

```
int n;  一个整数
```

1 数学

1.1 高精度

```
1  const int e[9] = {1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000};
2  const int bs = 8;
3  const int maxlen = 20;
4
5  struct bign {
6      int a[maxlen]; int n;
7
8      void suppress() { while (n > 1 && !a[n-1]) -- n; }
9
10     bign() { }
11     bign(char* s) {
12         int m = strlen(s);
13
14         n = 0;
15         for (int i = m-1; i >= 0; i -= bs) {
16             int t = 0;
17             for (int j = 0; j < bs && (i-j) >= 0; ++ j) {
18                 t = t + (s[i-j]-'0') * e[j];
19             }
20             a[n++] = t;
21         }
22         suppress();
```

```
23     }
24     bign(LL x) { n = 0; for (; x; x /= e[bs]) a[n++] = x % e[bs]; }
25     bign(const bign& A) { n = A.n; for (int i = 0; i < n; ++ i) a[i] = A.a[i]; }
26     bool operator ==(const bign& A) const {
27         if (n != A.n) return 0;
28         for (int i = n-1; i >= 0; -- i) if (a[i] != A.a[i]) return 0;
29         return 1;
30     }
31     bool operator <(bign& A) const {
32         if (n != A.n) return n < A.n;
33         for (int i = n-1; i >= 0; -- i) if (a[i] != A.a[i]) return a[i] < A.a[i];
34         return 0;
35     }
36
37     void print() {
38         printf("%d", a[n-1]); for (int i = n-2; i >= 0; -- i) printf("%0*d", bs, a[i]);
39     }
40 };
41
42 bign C;
43
44 bign operator +(const bign& A, const bign &B) {
45     C.n = std::max(A.n, B.n) + 1;
46
47     int t = 0;
48     for (int i = 0; i < C.n; ++ i) {
49         t = t + ((i < A.n ? A.a[i] : 0) + (i < B.n ? B.a[i] : 0));
50         if (t >= e[bs]) {
51             C.a[i] = t - e[bs];
```

```
52         t = 1;
53     } else {
54         C.a[i] = t;
55         t = 0;
56     }
57 }
58 C.suppress();
59 return C;
60 }
61 bign operator -(const bign &A, const bign &B) {
62     C.n = std::max(A.n, B.n);
63
64     int t = 0;
65     for (int i = 0; i < C.n; ++ i) {
66         t = t + ((i < A.n ? A.a[i] : 0) - (i < B.n ? B.a[i] : 0));
67         if (t < 0) {
68             C.a[i] = t + e[bs];
69             t = -1;
70         } else {
71             C.a[i] = t;
72             t = 0;
73         }
74     }
75     C.suppress();
76     return C;
77 }
78 bign operator *(const bign &A, const bign &B) {
79     C.n = A.n + B.n;
80     memset(C.a, 0, sizeof(int)*C.n);
```

```
81
82     LL t = 0;
83     for (int i = 0; i < A.n; ++ i) {
84         for (int j = 0; j <= B.n; ++ j) {
85             t = t + C.a[i+j] + (j < B.n ? ((LL)A.a[i] * B.a[j]) : 0);
86             C.a[i+j] = t % e[bs];
87             t = t / e[bs];
88         }
89     }
90
91     C.suppress();
92     return C;
93 }
94 bign operator /(const bign& A, LL b) {
95     C.n = A.n;
96
97     LL t = 0;
98     for (int i = C.n-1; i >= 0; -- i) {
99         t = (t%b)*e[bs] + A.a[i];
100        C.a[i] = t/b;
101    }
102
103    C.suppress();
104    return C;
105 }
106
107 bign operator +=(bign& A, const bign& B) { return A = A + B; }
```

1.2 模数

```

1  #include<cstdio>
2  typedef long long LL;
3  int MOD;
4  struct modn{
5      int n,MOD;
6      modn() {}
7      modn(LL x=0,int MOD):MOD(MOD){n=(x>=0?(x<MOD?x:x%MOD):(x%MOD+MOD)%MOD);}
8      // Attention: MOD is not participate in the comparison.
9      bool operator <(const modn& B){return n<B.n;}
10     bool operator ==(const modn& B){return n==B.n;}
11     bool operator !=(const modn& B){return n!=B.n;}
12     modn operator +(const modn& B){return modn(n+B.n>=MOD?n+B.n-MOD:n+B.n,MOD);}
13     modn operator -(const modn& B){return modn(n-B.n<0?n-B.n+MOD:n-B.n,MOD);}
14     modn operator *(const modn& B){return modn((LL)n*B.n,MOD);}
15     modn& operator +=(const modn& B){return *this=*this+B;}
16     modn& operator -=(const modn& B){return *this=*this-B;}
17     modn& operator *=(const modn& B){return *this=*this*B;}
18     modn& operator -() {n=MOD-n;return *this;}
19     void print()const{printf("%d\n",n);}
20 };
21 modn pow(const modn& A,LL x){
22     modn s=modn(1,A.MOD),t=A;for(;x;x>>=1){if(x&1)s*=t;t*=t;}return s;
23 }
24 modn zero; // zero should be initialed.
25 bool iszero(const modn& A){return A.n==0;}
26
27 /*
28 modn fact_mod[MOD];

```

```
29 modn fact(LL n) {
30     return n == 0 ? 1 : ((n/mod)&1) ? -1 : 1 * fact(n/mod) * fact_mod[n%mod];
31 }
32
33 modn mod_fact(LL n, int& e) {
34     if (n == 0) { e = 0; return modn(1); }
35     modn res = mod_fact(n/mod, e);
36     e += n/mod;
37     return ((n/mod)&1) ? -res : res * fact_mod[n%mod];
38 }
39 modn C_mod(LL n, LL k, int& e) {
40     if (n < 0 || k < 0 || n < k) return 0;
41     int e1, e2, e3;
42     modn n1 = mod_fact(n, e1), k1 = mod_fact(k, e2), k2 = mod_fact(n-k, e3);
43     e = e1 - (e2 + e3);
44     return n1 / (k1*k2);
45 }
46 */
```

1.3 分数

```
1 LL gcd(LL a, LL b) { LL c; while (b) { c = a; a = b; b = c%b; } return a; }
2
3 struct frac {
4     LL a, b;
5     frac() { }
6     frac(LL a) : a(a), b(1) { }
7     frac(LL a, LL b) : a(a), b(b) { }
8 }
```

```

9      frac normalize() const { LL d = gcd(a, b); return b/d < 0 ? frac(-a/d, -b/d) : frac(a/d, b/d); }
10
11     void print() {
12         if (b == 1) printf("%I64d", a);
13         else if (a < 0) printf("-\\frac{%I64d}{%I64d}", -a, b);
14         else printf("\\frac{%I64d}{%I64d}", a, b);
15     }
16 };
17 frac operator +(const frac& A, const frac& B) { return frac(A.a * B.b + A.b * B.a, A.b * B.b).normalize(); }
18 frac operator -(const frac& A, const frac& B) { return frac(A.a * B.b - A.b * B.a, A.b * B.b).normalize(); }
19 frac operator *(const frac& A, const frac& B) { return frac(A.a * B.a, A.b * B.b).normalize(); }
20 frac operator /(const frac& A, const frac& B) { return frac(A.a * B.b, A.b * B.a).normalize(); }
21 frac operator <(const frac& A, const frac& B) { return A.a * B.b - A.b * B.a < 0; }
22 bool operator ==(const frac& A, const frac& B) { return A.a == B.a && A.b == B.b; }
23 bool operator !=(const frac& A, const frac& B) { return !(A.a == B.a && A.b == B.b); }

```

1.4 矩阵

```

1  #include<cstdio>
2
3  const int msz = 3;
4
5  template<class T>
6  struct Mat {
7      int m, n;
8      T a[msz][msz];
9
10     Mat() {}
11     Mat(int m, int n) : m(m), n(n) {

```

```
12         for(int i=0;i<m;++i) for(int j=0;j<n;++j) a[i][j]=0;
13     }
14     Mat(int m,int n,T* A):m(m),n(n){
15         for(int i=0;i<m;++i) for(int j=0;j<n;++j) a[i][j]=A[i*n+j];
16     }
17     T* operator [](int i) { return a[i]; }
18     const T* operator [](int i) const { return a[i]; }
19
20     // assert(m==B.m&&n==B.n);
21     Mat operator +(const Mat& B){
22         static Mat C;new(&C)Mat(m,n);
23         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];
24         return C;
25     }
26     // zero(T) is needed
27     // assert(n==B.m);
28     Mat operator *(const Mat& B){
29         static Mat C;new(&C)Mat(m,B.n);
30         for(int i=0;i<m;++i)
31             for(int j=0;j<n;++j) if(zero(a[i][j]))
32                 for(int k=0;k<B.n;++k) if(zero(B[j][k]))
33                     C[i][k]+=a[i][j]*B[j][k];
34         return C;
35     }
36     void print() {
37         for (int i = 0; i < m; ++ i,putchar('\n'))
38             for (int j = 0; j < n; ++ j,putchar(' '))
39                 a[i][j].print();
40     }
```



```
41 };
42
43 template<class T>
44 Mat<T> power(Mat<T> a, int n) {
45     assert(a.m==a.n);
46     Mat<T> s(a.n,a.n),t=a;
47     for(int i=0;i<s.n;++i)s[i][i]=1;
48     for(;n;n>>=1){if(n&1)s=s*t;t=t*t;}
49     return s;
50 }
51
52 // Output: Det(A)
53 template<class T>
54 T Determinant(Mat<T> A){
55     assert(A.m==A.n);
56     int n=A.n;
57     T res=1;
58     for(int i=0;i<n;++i){
59         for(int j=i+1;j<n;++j){
60             T *p=A[i], *q=A[j];
61             while(q[i]!=0){
62                 int t=p[i]/q[i];
63                 for(int k=i;k<n;++k)
64                     p[k]=p[k]-q[k]*t;
65                 swap(p,q);
66             }
67             if(p!=A[i]){
68                 for(int k=i;k<n;++k)
69                     swap(p[k],q[k]);
```

```
70         swap(p,q);
71         res=-res;
72     }
73 }
74 if(A[i][i]==0) return 0;
75 res=res*A[i][i];
76 }
77 return res;
78 }
```

1.5 异或找基

```
1 LL b[100], bN;
2 void solve() {
3     bN = 0;
4     for (int i = 1; i <= tot; ++ i) {
5         for (int j = 0; j < bN; ++ j) {
6             if ((num[i]^b[j]) < num[i])
7                 num[i] = num[i] ^ b[j];
8         }
9         if (num[i])
10             b[bN ++] = num[i];
11     }
12
13     sort(b, b+bN, greater<LL>());
14
15     LL res = 0;
16     for (int i = 0; i < bN; ++ i)
17         if ((res ^ b[i]) > res)
```

```

18         res ^= b[i];
19     printf("%I64d\n", res);
20 }

```

1.6 k 阶等差数列

```

1     B[0][0] = 0; B[0][1] = 1;
2     for (int d = 1; d < maxn; ++ d) {
3         for (int j = 0; j <= d; ++ j) {
4             B[d][j] = 0;
5             for (int i = max(0, j-1); i <= d-1; ++ i)
6                 B[d][j] += - C[d+1][i] * B[i][j];
7             B[d][j] = B[d][j] / (d+1);
8         }
9         B[d][d+1] = frac(1) / (d+1);
10    }

```

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

```

1  int n;
2
3  int prime[maxp], pN;
4  int phi[maxn], mu[maxn];
5  // mf:minfactor, mfk:the num of mf, mfx:pow(mf,mfk)
6  int mf[maxn], mfk[maxn], mfx[maxn];
7  // divN:the num of divisors, divsN:the num of divisors set
8  int divN[maxn], divsN[maxn];
9  // f:any function that can be summarized by gen, sf:sum of f
10 // such as "multiplicative function" or some function that can be derived by (i|p) and/or !(i|p)
11 LL f[maxn], sf[maxn];
12 bool isp[maxn];
13 void gen(int n) {
14     pN = 0;
15     memset(isp, 1, sizeof isp);
16
17     isp[0]=isp[1]=0;
18     phi[1]=1, mu[1]=1;
19     mf[1]=1, mfk[1]=0, mfx[1]=1;
20     divN[1]=divsN[1]=1;
21     f[1]=1;
22     for(int i=2, p; i<n; ++i) {
23         if (isp[i]) {

```

```

24     prime[pN++] = p;
25     phi[p] = p - 1, mu[p] = -1;
26     mf[p] = p, mfk[p] = 1, mfx[p] = p;
27     divN[p] = divsN[p] = 2;
28     f[p] = 2LL * p - 1;
29 }
30
31 for(int j = 0, x; j < pN && (x = i * (p = prime[j])) < n; ++j) {
32     isp[x] = 0;
33
34     if(!(i % p)) {
35         phi[x] = phi[i] * p, mu[x] = 0;
36         mf[x] = p, mfk[x] = mfk[i] + 1, mfx[x] = mfx[i] * p;
37         divN[x] = divN[i] + divN[p], divsN[x] = divsN[i];
38         if(mfx[x] == x) // x = p^k
39             f[x] = x + mfk[x] * (x / p * (p - 1));
40         else
41             f[x] = f[x / mfx[x]] * f[mfx[x]];
42         break;
43     } else {
44         phi[x] = phi[i] * (p - 1), mu[x] = -mu[i];
45         mf[x] = p, mfk[x] = 1, mfx[x] = p;
46         divN[x] = divN[i] + divN[p], divsN[x] = divsN[i] + 1;
47         f[x] = f[i] * f[p];
48     }
49 }
50 }
51 }
52 int get_pri(int *pri, int* pri_num, int n) {

```

```
53     int pri_n = 0;
54     for (int i = 0, p; (p = prime[i]) <= n/p; ++ i) if (n%p == 0) {
55         pri[pri_n] = p; pri_num[pri_n] = 0;
56         while (n%p == 0) { n /= p; ++ pri_num[pri_n]; }
57         ++ pri_n;
58     }
59     if (n > 1) { pri[pri_n] = n; pri_num[pri_n] = 1; ++ pri_n; }
60     return pri_n;
61 }
62 void dfs_divisor(int d, int div) {
63     if (d == pri_n) {
64         /* do something here */
65         return;
66     }
67     for (int i = 0; i <= pri_num[d]; ++ i) {
68         dfs_divisor(d+1, div);
69         div = div * pri[d];
70     }
71 }
72 dfs_divisor(0, 1);
73
74 int phi(int n) {
75     int res = n;
76     for (int i = 0, j; (j = prime[i]) <= n/j; ++ i) if (n%j == 0) {
77         res = res / j * (j-1);
78         while (n%j == 0) n /= j;
79     }
80     if (n > 1) res = res / n * (n-1);
81     return res;
```

```
82 }
83
84 int get_primitive_root(int MOD, int phi) {
85     vector<int> factors;
86     int n = phi;
87     for(int i = 2; i <= n/i; i++) {
88         if(n % i != 0) continue;
89         factors.push_back(i);
90         while(n % i == 0) n /= i;
91     }
92     if(n > 1) factors.push_back(n);
93
94     for (int m = 2; ; ++ m) {
95         bool ok = true;
96         for(int i = 0; i < factors.size(); i++)
97             if(pow_mod(m, phi/factors[i], MOD) == 1) { ok = false; break; }
98         if(ok) return m;
99     }
100 }
101
102 inv_mod[1] = 1; for (int i = 2; i < mod; ++ i) inv_mod[i] = (mod-mod/i) * inv_mod[mod%i];
103 fact_mod[0] = 1; for (int i = 1; i < maxn; ++ i) fact_mod[i] = fact_mod[i-1] * i;
104 ifact_mod[0] = 1; for (int i = 1; i < maxn; ++ i) ifact_mod[i] = ifact_mod[i-1] * inv_mod[i];
105
106 gen(n+1);
```

2.2 素数和

```
1 LL N(int n, int _p) {
```

```
2         if (_p < 0) return n-1;
3         int p = prime[_p];
4         if (p*p > n) return N(n, _p-1);
5         return N(n, _p-1) - (N(n/p, _p-1) - N(p-1, _p-1));
6     }
7     LL S(int n, int _p) {
8         if (_p < 0) return (LL)n*(n+1)/2 - 1;
9         int p = prime[_p];
10        if (p*p > n) return S(n, _p-1);
11        return S(n, _p-1) - (LL)p * (S(n/p, _p-1) - S(p-1, _p-1));
12    }
13    LL fact[maxn];
14    LL P(int n, int _p) {
15        if (_p < 0) return fact[n];
16        int p = prime[_p];
17        if (p*p > n) return P(n, _p-1);
18        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));
19    }
20
21    int n, p;
22
23    void solve() {
24        scanf("%d", &n);
25        p = std::upper_bound(prime, prime+prime_N, (int)(sqrt(n)+1)) - prime - 1;
26
27        fact[0] = 1; for (int i = 1; i <= 15; ++ i) fact[i] = fact[i-1] * i;
28        printf("%I64d\n", N(n, p));
29    }
```


2.3 NTT

```
1  /*
2  find_NTT_prime
3
4      gen(1000000001);
5
6      for (int i = 1; i < prime_N; ++ i) {
7          int p = prime[i]; -- p;
8          int s = 0;
9          for (; !(p&1); p >>= 1) ++ s;
10
11          p = prime[i];
12          int PR = get_primitive_root(p, p-1);
13          if (s >= 20) printf("%d %d %d %d\n", p, s, PR, pow_mod(PR, p-2, p));
14      }
15
16  167772161 25 3
17  377487361 23 7
18  469762049 26 3
19  595591169 23 3
20  645922817 23 3
21  754974721 24 11
22  880803841 23 26
23  897581057 23 3
24  998244353 23 3
25  */
26
27  typedef long long LL;
28
```

```
29 const int mod = 998244353;
30 const int m2k = 23;
31 const int PR = 3;
32 const int inv_PR = 332748118;
33
34 modn PRw[30], inv_PRw[30];
35
36 typedef long long LL;
37 typedef std::vector<modn> Vec;
38
39 int n;
40
41 void NTT(Vec& A, bool inv) {
42     int n = A.size();
43
44     for (int i = 0, j = 0; i < n; ++ i) {
45         if (j > i) std::swap(A[i], A[j]);
46         int k = n;
47         while (j & (k >>= 1)) j &= ~k;
48         j |= k;
49     }
50
51     for (int step = 1, j = 1; step < n; step <= 1, ++ j) {
52         modn wn = inv ? PRw[j] : inv_PRw[j];
53         modn wnk = 1;
54         for (int k = 0; k < step; ++ k) {
55             for (int Ek = k; Ek < n; Ek += (step<<1)) {
56                 int Ok = Ek + step;
57                 modn t = wnk * A[Ok];
```

```

58         A[Ok] = A[Ek] - t;
59         A[Ek] = A[Ek] + t;
60     }
61     wnk = wnk * wn;
62 }
63 }
64 modn inv_n = pow_mod(n, mod-2);
65 if (inv)
66     for (int i = 0; i < n; ++ i) A[i] = A[i] * inv_n;
67 }
68
69 Vec t1, t2;
70 Vec operator *(const Vec& A, const Vec& B) {
71     int a = A.size(), b = B.size(), S = 2;
72     for (; S <= (a+b-1); S <= 1);
73     t1.resize(S); for (int i = 0; i < S; ++ i) t1[i] = (i < a ? A[i] : 0); NTT(t1, 0);
74     t2.resize(S); for (int i = 0; i < S; ++ i) t2[i] = (i < b ? B[i] : 0); NTT(t2, 0);
75     for (int i = 0; i < S; ++ i) t1[i] = t1[i] * t2[i]; NTT(t1, 1); t1.resize(a+b-1);
76     return t1;
77 }
78
79 Vec a, b;
80 void solve() {
81     PRw[m2k] = pow_mod(PR, (mod-1)/(1<<m2k));
82     for (int i = m2k-1; i >= 0; -- i) PRw[i] = PRw[i+1] * PRw[i+1];
83     inv_PRw[m2k] = pow_mod(inv_PR, (mod-1)/(1<<m2k));
84     for (int i = m2k-1; i >= 0; -- i) inv_PRw[i] = inv_PRw[i+1] * inv_PRw[i+1];
85
86     a.resize(2); a[0] = 1; a[1] = 2;

```

```
87     b.resize(2); b[0] = 2; b[1] = 0;
88     a = a * b;
89     for (int i = 0; i < a.size(); ++ i) a[i].print(); putchar('\n');
90 }
```

2.4 一些结论

可逆矩阵循环节 从组合的角度可以算出来 \mathbb{Z}_p 下的 n 阶可逆阵形成的群的阶为: $(p^n - 1)(p^n - p)(p^n - p^2)(p^n - p^3) \dots (p^n - p^{n-1})$ 。

对于 \mathbb{Z}_p 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

```
1 #include <algorithm>
2 #include <cassert>
3 #include <cmath>
4 #include <cstdio>
5 #include <cstring>
6 #include <deque>
7 #include <iostream>
8 #include <map>
9 #include <queue>
10 #include <vector>
11
12 typedef unsigned long long LL;
13 typedef std::pair<int, int> P;
14
```

```

15 const int hash = 123;
16 const int INF = 0x3f3f3f3f;
17 const int maxn = 500010;
18
19 struct StringHash{
20     int MOD;
21     char s[maxn]; int n; // the string is from 1 !!!!!
22     LL H[maxn], iH[maxn], xk[maxn];
23     LL Hash(int l, int r){ // the Hash of [l, r)
24         return ((H[l] - (H[r] * xk[r - l] % MOD)) % MOD + MOD) % MOD;
25     }
26     LL iHash(int l, int r){ // the inverseHash of (r, l]
27         return ((iH[l] - (iH[r] * xk[l - r] % MOD)) % MOD + MOD) % MOD;
28     }
29     void init(int MOD, int n, char*s){
30         this->MOD=MOD; strcpy(this->s+1, s); this->n=n;
31         H[n+1]=0; for(int i=n; i>0; --i) H[i] = (H[i+1] * hash + this->s[i]) % MOD;
32         iH[0]=0; for(int i=1; i<=n; ++i) iH[i] = (iH[i-1] * hash + this->s[i]) % MOD;
33         xk[0]=1; for(int i=1; i<=n; ++i) xk[i] = (xk[i-1] * hash) % MOD;
34         // printf("%I64d %I64d\n", Hash(3, 5), iHash(2, 0));
35     }
36     bool equal(int xl, int xr, int yl, int yr){
37         // Determin [xl, xr) substring and [yl, yr) substring is equal
38         // (the reverse string is accepted.)
39         ++xl, ++xr, ++yl, ++yr;
40         if(abs(xr - xl) != abs(yr - yl)) return 0;
41         return (xl <= xr ? Hash(xl, xr) : iHash(xl, xr)) == (yl <= yr ? Hash(yl, yr) : iHash(yl, yr));
42     }
43 };

```

```
44
45 int n;
46 char s[maxn];
47 StringHash H7,H9;
48 bool equal(int xl,int xr,int yl,int yr){
49     return H7.equal(xl,xr,yl,yr) && H9.equal(xl,xr,yl,yr);
50 }
51
52 // 0:even, 1:odd;
53 bool check(int type, int L) {
54     if (type == 0) {
55         for (int i = L-1; i <= n-(L+1); ++ i)
56             if (equal(i,i-L,i+1,i+1+L))
57                 return 1;
58         return 0;
59     } else {
60         for (int i = L-1; i <= n-L; ++ i)
61             if (equal(i,i-L,i,i+L))
62                 return 1;
63         return 0;
64     }
65 }
66
67 void solve() {
68     scanf("%s", s); n = strlen(s);
69     H7.init(1000000007,n,s);
70     H9.init(1000000009,n,s);
71     for (int t = 0; t < 2; ++ t) {
72         int l = 0, r = n/2+1;
```

```
73         while (l + 1 < r) {
74             int m = (l + r) >> 1;
75             if (check(t, m)) l = m; else r = m;
76         }
77         printf("%d_ %d\n", t, l*2-t);
78     }
79 }
80
81 int main() {
82     freopen("hash.in", "r", stdin);
83     // freopen(".out", "w", stdout);
84     solve();
85     for(;;);
86
87     return 0;
88 }
```

3.2 Manacher

```
1  int n, m;
2
3  char str[maxn], s[maxn << 1];
4  int f[maxn << 1];
5
6  void init() {
7      scanf("%s", str); n = strlen(str);
8      for (int i = 0; i < n; ++ i) {
9          s[i<<1] = '*';
10         s[i<<1|1] = str[i];
```

```
11     }
12     m = n<<1; s[m] = '*'; m ++;
13
14     int id = 1, mx = 0; f[0] = 1;
15     for (int i = 1; i < m; ++ i) {
16         int p;
17         if (mx > i)
18             p = std::min(f[id*2-i], mx-i);
19         else
20             p = 1;
21
22         while (i-p >= 0 && s[i-p] == s[i+p]) ++ p;
23         f[i] = p;
24         if (i+f[i]-1 > mx) {
25             id = i;
26             mx = i+f[i]-1;
27         }
28     }
29
30     for (int i = 0; i < m; ++ i)
31         printf("%d_", f[i]);
32     putchar('\n');
33 }
```

3.3 AC

```
1 const int maxsigma = 26;
2 const int maxnode = 500010;
3 const int maxs = 60;
```



```
4  const int maxS = 100010;
5
6  int idx(char x) { return x - 'a'; }
7
8  struct Trie {
9      Trie *ch[maxsigma], *pre, *lst;
10     int v;
11     Trie() { memset(ch, 0, sizeof ch); lst = 0; v = 0; }
12
13     int calc() { int x = (lst ? lst->calc() : 0) + v; v = 0; return x;}
14 } trie[maxnode], *rot, *trieR;
15
16 void insert(char *s) {
17     int n = strlen(s);
18     Trie *p = rot;
19     for (int i = 0; i < n; ++ i) {
20         int x = idx(s[i]);
21         if (!p->ch[x]) {
22             p->ch[x] = trieR++;
23             *p->ch[x] = Trie();
24         }
25         p = p->ch[x];
26     }
27     ++ p->v;
28 }
29
30 Trie* Q[maxnode]; int l, r;
31 void getFail() {
32     rot->pre = rot;
```

```
33
34     l = r = 0;
35     for (int x = 0; x < maxsigma; ++ x) if (rot->ch[x]) {
36         rot->ch[x]->pre = rot;
37         Q[r ++] = rot->ch[x];
38     } else
39         rot->ch[x] = rot;
40
41     for (; l != r; ++ l) {
42         Trie *u = Q[l];
43         for (int x = 0; x < maxsigma; ++ x) if (u->ch[x]) {
44             Trie *v = u->ch[x], *p = u->pre->ch[x];
45             v->pre = p;
46             v->lst = p->v ? p : p->lst;
47             Q[r ++] = v;
48         } else
49             u->ch[x] = u->pre->ch[x];
50     }
51 }
52
53 int n, m;
54 char s[maxs], S[maxS];
55
56 void init() {
57     rot = trieR = trie; trieR ++;
58     *rot = Trie();
59     for (int i = 0; i < m; ++ i) {
60         scanf("%s", s); insert(s);
61     }
```

```
62         getFail();
63     }
64
65     void solve() {
66         scanf("%d", &m);
67         init();
68         scanf("%s", S); n = strlen(S);
69
70         int res = 0;
71         Trie *p = rot;
72         for (int i = 0; i < n; ++ i)
73             res += (p = p->ch[idx(S[i])])->calc();
74         printf("%d\n", res);
75     }
76
77     int main() { }
```

3.4 SA

```
1  #include<algorithm>
2  #include<cstdio>
3  #include<cstring>
4  using namespace std;
5
6  typedef long long LL;
7
8  const int MAXN=100010;
9  const int MAXLG=18;
10
```

```

11 int lg2[MAXN];
12 int icmp(int x){ return x==0 ? 0 : (x<0 ? -1 : 1); }
13 const int MAXM=256;
14 int n;
15 struct SA{
16     int n;
17     char s[MAXN];
18     int sa[MAXN],rk[MAXN],ht[MAXLG][MAXN];
19     LL subs[MAXN]; // subs[n][j]: 字符串s中从位置n开始,长度为j的子串在s中出现的次数
20
21     //=====
22     /*
23     struct cmp{
24         int* s;
25         cmp(int* s):s(s){}
26         bool operator()(const int& a,const int& b){
27             return s[a]<s[b];
28         }
29     };
30     */
31     //=====
32
33     void init(int n,char *a){
34         this->n=n; for(int i=0;i<n;++i)s[i]=a[i];
35
36         static int t[MAXN], c[MAXN];
37         s[n++]=0;
38         int *x=sa, *y=rk, *z=t;
39         //=====

```

```

40     for(int i=0;i<MAXM;++i) c[i]=0;
41     for(int i=0;i<n;++i) c[y[i]=s[i]]++;
42     for(int i=1;i<MAXM;++i) c[i]+=c[i-1];
43     for(int i=0;i<n;++i) x[--c[y[i]]]=i;
44     //=====
45     /*
46     for(int i=0;i<n;++i) x[i]=i;
47     sort(x,x+n,cmp(s));
48     c[y[x[0]]=0]=0; int p=1;
49     for(int i=1;i<n;++i){
50         if(!(s[x[i]]==s[x[i-1]]))c[p++]=i;
51         y[x[i]]=p-1;
52     }
53     */
54     //=====
55     for(int k=1;k<n;k<=1){
56         swap(x,z);
57         for(int i=0;i<n;++i){
58             int j=z[i]-k;
59             if(j<0) j+=n;
60             x[c[y[j]]++]=j;
61         }
62         swap(y,z);
63         c[y[x[0]]=0]=0; int p=1;
64         for(int i=1;i<n;++i){
65             if(!(z[x[i-1]]==z[x[i]]&&z[x[i-1]+k]==z[x[i]+k])) c[p++]=i; // 00h 鵠 0000 磬 00İjh00v00000000h
66             y[x[i]]=p-1;
67         }
68         if(p==n) break;

```

```

69     }
70     if(x!=sa){ memcpy(c,sa,sizeof(int)*n); memcpy(sa,x,sizeof(int)*n); if(y==sa)y=c; }
71     if(y!=rk) memcpy(rk,y,sizeof(int)*n);
72
73     ht[0][0]=0;
74     for(int i=0,j=rk[0],k=0;i<n-1;++i,++k)
75         while(~k&& s[i]!=s[sa[j-1]+k]) ht[0][j]=k--,j=rk[sa[j]+1];
76     for(int j=1;j<MAXLG;++j)
77         for(int i=1;i+(1<<j)<=n;++i)
78             ht[j][i]=min(ht[j-1][i],ht[j-1][i+(1<<j-1)]);
79
80     subs[0]=0; for(int i=1;i<=n;++i) subs[i]=subs[i-1]+(n-sa[i]-ht[0][i]);
81 }
82 int LCP(int x,int y){ // 计算x与y的LCP
83     if(x==y) return n-x;
84     x=rk[x]+1,y=rk[y];
85     if(x>y) swap(x,y);
86     int k=lg2[y-x+1];
87     return min(ht[k][x],ht[k][y-(1<<k)+1]);
88 }
89 void kth(LL k,int &sl,int &sr){ // 求第k小的子串, (k'0%) 求第k小的子串 Input: k Output: sl,sr
90     int p=upper_bound(subs,subs+n+1,k)-subs;
91     if(p>n) sl=sr=-1;
92     else sl=sa[p],sr=sa[p]+ht[0][p]+(k-sub[sr-1])+1;
93 }
94 int scmp(int xl,int xr,int yl,int yr){ // 比较[xl,xr)与[yl,yr) (-1,0,1)
95     int t=LCP(xl,yl);
96     if(t>=min(xr-xl,yr-yl)) return icmp((xr-xl)-(yr-yl));
97     return icmp(s[xl+t]-s[yl+t]);

```

```
98     }
99     bool check(LL lam, int K) {
100         int sl, sr; kth(lam, sl, sr);
101         int k=0;
102         for(int r=n, l; r; r=l+1, ++k) {
103             for(l=r-1; ~l; --l) if(scmp(sl, sr, l, r)<0) break;
104             if(l==r-1) return 0;
105         }
106         return k<=K;
107     }
108
109     void solve(int K) {
110         LL L=-1, R=subs[n]-1;
111         while(L+1<R) {
112             LL M=(L+R)>>1;
113             if(check(M, K)) R=M; else L=M;
114         }
115         check(R, K);
116         int sl, sr; kth(R, sl, sr);
117         for(int i=sl; i<sr; ++i) putchar(s[i]); putchar('\n');
118     }
119     void print() {
120         puts(s); for(int i=0; i<n; ++i) printf("%d", i%10); putchar('\n'); puts("=====");
121     }
122 } sa;
123
124 char s[MAXN];
125 void solve() {
126     int K; scanf("%d", &K);
```

```
127     scanf("%s",s); n=strlen(s);
128     sa.init(n,s);
129     sa.solve(K);
130 }
131
132 int main(){
133     //     freopen("in.txt","r",stdin);
134     lg2[0]=-1; for(int i=1;i<MAXN;++i) lg2[i]=lg2[i>>1]+1;
135     solve();
136     //     for(;;);
137     return 0;
138 }
```

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

```
1 #include<algorithm>
2 #include<cstdio>
3 #include<cstring>
4 using namespace std;
5
6 const int maxn=100010;
7
8 char s[maxn];
9 int n;
10
```



```

11 // mx: 0000'® v: 0000'®³ fst: '® lst: iY0000q\0'®
12 struct Sam {
13     Sam *pre, *ch[26];
14     int mx, v, fst, lst;
15 } sam[maxn << 1], *samR, *r[maxn << 1], *rot, *now;
16 int num;
17
18 void insert(int x) {
19     Sam *p = now, *np = samR ++;
20     memset(np->ch, 0, sizeof np->ch);
21     np->mx = p->mx + 1; np->v = 0; np->fst = n; np->lst = 0;
22
23     for (; p && !p->ch[x]; p = p->pre) p->ch[x] = np;
24
25     if (!p) np->pre = rot;
26     else {
27         Sam *q = p->ch[x];
28         if (q->mx == p->mx + 1) np->pre = q;
29         else {
30             Sam *nq = samR ++;
31             *nq = *q;
32             nq->mx = p->mx + 1;
33             np->pre = q->pre = nq;
34             for (; p && p->ch[x] == q; p = p->pre) p->ch[x] = nq;
35         }
36     }
37
38     now = np;
39 }

```

```
40
41 char ss[maxn];
42 void dfs(int d, Sam* p) {
43     if(d) puts(ss);
44     for(int x=0; x<26; ++x) if(p->ch[x]) {
45         ss[d]=x+'a';
46         dfs(d+1, p->ch[x]);
47         ss[d]=0;
48     }
49 }
50
51 int c[maxn];
52 void solve() {
53     scanf("%s", s); n = strlen(s);
54
55     rot = now = samR = sam; samR ++;
56     memset(rot->ch, 0, sizeof rot->ch);
57     rot->mx = rot->v = 0; rot->fst = n; rot->lst = 0;
58
59     for (int i = 0; i < n; ++ i) insert(s[i]-'a');
60     num = samR - sam;
61
62     for (int i = 0; i <= n; ++ i) c[i] = 0;
63     for (int i = 0; i < num; ++ i) ++ c[sam[i].mx];
64     for (int i = 1; i <= n; ++ i) c[i] += c[i-1];
65     for (int i = num-1; i >= 0; -- i) r[-- c[sam[i].mx]] = &sam[i];
66
67     Sam *p = rot;
68     for (int i = 0; i < n; ++ i) {
```

```
69         p = p->ch[s[i]-'a'];
70         ++ p->v; p->fst = p->lst = i;
71     }
72     for (int i = num-1; i > 0; -- i) {
73         r[i]->pre->v += r[i]->v;
74         r[i]->pre->fst = min(r[i]->pre->fst, r[i]->fst);
75         r[i]->pre->lst = max(r[i]->pre->lst, r[i]->lst);
76     }
77
78     dfs(0,rot);
79 }
80 int main(){
81     freopen("in.txt","r",stdin);
82     freopen("out.txt","w",stdout);
83     int kase;scanf("%d",&kase);
84     while(kase--){solve();}
85     return 0;
86 }
```

4 图论

4.1 最大流 Dinic

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

```
7     int d[MAXN];
8     map<P,int> idx;
9     int su[MAXN];
10
11     void init(int n) {
12         this->n = n;
13         memset(st, 0, sizeof st); tot = 1;
14         memset(su, 0, sizeof su);
15         idx.clear();
16     }
17
18     void addedge(int u, int v, int w) {
19 //         printf("%d %d %d\n",u,v,w);
20         if (!idx.count(P(u,v))) {
21             lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; del[tot] = 0; st[u] = tot;
22             lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; del[tot] = 0; st[v] = tot;
23             idx[P(u,v)] = tot-1; idx[P(v,u)] = tot;
24         } else {
25             f[idx[P(u,v)]] += w;
26         }
27         su[u] += w; su[v] += w;
28     }
29     bool BFS() {
30         memset(d, 0, sizeof d);
31         l = r = 0;
32         d[ Q[r++] = s ] = 1;
33         for (; l != r; ++ l) {
34             int u = Q[l];
35             for (int i = st[u]; i; i = lk[i]) if (!del[i]) {
```

```
36         int v = b[i];
37         if (f[i] && !d[v]) {
38             d[v] = d[u] + 1;
39             Q[r++] = v;
40         }
41     }
42 }
43 return d[t];
44 }
45 int DFS(int u, int a) {
46     if (u == t || a == 0) return a;
47     int flow = 0, df;
48     for (int& i = st0[u]; i; i = lk[i]) if (!del[i]) {
49         int v = b[i];
50         if (d[v] == d[u] + 1 && (df = DFS(v, std::min(a, f[i])))) {
51             f[i] -= df; f[i^1] += df;
52             a -= df; flow += df;
53
54             if (a == 0) break;
55         }
56     }
57     return flow;
58 }
59 void solve(int s, int t, int& flow) {
60     this->s = s; this->t = t;
61     while (BFS()) {
62         memcpy(st0, st, sizeof st0);
63         flow += DFS(s, INF);
64     }
```

```
65     }
66     bool aug(int s, int t) {
67         this->s = s; this->t = t;
68         if (BFS()) {
69             memcpy(st0, st, sizeof st0);
70             DFS(s, 1);
71             return 1;
72         }
73         return 0;
74     }
75     int flow(int u, int v) {
76         return f[idx[make_pair(u,v)]^1];
77     }
78     void dt(int e) {
79         del[e] ^= 1; del[e^1] ^= 1;
80     }
81     void dt(int u, int v) {
82         dt(idx[make_pair(u,v)]);
83     }
84     int cap(int u) {
85         return su[u];
86     }
87     int isCut(int u, int v) {
88         if(!idx[make_pair(u,v)]) return 1;
89         return d[u] && !d[v] && f[idx[make_pair(u,v)]^1];
90     }
91     bool inS(int u){
92         return d[u];
93     }
```

```
94     void clear() {
95         for (int i = 2; i <= tot; i += 2) {
96             f[i] += f[i^1]; f[i^1] = 0;
97         }
98     }
99 } solver;
```

4.2 最大流 ISAP

```
1 struct ISAP {
2     int n, tot, s, t;
3     int st[MAXN], st0[MAXN];
4     int lk[MAXM << 1], b[MAXM << 1], f[MAXM << 1];
5     int Q[MAXN]; int l, r;
6     int d[MAXN], p[MAXN];
7     int num[MAXN];
8
9     void init(int n) { // all indices should < n
10         this->n = n;
11         memset(st, 0, sizeof st); tot = 1;
12     }
13     void addedge(int u, int v, int w) {
14         lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; st[u] = tot;
15         lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; st[v] = tot;
16     }
17     bool BFS() {
18         for (int u = 0; u < n; ++ u) d[u] = n;
19         l = r = 0;
20         d[ Q[r++] = t ] = 0;
```

```
21     for (; l != r; ++ l) {
22         int u = Q[l];
23         for (int i = st[u]; i; i = lk[i]) {
24             int v = b[i];
25             if (f[i^1] && d[v] >= n) {
26                 d[v] = d[u] + 1;
27                 Q[r++] = v;
28             }
29         }
30     }
31     return ~d[s];
32 }
33
34 int augment() {
35     int a = INF;
36     for (int u = t; u != s; ) {
37         int i = p[u];
38         a = min(a, f[i]);
39         u = b[i^1];
40     }
41     for (int u = t; u != s; ) {
42         int i = p[u];
43         f[i] -= a; f[i^1] += a;
44         u = b[i^1];
45     }
46     return a;
47 }
48
49 void solve(int s, int t, int& flow) {
```



```
50     this->s = s; this->t = t;
51     BFS();
52     memset(num, 0, sizeof num);
53     for (int i = 0; i < n; ++ i) ++ num[d[i]];
54     memcpy(st0, st, sizeof st0);
55     for (int u = s; d[s] < n; ) {
56         if (u == t) {
57             flow += augment();
58             u = s;
59         }
60         int ok = 0;
61         for (int& i = st0[u]; i; i = lk[i]) {
62             int v = b[i];
63             if (f[i] && d[u] == d[v] + 1) {
64                 ok = 1;
65                 p[v] = i;
66                 u = v;
67                 break;
68             }
69         }
70         if (!ok) {
71             int mn = n-1;
72             for (int i = st0[u] = st[u]; i; i = lk[i]) {
73                 int v = b[i];
74                 if (f[i] && d[v] < mn) mn = d[v];
75             }
76             if ((-- num[d[u]]) == 0) break;
77             ++ num[ d[u] = mn+1 ];
78             if (u != s)
```

```

79         u = b[p[u]^1];
80     }
81 }
82 }
83 } solver;

```

4.3 最小费用最大流

```

1 struct MCMF {
2     int n, tot, s, t;
3     int st[MAXN], lk[MAXM << 1], b[MAXM << 1], c[MAXM << 1], f[MAXM << 1];
4     int Q[MAXN]; int l, r;
5     int inq[MAXN];
6     int d[MAXN], p[MAXN], a[MAXN];
7
8     void init(int n) {
9         this->n = n;
10        memset(st, 0, sizeof st); tot = 1;
11    }
12    void addedge(int u, int v, int w, int x) {
13        lk[++ tot] = st[u]; b[tot] = v; f[tot] = w; c[tot] = x; st[u] = tot;
14        lk[++ tot] = st[v]; b[tot] = u; f[tot] = 0; c[tot] = -x; st[v] = tot;
15        // printf("%d %d %d %d\n", u, v, w, x);
16    }
17    bool SPFA(int& flow, int& cost) {
18        l = r = 0;
19        memset(d, 0x3f, sizeof d);
20        memset(inq, 0, sizeof inq);
21        d[ Q[r++] = s ] = 0; inq[s] = 1; a[s] = INF;

```

```
22     for (; l != r; ) {
23         int u = Q[l]; l = (l+1 == MAXN ? 0 : l+1); inq[u] = 0;
24         for (int i = st[u]; i; i = lk[i]) {
25             int v = b[i];
26             if (f[i] && d[v] > d[u] + c[i]) {
27                 d[v] = d[u] + c[i];
28                 p[v] = i;
29                 a[v] = min(a[u], f[i]);
30                 if (!inq[v]) {
31                     if (l == r || d[v] < d[Q[l]]) {
32                         l = (l-1 == -1 ? MAXN-1 : l-1);
33                         Q[l] = v;
34                     } else {
35                         Q[r] = v;
36                         r = (r+1 == MAXN ? 0 : r+1);
37                     }
38                     inq[v] = 1;
39                 }
40             }
41         }
42     }
43
44     if (d[t] == INF) return 0;
45     flow += a[t];
46     cost += a[t] * d[t];
47
48     for (int u = t; u != s; ) {
49         int i = p[u];
50         f[i] -= a[t]; f[i^1] += a[t];
```

```
51         u = b[i^1];
52     }
53     return 1;
54 }
55 void solve(int s, int t, int& flow, int& cost) {
56     this->s = s; this->t = t;
57     while (SPFA(flow, cost));
58 }
59 } solver;
```

4.4 KM

```
1  #define FOR(i, a, b) for (int i = (a); i <= (b); ++ i)
2
3  int n;
4  int W[maxn][maxn];
5  int Lx[maxn], Ly[maxn], slack[maxn];
6  int left[maxn];
7  bool S[maxn], T[maxn];
8
9  bool match(int i) {
10     S[i] = 1;
11     FOR(j, 1, n) if (!T[j]) {
12         int tmp = Lx[i]+Ly[j] - W[i][j];
13         if (tmp == 0) {
14             T[j] = 1;
15             if (!left[j] || match(left[j])) {
16                 left[j] = i;
17                 return 1;
18             }
19         }
20     }
21     return 0;
22 }
```

```
18         }
19     }
20     else if (tmp < slack[j]) slack[j] = tmp;
21 }
22 return 0;
23 }
24
25 void update() {
26     int a = INF;
27     FOR(i, 1, n) if (!T[i] && slack[i] < a) a = slack[i];
28     FOR(i, 1, n) { if (S[i]) Lx[i] -= a; if (T[i]) Ly[i] += a; }
29 }
30
31 void KM() {
32     FOR(i, 1, n) {
33         left[i] = 0; Lx[i] = Ly[i] = 0;
34         FOR(j, 1, n) Lx[i] = max(Lx[i], W[i][j]);
35     }
36     FOR(i, 1, n) for (;;) {
37         FOR(j, 1, n) { S[j] = T[j] = 0; slack[j] = INF; }
38         if (match(i)) break; else update();
39     }
40 }
41
42 void init() {
43     scanf("%d", &n);
44     for (int i = 1; i <= n; ++ i)
45         for (int j = 1; j <= n; ++ j)
46             scanf("%d", &W[i][j]);
```

```
47 }
48
49 void solve() {
50     KM();
51     int ans = 0;
52     for (int i = 1; i <= n; ++ i)
53         ans += W[i][left[i]];
54     printf("%d\n", ans);
55 }
```

4.5 2-SAT

```
1 int n;
2
3 struct TwoSAT {
4     int n;
5     vector<int> G[maxn<<1];
6     bool mark[maxn];
7     int a[maxn<<1], aN;
8
9     void init(int n) : n(n) {
10         for (int i = 0; i < n*2; ++ i) G[i].clear();
11         memset(mark, 0, sizeof mark);
12     }
13     void addedge(int u, int uval, int v, int vval) {
14         u = u * 2 + uval; v = v * 2 + vval;
15         G[u].push_back(v);
16     }
17     bool dfs(int x) {
```

```
18         if (mark[x^1]) return 0;
19         if (mark[x]) return 1;
20         mark[x] = 1;
21         a[aN++] = x;
22         for (int i = 0; i < G[x].size(); ++ i) if (!dfs(G[x][i])) return 0;
23         return 1;
24     }
25     bool solve() {
26         for (int i = 0; i < n*2; i += 2) if (!mark[i] && !mark[i^1]) {
27             aN = 0;
28             if (!dfs(i)) {
29                 for (int i = 0; i < aN; ++ i) mark[a[i]] = 0;
30                 if (!dfs(i^1)) return 0;
31             }
32         }
33         return 1;
34     }
35 } TS;
```

4.6 割点、割边

```
1 int n, m;
2 int st[maxn], lk[maxm << 1], b[maxm << 1];
3 int tot;
4 void addedge(int u, int v) {
5     lk[++ tot] = st[u]; b[tot] = v; st[u] = tot;
6 }
7
8 void init() {
```

```
9     scanf("%d%d", &n, &m);
10
11     memset(st, 0, sizeof st); tot = 1;
12     for (int i = 1; i <= m; ++ i) {
13         int u, v; scanf("%d%d", &u, &v);
14         addedge(u, v); addedge(v, u);
15     }
16 }
17
18 int isnode[maxn];
19 int dfn[maxn], low[maxn];
20 int dfs_clock;
21 void dfs1(int u, int fa) {
22     int ch = 0;
23     dfn[u] = low[u] = ++ dfs_clock;
24     for (int i = st[u]; i; i = lk[i]) if ((i^1) != fa) {
25         int v = b[i];
26         if (!dfn[v]) {
27             ++ ch;
28             dfs1(v, i);
29             if (low[v] >= dfn[u])
30                 isnode[u] = 1;
31             low[u] = std::min(low[u], low[v]);
32         } else {
33             low[u] = std::min(low[u], dfn[v]);
34         }
35     }
36     if (dfn[u] == 1 && ch <= 1) isnode[u] = 0;
37 }
```



```
38 void cut_node() {
39     memset(dfn, 0, sizeof dfn); dfs_clock = 0;
40     dfs1(1, 0);
41
42     int s = 0; for (int i = 1; i <= n; ++ i) if (isnode[i]) ++ s;
43
44     if (s == 0) puts("Null");
45     else {
46         int t = 0;
47         for (int i = 1; i <= n; ++ i) if (isnode[i]) {
48             ++ t; printf("%d%c", i, t == s ? '\n' : ' ');
49         }
50     }
51 }
52
53 std::vector<P> edges;
54
55 void dfs2(int u, int fa) {
56     dfn[u] = low[u] = ++ dfs_clock;
57     for (int i = st[u]; i; i = lk[i]) if ((i^1) != fa){
58         int v = b[i];
59         if (!dfn[v]) {
60             dfs2(v, i);
61             if (low[v] == dfn[v]) {
62                 if (u <= v)
63                     edges.push_back(P(u, v));
64                 else
65                     edges.push_back(P(v, u));
66             }
67         }
68     }
```

```
67         low[u] = std::min(low[u], low[v]);
68     } else {
69         low[u] = std::min(low[u], dfn[v]);
70     }
71 }
72 }
73
74 void cut_edge() {
75     memset(dfn, 0, sizeof dfn); dfs_clock = 0;
76     dfs2(1, 0);
77
78     sort(edges.begin(), edges.end());
79     for (int i = 0; i < (int)edges.size(); ++ i)
80         printf("%d_%d\n", edges[i].first, edges[i].second);
81 }
```

5 计算几何

5.1 模板

```
1 #include<algorithm>
2 #include<cmath>
3 #include<vector>
4 using namespace std;
5
6 const double PI = acos(-1.0);
7 const double eps = 1e-5;
8 const double INF = 1e5;
9
```

```

10 int dcmp(double x) { if (fabs(x) < eps) return 0; else return x < 0 ? -1 : 1; }
11
12 // Point Vector
13 struct Point {
14     double x, y;
15     Point() { }
16     Point(double x, double y) : x(x), y(y) { }
17     bool operator < (const Point &A) const {
18         return dcmp(x-A.x) < 0 || (dcmp(x-A.x) == 0 && dcmp(y-A.y) < 0);
19     }
20     void read(){
21         double x,y; scanf("%lf%lf",&x,&y);
22         this->x=x, this->y=y;
23     }
24 };
25 typedef Point Angle;
26 typedef Point Vector;
27 Point operator + (const Point &A, const Point &B) { return Point(A.x+B.x, A.y+B.y); }
28 Point operator - (const Point &A, const Point &B) { return Point(A.x-B.x, A.y-B.y); }
29 Point operator * (const Point &A, double b) { return Point(A.x*b, A.y*b); }
30 Point operator / (const Point &A, double b) { return Point(A.x/b, A.y/b); }
31 bool operator == (const Point &A, const Point &B) { return dcmp(A.x-B.x) == 0 && dcmp(A.y-B.y) == 0; }
32 double Dot(const Point &A, const Point &B) { return A.x*B.x + A.y*B.y; }
33 double Cross(const Point &A, const Point &B) { return A.x*B.y - A.y*B.x; }
34
35 double Length(const Point &A) { return sqrt(Dot(A, A)); }
36 double Area2(const Point &A, const Point &B, const Point &C) { return Cross(B-A, C-A); }
37 Point Rotate(const Point &A, double ang) {
38     return Point(cos(ang)*A.x - sin(ang)*A.y, sin(ang)*A.x + cos(ang)*A.y);

```

```

39 }
40 Point Normal(const Point &A) {
41     double L = Length(A); return Point(-A.y/L, A.x/L);
42 }
43
44 // Line Segment
45 struct Line {
46     Point A, B, v;
47     double ang;
48
49     Line() { }
50     Line(Point A, Point B): A(A), B(B) { v = B-A; ang = atan2(v.y, v.x); }
51     // ax + by + c > 0
52     Line(double a, double b, double c) {
53         v = Point(b, -a);
54         if (fabs(a) > fabs(b)) A = Point(-c/a, 0); else A = Point(0, -c/b);
55         B = A + v;
56         ang = atan2(v.y, v.x);
57     }
58     bool operator < (const Line &A) const { return ang < A.ang; }
59     Point point(double t) const { return A + v*t; }
60     double pos(Point p) const { return Dot(p-A,v)/Dot(v,v); }
61 };
62
63 // t[0] is the t of L1
64 int LineLineIntersection(const Line& L1, const Line& L2, Point* p, double* t=0) {
65     if(dcmp(Cross(L1.v,L2.v))==0)
66         return dcmp(Cross(L1.v,L2.A-L1.A))==0 ? -1 : 0;
67     p[0]=L1.point(Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v));

```

```

68         if(t) t[0]=Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v);
69         return 1;
70     }
71
72     double DistanceToLine(const Point &P, const Line &L) {
73         return fabs(Cross(L.v, P-L.A)) / Length(L.v);
74     }
75     Point GetLineProjection(const Point &P, const Line &L) {
76         return L.point(Dot(P-L.A, L.v) / Dot(L.v, L.v));
77     }
78     int Position(const Point &P, const Line &L) { return dcmp(Cross(L.v, P-L.A)); }
79
80     double DistanceToSegment(const Point &P, const Line &L) {
81         if (dcmp(Dot(L.v, P-L.A)) < 0) return Length(P-L.A);
82         else if (dcmp(Dot(L.v, P-L.B) > 0)) return Length(P-L.B);
83         else return DistanceToLine(P, L);
84     }
85     bool SegmentProperIntersection(const Line &L1, const Line &L2) {
86         int c1 = Position(L2.A, L1), c2 = Position(L2.B, L1),
87             c3 = Position(L1.A, L2), c4 = Position(L1.B, L2);
88         return c1*c2 < 0 && c3*c4 < 0;
89     }
90
91     int SegmentSegmentIntersection(const Line& L1, const Line& L2, Point* p, double* t=0) {
92         if(dcmp(Cross(L1.v,L2.v))==0)
93             return dcmp(Cross(L1.v,L2.A-L1.A))==0 ? -1 : 1;
94         p[0]=L1.point(Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v));
95         if(t) t[0]=Cross(L2.v, L1.A-L2.A) / Cross(L1.v, L2.v);
96         return 1;

```

```
97 }
98
99 bool OnSegment(const Point &P, const Line &L) {
100     return dcmp(Cross(L.A-P, L.B-P)) == 0 &&
101           dcmp(Dot(L.A-P, L.B-P)) <= 0;
102 }
103 bool OnSegment2(const Point &P, const Line &L) {
104     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;
105 }
106
107 // Polygon
108 typedef vector<Point> Polygon;
109 typedef std::vector<Line> Lines;
110 void print(const Polygon& A) {
111     for (int i = 0; i < A.size(); ++ i) printf("%.2lf_%.2lf\n", A[i].x, A[i].y);
112 }
113
114 double PolygonArea(const Polygon &p) {
115     int n = p.size();
116     double area = 0;
117     for (int i = 1; i < n-1; ++ i)
118         area += Area2(p[0], p[i], p[i+1]);
119     return area / 2.0;
120 }
121 int isPointInPolygon(const Point &P, const Polygon &p) {
122     int n = p.size();
123     int wn = 0;
124     for (int i = 0; i < n; ++ i) {
125         const Point &p1 = p[i]; const Point &p2 = p[(i+1)%n];
```

```

126         if (P == p1 || P == p2 || OnSegment(P, Line(p1, p2))) return -1;
127         int k = dcmp(Cross(p2-p1, P-p1));
128         int d1 = dcmp(p1.y-P.y);
129         int d2 = dcmp(p2.y-P.y);
130         if (k > 0 && d1 <= 0 && d2 > 0) ++ wn;
131         if (k < 0 && d2 <= 0 && d1 > 0) -- wn;
132     }
133     return wn != 0;
134 }
135 // ConvexHull
136 Polygon ConvexHull(Polygon p) {
137     int n = p.size();
138     sort(p.begin(), p.end());
139
140     // int n = p.erase(std::unique(p.begin(), p.end()), p.end());
141     Polygon q(n+1);
142     int m = 0;
143     for (int i = 0; i < n; ++ i) {
144         while (m > 1 && Cross(q[m-1]-q[m-2], p[i]-q[m-2]) <= 0) -- m;
145         q[m++] = p[i];
146     }
147     int k = m;
148     for (int i = n-2; i >= 0; -- i) {
149         while (m > k && Cross(q[m-1]-q[m-2], p[i]-q[m-2]) <= 0) -- m;
150         q[m++] = p[i];
151     }
152     if (n > 1) m--;
153     q.resize(m);
154     return q;

```

```
155 }
156 double ConvexHullMaxDist(const Polygon &poly) {
157     int n = poly.size();
158     if (n == 2) return Length(poly[1]-poly[0]);
159
160     int i = min_element(poly.begin(), poly.end()) - poly.begin(),
161         j = max_element(poly.begin(), poly.end()) - poly.begin();
162     double res = 0.0;
163     for (int si = i, sj = j; i != sj || j != si; ) {
164         res = max(res, Length(poly[j]-poly[i]));
165         if (Cross(poly[(i+1)%n]-poly[i], poly[(j+1)%n]-poly[j]) < 0) {
166             i = (i+1) % n;
167         } else {
168             j = (j+1) % n;
169         }
170     }
171     return res;
172 }
173 double ConvexHullMinDist(const Polygon &poly) {
174     int n = poly.size();
175     if (n == 2) return Length(poly[1]-poly[0]);
176
177     int i = min_element(poly.begin(), poly.end()) - poly.begin(),
178         j = max_element(poly.begin(), poly.end()) - poly.begin();
179     double res = INF;
180     for (int si = i, sj = j; i != sj || j != si; ) {
181         if (Cross(poly[(i+1)%n]-poly[i], poly[(j+1)%n]-poly[j]) < 0) {
182             res = min(res, DistanceToSegment(poly[j], Line(poly[i], poly[(i+1)%n])));
183             i = (i+1) % n;
```



```
184         } else {
185             res = min(res, DistanceToSegment(poly[i], Line(poly[j], poly[(j+1)%n])));
186             j = (j+1) % n;
187         }
188     }
189     return res;
190 }
191 double MinDist(const Polygon& A, const Polygon& B){
192     int n=A.size(), m=B.size();
193     // if (n<3||m<3) for(;;);
194
195     int i = min_element(A.begin(), A.end()) - A.begin(),
196         j = max_element(B.begin(), B.end()) - B.begin();
197     double res = INF;
198     int si = i, sj = j;
199     do {
200         if (Cross(A[(i+1)%n]-A[i], B[(j+1)%m]-B[j]) < 0) {
201             res = min(res, DistanceToSegment(B[j], Line(A[i], A[(i+1)%n])));
202             i = (i+1) % n;
203         } else {
204             res = min(res, DistanceToSegment(A[i], Line(B[j], B[(j+1)%m])));
205             j = (j+1) % m;
206         }
207     } while (i != si || j != sj);
208     return res;
209 }
210
211 Polygon simplify(const Polygon& poly){
212     Polygon ans;
```

```
213     int n=poly.size();
214     for(int i=0;i<n;++i){
215         Point a=poly[i], b=poly[(i+1)%n], c=poly[(i+2)%n];
216         if(dcmp(Cross(a-b,c-b))!=0) ans.push_back(b);
217     }
218     return ans;
219 }
220
221 // HalfplaneIntersection
222 Polygon CutPolygon(const Polygon &poly, Line L) {
223     Polygon newpoly;
224     int n = poly.size();
225     for (int i = 0; i < n; ++ i) {
226         const Point &p1 = poly[i]; const Point &p2 = poly[(i+1)%n];
227         const Line &l = Line(p1, p2);
228         if (Position(p1, L) >= 0) newpoly.push_back(p1);
229
230         Point ip; int x=LineLineIntersection(L,l,&ip);
231         if (x==1){
232             if (OnSegment2(ip, l)) newpoly.push_back(ip);
233         }
234     }
235     return newpoly;
236 }
237 Polygon HalfplaneIntersection(Lines L) {
238     int n = L.size();
239     sort(L.begin(), L.end());
240
241     Lines Q(n); int l, r;
```

```

242     Polygon P(n);
243     Polygon ans;
244
245     l = r = 0; Q[r++] = L[0];
246     for (int i = 1; i < n; ++ i) {
247         while (l+1 < r && Position(P[r-2], L[i]) <= 0) -- r;
248         while (l+1 < r && Position(P[l], L[i]) <= 0) ++ l;
249         Q[r++] = L[i];
250         if (dcmp(Cross(Q[r-2].v, Q[r-1].v)) == 0) {
251             -- r;
252             if (Position(L[i].A, Q[r-1]) > 0)
253                 Q[r-1] = L[i];
254         }
255         if (l+1 < r) LineLineIntersection(Q[r-2], Q[r-1], &P[r-2]);
256     }
257     while (l+1 < r && Position(P[r-2], Q[l]) <= 0) -- r;
258     if (l+2 >= r) return ans;
259     LineLineIntersection(Q[r-1], Q[l], &P[r-1]);
260
261     for (int i = 1; i < r; ++ i) ans.push_back(P[i]);
262     return ans;
263 }
264
265
266 //Angle
267 typedef Point Angle;
268 Point Rotate(const Point& A, const Angle& a) {
269     return Point(a.x*A.x-a.y*A.y, a.y*A.x+a.x*A.y);
270 }

```

```
271
272 //Circle
273 struct Circle{
274     Point c;double r;
275     Circle(){}
276     Circle(Point c,double r):c(c),r(r){}
277     Point point(double ang) const{
278         return Point(c.x+cos(ang)*r,c.y+sin(ang)*r);
279     }
280     Point point(double cosa,double sina) const{
281         return Point(c.x+cosa*r,c.y+sina*r);
282     }
283 };
284 int LineCircleIntersection(const Line& L,const Circle& C,Point* p,double* t=0){
285     double a=L.v.x, b=L.A.x-C.c.x, c=L.v.y, d=L.A.y-C.c.y;
286     double e=a*a+c*c, f=2*(a*b+c*d), g=b*b+d*d-C.r*C.r;
287     double delta=f*f-4*e*g;
288     if(dcmp(delta)<0) return 0;
289     if(dcmp(delta)==0){
290         p[0]=L.point(-f/(2*e));
291         if(t) t[0]=-f/(2*e);
292         return 1;
293     }
294     p[0]=L.point((-f-sqrt(delta))/(2*e)),p[1]=L.point((-f+sqrt(delta))/(2*e));
295     if(t) t[0]=(-f-sqrt(delta))/(2*e), t[1]=(-f+sqrt(delta))/(2*e);
296     return 2;
297 }
298 int CircleCircleIntersection(const Circle& C1,const Circle& C2,Point* p,Angle* a=0){
299     double d=Length(C2.c-C1.c);
```

```

300     if (dcmp(d)==0) {
301         if (dcmp(C1.r-C2.r)==0) return -1; // 点重合
302         return 0; // 点在圆上(1)
303     }
304     if (dcmp(C1.r+C2.r-d)<0) return 0; // 圆外
305     if (dcmp(fabs(C1.r-C2.r)-d)>0) return 0; // 圆外(2)
306     double cosa=(C1.r*C1.r+d*d-C2.r*C2.r)/(2*C1.r*d), sina=sqrt(1-cosa*cosa);
307
308     Point v=(C2.c-C1.c)/Length(C2.c-C1.c)*C1.r;
309
310     p[0]=C1.c+Rotate(v,Angle(cosa,-sina)), p[1]=C1.c+Rotate(v,Angle(cosa,sina));
311     if (a) a[0]=Rotate(v,Angle(cosa,-sina))/Length(v), a[1]=Rotate(v,Angle(cosa,sina))/Length(v);
312     if (p[0]==p[1]) return 1; else return 2;
313 }
314 Circle CircumscribedCircle(const Point& A, const Point& B, const Point& C) {
315     double da=Dot(B-A,C-A),db=Dot(A-B,C-B),dc=Dot(A-C,B-C);
316     double ka=db*dc,kb=da*dc,kc=da*db;
317     Point D=(A*(kb+kc)+B*(ka+kc)+C*(ka+kb))/(2*(ka+kb+kc));
318     return Circle(D,Length(A-D));
319 }
320 Circle InscribedCircle(const Point& A, const Point& B, const Point& C) {
321     double a=Length(C-B),b=Length(C-A),c=Length(B-A);
322     Point p=(A*a+B*b+C*c)/(a+b+c);
323     return Circle(C,fabs(Area2(A,B,C))/(a+b+c));
324 }
325 int CirclePointTangents(const Circle& C,const Point& P,Point* p,Angle* ang=0) {
326     Point v=C.c-P; double d=Length(v);
327     if (dcmp(d-C.r)<0) return 0;
328     if (dcmp(d-C.r)==0) {

```

```

329         p[0]=P;
330         if(ang) ang[0]=Rotate(v,PI/2);
331         return 1;
332     }
333
334     double a=C.r,c=d,b=sqrt(c*c-a*a);
335     double sina=a/c, cosa=b/c;
336     p[0]=P+Rotate(v,Angle(cosa,-sina))/c*b,p[1]=P+Rotate(v,Angle(cosa,sina))/c*b;
337     if(ang) ang[0]=Rotate(v,Angle(cosa,-sina))/c, ang[1]=Rotate(v,Angle(cosa,sina))/c;
338     return 2;
339 }
340
341 // ===== CirclePolygonIntersectionArea =====
342 double SectorArea(const Circle& C, const Point& A, const Point& B) {
343     double ang=atan2(A.y,A.x)-atan2(B.y,B.x);
344     while(ang<=0) ang+=2*PI;
345     while(ang>2*PI) ang-=2*PI;
346     ang=min(ang,2*PI-ang);
347     return C.r*C.r*ang/2;
348 }
349 double CircleTriangleIntersectionArea(Circle C, Point A, Point B) {
350     A=A-C.c, B=B-C.c; C.c=Point(0,0);
351     int sgn=dcmp(Cross(A, B));
352     if(sgn==0) return 0;
353
354     Line L=Line(A,B);
355     Point p[2];
356     int num=0;
357     int ina=dcmp(Length(A)-C.r)<0;

```

```

358     int inb=dcmp (Length (B)-C.r)<0;
359     if (ina){
360         if (inb){
361             return sgn*(fabs (Cross (A,B))/2.0);
362         }else{
363             LineCircleIntersection (L,C,p);
364             return sgn*(fabs (Cross (A,p[1]))/2.0+SectorArea (C,p[1],B));
365         }
366     }else{
367         if (inb) {
368             LineCircleIntersection (L,C,p);
369             return sgn*(fabs (Cross (B,p[0]))/2.0+SectorArea (C,p[0],A));
370         }else{
371             int num=LineCircleIntersection (L,C,p);
372             if (num==2 && OnSegment2 (p[0],L) && OnSegment2 (p[1],L)){
373                 return sgn*(SectorArea (C,A,p[0])+SectorArea (C,p[1],B)+fabs (Cross (p[0],p[1]))/2.0);
374             }else{
375                 return sgn*(SectorArea (C,A,B));
376             }
377         }
378     }
379 }
380 double CirclePolygonIntersectionArea (const Circle& C,const Polygon& P) {
381     int n=P.size();
382     double res=0;
383     for (int i=0;i<n;++i){
384         res+=CircleTriangleIntersectionArea (C,P[i],P[(i+1)%n]);
385     }
386     return res;

```

```

387 }
388
389 // 计算n个圆的交集面积
390 void CirclesIntersectionArea(vector<Circle>& vc, double* ans) {
391     int n=vc.size();
392     for(int i=1;i<=n;++i) ans[i]=0;
393     static vector< pair<double, int> > ev;
394     for(int i=0;i<n;++i){
395         int cv=0;
396         ev.clear();
397         for(int j=0;j<n;++j) if(j!=i){
398             Point sol[2], ang[2];
399             int t=CircleCircleIntersection(vc[i],vc[j],sol,ang);
400             if(t==2) {
401                 double a1=atan2(ang[0].y,ang[0].x);
402                 double a2=atan2(ang[1].y,ang[1].x);
403                 if(a1<a2) ev.push_back(make_pair(a1,1)),ev.push_back(make_pair(a2,-1));
404                 else     ev.push_back(make_pair(a1,1)),ev.push_back(make_pair(PI,-1)),
405                         ev.push_back(make_pair(-PI,1)),ev.push_back(make_pair(a2,-1));
406             } else if(t==-1){
407                 if(i<j) ++ cv;
408             } else{
409                 int rd=dcmp(Length(vc[i].c-vc[j].c)-(vc[j].r-vc[i].r));
410                 if(rd<=0) ++ cv;
411             }
412         }
413         ev.push_back(make_pair(-PI,1)),ev.push_back(make_pair(PI,-1));
414         sort(ev.begin(),ev.end());
415     }

```



```

416     double lst=-PI;
417     for(int l=0,r;l<ev.size();){
418         double a=ev[l].first-lst;
419         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);
420         lst=ev[l].first;
421
422         for(r=l+1;r<ev.size()&&ev[r].first<=ev[l].first;++r);
423         for(int p=l;p<r;++p) cv+=ev[p].second;
424         l=r;
425     }
426 }
427 }
428
429 // 计算多边形面积
430 void PolygonsIntersectionArea(vector<Polygon>& ps,double* ans) {
431     int n=ps.size();
432     for(int i=0;i<n+1;++i) ans[i]=0;
433
434     static vector< pair<double, int> > ev;
435     for(int i=0;i<n;++i) for(int pi=0;pi<ps[i].size();++pi){
436         Line L0=Line(ps[i][pi],ps[i][(pi+1)%ps[i].size()]);
437         ev.clear();
438         for(int j=0;j<n;++j) if(i!=j) for(int pj=0;pj<ps[j].size();++pj){
439             Line L=Line(ps[j][pj],ps[j][(pj+1)%ps[j].size()]);
440
441             int p1=Position(L.A,L0), p2=Position(L.B,L0);
442             if(!p1 && !p2){
443                 if(i<j && dcmp(Dot(L0.v,L.v))>0){
444                     ev.push_back(make_pair(min(max(L0.pos(L.A),0.0),1.0), 1));

```

```

445         ev.push_back(make_pair(min(max(L0.pos(L.B),0.0),1.0), -1));
446     }
447     }else{
448         Point p; double t;
449         int x=LineLineIntersection(L0,L,&p,&t);
450         if(p1>=0 && p2<0) ev.push_back(make_pair(min(max(t,0.0),1.0), 1));
451         if(p1<0 && p2>=0) ev.push_back(make_pair(min(max(t,0.0),1.0), -1));
452     }
453 }
454 ev.push_back(make_pair(0.0,1));
455 ev.push_back(make_pair(1.0,-1));
456 sort(ev.begin(),ev.end());
457
458 int cv=0;
459 double S0=Cross(L0.A,L0.B)/2.0;
460 double lst=0;
461 for(int l=0,r;l<ev.size();){
462     ans[cv]+=S0*(ev[l].first-lst);
463     lst=ev[l].first;
464
465     for(r=l+1;r<ev.size()&&ev[r].first<=ev[l].first;++r);
466     for(int p=l;p<r;++p) cv+=ev[p].second;
467     l=r;
468 }
469 }
470 }
471
472 // examples
473 int CircleThroughAPointAndTangentToALineWithRadius(const Point& P, const Line& L, double r, Point* p){

```

```
474     Point v=Normal(L.v)*r;
475     Line l1=Line(L.A+v, L.v), l2=Line(L.A-v, L.v);
476     Circle CP=Circle(P,r);
477     int x=0;
478     x+=LineCircleIntersection(l1,CP,p+x);
479     x+=LineCircleIntersection(l2,CP,p+x);
480     return x;
481 }
482
483 int CircleTangentToTwoLinesWithRadius(const Line& L1, const Line& L2, double r, Point* p){
484     Point v1=Normal(L1.v)*r,v2=Normal(L2.v)*r;
485     int x=0;
486     x+=LineLineIntersection(Line(L1.A+v1,L1.v), Line(L2.A+v2,L2.v),p);
487     x+=LineLineIntersection(Line(L1.A-v1,L1.v), Line(L2.A+v2,L2.v),p);
488     x+=LineLineIntersection(Line(L1.A+v1,L1.v), Line(L2.A-v2,L2.v),p);
489     x+=LineLineIntersection(Line(L1.A-v1,L1.v), Line(L2.A-v2,L2.v),p);
490     return x;
491 }
492 int CircleTangentToTwoDisjointCirclesWithRadius(const Circle& C1, const Circle& C2, double r, Point* p){
493     int x=0;
494     x+=CircleCircleIntersection(Circle(C1.c,C1.r+r), Circle(C2.c,C2.r+r), p);
495     return x;
496 }
497
498 int main() {
499     return 0;
500 }
```

5.2 最小覆盖圆

```
1 int n;
2
3 struct Point {
4     double x, y;
5 } p[maxn];
6
7 Point C; double r;
8
9 Point getCir(const Point &A, const Point &B, const Point &C) {
10     Point tmp;
11     double a1 = B.x-A.x, b1 = B.y-A.y, c1 = (a1*a1 + b1*b1) / 2.0,
12           a2 = C.x-A.x, b2 = C.y-A.y, c2 = (a2*a2 + b2*b2) / 2.0,
13           d = a1*b2 - a2*b1;
14     tmp.x = A.x + (c1*b2-c2*b1) / d;
15     tmp.y = A.y + (a1*c2-a2*c1) / d;
16     return tmp;
17 }
18
19 void minCircle() {
20     std::random_shuffle(p+1, p+1+n);
21
22     C = p[1]; r = 0;
23     for (int i = 2; i <= n; ++ i) if (Length(p[i]-C) > r + eps) {
24         C = p[i];
25         r = 0;
26         for (int j = 1; j < i; ++ j) if (Length(p[j]-C) > r + eps) {
27             C = (p[i]+p[j]) / 2.0;
28             r = Length(p[j]-p[i]) / 2.0;
```

```

29         for (int k = 1; k < j; ++ k) if (Length(p[k]-C) > r + eps) {
30             C = getCir(p[i], p[j], p[k]);
31             r = Length(p[i]-C);
32         }
33     }
34 }
35 }
36 void init() {
37     for (int i = 1; i <= n; ++ i)
38         scanf("%lf%lf", &p[i].x, &p[i].y);
39 }
40 void solve() {
41     minCircle();
42     printf("%.2lf_%.2lf_%.2lf\n", C.x, C.y, r);
43 }

```

6 DP

6.1 数位 dp

```

1 int base = 10;
2
3 int memo[20][10][2];
4 int f(int d, int s, bool zero) {
5     int &ans = memo[d][s][zero];
6     if (ans != -1) return ans;
7
8     if (d == 0) return ans = 1;
9     ans = 0;

```

```
10     for (int x = 0; x < base; ++ x) if (zero || abs(s - x) >= 2)
11         ans += f(d-1, x, (x == 0) && zero);
12     return ans;
13 }
14
15 int digits[20];
16 int sumf(unsigned n) {
17     int m = 0;
18     for (; n; n /= base)
19         digits[m ++] = n % base;
20
21     int ans = 0;
22     for (int i = m-1; i >= 0; -- i) {
23         int j = i, x;
24         for (x = 0; x < digits[i]; ++ x) if (i == m-1 || abs((digits[i+1]) - x) >= 2)
25             ans += f(j, x, (x == 0) && (i == m-1));
26         if (!(i == m-1 || abs(digits[i+1] - x) >= 2)) break;
27         /* do something here */
28     }
29     return ans;
30 }
31
32 int a, b;
33
34 void solve() {
35     scanf("%d%d", &a, &b);
36     memset(memo, 255, sizeof memo);
37     printf("%d\n", sumf((unsigned)b+1) - sumf((unsigned)a));
38 }
```

6.2 状压 dp

```
1      int S;  
2      // sub  
3      int SS = S;  
4      do {  
5          // solve  
6          SS = (SS-1) & S;  
7      } while (SS != S);  
8  
9      //k-sub  
10     int k;  
11     SS = (1<<k)-1;  
12     while (SS < (1<<n)) {  
13         //solve  
14         int x = SS & -SS, y = SS + x;  
15         SS = (((SS & ~y) / x) >> 1) | y;  
16     }
```

7 数据结构

7.1 Hash

```
1  #include <cstring>  
2  const int Hmod = 4000001;  
3  
4  template <typename _Value>  
5  struct HASH {  
6      LL a[Hmod + 1000]; _Value b[Hmod + 1000];  
7  }
```

```
7
8     void clear() {
9         memset(a, 0xff, sizeof a);
10    }
11    void insert(LL key, _Value value) {
12        int p; for (p = key % Hmod; a[p] != -1; ++ p);
13        a[p] = key; b[p] = value;
14    }
15    int count(LL key) {
16        int p; for (p = key % Hmod; a[p] != -1 && a[p] != key; ++ p);
17        return a[p] != -1;
18    }
19    _Value query(LL key) {
20        int p; for (p = key % Hmod; a[p] != -1 && a[p] != key; ++ p);
21        return b[p];
22    }
23 };
24
25 HASH<int> hash;
```

7.2 线段树

```
1 #include<algorithm>
2
3 const int MAXN=100000;
4
5 template<class ct,class mt>struct Seg {
6     ct c;mt x;
7     Seg *ch[2];
```



```

8         Seg() {}
9         Seg(ct c,mt x):c(c),x(x) {}
10 // void* operator new(unsigned,void* p){ return p; }
11 void pass() {
12     if(x.empty()) return;
13     if(ch[0]){ merge(ch[0]->x, x); merge(ch[0]->c, x); }
14     if(ch[1]){ merge(ch[1]->x, x); merge(ch[1]->c, x); }
15     new(&x)mt();
16 }
17 };
18 template<class ct,class mt>struct SegSeq {
19     typedef Seg<ct, mt> _Seg;
20     int n,ql,qr;
21     ct cv;mt mv;
22     _Seg seg[MAXN],*segR,*rt;
23
24     int *A;
25     _Seg* build(int l, int r) {
26         int m=(l+r)>>1;
27         _Seg *o=new(segR++)_Seg(ct(),mt());
28         if(l==r){ new(&o->c)ct(A[l]); o->ch[0] = o->ch[1] = 0; return o; }
29         o->ch[0]=build(l,m); o->ch[1]=build(m+1,r);
30         o->c=merge(o->ch[0]->c,o->ch[1]->c);
31         return o;
32     }
33     void init(int n=MAXN,int *A=0){
34         this->n=n; this->A=A;
35         segR=seg; rt=build(1,n);
36     }

```

```

37     void update(_Seg*& o,int l,int r){
38         int m=(l+r)>>1;
39         o->pass();
40         if (ql<=l&&qr<=qr){ merge(o->c, mv); merge(o->x, mv); return; }
41         if (ql<=m) update(o->ch[0],l,m);if(m<qr) update(o->ch[1],m+1,r);
42         o->c=merge(o->ch[0] ? o->ch[0]->c : ct(),o->ch[1] ? o->ch[1]->c : ct());
43     }
44     void query(_Seg*& o,int l,int r){
45         int m=(l+r)>>1;
46         if(!o)o=new(segR++)_Seg();
47         o->pass();
48         if (ql<=l && r<=qr) { cv=merge(cv, o->c); return; }
49         if (ql<=m) query(o->ch[0],l,m);if(m<qr) query(o->ch[1],m+1,r);
50     }
51     void update(int ql, int qr, const mt& mv) {
52         this->ql=ql,this->qr = qr; this->mv=mv; update(rt,l,n);
53     }
54     ct query(int ql, int qr) {
55         this->ql=ql,this->qr=qr; new(&cv)ct(); query(rt,l,n); return cv;
56     }
57 };

```

7.3 Splay

```

1 #include<algorithm>
2 #include<new>
3 using namespace std;
4
5 //=====Splay Begin=====

```

```

6 // pool template is needed.
7 // example:
8
9 //please init() before use
10 template<class T>struct Pool{
11     T *a,**q; int pa,pq;
12     Pool(int MAXN){a=new T[MAXN];q=new T*[MAXN];}
13     // ~Pool(){delete[] a; delete[] q;}
14     void init(){pa=0;pq=0;}
15     T* NEW(){return new(pq ? q[--pq] : &a[pa++])T();}
16     void DELETE(T* x){q[pq++]=x; return;}
17 };
18
19 // ct & mt & it is needed
20 // ct::rev() is needed.
21 // example:
22 struct ct{int mx;ct(){mx=-1;}ct(int mx):mx(mx){}ct rev(){return *this;}
23 bool empty()const{return mx==-1;}};
24 struct mt{int ept,v;mt(){ept=1;}mt(int v):v(v){ept=0;}
25 bool empty()const{return ept;}mt rev(){return *this;}};
26 ct operator+(const ct& A,const ct& B){
27     if(B.empty())return A;if(A.empty())return B;return ct(max(A.mx,B.mx));
28 }
29 void operator+=(ct& A,const mt& B){if(!B.empty()) new(&A)ct(B.v);}
30 void operator+=(mt& A,const mt& B){if(!B.empty()) new(&A)mt(B.v);}
31
32 //=====Splay Begin=====
33
34 namespace Splay{

```

```

35     const int MAXNODE=50010; // total number of SplayNode
36     const int MAXN=50010; // size of one SplayTree
37     struct node{
38         node *ch[2],*p;
39         int sz,int rev;
40         ct c,v;mt x;
41         node(){new(&c)ct();new(&x)mt();ch[0]=ch[1]=p=0;v=sz=rev=0;}
42         node(ct c,mt x):c(c),x(x){}
43         bool isRoot(){return !p || (p->ch[0]!=this && p->ch[1]!=this);} // LCT sp
44         int getlr(){return p->ch[1]==this; } // "this" mustn't be root
45         node* link(int x,node* o){ch[x]=o;if(o)o->p=this;return this;}
46         node* unlink(int x){if(ch[x])ch[x]->p=0;ch[x]=0;return this;}
47         node* reverse(){rev^=1;swap(ch[0],ch[1]);c=c.rev();x=x.rev();return this;}
48         node* modify(const mt& X){
49             // if it is a Point modify, please erase x+=X
50             x+=X;c+=X;v+=X;return this;
51         }
52         node* upd(){
53             if(!ch[0])if(!ch[1]){c=v;sz=1;}
54             else{c=v+ch[1]->c;sz=1+ch[1]->sz;}
55             else if(!ch[1]){c=ch[0]->c+v;sz=ch[0]->sz+1;}
56             else{c=ch[0]->c+v+ch[1]->c;sz=ch[0]->sz+1+ch[1]->sz;}
57             return this;
58         }
59         node* pass(){
60             if(rev){if(ch[0])ch[0]->reverse();if(ch[1])ch[1]->reverse();rev=0;}
61             if(!x.empty()){if(ch[0])ch[0]->modify(x);if(ch[1])ch[1]->modify(x);new(&x)mt();}
62             return this;
63         }

```

```

64         node* rotate(){ // p mustn't be root
65             node* q=p->p; int x=getlr(),y=p->isRoot() ? -1 : p->getlr();
66             link(x^1,p->link(x,ch[x^1]));
67             p->upd();
68             if(y== -1)p=q;else q->link(y,this);
69             return this;
70         }
71     };
72     Pool<node> pool(MAXNODE);
73     void passAll(node* o,node* tar){
74         static node* sk[MAXN];
75         static int tp;
76         tp=0;
77         for(;o->p!=tar;o=o->p) sk[tp++]=o;
78         sk[tp++]=o;
79         for(;tp;--tp) sk[tp-1]->pass();
80     }
81     // every node in [o,tar) should have already been passed.
82     node* splay(node* o,node* tar,bool passed=0){
83         if(!passed)passAll(o,tar);
84         if(o->p==tar) return o;
85         while(o->p!=tar&&o->p->p!=tar)
86             o->getlr()==o->p->getlr() ? (o->p->rotate(),o->rotate())
87             : (o->rotate(),o->rotate());
88         if(o->p!=tar)o->rotate();
89         return o->upd();
90     }
91     void passAllLCT(node* o){ // for LCT
92         static node* sk[MAXN];

```

```

93         static int tp;
94         tp=0;
95         for(;!o->isRoot();o=o->p) sk[tp++]=o;
96         sk[tp++]=o;
97         for(;tp;--tp) sk[tp-1]->pass();
98     }
99     node* splayLCT(node* o,bool passed=0){ // for LCT
100         if(!passed)passAllLCT(o);
101         if(o->isRoot())return o;
102         while(!o->isRoot()&&!o->p->isRoot())
103             o->getlr()==o->p->getlr()? (o->p->rotate(),o->rotate())
104             : (o->rotate(),o->rotate());
105         if(!o->isRoot())o->rotate();
106         return o->upd();
107     }
108     node* build(int* A,int l, int r) {
109         if(l>r)return 0;
110         int m=(l+r)>>1;
111         node* o=pool.NEW();
112         o->link(0,build(A,l,m-1));o->link(1,build(A,m+1,r));
113         o->v=A[m];o->upd();new(&o->x)mt();
114         return o;
115     }
116     void erase(node* o){
117         if(o->ch[0])erase(o->ch[0]);
118         if(o->ch[1])erase(o->ch[1]);
119         pool.DELETE(o);
120     }
121     void eraseAll(node* o){splay(o,0);erase(o);}

```

```

122 // k is from 1
123 node* splayk(node* o, int k) {
124     node *p=o; int w;
125     for(;p->pass(), (w=p->ch[0]?p->ch[0]->sz:0)+1!=k;)
126         if(k<=w)p=p->ch[0];else{k-=w+1;p=p->ch[1];}
127     if((w=p->ch[0]?p->ch[0]->sz:0)+1!=k)o=0;else o=p;
128     splay(o,0,1);
129     return o;
130 }
131 int rank(node* o){splay(o,0);return (o->ch[0] ? o->ch[0]->sz : 0)+1;}
132 // split k elements to o and others to R(when k==0 remain nothing)
133 void split(node* o, int k, node* R){if(k==0){R=o;o=0;return;}
134 splayk(o,k);R=o->ch[1];o->unlink(1)->upd();}
135 void merge(node* L, node* R){if(!L){L=R;return;}
136 splayk(L,L->sz);L->link(1,R)->upd();}
137 // some functions refer to sequence options
138 node* pick(node* o, int l, int r){
139     splay(o,0);
140     if(l<0 || r>o->sz || l>r)return 0;
141     if(l==1)if(r==o->sz)return o;
142         else return splayk(o,r+1)->ch[0];
143     else if(r==o->sz) return splayk(o,l-1)->ch[1];
144         else{
145         node*p=splayk(o,l-1),*q=splayk(o,r+1);
146         splay(p,q);return p->ch[1];
147     }
148 }
149 void reverse(node* o, int l, int r){
150     if(l>r)return; splay(pick(o,l,r)->reverse(),0);

```

```
151     }
152     void update(node* o,int l,int r,const int& v){
153         if(l>r) return; splay(pick(o,l,r)->modify(v),0);
154     }
155
156     //debugging....
157     void print(node* o){
158         o->pass();
159         if(o->ch[0]) print(o->ch[0]);
160         printf("%p\\",o);
161         if(o->ch[1]) print(o->ch[1]);
162     }
163     void reupd(node* o){
164         o->pass();
165         if(o->ch[0]) reupd(o->ch[0]);
166         if(o->ch[1]) reupd(o->ch[1]);
167         o->upd();
168     }
169 }
170
171 using namespace Splay;
172
173 //=====Splay End=====
174
175 //examples:
176
177 //BZOJ 1500 ㄅㄅㄅㄅㄅㄅ
178 //struct myDS{
179 //     typedef SplayNode node;
```



```
180 //      Splay t;
181 //      void insert(int p,int n,int* a){
182 //          Splay R=split(p);merge(build(a,l,n));merge(R);
183 //      }
184 //      void erase(int l,int r){
185 //          if(l>r)return;
186 //          Splay M,R;
187 //          M=t.split(l-1);R=M.split(r-l+1);
188 //          t.merge(R);M.erase();
189 //      }
190 //      int query(int Tp,int l=0,int r=0){
191 //          if(l>r)return 0;
192 //          if(Tp==0)return t.rt->c.mxs;
193 //          node* p=t.pick(l,r);
194 //          int v=p->c.s;
195 //          t.splay(p->pass(),0);
196 //          return v;
197 //      }
198 //      void print(){
199 //          t.print();
200 //      }
201 //}BST;
202
203 //LA 3961
204 //struct myDS{
205 //      Splay t;
206 //      pair<int,int>Y[maxn];
207 //      int n;int A[maxn];
208 //      SplayNode* pos[maxn];
```

```
209 //
210 //      void getpos(SplayNode* o){
211 //          pos[o->v]=o;
212 //          if(o->ch[0])getpos(o->ch[0]);
213 //          if(o->ch[1])getpos(o->ch[1]);
214 //      }
215 //      bool solve(){
216 //          if(!(scanf("%d",&n)==1&&n))return 0;
217 //          for(int i=1;i<=n;++i){
218 //              int x;scanf("%d",&x);
219 //              Y[i]=make_pair(x,i);
220 //          }
221 //          sort(Y+1,Y+1+n);
222 //          for(int i=1;i<=n;++i)A[Y[i].second]=i;
223 //
224 //          splayPool.init();new(&t)Splay();
225 //          t.merge(t.build(A,1,n));
226 //          getpos(t.rt);
227 //          for(int i=1;i<=n;++i){
228 //              int x=t.rank(pos[i]);
229 //              printf("%d%c",x,i==n?'\\n':' ');
230 //              t.reverse(i,x);
231 //          }
232 //          return 1;
233 //      }
234 //}BST;
```

7.4 KD-Tree

```
1 // ===== KD-Tree =====
2 const int MAXD=2;
3
4 struct KDPoint{
5     LL x[MAXD];
6     void read(){
7         for(int i=0;i<MAXD;++i){
8             int v;scanf("%d",&v);
9             x[i]=v;
10        }
11    }
12 };
13 typedef pair<KDPoint,int> KDPTYPE;
14
15 struct ct{
16     LL mxD[MAXD], mnD[MAXD];
17     ct(){ mnD[0]=INF; }
18     ct(KDPoint P){ for(int i=0;i<MAXD;++i) mxD[i]=mnD[i]=P.x[i]; }
19     bool empty()const{ return mnD[0]==INF; }
20 };
21
22 ct merge(const ct& A, const ct& B){
23     if(A.empty()) return B;
24     if(B.empty()) return A;
25     ct C;
26     for(int i=0;i<MAXD;++i) {
27         C.mxD[i]=max(A.mxD[i], B.mxD[i]);
28         C.mnD[i]=min(A.mnD[i], B.mnD[i]);
29     }
```

```
30         return C;
31     }
32
33     struct KDTNode{
34         KDTNode *ch[2];
35         KDPTYPE v;
36         ct c;
37
38         KDTNode() {}
39         KDTNode(KDPTYPE v):v(v) {}
40     };
41
42     const int MAXN=100010;
43
44     struct cmp{
45         int D;
46         cmp(int D):D(D) {}
47         bool operator()(const KDPTYPE& A, const KDPTYPE& B) const{
48             return A.first.x[D]<B.first.x[D];
49         }
50     };
51
52     struct KDT{
53         KDTNode kdt[MAXN], *kdtR, *rt;
54
55         KDPTYPE *A;
56         KDPoint qP;
57
58         KDTNode* build(int d,int l,int r){
```

```

59         if(l>r) return 0;
60
61         int m = (l+r)>>1;
62         nth_element(A+l,A+m,A+r+1,cmp(d));
63
64         KDTreeNode *o=new(kdtR++)KDTreeNode(A[m]);
65         o->ch[0]=build((d+1)%MAXD, l, m-1); o->ch[1] = build((d+1)%MAXD, m+1, r);
66         o->c=merge(merge(o->ch[0] ? o->ch[0]->c : ct(), o->v.first), o->ch[1] ? o->ch[1]->c : ct());
67         return o;
68     }
69     void init(int n,KDPTYPE* A) {
70         this->A = A; kdtR = kdt; rt = build(0,1,n);
71     }
72
73     // insert
74     KDPTYPE iP;
75     void insert(int d, KDTreeNode*& o){
76         if(!o){ o=new(kdtR++)KDTreeNode(iP); }
77         else { if(cmp(d) (iP,o->v))insert(d^1,o->ch[0]);else insert(d^1,o->ch[1]); }
78         o->c=merge(merge(o->ch[0] ? o->ch[0]->c : ct(), o->v.first), o->ch[1] ? o->ch[1]->c : ct());
79     }
80     void insert(KDPTYPE P){
81         iP=P; insert(0,rt);
82     }
83
84     // query
85     LL getdis(KDTreeNode* o){ // getdis¹ǎ<□□Çğ□L □□□д□□□□□□□□Ž□̄•±±£¬³⁹□>©µ
86         LL res=0;
87         for(int d=0;d<MAXD;++d) {

```

```

88         if(qP.x[d]<o->c.mnD[d]) res+=sqr(o->c.mnD[d]-qP.x[d]);
89         if(o->c.mxD[d]<qP.x[d]) res+=sqr(qP.x[d]-o->c.mxD[d]);
90     }
91     return res;
92 }
93
94 vector<LL> res; vector<int> resi;
95 void query(int d, KDTNode* o){
96     LL d0=0; for(int i=0;i<MAXD;++i) d0+=sqr(o->v.first.x[i]-qP.x[i]);
97     if (dcmp(d0-res[0])<0 || (dcmp(d0-res[0])==0 && o->v.second<resi[0]))
98         res[1]=res[0], resi[1]=resi[0], res[0]=d0, resi[0]=o->v.second;
99     else if(dcmp(d0-res[1])<0 || (dcmp(d0-res[1])==0 && o->v.second<resi[1]))
100         res[1]=d0, resi[1]=o->v.second;
101
102     LL dl=o->ch[0]?getdis(o->ch[0]):INF;
103     LL dr=o->ch[1]?getdis(o->ch[1]):INF;
104     if(dl<dr){
105         if(dl<=res[1])query((d+1)%MAXN,o->ch[0]);
106         if(dr<=res[1])query((d+1)%MAXN,o->ch[1]);
107     }else{
108         if(dr<=res[1])query((d+1)%MAXN,o->ch[1]);
109         if(dl<=res[1])query((d+1)%MAXN,o->ch[0]);
110     }
111 }
112
113 vector<int> query(KDPoint P){
114     qP=P;
115     res.resize(2); res[0]=res[1]=INF;
116     resi.resize(2);

```

```

117         query(0,rt);
118         return resi;
119     }
120 }solver;

```

7.5 内存池

```

1 //please init() before use
2 template<class T>struct Pool{
3     T *a,**q; int pa,pq;
4     Pool(int MAXN){a=new T[MAXN];q=new T*[MAXN];}
5     // ~Pool(){delete[] a; delete[] q;}
6     void init(){pa=0;pq=0;}
7     T* NEW(){return new(pq ? q[--pq] : &a[pa++])T();}
8     void DELETE(T* x){q[pq++]=x; return;}
9 };

```

7.6 轻重链剖分

```

1 #include<cstdio>
2
3 const int maxn=10010;
4
5 //Heavy-Light Decomposition
6 //help:1.init() 2.addedge() 3.pre() 4.update()/query()
7 struct HLD{
8     int st[maxn],lk[maxn<<1],b[maxn<<1];
9     int tot;
10    void init(){

```

```
11         tot=1;memset(st,0,sizeof st);
12     }
13     void addedge(int u,int v){
14         lk[++tot]=st[u];b[tot]=v;st[u]=tot;
15     }
16     int fa[maxn],d[maxn],sz[maxn],son[maxn],top[maxn];
17     int seq[maxn],bg[maxn],ed[maxn];
18     int dfN;
19
20     int A[maxn];
21     SegSeq<ct,mt,maxn> seg;
22
23     void dfs1(int u){
24         sz[u]=1;son[u]=0;
25         for(int i=st[u];i;i=lk[i]){
26             int v=b[i];
27             if(v==fa[u])continue;
28             fa[v]=u; d[v]=d[u]+1;
29             dfs1(v);
30             sz[u]+=sz[v];
31             if(!son[u]||sz[son[u]]<=sz[v]) son[u]=v;
32         }
33     }
34     void dfs2(int u){
35         seq[bg[u]=++dfN]=u;
36         if(son[u]){ top[son[u]]=top[u]; dfs2(son[u]); }
37         for(int i=st[u];i;i=lk[i]){
38             int v=b[i];
39             if(v==fa[u]||v==son[u])continue;
```



```

40         top[v]=v; dfs2(v);
41     }
42     ed[u]=dfN;
43 }
44 void pre(int n,int* a){
45     int rt=1;
46     dfN=0; fa[rt]=0,d[rt]=1; dfs1(rt); top[rt]=rt; dfs2(rt);
47     for(int i=1;i<=n;++i) A[i]=a[seq[i]];
48     seg.init(n,A);
49 }
50 void update(int x,int y,int& v){
51     int p,q;
52     for(; (p=top[x])!=(q=top[y]);){
53         if(d[p]<d[q]) swap(x,y),swap(p,q);
54         seg.update(bg[p],bg[x],v),x=fa[p];
55     }
56     if(d[x]<d[y]) swap(x,y);
57     seg.update(bg[y],bg[x],v);
58 }
59 /*
60 Pay attantion to the order of merging. The following is according to x->y. The rev() function is required.
61 You can also imitate update() to speed up it.
62 */
63 void query(int x,int y,int& v){
64     ct cx,cy;
65     int p,q;
66     for(; (p=top[x])!=(q=top[y]);){
67         if(d[p]<d[q]) swap(x,y),swap(p,q),swap(cx,cy);
68         cx=cx+seg.query(bg[p],bg[x]).rev();

```

```

69         x=fa[p];
70     }
71     if(d[x]<d[y]) swap(x,y), swap(cx,cy);
72     cx=cx+seg.query(bg[y],bg[x]).rev();
73     v=(cx+cy.inv()).s;
74 }
75 }hld;

```

7.7 LCT

```

1  const int maxn=100010;
2
3  //LCT
4  //splay is needed
5  struct LCT{
6      // in LCT, unlink() is a dangerous operation.
7      int n;
8      node* T[maxn];
9      int d[maxn]; // d[u]: the depth of u (used for location)
10
11     void init(int n,int* A){
12         this->n=n;for(int i=1;i<=n;++i)T[i]=build(A,i,i);
13     }
14     void access(node* o){
15         for(node *p=o,*q=0;p;q=p,p=p->p){splayLCT(p);p->link(1,q)->upd();}
16         splayLCT(o);
17     }
18     node* findRoot(node* o){access(o);return splayk(o,1);}
19     void link(node* o,node* p){access(o);access(p);o->p=p;}

```

```
20 void cut(node* o){access(o);o->unlink(0);o->upd();}
21 void evert(node *o){access(o);o->reverse();}
22
23 // you should also modify "Splay" when you choose to "modify".
24 void modify_route(node* o,const mt& v){access(o);o->modify(v);}
25 void modify(node* o,const mt& v){splayLCT(o)->modify(v);}
26
27 ct query(node* x,node* y){
28     ct res;
29     access(y);
30     for(node *p=x,*q=0;p;q=p,p=p->p){
31         splayLCT(p);
32         if(!p->p){
33             if(p->ch[1])
34                 if(q)res=p->ch[1]->c+q->c.rev();
35                 else res=p->ch[1]->c;
36             else res=q->c.rev();
37         }
38         p->link(1,q);
39         p->upd();
40     }
41     splayLCT(x);
42     return res;
43 }
44 ct query(node *x){access(x);return x->c;}
45 void link(int o,int p){link(T[o],T[p]);}
46 void cut(int o){cut(T[o]);}
47 void evert(int o){evert(T[o]);}
48 int query(int x,int y){return query(T[x],T[y]).mx;}
```

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