area2(p[0], p[i], p[i+1]);
        return area / 2.0;
}
int isPointInPolygon(const Point &P, const Polygon &p) {
        int n = p.size();
        int wn = 0;
        for (int i = 0; i < n; ++ i) {
                const Point &p1 = p[i]; const Point &p2 = p[(i+1)%n];
                if (P == p1 || P == p2 || OnSegment(P, Line(p1, p2))) return -1;
                int k = dcmp(Cross(p2-p1, P-p1));
                int d1 = dcmp(p1.y-P.y);
                int d2 = dcmp(p2.y-P.y);
                if (k > 0 && d1 <= 0 && d2 > 0) ++ wn;
                if (k < 0 && d2 <= 0 && d1 > 0) -- wn;
        }
        return wn != 0;
}
// ConvexHull
Polygon ConvexHull(Polygon p) {
        int n = p.size();
        sort(p.begin(), p.end());

//      int n = p.erase(std::unique(p.begin(), p.end()), p.end());
        Polygon q(n+1);
        int m = 0;
        for (int i = 0; i < n; ++ i) {
                while (m > 1 && Cross(q[m-1]-q[m-2], p[i]-q[m-2]) <= 0) -- m;
                q[m ++] = p[i];
        }
        int k = m;
        for (int i = n-2; i >= 0; -- i) {
                while (m > k && Cross(q[m-1]-q[m-2], p[i]-q[m-2]) <= 0) -- m;
                q[m ++] = p[i];
        }
        if (n > 1) m --;
        q.resize(m);
        return q;
```

```
}
double ConvexHullMaxDist(const Polygon &poly) {
        int n = poly.size();
        if (n == 2) return Length(poly[1]-poly[0]);

        int i = min_element(poly.begin(), poly.end()) - poly.begin(),
                j = max_element(poly.begin(), poly.end()) - poly.begin();
        double res = 0.0;
        for (int si = i, sj = j; i != sj || j != si; ) {
                res = max(res, Length(poly[j]-poly[i]));
                if (Cross(poly[(i+1)%n]-poly[i], poly[(j+1)%n]-poly[j]) < 0) {
                        i = (i+1) % n;
                } else {
                        j = (j+1) % n;
                }
        }
        return res;
}
double ConvexHullMinDist(const Polygon &poly) {
        int n = poly.size();
        if (n == 2) return Length(poly[1]-poly[0]);

        int i = min_element(poly.begin(), poly.end()) - poly.begin(),
                j = max_element(poly.begin(), poly.end()) - poly.begin();
        double res = INF;
        for (int si = i, sj = j; i != sj || j != si; ) {
                if (Cross(poly[(i+1)%n]-poly[i], poly[(j+1)%n]-poly[j]) < 0) {
                        res = min(res, DistanceToSegment(poly[j], Line(poly[i], poly[(i+1)%n])));
                        i = (i+1) % n;
                } else {
                        res = min(res, DistanceToSegment(poly[i], Line(poly[j], poly[(j+1)%n])));
                        j = (j+1) % n;
                }
        }
        return res;
}
double MinDist(const Polygon& A, const Polygon& B){
        int n=A.size(), m=B.size();
//      if (n<3||m<3) for(;;);

        int i = min_element(A.begin(), A.end()) - A.begin(),
                j = max_element(B.begin(), B.end()) - B.begin();
        double res = INF;
        int si = i, sj = j;
        do {
                if (Cross(A[(i+1)%n]-A[i], B[(j+1)%m]-B[j]) < 0) {
                        res = min(res, DistanceToSegment(B[j], Line(A[i], A[(i+1)%n])));
                        i = (i+1) % n;
                } else {
                        res = min(res, DistanceToSegment(A[i], Line(B[j], B[(j+1)%m])));
                        j = (j+1) % m;
                }
        } while (i != si || j != sj);
        return res;
}

Polygon simplify(const Polygon& poly){
        Polygon ans;
        int n=poly.size();
        for(int i=0;i<n;++i){
                Point a=poly[i], b=poly[(i+1)%n], c=poly[(i+2)%n];
                if(dcmp(Cross(a-b,c-b))!=0) ans.push_back(b);
        }
        return ans;
}

// HalfplaneIntersection
Polygon CutPolygon(const Polygon &poly, Line L) {
        Polygon newpoly;
        int n = poly.size();
        for (int i = 0; i < n; ++ i) {
                const Point &p1 = poly[i]; const Point &p2 = poly[(i+1)%n];
                const Line &l = Line(p1, p2);
                if (Position(p1, L) >= 0) newpoly.push_back(p1);

                Point ip; int x=LineLineIntersection(L,l,&ip);
                if (x==1){
                        if (OnSegment2(ip, l)) newpoly.push_back(ip);
                }
        }
        return newpoly;
}
Polygon HalfplaneIntersection(Lines L) {
        int n = L.size();
        sort(L.begin(), L.end());

        Lines Q(n); int l, r;
        Polygon P(n);
        Polygon ans;

        l = r = 0; Q[r ++] = L[0];
        for (int i = 1; i < n; ++ i) {
                while (l+1 < r && Position(P[r-2], L[i]) <= 0) -- r;
                while (l+1 < r && Position(P[l], L[i]) <= 0) ++ l;
                Q[r ++] = L[i];
                if (dcmp(Cross(Q[r-2].v, Q[r-1].v)) == 0) {
                        -- r;
                        if (Position(L[i].A, Q[r-1]) > 0)
                                Q[r-1] = L[i];
                }
                if (l+1 < r) LineLineIntersection(Q[r-2], Q[r-1], &P[r-2]);
        }
        while (l+1 < r && Position(P[r-2], Q[l]) <= 0) -- r;
        if (l+2 >= r) return ans;
        LineLineIntersection(Q[r-1], Q[l], &P[r-1]);

        for (int i = l; i < r; ++ i) ans.push_back(P[i]);
        return ans;
}


//Angle
typedef Point Angle;
Point Rotate(const Point& A, const Angle& a) {
        return Point(a.x*A.x-a.y*A.y, a.y*A.x+a.x*A.y);
}
```

```

//Circle
struct Circle{
        Point c;double r;
        Circle(){}
        Circle(Point c,double r):c(c),r(r){}
        Point point(double ang) const{
                return Point(c.x+cos(ang)*r,c.y+sin(ang)*r);
        }
        Point point(double cosa,double sina) const{
                return Point(c.x+cosa*r,c.y+sina*r);
        }
};
int LineCircleIntersection(const Line& L,const Circle& C,Point* p,double* t=0){
        double a=L.v.x, b=L.A.x-C.c.x, c=L.v.y, d=L.A.y-C.c.y;
        double e=a*a+c*c, f=2*(a*b+c*d), g=b*b+d*d-C.r*C.r;
        double delta=f*f-4*e*g;
        if(dcmp(delta)<0)return 0;
        if(dcmp(delta)==0){
                p[0]=L.point(-f/(2*e));
                if(t) t[0]=-f/(2*e);
                return 1;
        }
        p[0]=L.point((-f-sqrt(delta))/(2*e)),p[1]=L.point((-f+sqrt(delta))/(2*e));
        if(t) t[0]=(-f-sqrt(delta))/(2*e), t[1]=(-f+sqrt(delta))/(2*e);
        return 2;
}
int CircleCircleIntersection(const Circle& C1,const Circle& C2,Point* p,Angle* a=0){
        double d=Length(C2.c-C1.c);
        if(dcmp(d)==0){
                if(dcmp(C1.r-C2.r)==0) return -1; // □غ□
                return 0; // ں□¬(1)
        }
        if(dcmp(C1.r+C2.r-d)<0) return 0; // □□□□
        if(dcmp(fabs(C1.r-C2.r)-d)>0) return 0; // ں□¬(2)
        double cosa=(C1.r*C1.r+d*d-C2.r*C2.r)/(2*C1.r*d), sina=sqrt(1-cosa*cosa);

        Point v=(C2.c-C1.c)/Length(C2.c-C1.c)*C1.r;

        p[0]=C1.c+Rotate(v,Angle(cosa,-sina)), p[1]=C1.c+Rotate(v,Angle(cosa,sina));
        if (a) a[0]=Rotate(v,Angle(cosa,-sina))/Length(v), a[1]=Rotate(v,Angle(cosa,sina))/Length(v);
        if(p[0]==p[1]) return 1; else return 2;
}
Circle CircumscribedCircle(const Point& A, const Point& B, const Point& C){
        double da=Dot(B-A,C-A),db=Dot(A-B,C-B),dc=Dot(A-C,B-C);
        double ka=db*dc,kb=da*dc,kc=da*db;
        Point D=(A*(kb+kc)+B*(ka+kc)+C*(ka+kb))/(2*(ka+kb+kc));
        return Circle(D,Length(A-D));
}
Circle InscribedCircle(const Point& A, const Point& B, const Point& C){
        double a=Length(C-B),b=Length(C-A),c=Length(B-A);
        Point p=(A*a+B*b+C*c)/(a+b+c);
        return Circle(C,fabs(Area2(A,B,C))/(a+b+c));
}
int CirclePointTangents(const Circle& C,const Point& P,Point* p,Angle* ang=0){
        Point v=C.c-P; double d=Length(v);
        if(dcmp(d-C.r)<0) return 0;
        if(dcmp(d-C.r)==0){
                p[0]=P;
                if(ang) ang[0]=Rotate(v,PI/2);
                return 1;
        }

        double a=C.r,c=d,b=sqrt(c*c-a*a);
        double sina=a/c, cosa=b/c;
        p[0]=P+Rotate(v,Angle(cosa,-sina))/c*b,p[1]=P+Rotate(v,Angle(cosa,sina))/c*b;
        if(ang) ang[0]=Rotate(v,Angle(cosa,-sina))/c, ang[1]=Rotate(v,Angle(cosa,sina))/c;
        return 2;
}

// ==================== CirclePolygonIntersectionArea ====================
double SectorArea(const Circle& C, const Point& A, const Point& B) {
        double ang=atan2(A.y,A.x)-atan2(B.y,B.x);
        while(ang<=0) ang+=2*PI;
        while(ang>2*PI) ang-=2*PI;
        ang=min(ang,2*PI-ang);
        return C.r*C.r*ang/2;
}
double CircleTriangleIntersectionArea(Circle C, Point A, Point B) {
        A=A-C.c, B=B-C.c; C.c=Point(0,0);
        int sgn=dcmp(Cross(A, B));
        if(sgn==0) return 0;

        Line L=Line(A,B);
        Point p[2];
        int num=0;
        int ina=dcmp(Length(A)-C.r)<0;
```

```
		int inb=dcmp(Length(B)-C.r)<0;
		if(ina){
			if(inb){
				return sgn*(fabs(Cross(A,B))/2.0);
			}else{
				LineCircleIntersection(L,C,p);
				return sgn*(fabs(Cross(A,p[1]))/2.0+SectorArea(C,p[1],B));
			}
		}else{
			if(inb) {
				LineCircleIntersection(L,C,p);
				return sgn*(fabs(Cross(B,p[0]))/2.0+SectorArea(C,p[0],A));
			}else{
				int num=LineCircleIntersection(L,C,p);
				if(num==2 && OnSegment2(p[0],L) && OnSegment2(p[1],L)){
					return sgn*(SectorArea(C,A,p[0])+SectorArea(C,p[1],B)+fabs(Cross(p[0],p[1]))/2.0);
				}else{
					return sgn*(SectorArea(C,A,B));
				}
			}
		}
}
double CirclePolygonIntersectionArea(const Circle& C,const Polygon& P) {
		int n=P.size();
		double res=0;
		for(int i=0;i<n;++i){
			res+=CircleTriangleIntersectionArea(C,P[i],P[(i+1)%n]);
		}
		return res;
```

```
}

// □□□□□□□□□□□□n¸£¬ans_k±´>=□□□□kµ
void CirclesIntersectionArea(vector<Circle>& vc,double* ans) {
        int n=vc.size();
        for(int i=1;i<=n;++i) ans[i]=0;
        static vector< pair<double, int> > ev;
        for(int i=0;i<n;++i){
                int cv=0;
                ev.clear();
                for(int j=0;j<n;++j)if(j!=i){
                        Point sol[2], ang[2];
                        int t=CircleCircleIntersection(vc[i],vc[j],sol,ang);
                        if(t==2) {
                                double a1=atan2(ang[0].y,ang[0].x);
                                double a2=atan2(ang[1].y,ang[1].x);
                                if(a1<a2)ev.push_back(make_pair(a1,1)),ev.push_back(make_pair(a2,-1));
                                else    ev.push_back(make_pair(a1,1)),ev.push_back(make_pair(PI,-1)),
                                                ev.push_back(make_pair(-PI,1)),ev.push_back(make_pair(a2,-1));
                        }else if(t==-1){
                                if(i<j) ++ cv;
                        }else{
                                int rd=dcmp(Length(vc[i].c-vc[j].c)-(vc[j].r-vc[i].r));
                                if(rd<=0) ++ cv;
                        }
                }
                ev.push_back(make_pair(-PI,1)),ev.push_back(make_pair(PI,-1));
                sort(ev.begin(),ev.end());

                double lst=-PI;
                for(int l=0,r;l<ev.size();){
                        double a=ev[l].first-lst;
                        ans[cv]+=vc[i].r*vc[i].r*(a-sin(a))/2.0+(Cross(vc[i].point(lst),vc[i].point(ev[l].first))/2.0);
                        lst=ev[l].first;

                        for(r=l+1;r<ev.size()&&ev[r].first<=ev[l].first;++r);
                        for(int p=l;p<r;++p) cv+=ev[p].second;
                        l=r;
                }
        }
}

// □□□□□□□□□ɋ□□□□□□□□□²¢¬µ±°¿´n±(n□□□□->)ơγ□o□□□□□□□□□ɦ□□£¬°-´£¬□□□□□h}□µ
void PolygonsIntersectionArea(vector<Polygon>& ps,double* ans) {
        int n=ps.size();
        for(int i=0;i<n+1;++i) ans[i]=0;

        static vector< pair<double, int> > ev;
        for(int i=0;i<n;++i) for(int pi=0;pi<ps[i].size();++pi){
                Line L0=Line(ps[i][pi],ps[i][(pi+1)%ps[i].size()]);
                ev.clear();
                for(int j=0;j<n;++j)if(i!=j) for(int pj=0;pj<ps[j].size();++pj){
                        Line L=Line(ps[j][pj],ps[j][(pj+1)%ps[j].size()]);

                        int p1=Position(L.A,L0), p2=Position(L.B,L0);
                        if(!p1 && !p2){
                                if(i<j && dcmp(Dot(L0.v,L.v))>0){
                                        ev.push_back(make_pair(min(max(L0.pos(L.A),0.0),1.0), 1));
```

```
					ev.push_back(make_pair(min(max(L0.pos(L.B),0.0),1.0), -1));
				}
			}else{
				Point p; double t;
				int x=LineLineIntersection(L0,L,&p,&t);
				if(p1>=0 && p2<0) ev.push_back(make_pair(min(max(t,0.0),1.0), 1));
				if(p1<0 && p2>=0) ev.push_back(make_pair(min(max(t,0.0),1.0), -1));
			}
		}
		ev.push_back(make_pair(0.0,1));
		ev.push_back(make_pair(1.0,-1));
		sort(ev.begin(),ev.end());

		int cv=0;
		double S0=Cross(L0.A,L0.B)/2.0;
		double lst=0;
		for(int l=0,r;l<ev.size();){
			ans[cv]+=S0*(ev[l].first-lst);
			lst=ev[l].first;

			for(r=l+1;r<ev.size()&&ev[r].first<=ev[l].first;++r);
			for(int p=l;p<r;++p) cv+=ev[p].second;
			l=r;
		}
	}
}

// examples
int CircleThroughAPointAndTangentToALineWithRadius(const Point& P, const Line& L, double r, Point* p){
        Point v=Normal(L.v)*r;
        Line l1=Line(L.A+v, L.v), l2=Line(L.A-v, L.v);
        Circle CP=Circle(P,r);
        int x=0;
        x+=LineCircleIntersection(l1,CP,p+x);
        x+=LineCircleIntersection(l2,CP,p+x);
        return x;
}

int CircleTangentToTwoLinesWithRadius(const Line& L1, const Line& L2, double r, Point* p){
        Point v1=Normal(L1.v)*r,v2=Normal(L2.v)*r;
        int x=0;
        x+=LineLineIntersection(Line(L1.A+v1,L1.v), Line(L2.A+v2,L2.v),p);
        x+=LineLineIntersection(Line(L1.A-v1,L1.v), Line(L2.A+v2,L2.v),p);
        x+=LineLineIntersection(Line(L1.A+v1,L1.v), Line(L2.A-v2,L2.v),p);
        x+=LineLineIntersection(Line(L1.A-v1,L1.v), Line(L2.A-v2,L2.v),p);
        return x;
}
int CircleTangentToTwoDisjointCirclesWithRadius(const Circle& C1, const Circle& C2, double r, Point* p){
        int x=0;
        x+=CircleCircleIntersection(Circle(C1.c,C1.r+r), Circle(C2.c,C2.r+r), p);
        return x;
}

int main() {
        return 0;
}
```

5.2 最小覆盖圆

```
int n;

struct Point {
        double x, y;
} p[maxn];

Point C; double r;

Point getCir(const Point &A, const Point &B, const Point &C) {
        Point tmp;
        double  a1 = B.x-A.x, b1 = B.y-A.y, c1 = (a1*a1 + b1*b1) / 2.0,
                        a2 = C.x-A.x, b2 = C.y-A.y, c2 = (a2*a2 + b2*b2) / 2.0,
                        d = a1*b2 - a2*b1;
        tmp.x = A.x + (c1*b2-c2*b1) / d;
        tmp.y = A.y + (a1*c2-a2*c1) / d;
        return tmp;
}

void minCircle() {
        std::random_shuffle(p+1, p+1+n);

        C = p[1]; r = 0;
        for (int i = 2; i <= n; ++ i) if (Length(p[i]-C) > r + eps) {
                C = p[i];
                r = 0;
                for (int j = 1; j < i; ++ j) if (Length(p[j]-C) > r + eps) {
                        C = (p[i]+p[j]) / 2.0;
                        r = Length(p[j]-p[i]) / 2.0;
                        for (int k = 1; k < j; ++ k) if (Length(p[k]-C) > r + eps) {
                                C = getCir(p[i], p[j], p[k]);
                                r = Length(p[i]-C);
                        }
                }
        }
}
void init() {
        for (int i = 1; i <= n; ++ i)
                scanf("%lf%lf", &p[i].x, &p[i].y);
}
void solve() {
        minCircle();
        printf("%.2lf␣%.2lf␣%.2lf\n", C.x, C.y, r);
}
```

6 DP

6.1 数位 dp

```
int base = 10;

int memo[20][10][2];
int f(int d, int s, bool zero) {
        int &ans = memo[d][s][zero];
        if (ans != -1) return ans;

        if (d == 0) return ans = 1;
        ans = 0;
        for (int x = 0; x < base; ++ x) if (zero || abs(s - x) >= 2)
                ans += f(d-1, x, (x == 0) && zero);
        return ans;
}

int digits[20];
int sumf(unsigned n) {
        int m = 0;
        for (; n; n /= base)
                digits[m ++] = n % base;

        int ans = 0;
        for (int i = m-1; i >= 0; -- i) {
                int j = i, x;
                for (x = 0; x < digits[i]; ++ x) if (i == m-1 || abs((digits[i+1]) - x) >= 2)
                        ans += f(j, x, (x == 0) && (i == m-1));
                if (!(i == m-1 || abs(digits[i+1] - x) >= 2)) break;
                /* do something here */
        }
        return ans;
}

int a, b;

void solve() {
        scanf("%d%d", &a, &b);
        memset(memo, 255, sizeof memo);
        printf("%d\n", sumf((unsigned)b+1) - sumf((unsigned)a));
}
```

6.2 状压 dp

```
        int S;
        // sub
        int SS = S;
        do {
                // solve
                SS = (SS-1) & S;
        } while (SS != S);

        //k-sub
        int k;
        SS = (1<<k)-1;
        while (SS < (1<<n)) {
                //solve
                int x = SS & -SS, y = SS + x;
                SS = (((SS & ~y) / x) >> 1) | y;
        }
```

7 数据结构

7.1 Hash

```
#include <cstring>
const int Hmod = 4000001;

template <typename _Value>
struct HASH {
        LL a[Hmod + 1000]; _Value b[Hmod + 1000];
```

```cpp

        void clear() {
                memset(a, 0xff, sizeof a);
        }
        void insert(LL key, _Value value) {
                int p; for (p = key % Hmod; a[p] != -1; ++ p);
                a[p] = key; b[p] = value;
        }
        int count(LL key) {
                int p; for (p = key % Hmod; a[p] != -1 && a[p] != key; ++ p);
                return a[p] != -1;
        }
        _Value query(LL key) {
                int p; for (p = key % Hmod; a[p] != -1 && a[p] != key; ++ p);
                return b[p];
        }
};

HASH<int> hash;
```

7.2 线段树

```cpp
#include<algorithm>

const int MAXN=100000;

template<class ct,class mt>struct Seg {
        ct c;mt x;
        Seg *ch[2];
        Seg(){}
        Seg(ct c,mt x):c(c),x(x){}
//      void* operator new(unsigned,void* p){ return p; }
        void pass(){
                if(x.empty()) return;
                if(ch[0]){ merge(ch[0]->x, x); merge(ch[0]->c, x); }
                if(ch[1]){ merge(ch[1]->x, x); merge(ch[1]->c, x); }
                new(&x)mt();
        }
};
template<class ct,class mt>struct SegSeq {
        typedef Seg<ct, mt> _Seg;
        int n,ql,qr;
        ct cv;mt mv;
        _Seg seg[MAXN],*segR,*rt;

        int *A;
        _Seg* build(int l, int r) {
                int m=(l+r)>>1;
                _Seg *o=new(segR ++)_Seg(ct(),mt());
                if(l==r){ new(&o->c)ct(A[l]); o->ch[0] = o->ch[1] = 0; return o; }
                o->ch[0]=build(l,m); o->ch[1]=build(m+1,r);
                o->c=merge(o->ch[0]->c,o->ch[1]->c);
                return o;
        }
        void init(int n=MAXN,int *A=0){
                this->n=n; this->A=A;
                segR=seg; rt=build(1,n);
        }
        void update(_Seg*& o,int l,int r){
                int m=(l+r)>>1;
                o->pass();
                if (ql<=l&&r<=qr){ merge(o->c, mv); merge(o->x, mv); return; }
                if (ql<=m) update(o->ch[0],l,m);if(m<qr) update(o->ch[1],m+1,r);
                o->c=merge(o->ch[0] ? o->ch[0]->c : ct(),o->ch[1] ? o->ch[1]->c : ct());
        }
        void query(_Seg*& o,int l,int r){
                int m=(l+r)>>1;
                if(!o)o=new(segR++)_Seg();
                o->pass();
                if (ql<=l && r<=qr) { cv=merge(cv, o->c); return; }
                if (ql<=m) query(o->ch[0],l,m);if(m<qr) query(o->ch[1],m+1,r);
        }
        void update(int ql, int qr, const mt& mv) {
                this->ql=ql,this->qr = qr; this->mv=mv; update(rt,1,n);
        }
        ct query(int ql, int qr) {
                this->ql=ql,this->qr=qr; new(&cv)ct(); query(rt,1,n); return cv;
        }
};
```

7.3 Splay

```
#include<algorithm>
#include<new>
using namespace std;

//=====Splay Begin=====
// pool template is needed.
// example:

//please init() before use
template<class T>struct Pool{
        T *a,**q; int pa,pq;
        Pool(int MAXN){a=new T[MAXN];q=new T*[MAXN];}
//      ~Pool(){delete[] a; delete[] q;}
        void init(){pa=0;pq=0;}
        T* NEW(){return new(pq ? q[--pq] : &a[pa++])T();}
        void DELETE(T* x){q[pq++]=x; return;}
};

// ct & mt & it is needed
// ct::rev() is needed.
// example:
struct ct{int mx;ct(){mx=-1;}ct(int mx):mx(mx){}ct rev(){return *this;}
bool empty()const{return mx==-1;}};
struct mt{int ept,v;mt(){ept=1;}mt(int v):v(v){ept=0;}
bool empty()const{return ept;}mt rev(){return *this;}};
ct operator+(const ct& A,const ct& B){
        if(B.empty())return A;if(A.empty())return B;return ct(max(A.mx,B.mx));
}
void operator+=(ct& A,const mt& B){if(!B.empty()) new(&A)ct(B.v);}
void operator+=(mt& A,const mt& B){if(!B.empty()) new(&A)mt(B.v);}

//=====Splay Begin=====

namespace Splay{
        const int MAXNODE=50010; // total number of SplayNode
        const int MAXN=50010; // size of one SplayTree
        struct node{
                node *ch[2],*p;
                int sz;int rev;
                ct c,v;mt x;
                node(){new(&c)ct();new(&x)mt();ch[0]=ch[1]=p=0;v=sz=rev=0;}
                node(ct c,mt x):c(c),x(x){}
                bool isRoot(){return !p || (p->ch[0]!=this && p->ch[1]!=this);}// LCT sp
                int getlr(){return p->ch[1]==this; } // "this" mustn't be root
                node* link(int x,node* o){ch[x]=o;if(o)o->p=this;return this;}
                node* unlink(int x){if(ch[x])ch[x]->p=0;ch[x]=0;return this;}
                node* reverse(){rev^=1;swap(ch[0],ch[1]);c=c.rev();x=x.rev();return this;}
                node* modify(const mt& X){
                        // if it is a Point modify, please erase x+=X
                        x+=X;c+=X;v+=X;return this;
                }
                node* upd(){
                        if(!ch[0])if(!ch[1]){c=v;sz=1;}
                                else{c=v+ch[1]->c;sz=1+ch[1]->sz;}
                        else if(!ch[1]){c=ch[0]->c+v;sz=ch[0]->sz+1;}
                                else{c=ch[0]->c+v+ch[1]->c;sz=ch[0]->sz+1+ch[1]->sz;}
                        return this;
                }
                node* pass(){
                        if(rev){if(ch[0])ch[0]->reverse();if(ch[1])ch[1]->reverse();rev=0;}
                        if(!x.empty()){if(ch[0])ch[0]->modify(x);if(ch[1])ch[1]->modify(x);new(&x)mt();}
                        return this;
                }
```

```
                node* rotate(){ // p mustn't be root
                        node* q=p->p; int x=getlr(),y=p->isRoot() ? -1 : p->getlr();
                        link(x^1,p->link(x,ch[x^1]));
                        p->upd();
                        if(y==-1)p=q;else q->link(y,this);
                        return this;
                }
        };
        Pool<node> pool(MAXNODE);
        void passAll(node* o,node* tar){
                static node* sk[MAXN];
                static int tp;
                tp=0;
                for(;o->p!=tar;o=o->p)sk[tp++]=o;
                sk[tp++]=o;
                for(;tp;--tp)sk[tp-1]->pass();
        }
        // every node in [o,tar) should have already been passed.
        node* splay(node* o,node* tar,bool passed=0){
                if(!passed)passAll(o,tar);
                if(o->p==tar)return o;
                while(o->p!=tar&&o->p->p!=tar)
                        o->getlr()==o->p->getlr() ? (o->p->rotate(),o->rotate())
                        : (o->rotate(),o->rotate());
                if(o->p!=tar)o->rotate();
                return o->upd();
        }
        void passAllLCT(node* o){ // for LCT
                static node* sk[MAXN];
```

```
            static int tp;
            tp=0;
            for(;!o->isRoot();o=o->p)sk[tp++]=o;
            sk[tp++]=o;
            for(;tp;--tp)sk[tp-1]->pass();
      }
      node* splayLCT(node* o,bool passed=0){ // for LCT
            if(!passed)passAllLCT(o);
            if(o->isRoot())return o;
            while(!o->isRoot()&&!o->p->isRoot())
                  o->getlr()==o->p->getlr() ? (o->p->rotate(),o->rotate())
                  : (o->rotate(),o->rotate());
            if(!o->isRoot())o->rotate();
            return o->upd();
      }
      node* build(int* A,int l, int r) {
            if(l>r)return 0;
            int m=(l+r)>>1;
            node* o=pool.NEW();
            o->link(0,build(A,l,m-1));o->link(1,build(A,m+1,r));
            o->v=A[m];o->upd();new(&o->x)mt();
            return o;
      }
      void erase(node* o){
            if(o->ch[0])erase(o->ch[0]);
            if(o->ch[1])erase(o->ch[1]);
            pool.DELETE(o);
      }
      void eraseAll(node* o){splay(o,0);erase(o);}
```

```
	// k is from 1
	node* splayk(node*& o,int k){
		node *p=o; int w;
		for(;p->pass(),(w=p->ch[0]?p->ch[0]->sz:0)+1!=k;)
			if(k<=w)p=p->ch[0];else{k-=w+1;p=p->ch[1];}
		if((w=p->ch[0]?p->ch[0]->sz:0)+1!=k)o=0;else o=p;
		splay(o,0,1);
		return o;
	}
	int rank(node* o){splay(o,0);return (o->ch[0] ? o->ch[0]->sz : 0)+1;}
	// split k elements to o and others to R(when k==0 remain nothing)
	void split(node*& o,int k,node*& R){if(k==0){R=o;o=0;return;}
	splayk(o,k);R=o->ch[1];o->unlink(1)->upd();}
	void merge(node*& L,node* R){if(!L){L=R;return;}
	splayk(L,L->sz);L->link(1,R)->upd();}
	// some functions refer to sequence options
	node* pick(node* o,int l,int r){
		splay(o,0);
		if(l<0 || r>o->sz || l>r)return 0;
		if(l==1)if(r==o->sz)return o;
			else return splayk(o,r+1)->ch[0];
		else if(r==o->sz) return splayk(o,l-1)->ch[1];
			else{
				node*p=splayk(o,l-1),*q=splayk(o,r+1);
				splay(p,q);return p->ch[1];
			}
	}
	void reverse(node* o,int l,int r){
		if(l>r)return; splay(pick(o,l,r)->reverse(),0);
```

```
        }
        void update(node* o,int l,int r,const int& v){
                if(l>r)return; splay(pick(o,l,r)->modify(v),0);
        }

        //debugging....
        void print(node* o){
                o->pass();
                if(o->ch[0]) print(o->ch[0]);
                printf("%p␣",o);
                if(o->ch[1]) print(o->ch[1]);
        }
        void reupd(node* o){
                o->pass();
                if(o->ch[0]) reupd(o->ch[0]);
                if(o->ch[1]) reupd(o->ch[1]);
                o->upd();
        }
}

using namespace Splay;

//=====Splay End=====

//examples:

//BZOJ 1500 ά□□□□□□
//struct myDS{
//      typedef SplayNode node;
```

```
//	Splay t;
//	void insert(int p,int n,int* a){
//		Splay R=split(p);merge(build(a,1,n));merge(R);
//	}
//	void erase(int l,int r){
//		if(l>r)return;
//		Splay M,R;
//		M=t.split(l-1);R=M.split(r-l+1);
//		t.merge(R);M.erase();
//	}
//	int query(int Tp,int l=0,int r=0){
//		if(l>r)return 0;
//		if(Tp==0)return t.rt->c.mxs;
//		node* p=t.pick(l,r);
//		int v=p->c.s;
//		t.splay(p->pass(),0);
//		return v;
//	}
//	void print(){
//		t.print();
//	}
//}BST;

//LA 3961
//struct myDS{
//	Splay t;
//	pair<int,int>Y[maxn];
//	int n;int A[maxn];
//	SplayNode* pos[maxn];
```

```
//
//      void getpos(SplayNode* o){
//              pos[o->v]=o;
//              if(o->ch[0])getpos(o->ch[0]);
//              if(o->ch[1])getpos(o->ch[1]);
//      }
//      bool solve(){
//              if(!(scanf("%d",&n)==1&&n))return 0;
//              for(int i=1;i<=n;++i){
//                      int x;scanf("%d",&x);
//                      Y[i]=make_pair(x,i);
//              }
//              sort(Y+1,Y+1+n);
//              for(int i=1;i<=n;++i)A[Y[i].second]=i;
//
//              splayPool.init();new(&t)Splay();
//              t.merge(t.build(A,1,n));
//              getpos(t.rt);
//              for(int i=1;i<=n;++i){
//                      int x=t.rank(pos[i]);
//                      printf("%d%c",x,i==n?'\n':' ');
//                      t.reverse(i,x);
//              }
//              return 1;
//      }
//}BST;
```

7.4 KD-Tree

```
// =============== KD-Tree ===============
const int MAXD=2;

struct KDPoint{
        LL x[MAXD];
        void read(){
                for(int i=0;i<MAXD;++i){
                        int v;scanf("%d",&v);
                        x[i]=v;
                }
        }
};
typedef pair<KDPoint,int> KDPType;

struct ct{
        LL mxD[MAXD], mnD[MAXD];
        ct(){ mnD[0]=INF; }
        ct(KDPoint P){ for(int i=0;i<MAXD;++i) mxD[i]=mnD[i]=P.x[i]; }
        bool empty()const{ return mnD[0]==INF; }
};

ct merge(const ct& A, const ct& B){
        if(A.empty()) return B;
        if(B.empty()) return A;
        ct C;
        for(int i=0;i<MAXD;++i) {
                C.mxD[i]=max(A.mxD[i], B.mxD[i]);
                C.mnD[i]=min(A.mnD[i], B.mnD[i]);
        }
        return C;
}

struct KDTNode{
        KDTNode *ch[2];
        KDPType v;
        ct c;

        KDTNode(){}
        KDTNode(KDPType v):v(v){}
};

const int MAXN=100010;

struct cmp{
        int D;
        cmp(int D):D(D){}
        bool operator()(const KDPType& A, const KDPType& B)const{
                return A.first.x[D]<B.first.x[D];
        }
};

struct KDT{
        KDTNode kdt[MAXN], *kdtR, *rt;

        KDPType *A;
        KDPoint qP;

        KDTNode* build(int d,int l,int r){
                if(l>r) return 0;

                int m = (l+r)>>1;
                nth_element(A+l,A+m,A+r+1,cmp(d));

                KDTNode *o=new(kdtR ++)KDTNode(A[m]);
                o->ch[0]=build((d+1)%MAXD, l, m-1); o->ch[1] = build((d+1)%MAXD, m+1, r);
                o->c=merge(merge(o->ch[0] ? o->ch[0]->c : ct(), o->v.first), o->ch[1] ? o->ch[1]->c : ct());
                return o;
        }
        void init(int n,KDPType* A) {
                this->A = A; kdtR = kdt; rt = build(0,1,n);
        }

        // insert
        KDPType iP;
        void insert(int d, KDTNode*& o){
                if(!o){ o=new(kdtR ++)KDTNode(iP); }
                else { if(cmp(d)(iP,o->v))insert(d^1,o->ch[0]);else insert(d^1,o->ch[1]); }
                o->c=merge(merge(o->ch[0] ? o->ch[0]->c : ct(), o->v.first), o->ch[1] ? o->ch[1]->c : ct());
        }
        void insert(KDPType P){
                iP=P; insert(0,rt);
        }

        // query
        LL getdis(KDTNode* o){ // getdis¹ش<□□Çǵ□Ŀ    □□□д□□□□□□□□□Ž□¯•±±£¬¾¶»©µ
                LL res=0;
                for(int d=0;d<MAXD;++d){
```

```
                if(qP.x[d]<o->c.mnD[d]) res+=sqr(o->c.mnD[d]-qP.x[d]);
                if(o->c.mxD[d]<qP.x[d]) res+=sqr(qP.x[d]-o->c.mxD[d]);
            }
            return res;
        }

        vector<LL> res; vector<int> resi;
        void query(int d, KDTNode* o){
            LL d0=0; for(int i=0;i<MAXD;++i) d0+=sqr(o->v.first.x[i]-qP.x[i]);
            if (dcmp(d0-res[0])<0 || (dcmp(d0-res[0])==0 && o->v.second<resi[0]))
                res[1]=res[0], resi[1]=resi[0], res[0]=d0, resi[0]=o->v.second;
            else if(dcmp(d0-res[1])<0 || (dcmp(d0-res[1])==0 && o->v.second<resi[1]))
                res[1]=d0, resi[1]=o->v.second;

            LL dl=o->ch[0]?getdis(o->ch[0]):INF;
            LL dr=o->ch[1]?getdis(o->ch[1]):INF;
            if(dl<dr){
                if(dl<=res[1])query((d+1)%MAXN,o->ch[0]);
                if(dr<=res[1])query((d+1)%MAXN,o->ch[1]);
            }else{
                if(dr<=res[1])query((d+1)%MAXN,o->ch[1]);
                if(dl<=res[1])query((d+1)%MAXN,o->ch[0]);
            }
        }

        vector<int> query(KDPoint P){
            qP=P;
            res.resize(2); res[0]=res[1]=INF;
            resi.resize(2);
```

```
                query(0,rt);
                return resi;
        }
}solver;
```

7.5 内存池

```
//please init() before use
template<class T>struct Pool{
        T *a,**q; int pa,pq;
        Pool(int MAXN){a=new T[MAXN];q=new T*[MAXN];}
//      ~Pool(){delete[] a; delete[] q;}
        void init(){pa=0;pq=0;}
        T* NEW(){return new(pq ? q[--pq] : &a[pa++])T();}
        void DELETE(T* x){q[pq++]=x; return;}
};
```

7.6 轻重链剖分

```
#include<cstdio>

const int maxn=10010;

//Heavy-Light Decomposition
//help:1.init() 2.addedge() 3.pre() 4.update()/query()
struct HLD{
        int st[maxn],lk[maxn<<1],b[maxn<<1];
        int tot;
        void init(){
```

```
		tot=1;memset(st,0,sizeof st);
	}
	void addedge(int u,int v){
		lk[++tot]=st[u];b[tot]=v;st[u]=tot;
	}
	int fa[maxn],d[maxn],sz[maxn],son[maxn],top[maxn];
	int seq[maxn],bg[maxn],ed[maxn];
	int dfN;

	int A[maxn];
	SegSeq<ct,mt,maxn> seg;

	void dfs1(int u){
		sz[u]=1;son[u]=0;
		for(int i=st[u];i;i=lk[i]){
			int v=b[i];
			if(v==fa[u])continue;
			fa[v]=u; d[v]=d[u]+1;
			dfs1(v);
			sz[u]+=sz[v];
			if(!son[u]||sz[son[u]]<=sz[v]) son[u]=v;
		}
	}
	void dfs2(int u){
		seq[bg[u]=++dfN]=u;
		if(son[u]){ top[son[u]]=top[u]; dfs2(son[u]); }
		for(int i=st[u];i;i=lk[i]){
			int v=b[i];
			if(v==fa[u]||v==son[u])continue;
```

```
                top[v]=v; dfs2(v);
            }
            ed[u]=dfN;
        }
        void pre(int n,int* a){
            int rt=1;
            dfN=0; fa[rt]=0,d[rt]=1; dfs1(rt); top[rt]=rt; dfs2(rt);
            for(int i=1;i<=n;++i) A[i]=a[seq[i]];
            seg.init(n,A);
        }
        void update(int x,int y,int& v){
            int p,q;
            for(;(p=top[x])!=(q=top[y]);){
                if(d[p]<d[q]) swap(x,y),swap(p,q);
                seg.update(bg[p],bg[x],v),x=fa[p];
            }
            if(d[x]<d[y])swap(x,y);
            seg.update(bg[y],bg[x],v);
        }
        /*
        Pay attention to the order of merging. The following is according to x->y. The rev() function is required.
        You can also imitate update() to speed up it.
        */
        void query(int x,int y,int& v){
            ct cx,cy;
            int p,q;
            for(;(p=top[x])!=(q=top[y]);){
                if(d[p]<d[q]) swap(x,y),swap(p,q),swap(cx,cy);
                cx=cx+seg.query(bg[p],bg[x]).rev();
```

```
                        x=fa[p];
                }
                if(d[x]<d[y])swap(x,y),swap(cx,cy);
                cx=cx+seg.query(bg[y],bg[x]).rev();
                v=(cx+cy.inv()).s;
        }
}hld;
```

7.7 LCT

```
const int maxn=100010;

//LCT
//splay is needed
struct LCT{
        // in LCT, unlink() is a dangerous operation.
        int n;
        node* T[maxn];
        int d[maxn]; // d[u]: the depth of u (used for location)

        void init(int n,int* A){
                this->n=n;for(int i=1;i<=n;++i)T[i]=build(A,i,i);
        }
        void access(node* o){
                for(node *p=o,*q=0;p;q=p,p=p->p){splayLCT(p);p->link(1,q)->upd();}
                splayLCT(o);
        }
        node* findRoot(node* o){access(o);return splayk(o,1);}
        void link(node* o,node* p){access(o);access(p);o->p=p;}
```

```
	void cut(node* o){access(o);o->unlink(0);o->upd();}
	void evert(node *o){access(o);o->reverse();}

	// you should also modify "Splay" when you choose to "modify".
	void modify_route(node* o,const mt& v){access(o);o->modify(v);}
	void modify(node* o,const mt& v){splayLCT(o)->modify(v);}

	ct query(node* x,node* y){
		ct res;
		access(y);
		for(node *p=x,*q=0;p;q=p,p=p->p){
			splayLCT(p);
			if(!p->p){
				if(p->ch[1])
					if(q)res=p->ch[1]->c+q->c.rev();
					else res=p->ch[1]->c;
				else res=q->c.rev();
			}
			p->link(1,q);
			p->upd();
		}
		splayLCT(x);
		return res;
	}
	ct query(node *x){access(x);return x->c;}
	void link(int o,int p){link(T[o],T[p]);}
	void cut(int o){cut(T[o]);}
	void evert(int o){evert(T[o]);}
	int query(int x,int y){return query(T[x],T[y]).mx;}
```

```
        void update(int x,int v){modify(T[x],v);}
}lct;
```