XCPC 模板

First Come I!

2022 年 10 月

目录

1 代码结构性

1.1 io

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

```
typedef long long ll;
char In[1 << 20], *ss = In, *tt = In;
#define getchar() (ss == tt && (tt = (ss = In) + fread(In, 1, 1 << 20, stdin), ss ==
     tt) ? EOF : *ss++)
ll read() {
   ll x = 0, f = 1; char ch = getchar();
   for(; ch < '0' || ch > '9'; ch = getchar()) if(ch == '-') f = -1;
   for(; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + int(ch - '0');
   return x * f;
}
```

1.2 高精

```
const int MAXL = 2e4 + 5, N = 8, NN = 100000000;
char buff[MAXL];
struct Bigint {
   int n; ll A[MAXL / N + 5];
   Bigint(ll x = 0) {
      n = 0; memset(A, 0x00, sizeof A);
      if(x == 0) n = 1;
      else {
         while(x) {
            A[n++] = x % NN;
            x /= NN;
         }
      }
   }
   void Read() {
      n = 0; memset(A, 0x00, sizeof A);
      scanf("%s", buff+1); int len = strlen(buff+1);
      for(int i = len; i >= 1; i -= N) {
         int j = max(1, i - N + 1);
         for(int k = j; k <= i; k++) A[n] = A[n] * 10 + int(buff[k] - '0');
         n++;
      }
   }
   void Write() {
      printf("%lld", A[n-1]);
      for(int i = n-2; i >= 0; i--) printf("%08lld", A[i]);
   }
   void clrpre0() {while(n > 1 && A[n-1] == 0) n--;}
   ll& operator [] (const int& k) {return A[k];}
   const ll& operator [] (const int& k)const {return A[k];}
   Bigint operator + (const Bigint& B)const {
      Bigint C; C.n = max(n, B.n);
```

```
    for(int i = 0; i < C.n; i++) {
        C[i] += A[i] + B[i];
        if(C[i] >= NN) C[i+1]++, C[i] -= NN;
    }
    if(C[C.n]) C.n++;
    C.clrpre0();
    return C;
}
bool operator < (const Bigint& B)const {
    if(n != B.n) return n < B.n;
    for(int i = n-1; i >= 0; i--)
        if(A[i] != B[i]) return A[i] < B[i];
    return 0;
}
Bigint operator - (const Bigint& B)const {//be sure that *this >= B
    Bigint C; C.n = n;
    for(int i = 0; i < C.n; i++) {
        C[i] += A[i] - B[i];
        if(C[i] < 0) C[i+1]--, C[i] += NN;
    }
    C.clrpre0();
    return C;
}
Bigint operator * (const Bigint& B)const {
    Bigint C; C.n = n + B.n - 1;
    for(int i = 0; i < n; i++)
        for(int j = 0; j < B.n; j++)
            C[i+j] += A[i] * B[j], C[i+j+1] += C[i+j] / NN, C[i+j] %= NN;
    if(C[C.n]) C.n++;
    C.clrpre0();
    return C;
}
Bigint operator * (const ll& k) const {
    Bigint C; C.n = n;
    for(int i = 0; i < C.n; i++)
        C[i] += A[i] * k, C[i+1] += C[i] / NN, C[i] %= NN;
    if(C[C.n]) C.n++;
    C.clrpre0();
    return C;
}
pair<Bigint, Bigint> div(const Bigint& B)const {
    Bigint C, D;
    C.n = n; D.n = 0;
    for(int i = C.n-1; i >= 0; i--) {
        D.n++; for(int j = D.n-1; j >= 1; j--) D[j] = D[j-1];
        D[0] = A[i];
```

```
            ll l = 0, r = NN-1;
            while(l <= r) {
                ll m = (l + r) >> 1;
                if(!(D < B * m)) l = m + 1, C[i] = m;
                else r = m - 1;
            }
            D = D - B * C[i];
        }
        C.clrpre0(), D.clrpre0();
        return make_pair(C, D);
    }
    Bigint operator / (const Bigint& B)const {
        return div(B).first;
    }
    Bigint operator % (const Bigint& B)const {
        return div(B).second;
    }
};
```

2 数学

2.1 二元一次不定方程

```
ll gcd(ll a, ll b) {
    if(!b) return a;
    return gcd(b, a % b);
}
void exgcd(ll a, ll b, ll& x, ll& y) {
    if(!b) x = 1, y = 0;
    else exgcd(b, a % b, y, x), y -= (a / b) * x;
}

int main() {
    int T = read();
    while(T--) {
        ll a = read(), b = read(), c = read(), x, y, g, xmin, ymin, xmax, ymax;
        g = gcd(a, b);
        if(c % g != 0) {
            printf("-1\n");
            continue;
        }
        a /= g, b /= g, c /= g;
        exgcd(a, b, x, y);
        x *= c; y *= c;
        xmin = (x % b + b) % b; if(xmin == 0) xmin += b;
        ymax = (c - a * xmin) / b;
        ymin = (y % a + a) % a; if(ymin == 0) ymin += a;
        xmax = (c - b * ymin) / a;
        if(ymax > 0) // has a int solution
        {
            printf("%lld %lld %lld %lld %lld\n", (ymax - ymin) / a + 1, xmin, ymin, xmax
                 , ymax);
        } else {
            printf("%lld %lld\n", xmin, ymin);
        }
    }
    return 0;
}
```

2.2 中国剩余定理

```
const int MAXN = 15;
int n;
ll b[MAXN], m[MAXN], prod_m, ans;
ll mul(ll a, ll b, ll M) {
  ll c = (long double)a / M * b;
  ll d = (unsigned long long)a * b- (unsigned long long)c * M;
  return d;
}
void exgcd(ll a, ll b, ll &x, ll &y) {
  if(!b) x = 1, y = 0;
  else exgcd(b, a % b, y, x), y -= a / b * x;
}
ll inv(ll a, ll M) {
  ll x, y;
  exgcd(a, M, x, y);
  return (x % M + M) % M;
}
int main() {
  n = read();
  prod_m = 1;
  for(int i = 1; i <= n; i++) {
    m[i] = read(), b[i] = read();
    prod_m *= m[i];
  }
  ans = 0;
  for(int i = 1; i <= n; i++) {
    ll M = prod_m / m[i];
    ans = (ans + mul(mul(b[i], inv(M, m[i]), m[i]), M, prod_m)) % prod_m;
  }
  printf("%lld\n", ans);
  return 0;
}
```

2.3 扩展中国剩余定理

```
const int MAXN = 1e5 + 5;
ll mul(ll a, ll b, ll P) {
  ll c = (ldb)a / P * b;
  ll res = (ull)a * b - (ull)c * P;
  return res;
}
ll gcd(ll a, ll b) {
  return !b ? a : gcd(b, a % b);
}
ll lcm(ll a, ll b) {
  return a / gcd(a, b) * b;
}
void exgcd(ll a, ll b, ll &x, ll &y) {
  if(!b) x = 1, y = 0;
  else exgcd(b, a % b, y, x), y -= a / b * x;
}
int n;
ll b[MAXN], m[MAXN];
/**
 * x=a mod n
 * x=b mod m
 * x=pn+a=qm+b
 * pn-qm=b-a
 * x=p_0n+a mod lcm(n,m)
**/
void solve(ll a, ll n, ll b, ll m, ll &retB, ll &retM) {
  ll A = n, B = m, C = (b - a) % B + B, G = gcd(A, B);
  if(C % G) exit(1);
  ll x, y, st = B / G;
  exgcd(A, B, x, y);
  x = mul(x % st + st, C / G, st);
  retM = lcm(n, m);
  retB = (a + mul(x, n, retM)) % retM;
}
void exCRT(ll &B, ll &M) {
  B = b[1], M = m[1];
  for(int i = 2; i <= n; i++)
    solve(B, M, b[i], m[i], B, M);
}
int main() {
  n = read();
  for(int i = 1; i <= n; i++) m[i] = read(), b[i] = read();
  ll B, M;
  exCRT(B, M);
  printf("%lld\n", B);
  return 0;
}
```

2.4 线性方程组（高斯消元）

```
constexpr int MAXN = 55;
constexpr db eps = 1e-6;
int n;
db a[MAXN][MAXN];
int main() {
  n = read();
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n+1; j++)
      a[i][j] = read();
  int r = 1;
  for(int c = 1; c <= n; c++) {
    int num = r;
    for(int i = r+1; i <= n; i++)
      if(abs(a[i][c]) > abs(a[num][c]))
        num = i;
    if(num != r)
      for(int i = c; i <= n+1; i++)
        swap(a[r][i], a[num][i]);
    if(abs(a[r][c]) < eps)
      continue;
    for(int i = r+1; i <= n; i++) {
      db k = a[i][c] / a[r][c];
      for(int j = c; j <= n+1; j++)
        a[i][j] -= k * a[r][j];
    }
    r++;
  }
  if(r <= n) { //[r,n] exist; then all are zero.
    for(int i = r; i <= n; i++)
      if(abs(a[i][n+1]) > eps) {
        printf("-1\n");
        return 0;
      }
    printf("0\n");
    return 0;
  }
  for(int i = n; i >= 1; i--) {
    for(int j = i+1; j <= n; j++)
      a[i][n+1] -= a[j][n+1] * a[i][j];
    a[i][n+1] /= a[i][i];
  }
  for(int i = 1; i <= n; i++)
    printf("x%d=%.2lf\n", i, (abs(a[i][n+1]) < eps ? eps : a[i][n+1]));
  return 0;
}
```

2.5 Lucas 定理

```
const int MAXN = 2e5 + 5;
int n, m, p;
int pls(int a, int b) {return a + b < p ? a + b : a + b - p;}
int mns(int a, int b) {return a < b ? a + p - b : a - b;}
int mul(int a, int b) {return 1ll * a * b % p;}
int qpow(int a, int n) {
   int ret = 1;
   for(; n; n >>= 1, a = mul(a, a))
      if(n & 1) ret = mul(ret, a);
   return ret;
}
int fac[MAXN], inv[MAXN];
int C(int n, int m) {//n,m < p
   if(n < 0 || m < 0 || m > n) return 0;
   return mul(mul(fac[n], inv[m]), inv[n-m]);
}
int Lucas(int n, int m) {
   return !m ? 1 : mul(C(n % p, m % p), Lucas(n / p, m / p));
}
void work() {
   n = read(), m = read(), p = read();
   memset(fac, 0x00, sizeof fac);
   memset(inv, 0x00, sizeof inv);
   fac[0] = 1;
   for(int i = 1; i < p; i++) fac[i] = mul(fac[i-1], i);
   inv[p-1] = qpow(fac[p-1], p-2);
   for(int i = p-2; i >= 0; i--) inv[i] = mul(inv[i+1], i+1);
   printf("%d\n", Lucas(n+m, n));

}
```

```
int main() {
    int T = read();
    while(T--) {
        work();
    }
    return 0;
}
```

2.6 扩展 Lucas

```
void exgcd(ll a, ll b, ll& x, ll& y) {
    if(!b) x = 1, y = 0;
    else exgcd(b, a % b, y, x), y -= a / b * x;
}
ll inv(ll a, ll m) {
    ll x, y;
    exgcd(a, m, x, y);
    return (x % m + m) % m;
}
ll qpow(ll a, ll n, ll p) {
    ll ret = 1;
    for(; n; n >>= 1, a = a * a % p)
        if(n & 1) ret = ret * a % p;
    return ret;
}
ll fac(ll n, ll p, ll pk) {
    if(!n) return 1;
    ll ret = 1;
    for(int i = 1; i <= pk; i++)
        if(i % p) ret = ret * i % pk;
    ret = qpow(ret, n / pk, pk);
    for(int i = 1; i <= n % pk; i++)
        if(i % p) ret = ret * i % pk;
    return ret * fac(n / p, p, pk) % pk;
}
ll C(ll n, ll m, ll p, ll pk) {
    ll x = 0, y = 0, z = 0;
    for(ll i = p; i <= n; i *= p)
        x += n / i;
    for(ll i = p; i <= m; i *= p)
        y += m / i;
    for(ll i = p; i <= n-m; i *= p)
        z += (n-m) / i;
    return fac(n, p, pk) * inv(fac(m, p, pk), pk) % pk * inv(fac(n-m, p, pk), pk ) %
         pk * qpow(p, x - y - z, pk) % pk;
}
ll a[55], b[55], tot;
ll CRT() {
   ll M = 1, ans = 0;
   for(int i = 1; i <= tot; i++) M *= b[i];
   for(int i = 1; i <= tot; i++) {
      ll Mi = M / b[i];
      ans = (ans + a[i] * inv(Mi, b[i]) * Mi) % M;
   }
   return ans % M;
}
ll exlucas(ll n, ll m, ll P) {
   for(int i = 2; 1ll * i * i <= P; i++)
      if(P % i == 0) {
         ll pk = 1;
         while(P % i == 0) pk *= i, P /= i;
         ++tot;
         a[tot] = C(n, m, i, pk);
         b[tot] = pk;
      }
   if(P > 1) {
      ++tot;
      a[tot] = C(n, m, P, P);
      b[tot] = P;
   }
   return CRT();
}
int main() {
   ll n = read(), m = read(), p = read();
   printf("%lld\n", exlucas(n, m, p));
   return 0;
}
```

2.7 BSGS

```
int P, a, b, B;
int qpow(int a, int n=P-2) {int ret = 1; for(; n; n >>= 1, a = 1ll * a * a % P) if(n &
      1) ret = 1ll * ret * a % P; return ret;}
map<int, int> mp;
int main() {
   P = read(), a = read(), b = read();
   while(1ll * B * B <= P) B++;
   int t = qpow(a), pro = 1;
   for(int i = 0; i < B; i++) {
      if(!mp.count(1ll * pro * b % P)) mp[1ll * pro * b % P] = i;
```

```
        pro = 1ll * pro * t % P;
    }
    t = qpow(a, B); pro = 1;
    for(int i = 0; i < B; i++) {
        if(mp.count(pro)) {
            printf("%d\n", i * B + mp[pro]);
            return 0;
        }
        pro = 1ll * pro * t % P;
    }
    printf("no solution\n");
    return 0;
}
```

2.8 扩展 BSGS

```
int MOD(int a, int p) {return a >= p ? a - p : a;}
int gcd(int a, int b) {return !b ? a : gcd(b, a % b);}
void exgcd(int a, int b, int& x, int& y) {
    if(!b) x = 1, y = 0;
    else exgcd(b, a % b, y, x), y -= a / b * x;
}
int inv(int a, int p) {
    int x, y; exgcd(a, p, x, y); return MOD(x % p + p, p);
}
int BSGS(int a, int b, int p) {
    a %= p; b %= p;
    if(!a) return !b ? 1 : -1;
    map<int, int> mp;
    int B = 0; while(1ll * B * B <= p) B++;
    int t = inv(a, p), pro = 1;
    for(int i = 0; i < B; i++) {
        if(!mp.count(1ll * pro * b % p)) mp[1ll * pro * b % p] = i;
        pro = 1ll * pro * t % p;
    }
    t = inv(pro, p); pro = 1;
    for(int i = 0; i < B; i++) {
        if(mp.count(pro)) return i * B + mp[pro];
        pro = 1ll * pro * t % p;
    }
    return -1;
}
int ExBSGS(int a, int b, int p) {
    a %= p; b %= p;
    if(b == 1 || p == 1) return 0;
    if(!a) return !b ? 1 : -1;
    int cnt = 0, pro = 1;
    while(1) {
        int d = gcd(a, p);
        if(d == 1) break;
        if(b % d) return -1;
        b /= d; p /= d; pro = 1ll * pro * (a / d) % p; ++cnt;
        if(pro == b) return cnt;
    }
    int ret = BSGS(a % p, 1ll * b * inv(pro, p) % p, p);
    if(ret == -1) return -1;
    return ret + cnt;
}
int main() {
    int a, p, b;
    while(1) {
        a = read(), p = read(), b = read();
        if(!a) return 0;
        int ans = ExBSGS(a, b, p);
        if(ans == -1) printf("No Solution\n");
        else printf("%d\n", ans);
    }
    return 0;
}
```

2.9 线性基

```
int n;
namespace Linearbasis {
    const int B = 51;
    ll t[B];
    void ins(ll x) {
        for(int i = B-1; i >= 0; i--)
            if((x >> i) & 1) {
                if(!t[i]) {
                    t[i] = x;
                    break;
                } else x ^= t[i];
            }
    }
    ll qry() {
        ll ans = 0;
        for(int i = B-1; i >= 0; i--)
            ans = max(ans, ans ^ t[i]);
        return ans;
```

```
   }
}
using namespace Linearbasis;
int main() {
   n = read();
   for(int i = 1; i <= n; i++) {
      ll k = read();
      ins(k);
   }
   printf("%lld\n", qry());
   return 0;
}
```

2.10 线性筛

```
#include<cstdio>
const int MAXN = 1e8+5;
int n, q;
bool npr[MAXN];
int pr[MAXN], _pr;
void get_prime(int n) {
  npr[1] = 1; _pr = 0;
  for(int i = 2; i <= n; i++) {
    if(!npr[i]) {
      pr[++_pr] = i;
    }
    for(int j = 1, _lim = n / i; j <= _pr && pr[j] <= _lim; j++) {
      int k = i * pr[j];
      npr[k] = 1;
      if(i % pr[j] == 0) break;
    }
  }
}
int main() {
  scanf("%d%d", &n, &q);
  get_prime(n);
  for(int i = 1; i <= q; i++) {
    int t; scanf("%d", &t);
    printf("%d\n", pr[t]);
  }
  return 0;
}
```

2.11 杜教筛

```
const ll MAXN = 1e10 + 5, PN = 1.7e6;
int ip[PN+5], tot;
int mu[PN+5], phi[PN+5], pr[PN+5];
ll smu[PN + 5], sphi[PN + 5];
map<int, ll> fmu, fphi;
void init(ll n) {
   ip[1] = 1; mu[1] = phi[1] = 1;
   for(int i = 2; i <= n; i++) {
      if(!ip[i]) {
         pr[++tot] = i;
         mu[i] = -1;
         phi[i] = i-1;
      }
      for(int j = 1; j <= tot && 1ll * i * pr[j] <= n; j++) {
         int k = pr[j] * i;
         ip[k] = 1;
         if(i % pr[j]) {
            mu[k] = -mu[i];
            phi[k] = phi[i] * (pr[j] - 1);
         } else {
            mu[k] = 0;
            phi[k] = phi[i] * pr[j];
            break;
         }
      }
   }
   for(int i = 1; i <= n; i++) sphi[i] = sphi[i-1] + phi[i], smu[i] = smu[i-1] + mu[i
       ];
}
ll calcphi(ll x) {
   if(x <= PN) return sphi[x];
   if(fphi.count(x)) return fphi[x];
   ll ans = 1ull * x * (1ull * x + 1) / 2;
   for(ll i = 2, j; i <= x; i = j+1) {
      j = x / (x/i);
      ans -= 1ll * (j - i + 1) * calcphi(x/i);
   }
   return fphi[x] = ans;
}
ll calcmu(ll x) {
   if(x <= PN) return smu[x];
   if(fmu.count(x)) return fmu[x];
   ll ans = 1;
   for(ll i = 2, j; i <= x; i = j+1) {
        j = x / (x/i);
        ans -= 1ll * (j - i + 1) * calcmu(x/i);
    }
    return fmu[x] = ans;
}
int main() {
    init(PN);
    int T = read();
    for(int i = 1; i <= T; i++) {
        int n = read();
        printf("%lld %lld\n", calcphi(n), calcmu(n));
    }
    return 0;
}
```

2.12 Min25 筛

```
const int P = 1e9 + 7, MAXK = 5e5+5, inv6 = 166666668, inv2 = 500000004;
const ll MAXN = 1e10+5;
ll n, sn, w[MAXK], pr[MAXK], sp1[MAXK], sp2[MAXK], g1[MAXK], g2[MAXK];
int ip[MAXK], tot, num, id1[MAXK], id2[MAXK];
void pre_gao(int n) {
    ip[1] = 1;
    for(int i = 2; i <= n; i++) {
        if(!ip[i]) {
            pr[++tot] = i;
            sp1[tot] = (sp1[tot-1] + 1ll * i * i) % P;
            sp2[tot] = (sp2[tot-1] + i) % P;
        }
        for(int j = 1; j <= tot && 1ll * i * pr[j] <= n; j++) {
            ip[i * pr[j]] = 1;
            if(i % pr[j] == 0) break;
        }
    }
}
ll S(ll x, int k) {
    if(pr[k] >= x) return 0;
    int t = x <= sn ? id1[x] : id2[n / x];
    ll ret = ((g1[t] - sp1[k] + P) - (g2[t] - sp2[k] + P) % P) % P;
    for(int j = k+1; j <= tot && pr[j] * pr[j] <= x; j++) {
        ll p = pr[j];
        for(int e = 1; p <= x; e++, p *= pr[j]) {
            ll tmp = p % P;
            ret = (ret + tmp * (tmp - 1) % P * (S(x / p, j) + int(e != 1))) % P;
        }
    }
    return ret;
}
int main() {
    n = read(); sn = sqrt(n) + 1;
    pre_gao(sn);
    for(ll i = 1, j; i <= n; i = j + 1) {
        ll t = n / i; j = n / t;
        w[++num] = t;
        ll tmp = t % P;
        g1[num] = (tmp * (tmp + 1) % P * (tmp + tmp + 1) % P * inv6 + P - 1) % P;
        g2[num] = (tmp * (tmp + 1) % P * inv2 + P - 1) % P;
        //the initial value of g(-,0)
        if(t <= sn) id1[t] = num;
        else id2[n / t] = num;
    }

    //calc from g(-,j-1) to g(-,j)
    for(int j = 1; j <= tot; j++)
        for(int i = 1; i <= num && pr[j] * pr[j] <= w[i]; i++) {
            ll nxt = w[i] / pr[j]; int t = nxt <= sn ? id1[nxt] : id2[n / nxt];
            g1[i] = (g1[i] - pr[j] * pr[j] % P * (g1[t] - sp1[j-1]) % P + P) % P;
            g2[i] = (g2[i] - pr[j] * (g2[t] - sp2[j-1]) % P + P) % P;
        }
    printf("%lld\n", (S(n, 0) + 1));
    return 0;
}
```

2.13 二次剩余

```
ll n;
ll P, a, I2;
struct F2 {ll x, y;};
F2 operator * (const F2& A, const F2& B) {return (F2){A.x * B.x % P + I2 * (A.y * B.y
     % P) % P, (A.x * B.y + A.y * B.x) % P};}
ll qpow(ll a, int n) {ll ret = 1; for(; n; n >>= 1, a = a * a % P) if(n & 1) ret = ret
      * a % P; return ret;}
F2 qpow(F2 a, int n) {F2 ret = (F2){1, 0}; for(; n; n >>= 1, a = a * a) if(n & 1) ret
     = ret * a; return ret;}
ll judge(ll a) {return qpow(a, (P-1) / 2);}
void work() {
    n = read(); P = read();
    ll k = judge(n);
    if(k == 0) {
        printf("0\n");
        return ;
    } else if(k == P-1) {
        printf("Hola!\n");
        return ;
    }
    while(1) {
        a = rand() % P;
        I2 = (a * a % P + P - n) % P;
        if(judge(I2) == P-1) break;
    }
    ll ans1 = qpow((F2){a, 1}, (P+1) / 2).x % P;
    ll ans2 = P-ans1;
    if(ans1 > ans2) swap(ans1, ans2);
    printf("%lld %lld\n", ans1, ans2);
}


int main() {
    int T = read();
    while(T--) work();
    return 0;
}
```

2.14 多项式

```
const int MAXN = (1 << 18) + 5, P = 998244353, BAS = 1 << 18;
int pls(int a, int b) {return a + b < P ? a + b : a + b - P;}
int mns(int a, int b) {return a < b ? a + P - b : a - b;}
int mul(int a, int b) {return 1ll * a * b % P;}
int qpow(int a, int n=P-2) {int ret = 1; for(; n; n >>= 1, a = mul(a, a)) if(n & 1)
     ret = mul(ret, a); return ret;}
namespace Poly {
    typedef vector<int> poly;
    int _g[MAXN], tr[MAXN], tf, inv[MAXN], fac[MAXN], ifac[MAXN];
    poly Rsz(poly f, int n) {
        f.resize(n); return f;
    }
    void init() {
        _g[0] = 1; _g[1] = qpow(3, (P-1) / BAS);
        for(int i = 2; i < BAS; i++) _g[i] = mul(_g[i-1], _g[1]);
        inv[1] = 1; for(int i = 2; i < MAXN; i++) inv[i] = mul(P - P / i, inv[P % i]);
        fac[0] = 1; for(int i = 1; i < MAXN; i++) fac[i] = mul(fac[i-1], i);
        ifac[0] = 1; for(int i = 1; i < MAXN; i++) ifac[i] = mul(ifac[i-1], inv[i]);
    }
    int glim(int n) {
```

```
    int lim = 1; for(; lim < n; lim <<= 1);
    return lim;
}
void tpre(int lim) {
    if(lim == tf) return;
    tf = lim; for(int i = 0; i < lim; i++) tr[i] = (tr[i >> 1] >> 1) | ((i & 1) ? (
         lim >> 1) : 0);
}
void DFT(poly& f, int lim) {
    tpre(lim); if((int)f.size() < lim) f.resize(lim);
    for(int i = 0; i < lim; i++) if(i < tr[i]) swap(f[i], f[tr[i]]);
    for(int l = 2, k = 1; l <= lim; l <<= 1, k <<= 1)
        for(int i = 0; i < lim; i += l)
            for(int j = i; j < i+k; j++) {
                int tt = mul(f[j+k], _g[BAS / l * (j-i)]);
                f[j+k] = mns(f[j], tt);
                f[j] = pls(f[j], tt);
            }
}
void IDFT(poly& f, int lim) {
    DFT(f, lim); reverse(f.begin()+1, f.begin()+lim);
    for(int i = 0; i < lim; i++) f[i] = mul(f[i], inv[lim]);
}
poly Mul(poly f, poly g) {
    int n = f.size() + g.size() - 1, lim = glim(n);
    DFT(f, lim); DFT(g, lim);
    for(int i = 0; i < lim; i++) f[i] = mul(f[i], g[i]);
    IDFT(f, lim); f.resize(n); return f;
}
poly Pls(poly f, const poly& g) {
    f.resize(max(f.size(), g.size()));
    for(int i = 0; i < (int)g.size(); i++) f[i] = pls(f[i], g[i]);
    return f;
}
poly Mns(poly f, const poly& g) {
    f.resize(max(f.size(), g.size()));
    for(int i = 0; i < (int)g.size(); i++) f[i] = mns(f[i], g[i]);
    return f;
}
poly Mul(poly f, int k) {
    for(int i = 0; i < (int)f.size(); i++) f[i] = mul(f[i], k);
    return f;
}
poly Deriv(poly f) {
    for(int i = 1; i < (int)f.size(); i++) f[i-1] = mul(f[i], i);
    f.pop_back(); return f;
```

```
}
poly Integ(poly f) {
    f.push_back(0);
    for(int i = f.size()-1; i >= 1; i--) f[i] = mul(f[i-1], inv[i]);
    f[0] = 0; return f;
}
poly Inv(poly f) {
    if(f.size() == 1) return poly(1, qpow(f[0]));
    int n = f.size(), lim = glim(n+n);
    poly g = f; g.resize((n+1)/2); g = Inv(g);
    DFT(f, lim); DFT(g, lim);
    for(int i = 0; i < lim; i++) f[i] = mul(g[i], mns(2, mul(f[i], g[i])));
    IDFT(f, lim); f.resize(n); return f;
}
poly Ln(const poly& f) {
    poly ans = Integ(Mul(Inv(f), Deriv(f)));
    ans.resize(f.size());
    return ans;
}
poly Exp(poly f) {
    if(f.size() == 1) return poly(1, 1);
    int n = f.size(), lim = glim(n+n);
    poly g = f; g.resize((n+1)/2); g = Exp(g); g.resize(n);//careful
    f = Mns(f, Ln(g)); f[0]++;
    DFT(f, lim); DFT(g, lim);
    for(int i = 0; i < lim; i++) f[i] = mul(f[i], g[i]);
    IDFT(f, lim); f.resize(n);
    return f;
}
poly Rev(poly f) {
    reverse(f.begin(), f.end());
    return f;
}
poly Div(const poly& f, const poly& g) {
    if(f.size() < g.size()) return poly();
    int n = f.size() - g.size() + 1;
    poly ans = Mul(Rev(f), Inv(Rsz(Rev(g), n)));//careful the Rsz
    ans.resize(n);
    return Rev(ans);
}
poly Mod(const poly& f, const poly& g) {
    poly ans = Mns(f, Mul(Div(f, g), g));
    ans.resize(g.size()-1);
    return ans;
}
namespace Fastcalc {
```

```
#define ls p << 1
#define rs p << 1 | 1
poly h[MAXN << 2];
poly TMul(poly f, poly g) {
   int n = f.size(), m = g.size(), lim = glim(n);
   g = Rev(g);
   DFT(f, lim); DFT(g, lim);
   for(int i = 0; i < lim; i++) f[i] = mul(f[i], g[i]);
   IDFT(f, lim);
   poly T(n-m+1);
   for(int i = 0; i < n-m+1; i++) T[i] = f[i+m-1];
   return T;
}
void build(int p, int l, int r, const poly& a) {
   if(l == r) {
      h[p].resize(2); h[p][0] = 1; h[p][1] = mns(0, a[l]);
      return;
   }
   int m = (l + r) >> 1;
   build(ls, l, m, a); build(rs, m+1, r, a);
   h[p] = Mul(h[ls], h[rs]);
}
void calc1(int p, int l, int r, poly& ans, const poly& now) {
   if(l == r) {
      ans[l] = now[0];
      return ;
   }
   int m = (l + r) >> 1;
   calc1(ls, l, m, ans, TMul(now, h[rs]));
   calc1(rs, m+1, r, ans, TMul(now, h[ls]));
}
poly Eva(poly F, poly a) {
   int n = F.size(), m = a.size();
   n = max(n, m); F.resize(n); a.resize(n);
   build(1, 0, n-1, a);
   F.resize(2 * n + 1);
   calc1(1, 0, n-1, a, TMul(F, Inv(h[1])));
   a.resize(m);
   return a;
}
poly calc2(int p, int l, int r, const poly& Y) {
   if(l == r) return poly(1, Y[l]);
   int m = (l + r) >> 1;
   return Pls(Mul(calc2(ls, l, m, Y), h[rs]), Mul(calc2(rs, m+1, r, Y), h[ls]))
       ;
}
```

```
        poly Ins(poly X, poly Y) {
            int n = X.size(); if(!n) return poly();
            build(1, 0, n-1, X);
            poly F = Deriv(Rev(h[1]));
            F.resize(2 * n + 1);
            calc1(1, 0, n-1, X, TMul(F, Inv(h[1])));
            for(int i = 0; i < n; i++) Y[i] = mul(Y[i], qpow(X[i]));
            return Rev(calc2(1, 0, n-1, Y));
        }
        #undef ls
        #undef rs
    }
}
using namespace Poly;
```

2.15 FWT

```
namespace Poly {
    typedef vector<mint> poly;
    void OR(poly& f, int lim, int ty) {
        for(int l = 2, k = 1; l <= lim; l <<= 1, k <<= 1)
            for(int i = 0; i < lim; i += l)
                for(int j = i; j < i + k; j++) {
                    if(ty) f[j+k] += f[j];
                    else f[j+k] -= f[j];
                }
    }
    void AND(poly& f, int lim, int ty) {
        for(int l = 2, k = 1; l <= lim; l <<= 1, k <<= 1)
            for(int i = 0; i < lim; i += l)
                for(int j = i; j < i + k; j++) {
                    if(ty) f[j] += f[j+k];
                    else f[j] -= f[j+k];
                }
    }
    void XOR(poly& f, int lim, int ty) {
        for(int l = 2, k = 1; l <= lim; l <<= 1, k <<= 1)
            for(int i = 0; i < lim; i += l)
                for(int j = i; j < i + k; j++) {
                    mint t = f[j+k];
                    f[j+k] = f[j] - t;
                    f[j] = f[j] + t;
                    if(!ty) f[j] *= inv2, f[j+k] *= inv2;
                }
    }
    poly MulOR(poly f, poly g) {
        int n = f.size();
        OR(f, n, 1); OR(g, n, 1);
        for(int i = 0; i < n; i++) f[i] = f[i] * g[i];
        OR(f, n, 0);
        return f;
    }
    poly MulAND(poly f, poly g) {
        int n = f.size();
        AND(f, n, 1); AND(g, n, 1);
        for(int i = 0; i < n; i++) f[i] = f[i] * g[i];
        AND(f, n, 0);
        return f;
    }
    poly MulXOR(poly f, poly g) {
        int n = f.size();
        XOR(f, n, 1); XOR(g, n, 1);
        for(int i = 0; i < n; i++) f[i] = f[i] * g[i];
        XOR(f, n, 0);
        return f;
    }
} using namespace Poly;
```

2.16 积分

```
db a, b, c, d, l, r;
db f(db x) {return (c * x + d) / (a * x + b);}
db simpson(db l, db r) {
    db m = (l + r) / 2;
    return (f(l) + f(r) + 4 * f(m)) * (r-l) / 6;
}
db asr(db l, db r, db eps, db ans) {
    db m = (l + r) / 2;
    db L = simpson(l, m), R = simpson(m, r);
    if(fabs(L + R - ans) <= 15*eps) return L + R + (L+R-ans)/15;
    return asr(l, m, eps/2, L) + asr(m, r, eps/2, R);
}
int main() {
    scanf("%lf%lf%lf%lf%lf%lf", &a, &b, &c, &d, &l, &r);
    printf("%.6lf\n", asr(l, r, 1e-6, simpson(l, r)));
    return 0;
}
```

3 图论

3.1 最短路

3.1.1 Dijkstra+heap

```
typedef pair<int, int> pr;
const int MAXN = 1e5 + 5;
const int MAXM = 2e5 + 5;
const int INF = 0x3f3f3f3f;

int n, m, s;
struct Edge {
  int v, w, nxt;
}e[MAXM];
int head[MAXN], cnt;
void addedge(int u, int v, int w) {
  e[++cnt] = (Edge){v, w, head[u]}; head[u] = cnt;
}

priority_queue<pr, vector<pr>, greater<pr> > pq;
int dist[MAXN], vis[MAXN];

int main() {
  n = read(); m = read(); s = read();
  for(int i = 1; i <= m; i++) {
    int u = read(), v = read(), w = read();
    addedge(u, v, w);
  }
  for(int i = 1; i <= n; i++) dist[i] = INF, vis[i] = 0;
  dist[s] = 0; pq.push(pr(0, s));
  while(pq.size()) {
    int u = pq.top().second; pq.pop();
    if(vis[u]) continue;
    vis[u] = 1;
    for(int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v, w = e[i].w;
      if(vis[v]) continue;
      if(dist[v] > dist[u] + w) {
        dist[v] = dist[u] + w;
        pq.push(pr(dist[v], v));
      }
    }
  }
  for(int i = 1; i <= n; i++) printf("%d ", dist[i]);
  puts("");
  return 0;
}
```

3.1.2 SPFA（负环）

```
const int MAXN = 2e3 + 5;
const int MAXM = 3e3 + 5;
const int INF = 0x3f3f3f3f;
int n, m;
struct Edge {int v, w, nxt;}e[MAXM * 2];
int head[MAXN], cnt;
void addedge(int u, int v, int w) {
  e[++cnt] = (Edge){v, w, head[u]}; head[u] = cnt;
  if(w >= 0) {
    e[++cnt] = (Edge){u, w, head[v]}; head[v] = cnt;
  }
}

int vis[MAXN], dist[MAXN], num[MAXN];
queue<int> que;

bool spfa() {
  dist[1] = 0; num[1] = 0; vis[1] = 1; que.push(1);
  while(que.size()) {
    int u = que.front(); que.pop(); vis[u] = 0;
    for(int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v, w = e[i].w;
      if(dist[v] > dist[u] + w) {
        dist[v] = dist[u] + w;
        num[v] = num[u] + 1;
        if(num[v] >= n) return 1;
        if(!vis[v]) vis[v] = 1, que.push(v);
      }
    }
  }
  return 0;
}

void clear() {
  cnt = 0;
  for(int i = 1; i <= n; i++) head[i] = vis[i] = 0, dist[i] = INF, num[i] = 0;
  que = queue<int>();
}
void work() {
  n = read(), m = read();
  clear();
  for(int i = 1; i <= m; i++) {
    int u = read(), v = read(), w = read();
    addedge(u, v, w);
  }
  if(spfa()) printf("YES\n");
  else printf("NO\n");
}
int main() {
  int T = read();
  while(T--) work();
  return 0;
}
```

3.1.3 Johnson 全源

```
const int MAXN = 3e3 + 5;
const int MAXM = 6e3 + 5;
const ll INF = 0x3f3f3f3f3f3f3f3fll;
int n, m;
struct Edge {
   int v, nxt;
   ll w;
}e[MAXM + MAXN];
int head[MAXN], cnt;
void addedge(int u, int v, ll w) {
   e[++cnt].v = v; e[cnt].w = w; e[cnt].nxt = head[u]; head[u] = cnt;
}

queue<int> que;
ll h[MAXN];
int inq[MAXN], tot[MAXN];
bool spfa() {
   que.push(0); inq[0] = 1;
   memset(h, 0x3f, sizeof h); h[0] = 0;
   memset(tot, 0x00, sizeof tot); tot[0] = 0;
   while(!que.empty()) {
      int u = que.front(); que.pop();
      inq[u] = 0;
      for(int i = head[u]; i; i = e[i].nxt) {
         int v = e[i].v;
         if(h[v] > h[u] + e[i].w) {
            h[v] = h[u] + e[i].w;
            tot[v] = tot[u] + 1;
            if(tot[v] >= n + 1) return 1;
            if(!inq[v]) inq[v] = 1, que.push(v);
         }
      }
   }
   return 0;
}

struct Node {
   ll d;
   int u;
   bool operator < (const Node& B)const {
      return d > B.d;
   }
};
ll dist[MAXN];
bool vis[MAXN];
priority_queue<Node> pq;
void dij(int s) {
   memset(dist, 0x3f, sizeof dist); dist[s] = 0;
   memset(vis, 0x00, sizeof vis);
   while(!pq.empty()) pq.pop();
   pq.push((Node){0, s});
   while(!pq.empty()) {
      int u = pq.top().u; pq.pop();
      if(vis[u]) continue;
      vis[u] = 1;
      for(int i = head[u]; i; i = e[i].nxt) {
         int v = e[i].v;
         if(dist[v] > dist[u] + e[i].w) {
            dist[v] = dist[u] + e[i].w;
            pq.push((Node){dist[v], v});
         }
      }
   }
}
int main() {
   n = read(), m = read();
   for(int i = 1; i <= m; i++) {
      int u = read(), v = read(), w = read();
      addedge(u, v, w);
   }
   for(int i = 1; i <= n; i++) addedge(0, i, 0);
   if(spfa()) {
      printf("-1\n");
      return 0;
   }
```

```
    for(int u = 1; u <= n; u++)
        for(int i = head[u]; i; i = e[i].nxt)
            e[i].w += h[u] - h[e[i].v];
    for(int s = 1; s <= n; s++) {
        dij(s);
        ll ans = 0;
        for(int i = 1; i <= n; i++)
            if(dist[i] == INF) ans += ll(1e9) * i;
            else ans += (dist[i] - (h[s] - h[i])) * i;
        printf("%lld\n", ans);
    }

    return 0;
}
```

3.1.4 最小斯坦纳树

```
const int MAXN = 105;
const int MAXM = 1005;
const int MAXK = 10, MAXU = 1 << 10;
const int INF = 0x3f3f3f3f;
int n, U, m, k, head[MAXM], ver[MAXM], nxt[MAXM], edg[MAXM], cnt, id[MAXK], dp[MAXN][
      MAXU];
void addedge(int u, int v, int w) {
    ver[++cnt] = v; nxt[cnt] = head[u]; edg[cnt] = w; head[u] = cnt;
}
struct QNode{int d, u;};
bool operator < (const QNode& a, const QNode& b) {return a.d > b.d;}
priority_queue<QNode> pq;
int vis[MAXN];
void dijkstra(int s) {
    memset(vis, 0x00, sizeof vis);
    while(pq.size()) {
        int u = pq.top().u; pq.pop();
        if(vis[u]) continue;
        vis[u] = 1;
        for(int i = head[u]; i; i = nxt[i]) {
            int v = ver[i];
            if(dp[v][s] > dp[u][s] + edg[i]) {
                dp[v][s] = dp[u][s] + edg[i];
                pq.push((QNode){dp[v][s], v});
            }
        }
    }
}
```

```
int main() {
    n = read(), m = read(), k = read(); U = (1 << k) - 1;
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read(), w = read();
        addedge(u, v, w); addedge(v, u, w);
    }
    for(int i = 0; i < k; i++) id[i] = read();
    memset(dp, 0x3f, sizeof dp);
    for(int i = 0; i < k; i++)
        dp[id[i]][1 << i] = 0;
    for(int s = 1; s <= U; s++) {
        for(int i = 1; i <= n; i++) {
            for(int t = s & (s-1); t; t = s & (t-1))
                dp[i][s] = min(dp[i][s], dp[i][t] + dp[i][s^t]);
            if(dp[i][s] != INF) pq.push((QNode){dp[i][s], i});
        }
        dijkstra(s);
    }
    printf("%d\n", dp[id[0]][U]);
    return 0;
}
```

3.1.5 K 短路

```
#include<algorithm>
#include<cstdio>
#include<queue>
using namespace std;
typedef long long ll;
typedef double db;
ll read() {
    ll x = 0, f = 1; char ch = getchar();
    for(; ch < '0' || ch > '9'; ch = getchar()) if(ch == '-') f = -1;
    for(; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + int(ch - '0');
    return x * f;
}
const int MAXN = 5005, MAXM = 4e5 + 5;
const db inf = 1e40, eps = 1e-8;
int dcmp(db x) {return x < -eps ? -1 : (x > eps ? 1 : 0);}
int n, m;
db E;
namespace Graph {
    struct Edge {
        int v, nxt;
        db w;
    }e[MAXM];
    int head[MAXN], cnt = 1; // 0 is positive, 1 is negative (mod 2)
    void addedge(int u, int v, db w) {
        e[++cnt] = (Edge){v, head[u], w}; head[u] = cnt;
        e[++cnt] = (Edge){u, head[v], w}; head[v] = cnt;
    }
} using namespace Graph;
namespace SP {
    struct QNode {
        db d; int u;
    };
    bool operator < (const QNode& a, const QNode& b) {return a.d > b.d;}
    priority_queue<QNode> que;
    db dist[MAXN];
    int vis[MAXN], pre[MAXN];
    void dijkstra() {
        for(int i = 1; i <= n; i++) dist[i] = inf, vis[i] = pre[i] = 0;
        dist[n] = 0; que.push((QNode){0.0, n});
        while(que.size()) {
            int u = que.top().u; que.pop();
            if(vis[u]) continue;
            vis[u] = 1;
            for(int i = head[u]; i; i = e[i].nxt) if(i & 1) {//neg
                int v = e[i].v;
                if(dcmp(dist[v] - (dist[u] + e[i].w)) > 0) {
                    dist[v] = dist[u] + e[i].w; pre[v] = i; que.push((QNode){dist[v], v})
                        ;
                }
            }
        }
    }
} using namespace SP;
namespace LeftyTree {
    struct Node {
        int ls, rs, dist, v; db val;
    }h[MAXM << 5];
    int tot;
    int crenode(int v, db val) {
        h[++tot] = (Node){0, 0, 1, v, val}; return tot;
    }
    int merge(int x, int y) {
        if(!x || !y) return x + y;
        if(dcmp(h[x].val - h[y].val) > 0) swap(x, y);
        int p = ++tot; h[p] = h[x];
        h[p].rs = merge(h[x].rs, y);
        if(h[h[p].ls].dist < h[h[p].rs].dist) swap(h[p].ls, h[p].rs);
        h[p].dist = h[h[p].rs].dist + 1;
        return p;
    }
    void ins(int& p, int v, db val) {
        p = merge(p, crenode(v, val));
    }
} using namespace LeftyTree;
int seq[MAXN], rt[MAXN];
bool cmp(int a, int b) {return dist[a] < dist[b];}
struct QN {
    db d; int u;
};
bool operator < (const QN& a, const QN& b) {return a.d > b.d;}
priority_queue<QN> pq;
int main() {
    n = read(), m = read(); scanf("%lf", &E);
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read(); db w; scanf("%lf", &w);
        if(u == n) continue;
        addedge(u, v, w);
    }
    dijkstra();
    for(int i = 1; i <= n; i++) seq[i] = i;
    sort(seq + 1, seq + 1 + n, cmp);
    for(int k = 1; k <= n; k++) {
        int u = seq[k];
        if(pre[u]) rt[u] = rt[e[pre[u]^1].v];
        for(int i = head[u]; i; i = e[i].nxt) if(~i & 1 && i != (pre[u]^1)) {
            int v = e[i].v; db delta = e[i].w - (dist[u] - dist[v]);
            ins(rt[u], v, delta);
        }
    }
    int ans = 0;
    if(dcmp(E-dist[1]) >= 0) ans++, E -= dist[1];
    if(rt[1]) pq.push((QN){h[rt[1]].val, rt[1]});
    while(pq.size()) {
        int u = pq.top().u; db ld = pq.top().d; pq.pop();
        //now sigma = dist[1] - ld
        if(dcmp(E-dist[1]-ld) < 0) break;
        E -= dist[1] + ld; ans++;
        int nxt = rt[h[u].v];
        if(nxt) pq.push((QN){ld + h[nxt].val, nxt});
        if(h[u].ls) pq.push((QN){ld - h[u].val + h[h[u].ls].val, h[u].ls});
        if(h[u].rs) pq.push((QN){ld - h[u].val + h[h[u].rs].val, h[u].rs});
    }
    printf("%d\n", ans);
    return 0;
}
```

3.2 最小生成树

3.2.1 Prim

```
const int MAXN = 5e3+5;
const int MAXM = 2e5+5;
const int INF = 1e9;
int n, m;
int e[MAXN][MAXN], dist[MAXN], vis[MAXN], ans;

int main() {
  scanf("%d%d", &n, &m);
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
      e[i][j] = INF;
  for(int i = 1; i <= m; i++) {
    int u, v, w; scanf("%d%d%d", &u, &v, &w);
    e[u][v] = e[v][u] = min(e[u][v], w);
  }
  for(int i = 1; i <= n; i++) dist[i] = INF, vis[i] = 0;
  dist[1] = 0;
  for(int i = 1; i <= n; i++) {
    int now = -1;
    for(int j = 1; j <= n; j++)
      if(!vis[j] && (now == -1 || dist[j] < dist[now]))
        now = j;
    if(now == -1 || dist[now] == INF) {
      printf("orz\n");
      return 0;
    }
    ans += dist[now]; vis[now] = 1;
    for(int j = 1; j <= n; j++)
      if(!vis[j])
        dist[j] = min(dist[j], e[now][j]);
  }
  printf("%d\n", ans);
  return 0;
}
```

3.2.2 Kruskal

```
const int MAXN = 5e3+5;
const int MAXM = 2e5+5;
int N, M, tot, ans;
struct Edge {
  int u, v, w;
}e[MAXM];
bool cmp(const Edge& a, const Edge& b) {
  return a.w < b.w;
}
int upto[MAXN];
int getup(int x) {
  return x == upto[x] ? x : upto[x] = getup(upto[x]);
}
void Kruskal() {
  for(int i = 1; i <= N; i++) upto[i] = i;
  sort(e + 1, e + 1 + M, cmp);
  ans = tot = 0;
  for(int i = 1; i <= M && tot < N-1; i++) {
    int fu = getup(e[i].u), fv = getup(e[i].v);
    if(fu == fv) continue;
    tot++; ans += e[i].w;
    upto[fu] = fv;
  }
}
int main() {
  #ifndef ONLINE_JUDGE
  freopen("main.in", "r", stdin);
  #endif
  scanf("%d%d", &N, &M);
  for(int i = 1; i <= M; i++) {
    scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);
  }
  Kruskal();
  if(tot < N-1) printf("orz\n");
  else printf("%d\n", ans);
  return 0;
}
```

3.2.3 Boruvka

```
   typedef pair<int, int> pii;
const int MAXN = 5005;
const int MAXM = 2e5 + 5;
const int INF = 0x3f3f3f3f;
```

```
int n, m, fl, ans, upto[MAXN];
struct Edge {int u, v, w;}e[MAXM];
pii E[MAXN];
int getup(int u) {return u == upto[u] ? u : upto[u] = getup(upto[u]);}
void solve() {
    for(int i = 1; i <= n; i++) upto[i] = i;
    int tot = 0;
    while(tot < n - 1) {
        int upd = 0;
        for(int i = 1; i <= n; i++) E[i] = pii(INF, INF);
        for(int i = 1; i <= m; i++) {
            int fu = getup(e[i].u), fv = getup(e[i].v);
            if(fu == fv) continue;
            upd = 1;
            E[fu] = min(E[fu], pii(e[i].w, i));
            E[fv] = min(E[fv], pii(e[i].w, i));
        }
        if(!upd) break;
        for(int i = 1; i <= m; i++) {
            int fu = getup(e[i].u), fv = getup(e[i].v);
            if(fu == fv) continue;
            if(E[fu].second == i || E[fv].second == i) {
                upto[fu] = fv; ans += e[i].w; tot++;
            }
        }
    }
    if(tot < n - 1) fl = 1;
}
int main() {
    n = read(), m = read();
    for(int i = 1; i <= m; i++) {
        e[i].u = read(); e[i].v = read(); e[i].w = read();
    }
    solve();
    if(fl) printf("orz\n");
    else printf("%d\n", ans);
    return 0;
}
```

3.3 tarjan

3.3.1 缩点

```
#include <iostream>
#include <iostream>
```

```
using namespace std;

constexpr int MAXN = 1e4 + 5;
constexpr int MAXM = 1e5 + 5;

int n, m, a[MAXN];
struct Graph {
   struct Edge {
      int v, nxt;
   }e[MAXM];
   int head[MAXN], cnt;
   void addedge(int u, int v) {
      e[++cnt] = {v, head[u]};
      head[u] = cnt;
   }
}G1, G2;

int dfn[MAXN], low[MAXN], _dfn, stk[MAXN], top, ins[MAXN], scc[MAXN], _scc, scc_sum[
    MAXN], deg[MAXN], que[MAXN], hd, tl, f[MAXN];

void tarjan(int u) {
   dfn[u] = low[u] = ++_dfn;
   stk[++top] = u;
   ins[u] = 1;
   for(int i = G1.head[u]; i; i = G1.e[i].nxt) {
      int v = G1.e[i].v;
      if(!dfn[v]) {
         tarjan(v);
         low[u] = min(low[u], low[v]);
      } else if(ins[v]) {
         low[u] = min(low[u], dfn[v]);
      }
   }
   if(dfn[u] == low[u]) {
      ++_scc;
      int t;
      do {
         t = stk[top--];
         ins[t] = 0;
         scc[t] = _scc;
         scc_sum[_scc] += a[t];
      }while(t != u);
   }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        cin >> a[i];
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        G1.addedge(u, v);
    }
    for(int i = 1; i <= n; i++)
        if(!dfn[i])
            tarjan(i);
    for(int u = 1; u <= n; u++)
        for(int i = G1.head[u]; i; i = G1.e[i].nxt) {
            int v = G1.e[i].v;
            if(scc[u] == scc[v]) continue;
            G2.addedge(scc[u], scc[v]);
            deg[scc[v]]++;
        }
    hd = 1; tl = 0;
    for(int i = 1; i <= _scc; i++)
        if(deg[i] == 0) {
            que[++tl] = i;
            f[i] = scc_sum[i];
        }
    while(hd <= tl) {
        int u = que[hd++];
        for(int i = G2.head[u]; i; i = G2.e[i].nxt) {
            int v = G2.e[i].v;
            f[v] = max(f[v], f[u] + scc_sum[v]);
            if(--deg[v] == 0)
                que[++tl] = v;
        }
    }
    int ans = 0;
    for(int i = 1; i <= _scc; i++)
        ans = max(ans, f[i]);
    cout << ans << '\n';
    return 0;
}
```

3.3.2 割点

```
#include <algorithm>
```

```
#include <iostream>
using namespace std;

constexpr int MAXN = 2e4 + 5;
constexpr int MAXM = 2e5 + 5;
int n, m;
struct Edge {
   int v, nxt;
}e[MAXM];
int head[MAXN], cnt;
void addedge(int u, int v) {
   e[++cnt] = {v, head[u]};
   head[u] = cnt;
}

int dfn[MAXN], low[MAXN], _dfn, cut[MAXN];

void tarjan(int u, int fa, int rt) {
   dfn[u] = low[u] = ++_dfn;
   int ch = 0;
   for(int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v;
      if(v == fa) continue;
      if(!dfn[v]) {
         ch++;
         tarjan(v, u, rt);
         low[u] = min(low[u], low[v]);
         if(low[v] >= dfn[u]) {
            if(u != rt || ch >= 2)
               cut[u] = 1;
         }
      } else
         low[u] = min(low[u], dfn[v]);
   }
}

int main() {
   ios::sync_with_stdio(false);
   cin.tie(nullptr);
   cin >> n >> m;
   for(int i = 1; i <= m; i++) {
      int u, v;
      cin >> u >> v;
      addedge(u, v);
      addedge(v, u);
   }
```

```
    for(int i = 1; i <= n; i++)
        if(!dfn[i])
            tarjan(i, 0, i);
    int ans = 0;
    for(int i = 1; i <= n; i++)
        if(cut[i])
            ++ans;
    cout << ans << '\n';
    for(int i = 1; i <= n; i++)
        if(cut[i])
            cout << i << ' ';
    cout << '\n';
    return 0;
}
```

3.3.3 桥/边双

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

constexpr int MAXN = 5e5 + 5;
constexpr int MAXM = 4e6 + 5;

int n, m;
struct Edge {
    int v, nxt;
}e[MAXM];
int head[MAXN], cnt = 1;
void addedge(int u, int v) {
    e[++cnt] = {v, head[u]};
    head[u] = cnt;
}

int dfn[MAXN], low[MAXN], _dfn, cut[MAXM];
int vis[MAXN], _bcc;
vector<int> bcc_con[MAXN];



void tarjan(int u, int toe) {
    dfn[u] = low[u] = ++_dfn;
    for(int i = head[u]; i; i = e[i].nxt) {
        if((i ^ 1) == toe)
```

```
            continue;
        int v = e[i].v;
        if(!dfn[v]) {
            tarjan(v, i);
            low[u] = min(low[u], low[v]);
            if(low[v] > dfn[u])
                cut[i] = cut[i^1] = 1;
        } else
            low[u] = min(low[u], dfn[v]);
    }
}

void dfs(int u) {
    vis[u] = 1;
    bcc_con[_bcc].push_back(u);
    for(int i = head[u]; i; i = e[i].nxt) {
        if(cut[i])
            continue;
        int v = e[i].v;
        if(vis[v])
            continue;
        dfs(v);
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        addedge(u, v);
        addedge(v, u);
    }
    for(int i = 1; i <= n; i++)
        if(!dfn[i])
            tarjan(i, 0);
    for(int i = 1; i <= n; i++)
        if(!vis[i]) {
            ++_bcc;
            dfs(i);
        }
    cout << _bcc << '\n';
    for(int i = 1; i <= _bcc; i++) {
        cout << bcc_con[i].size() << ' ';
        for(auto v : bcc_con[i])
            cout << v << ' ';
        cout << '\n';
    }
    return 0;
}
```

3.3.4 点双

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

constexpr int MAXN = 5e5 + 5;
constexpr int MAXM = 4e6 + 5;

int n, m;
struct Edge {
    int v, nxt;
}e[MAXM];
int head[MAXN], cnt;
void addedge(int u, int v) {
    e[++cnt] = {v, head[u]};
    head[u] = cnt;
}

int dfn[MAXN], low[MAXN], _dfn, stk[MAXN], top;
vector<vector<int> > dcc;//v-dcc

void tarjan(int u, int fa, int rt) {
    dfn[u] = low[u] = ++_dfn;
    stk[++top] = u;
    if(u == rt && !head[u]) {
        dcc.push_back(vector<int>{u});
        return ;
    }
    int ch = 0;
    for(int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].v;
        if(v == fa)
            continue;
        if(!dfn[v]) {
            ch++;
            tarjan(v, u, rt);
            low[u] = min(low[u], low[v]);
            if(low[v] >= dfn[u]) {
                dcc.push_back(vector<int>{});
                int t;
                do {
                    t = stk[top--];
                    dcc[(int)dcc.size()-1].push_back(t);
                }while(t != v);
                dcc[(int)dcc.size()-1].push_back(u);
            }
        } else
            low[u] = min(low[u], dfn[v]);
    }
}


int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    for(int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        if(u == v)
            continue; //cases that (u, u) and u is size 1
        addedge(u, v);
        addedge(v, u);
    }
    for(int i = 1; i <= n; i++)
        if(!dfn[i])
            tarjan(i, 0, i);
    cout << dcc.size() << '\n';
    for(const auto &t : dcc) {
        cout << t.size() << ' ';
        for(auto v : t)
            cout << v << ' ';
        cout << '\n';
    }
    return 0;
}
```

3.4 二分图

3.4.1 二分图最大匹配-匈牙利

```
const int MAXN = 505;
const int MAXM = 5e4 + 5;
int n, m, ed;
struct Edge {
    int v, nxt;
}e[MAXM];
int head[MAXN], cnt;
void addedge(int u, int v) {
    e[++cnt].v = v; e[cnt].nxt = head[u]; head[u] = cnt;
}
int match[MAXN], vis[MAXN];
bool dfs(int u) {
    for(int i = head[u]; i; i = e[i].nxt) {
        int v = e[i].v;
        if(vis[v]) continue;
        vis[v] = 1;
        if(!match[v] || dfs(match[v])) {
            match[v] = u;
            return 1;
        }
    }
    return 0;
}
int ans;
int main() {
    n = read(), m = read(), ed = read();
    for(int i = 1; i <= ed; i++) {
        int u = read(), v = read();
        addedge(u, v);
    }
    memset(match, 0x00, sizeof match);
    for(int i = 1; i <= n; i++) {
        memset(vis, 0x00, sizeof vis);
        if(dfs(i)) ans++;
    }
    printf("%d\n", ans);
    return 0;
}
```

3.4.2 二分图最大匹配-网络流

```
const int MAXN = 5e5;
const int MAXE = 5e4 + 5;
const int INF = 0x3f3f3f3f;
const int MAXNODE = MAXN * 2;
const int MAXEDGE = MAXN * 2 + MAXE;
int n, m, E, s, t;
struct Edge {
   int v, nxt, w;
}e[MAXEDGE * 2];
int head[MAXNODE], cnt = 1;
void addedge(int u, int v, int w) {
   e[++cnt].v = v; e[cnt].w = w; e[cnt].nxt = head[u]; head[u] = cnt;
   e[++cnt].v = u; e[cnt].w = 0; e[cnt].nxt = head[v]; head[v] = cnt;
}


int dep[MAXN];
int que[MAXN], hd, tl;
bool bfs() {
   hd = 1; tl = 0; que[++tl] = s;
   memset(dep, 0x00, sizeof dep);
   dep[s] = 1;
   while(hd <= tl) {
      int u = que[hd++];
      for(int i = head[u]; i; i = e[i].nxt) {
         int v = e[i].v;
         if(e[i].w && !dep[v]) {
            dep[v] = dep[u] + 1;
            que[++tl] = v;
            if(v == t) return 1;
         }
      }
   }
   return 0;
}
int dfs(int u, int flow) {
   if(u == t) return flow;
   int rest = flow;
   for(int i = head[u]; i && rest; i = e[i].nxt) {
      int v = e[i].v;
      if(e[i].w && dep[v] == dep[u] + 1) {
         int k = dfs(v, min(rest, e[i].w));
         if(!k) dep[v] = 0;
         e[i].w -= k;
         e[i^1].w += k;
         rest -= k;
      }
   }
   return flow - rest;
}
int dinic() {
    int ans = 0, flow;
    while(bfs())
        while(flow = dfs(s, INF))
            ans += flow;
    return ans;
}
int main() {
    n = read(), m = read(), E = read();
    //s : 0
    //left : 1 ~ n
    //right : n+1 ~ n+m
    //t : n+m+1
    s = 0, t = n+m+1;
    for(int i = 1; i <= E; i++) {
        int u = read(), v = read();
        addedge(u, v + n, 1);
    }
    for(int i = 1; i <= n; i++) addedge(s, i, 1);
    for(int i = n+1; i <= n+m; i++) addedge(i, t, 1);
    printf("%d\n", dinic());
    return 0;
}
```

3.4.3 KM

```
const int MAXN = 505;
const ll inf = 0x3f3f3f3f3f3f3f3fll;
int n, m;
ll e[MAXN][MAXN], lx[MAXN], ly[MAXN], slack[MAXN];
int px[MAXN], py[MAXN], vx[MAXN], vy[MAXN], pre[MAXN];
queue<int> que;
void aug(int v) {
    while(v) {
        int t = px[pre[v]];
        px[pre[v]] = v;
        py[v] = pre[v];
        v = t;
    }
}

void bfs(int s) {
    for(int i = 1; i <= n; i++) vx[i] = vy[i] = 0, slack[i] = inf;
    que = queue<int>(); que.push(s);
    while(1) {
        while(que.size()) {
            int u = que.front(); que.pop();
            vx[u] = 1;
            for(int v = 1; v <= n; v++) if(!vy[v]) {
                if(lx[u] + ly[v] - e[u][v] < slack[v]) {
                    slack[v] = lx[u] + ly[v] - e[u][v];
                    pre[v] = u;
                    if(!slack[v]) {
                        vy[v] = 1;
                        if(!py[v]) { aug(v); return; }
                        else que.push(py[v]);
                    }
                }
            }
        }
        ll d = inf;
        for(int v = 1; v <= n; v++) if(!vy[v]) d = min(d, slack[v]);
        for(int i = 1; i <= n; i++) {
            if(vx[i]) lx[i] -= d;
            if(vy[i]) ly[i] += d;
            else slack[i] -= d;
        }
        for(int v = 1; v <= n; v++) if(!vy[v] && !slack[v]) {
            vy[v] = 1;
            if(!py[v]) { aug(v); return ;}
            else que.push(py[v]);
        }
    }

}
void KM() {
    for(int i = 1; i <= n; i++) lx[i] = -inf, ly[i] = 0;
    for(int u = 1; u <= n; u++) for(int v = 1; v <= n; v++) lx[u] = max(lx[u], e[u][v
        ]);
    for(int i = 1; i <= n; i++) bfs(i);
}
int main() {
    n = read(), m = read();
    for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++) e[i][j] = -inf;
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read();
        e[u][v] = max(e[u][v], read());
    }
    KM();
    ll ans = 0;
    for(int i = 1; i <= n; i++) ans += lx[i] + ly[i];
    printf("%lld\n", ans);
    for(int i = 1; i <= n; i++) printf("%d ", py[i]);
    printf("\n");
    return 0;
}
```

3.5 Kruskal 重构树

```
const int MAXN = 30005;
const int MAXM = 60005;
int n, m, k, num, upto[MAXN], val[MAXN], head[MAXN], ver[MAXM], nxt[MAXM], cnt, dep[
     MAXN], sz[MAXN], fa[MAXN], son[MAXN], top[MAXN];
struct Edge {int u, v, w;}e[MAXM];
bool cmp(const Edge& a, const Edge& b) {return a.w < b.w;}
int getup(int u) {return u == upto[u] ? u : upto[u] = getup(upto[u]);}
void addedge(int u, int v) {ver[++cnt] = v; nxt[cnt] = head[u]; head[u] = cnt;}
void Kruskal() {
    num = n;
    sort(e + 1, e + 1 + m, cmp);
    for(int i = 1; i <= n; i++) upto[i] = i;
    for(int i = 1; i <= m; i++) {
        int fu = getup(e[i].u), fv = getup(e[i].v);
        if(fu == fv) continue;
        val[++num] = e[i].w; upto[num] = upto[fu] = upto[fv] = num;
        addedge(fu, num); addedge(num, fu); addedge(fv, num); addedge(num, fv);
    }
}
void dfs1(int u, int f) {
    dep[u] = dep[f] + 1; sz[u] = 1; fa[u] = f; son[u] = 0;
    for(int i = head[u]; i; i = nxt[i]) if(ver[i] != f) {
        int v = ver[i]; dfs1(v, u); sz[u] += sz[v];
        if(sz[v] > sz[son[u]]) son[u] = v;
    }
}
void dfs2(int u, int tprt) {
    top[u] = tprt; if(son[u]) dfs2(son[u], tprt);
    for(int i = head[u]; i; i = nxt[i]) if(ver[i] != fa[u] && ver[i] != son[u]) {
        int v = ver[i]; dfs2(v, v);
    }
}
int Lca(int u, int v) {
    while(top[u] != top[v]) {
        if(dep[top[u]] < dep[top[v]]) swap(u, v);
        u = fa[top[u]];
    }
    return dep[u] > dep[v] ? v : u;
}
int main() {
    n = read(); m = read(); k = read();
    for(int i = 1; i <= m; i++) e[i].u = read(), e[i].v = read(), e[i].w = read();
    Kruskal();
    int rt = getup(1);
    dfs1(rt, 0); dfs2(rt, rt);
    for(int i = 1; i <= k; i++) {
        int a = read(), b = read();
        printf("%d\n", val[Lca(a, b)]);
    }
    return 0;
}
```

3.6 网络流

3.6.1 最大流 Dinic

```
const int MAXN = 205;
const int MAXM = 5005;
const ll INF = 0x3f3f3f3f3f3f3f3fll;
int n, m, s, t;
struct Edge {
    int v, nxt;
    ll w;
}e[MAXM << 1];
int cur[MAXN], head[MAXN], cnt = 1;
void addedge(int u, int v, ll w) {
    e[++cnt].v = v; e[cnt].w = w; e[cnt].nxt = head[u]; head[u] = cnt;
    e[++cnt].v = u; e[cnt].w = 0; e[cnt].nxt = head[v]; head[v] = cnt;
}


int dep[MAXN], que[MAXN], hd, tl;
bool bfs() {
    memset(dep, 0x00, sizeof dep); hd = 1; tl = 0;
    memcpy(cur, head, sizeof cur);
    que[++tl] = s; dep[s] = 1;
    while(hd <= tl) {
        int u = que[hd++];
        for(int i = head[u]; i; i = e[i].nxt) {
            int v = e[i].v;
            if(e[i].w && !dep[v]) {
                dep[v] = dep[u] + 1;
                que[++tl] = v;
                if(v == t) return 1;
            }
        }
    }
    return 0;
}
ll dfs(int u, ll flow) {
    if(u == t) return flow;
    ll rest = flow;
    for(int i = cur[u]; i && rest; i = e[i].nxt) {
        cur[u] = i;
        int v = e[i].v;
        if(e[i].w && dep[v] == dep[u] + 1) {
            ll k = dfs(v, min(rest, e[i].w));
            if(!k) dep[v] = 0;
            e[i].w -= k;
            rest -= k;
            e[i ^ 1].w += k;
        }
    }
    return flow - rest;
}

ll dinic() {
    ll ans = 0, flow;
    while(bfs())
        while(flow = dfs(s, INF))
            ans += flow;
    return ans;
}
int main() {
    n = read(), m = read(), s = read(), t = read();
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read(); ll w = read();
        addedge(u, v, w);
    }
    printf("%lld\n", dinic());
    return 0;
}
```

3.6.2 费用流

```
const int MAXN = 5e3 + 5, MAXM = 1e5 + 5, INF = 0x3f3f3f3f;
```

```
int n, m, s, t, head[MAXN], cnt = 1, ver[MAXM << 1], nxt[MAXM << 1], edg[MAXM << 1],
     cap[MAXM << 1], dist[MAXN], inq[MAXN], vis[MAXN], cur[MAXN], maxflow, mincost;
void addedge(int u, int v, int w, int c) {
   ver[++cnt] = v; edg[cnt] = w; cap[cnt] = c; nxt[cnt] = head[u]; head[u] = cnt;
   ver[++cnt] = u; edg[cnt] = 0; cap[cnt] = -c; nxt[cnt] = head[v]; head[v] = cnt;
}
bool spfa() {
   queue<int> que;
   memset(dist, 0x3f, sizeof dist), memset(inq, 0x00, sizeof inq); memcpy(cur, head,
        sizeof head);
   dist[s] = 0; que.push(s); inq[s] = 1;
   while(que.size()) {
      int u = que.front(); que.pop();
      inq[u] = 0;
      for(int i = head[u]; i; i = nxt[i])
         if(edg[i] && dist[ver[i]] > dist[u] + cap[i]) {
            dist[ver[i]] = dist[u] + cap[i];
            if(!inq[ver[i]]) inq[ver[i]] = 1, que.push(ver[i]);
         }
   }
   return dist[t] != INF;
}
int dfs(int u, int flow) {
   if(u == t) return flow;
   vis[u] = 1; int rest = flow;
   for(int i = cur[u]; i && rest; i = nxt[i]) {
      cur[u] = i;
      if(!vis[ver[i]] && edg[i] && dist[ver[i]] == dist[u] + cap[i]) {
         int k = dfs(ver[i], min(rest, edg[i]));
         edg[i] -= k; edg[i^1] += k; rest -= k;
         mincost += cap[i] * k;
      }
   }
   vis[u] = 0;
   return flow - rest;
}
void MCMF() {
   while(spfa()) {
      maxflow += dfs(s, INF);
   }
}
int main() {
   n = read(); m = read(); s = read(); t = read();
   for(int i = 1; i <= m; i++) {
      int u = read(), v = read(), w = read(), c = read();
      addedge(u, v, w, c);
```

```
    }
    MCMF();
    printf("%d %d\n", maxflow, mincost);
    return 0;
}
```

4 数据结构

4.1 线段树

```
const int MAXN = 1e5 + 5;
int n, m;
ll a[MAXN];
namespace Sgt {
  #define ls o << 1
  #define rs o << 1 | 1
  ll val[MAXN << 2], tag[MAXN << 2];
  void pushup(int o) {
    val[o] = val[ls] + val[rs];
  }
  void build(int o, int l, int r, ll a[]) {
    if(l == r) {
      val[o] = a[l]; tag[o] = 0;
      return ;
    }
    int m = (l + r) >> 1;
    build(ls, l, m, a);
    build(rs, m+1, r, a);
    pushup(o);
  }
  void addpoint(int o, int l, int r, ll v) {
    val[o] += (r-l+1) * v; tag[o] += v;
  }
  void pushdown(int o, int l, int r) {
    if(tag[o]) {
      int m = (l + r) >> 1;
      addpoint(ls, l, m, tag[o]);
      addpoint(rs, m+1, r, tag[o]);
      tag[o] = 0;
    }
  }
  void addrange(int o, int l, int r, int x, int y, ll v) {
    if(x <= l && r <= y) {
      addpoint(o, l, r, v);
```

```
            return ;
        }
        pushdown(o, l, r);
        int m = (l + r) >> 1;
        if(x <= m) addrange(ls, l, m, x, y, v);
        if(y > m) addrange(rs, m+1, r, x, y, v);
        pushup(o);
    }
    ll sumrange(int o, int l, int r, int x, int y) {
        if(x <= l && r <= y) return val[o];
        pushdown(o, l, r);
        int m = (l + r) >> 1;
        if(y <= m) return sumrange(ls, l, m, x, y);
        else if(x > m) return sumrange(rs, m+1, r, x, y);
        else return sumrange(ls, l, m, x, m) + sumrange(rs, m+1, r, m+1, y);
    }
    #undef ls
    #undef rs
}
int main() {
    #ifdef LOCAL
    freopen("main.in", "r", stdin);
    #endif
    scanf("%d%d", &n, &m);
    for(int i = 1; i <= n; i++) scanf("%lld", &a[i]);
    Sgt::build(1, 1, n, a);
    for(int i = 1; i <= m; i++) {
        int opt, x, y;
        scanf("%d%d%d", &opt, &x, &y);
        if(opt == 1) {
            ll v; scanf("%lld", &v);
            Sgt::addrange(1, 1, n, x, y, v);
        } else {
            printf("%lld\n", Sgt::sumrange(1, 1, n, x, y));
        }
    }
    return 0;
}
```

4.2 可持久化线段树

```
const int MAXN = 2e5 + 5;
namespace Segment_tree {
    #define L e[o].ls
    #define R e[o].rs
    const int MAXNODE = MAXN << 5;
    struct Node {
        int ls, rs, num;
    }e[MAXNODE];
    int tot, rt[MAXN];
    void pushup(int o) {
        e[o].num = e[L].num + e[R].num;
    }
    void build(int& o, int l, int r) {
        o = ++tot;
        if(l == r) return ;
        int m = (l + r) >> 1;
        build(L, l, m); build(R, m+1, r);
    }
    void update(int& o, int pre, int l, int r, int x) {
        o = ++tot; e[o] = e[pre];
        if(l == r) {e[o].num++; return ;}
        int m = (l + r) >> 1;
        if(x <= m) update(L, e[pre].ls, l, m, x);
        else update(R, e[pre].rs, m+1, r, x);
        pushup(o);
    }
    int query(int u, int v, int l, int r, int k) {
        if(l == r) return l;
        int m = (l + r) >> 1;
        if(k <= e[e[v].ls].num - e[e[u].ls].num) return query(e[u].ls, e[v].ls, l, m, k
            );
        else return query(e[u].rs, e[v].rs, m+1, r, k - (e[e[v].ls].num - e[e[u].ls].
            num));
    }
    #undef L
    #undef R
}
using namespace Segment_tree;
int n, m;
int a[MAXN];
int lsh[MAXN], idx;
void LSH() {
    sort(lsh + 1, lsh + 1 + idx);
    idx = unique(lsh + 1, lsh + 1 + idx) - lsh - 1;
}
int LSH(int x) {
    return lower_bound(lsh + 1, lsh + 1 + idx, x) - lsh;
}

int main() {
    n = read(), m = read();
    for(int i = 1; i <= n; i++) a[i] = lsh[++idx] = read();
    LSH();
    build(rt[0], 1, idx);
    for(int i = 1; i <= n; i++) update(rt[i], rt[i-1], 1, idx, LSH(a[i]));
    for(int i = 1; i <= m; i++) {
        int l = read(), r = read(), k = read();
        printf("%d\n", lsh[query(rt[l-1], rt[r], 1, idx, k)]);
    }

    return 0;
}
```

4.3 树状数组

```
const int MAXN = 5e5 + 5;
int n, m, a[MAXN], s[MAXN];
namespace BIT {
   int t[MAXN];
   int lb(int x) {return x & -x;}
   void build() {
      for(int i = 1; i <= n; i++) t[i] = s[i] - s[i - lb(i)];
   }
   void add(int x, int v) {
      for(int i = x; i <= n; i += lb(i)) t[i] += v;
   }
   int sum(int x) {
      int s = 0;
      for(int i = x; i; i -= lb(i)) s += t[i];
      return s;
   }
   int sum(int x, int y) {
      return sum(y) - sum(x-1);
   }
}
using BIT::build; using BIT::sum; using BIT::add;
int main() {
   scanf("%d%d", &n, &m);
   for(int i = 1; i <= n; i++) {
      scanf("%d", &a[i]);
      s[i] = s[i-1] + a[i];
   }
   build();
   for(int i = 1; i <= m; i++) {
      int opt; scanf("%d", &opt);
```

```
    if(opt == 1) {
      int x, k; scanf("%d%d", &x, &k);
      add(x, k);
    } else {
      int x, y; scanf("%d%d", &x, &y);
      printf("%d\n", sum(x, y));
    }
  }
  return 0;
}
```

4.4 ST 表

```
const int MAXN = 1e5 + 5;
int n, m, st[MAXN][17], lg2[MAXN];

int main() {
  n = read(), m = read();
  for(int i = 1; i <= n; i++) st[i][0] = read();
  lg2[0] = -1; for(int i = 1; i <= n; i++) lg2[i] = lg2[i >> 1] + 1;
  for(int k = 1; (1 << k) <= n; k++)
    for(int i = 1; i + (1 << k) - 1 <= n; i++)
      st[i][k] = max(st[i][k-1], st[i+(1 << (k-1))][k-1]);
  for(int i = 1; i <= m; i++) {
    int l = read(), r = read();
    int k = lg2[r-l+1];
    printf("%d\n", max(st[l][k], st[r - (1 << k) + 1][k]));
  }
  return 0;
}
```

4.5 平衡树

4.5.1 Treap

```
const int MAXN = 1e5 + 5;
const int INF = 0x3f3f3f3f;
int n;
namespace Treap {
   struct Node {int ls, rs, s, c, k, v;}e[MAXN];
   int tot;
   void upd(int p) {e[p].s = e[e[p].ls].s + e[e[p].rs].s + e[p].c;}
   void lturn(int& p) {int s = e[p].rs; e[p].rs = e[s].ls; upd(p); e[s].ls = p; upd(s
       ); p = s;}
   void rturn(int& p) {int s = e[p].ls; e[p].ls = e[s].rs; upd(p); e[s].rs = p; upd(s
       ); p = s;}
   int myrand() {
      static unsigned int seed = 123213;
      seed ^= (seed << 2) * 1321;
      seed ^= (seed >> 2) * 2133;
      seed += 13234;
      return (int)seed;
   }
   void ins(int& p, int v) {
      if(!p) e[p = ++tot] = (Node){0, 0, 1, 1, myrand(), v};
      else if(e[p].v == v) e[p].c++, upd(p);
      else if(e[p].v > v)
         ins(e[p].ls, v), e[e[p].ls].k < e[p].k ? rturn(p) : upd(p);
      else
         ins(e[p].rs, v), e[e[p].rs].k < e[p].k ? lturn(p) : upd(p);
   }
   void del(int& p, int v) {
      if(e[p].v > v) del(e[p].ls, v), upd(p);
      else if(e[p].v < v) del(e[p].rs, v), upd(p);
      else if(e[p].c > 1) e[p].c--, upd(p);
      else if(!e[p].ls) p = e[p].rs;
      else if(!e[p].rs) p = e[p].ls;
      else if(e[e[p].ls].k < e[e[p].rs].k)
         rturn(p), del(e[p].rs, v), upd(p);
      else
         lturn(p), del(e[p].ls, v), upd(p);
   }
   int rank(int p, int v) {
      if(!p) return 1;
      else if(e[p].v == v) return e[e[p].ls].s + 1;
      else if(e[p].v > v) return rank(e[p].ls, v);
      else return e[e[p].ls].s + e[p].c + rank(e[p].rs, v);
   }
   int kth(int p, int x) {
      if(x > e[p].s) return -1;
      else if(x <= e[e[p].ls].s) return kth(e[p].ls, x);
      else if(x <= e[e[p].ls].s + e[p].c) return e[p].v;
      else return kth(e[p].rs, x - e[e[p].ls].s - e[p].c);
   }
   int pre(int p, int v) {
      if(!p) return -INF;
      else if(e[p].v < v) return max(pre(e[p].rs, v), e[p].v);
      else return pre(e[p].ls, v);
```

```
    }
    int nxt(int p, int v) {
        if(!p) return INF;
        else if(e[p].v > v) return min(nxt(e[p].ls, v), e[p].v);
        else return nxt(e[p].rs, v);
    }
}
int rt;
int main() {
    n = read();
    for(int i = 1; i <= n; i++) {
        int opt = read(), x = read();
        if(opt == 1) {
            Treap::ins(rt, x);
        } else if(opt == 2) {
            Treap::del(rt, x);
        } else if(opt == 3) {
            printf("%d\n", Treap::rank(rt, x));
        } else if(opt == 4) {
            printf("%d\n", Treap::kth(rt, x));
        } else if(opt == 5) {
            printf("%d\n", Treap::pre(rt, x));
        } else {
            printf("%d\n", Treap::nxt(rt, x));
        }
    }
    return 0;
}
```

4.5.2 Splay

```
const int MAXN = 1e5 + 5, INF = 0x3f3f3f3f;
namespace Splay {
#define ls ch[0]
#define rs ch[1]
    struct Node {int ch[2], fa, val, siz, rev;} e[MAXN];
    int tot, rt;
    void upd(int p) {e[p].siz = e[e[p].ls].siz + e[e[p].rs].siz + 1;}
    int idy(int p) {return e[e[p].fa].rs == p;}
    void psdrev(int p) {if(p) swap(e[p].ls, e[p].rs), e[p].rev ^= 1;}
    void psd(int p) {if(e[p].rev) e[p].rev = 0, psdrev(e[p].ls), psdrev(e[p].rs);}
    void rtt(int p) {
        int y = e[p].fa, z = e[y].fa, k = idy(p), s = e[p].ch[k^1];
        e[p].fa = z; e[z].ch[idy(y)] = p;
        e[s].fa = y; e[y].ch[k] = s;
```

```
      e[y].fa = p; e[p].ch[k^1] = y;
      upd(y); upd(p);
   }
   void splay(int p, int to) {
      for(int y = e[p].fa; e[p].fa != to; rtt(p), y = e[p].fa)
         if(e[y].fa != to) rtt(idy(p) == idy(y) ? y : p);
      if(!to) rt = p;
   }
   void build(int& p, int l, int r, int a[], int fa) {
      if(l > r) {p = 0; return ;}
      int m = (l + r) >> 1;
      e[p = ++tot] = (Node){{0, 0}, fa, a[m], 1, 0};
      build(e[p].ls, l, m-1, a, p);
      build(e[p].rs, m+1, r, a, p);
      upd(p);
   }
   int kth(int k) {
      k++;
      int p = rt;
      while(p) {
         psd(p);
         if(k <= e[e[p].ls].siz) p = e[p].ls;
         else if(k <= e[e[p].ls].siz + 1) return p;
         else {k -= e[e[p].ls].siz + 1; p = e[p].rs;}
      }
      return -1;
   }
   void rev(int l, int r) {
      int L = kth(l-1), R = kth(r+1);
      splay(L, 0); splay(R, L);
      int p = e[R].ls;
      psdrev(p);
   }
   void print(int p) {
      if(!p) return ;
      psd(p);
      print(e[p].ls);
      if(e[p].val != INF) printf("%d ", e[p].val);
      print(e[p].rs);
   }
#undef ls
#undef rs
} using namespace Splay;
int n, m, a[MAXN];
int main() {
   n = read(), m = read();
```

```
    for(int i = 1; i <= n; i++) a[i] = i;
    a[0] = INF, a[n+1] = INF;
    build(rt, 0, n+1, a, 0);
    for(int i = 1; i <= m; i++) {
        int l = read(), r = read();
        rev(l, r);
    }
    print(rt);
    return 0;
}
```

4.5.3 FHQ Treap

```
const int MAXN = 1e5 + 5;
namespace FHQTreap {
    struct Node {
        int ls, rs;
        int v, k, s;
    }e[MAXN];
    int cnt;
    void upd(int x) {e[x].s = e[e[x].ls].s + e[e[x].rs].s + 1;}
    int crenode(int v) {
        e[++cnt] = (Node){0, 0, v, rand(), 1};
        return cnt;
    }
    void split(int now, int& x, int& y, int v) {
        if(!now) {x = y = 0; return;}
        if(e[now].v <= v)
            x = now, split(e[now].rs, e[x].rs, y, v), upd(x);
        else
            y = now, split(e[now].ls, x, e[y].ls, v), upd(y);
    }
    int merge(int x, int y) {
        if(!x || !y) return x | y;
        if(e[x].k < e[y].k)
            return e[x].rs = merge(e[x].rs, y), upd(x), x;
        else
            return e[y].ls = merge(x, e[y].ls), upd(y), y;
    }
    int kth(int x, int k) {
        while(1) {
            if(k <= e[e[x].ls].s) x = e[x].ls;
            else {
                k -= e[e[x].ls].s + 1;
                if(!k) return x;
            x = e[x].rs;
         }
      }
   }
   void ins(int& now, int v) {
      int x, y;
      split(now, x, y, v);
      now = merge(merge(x, crenode(v)), y);
   }
   void del(int& now, int v) {
      int x, y, z;
      split(now, x, y, v-1); split(y, y, z, v);
      y = merge(e[y].ls, e[y].rs);
      now = merge(merge(x, y), z);
   }
   int rk(int& now, int v) {
      int x, y;
      split(now, x, y, v-1);
      int ans = e[x].s + 1;
      now = merge(x, y);
      return ans;
   }
   int atrank(int& now, int k) {
      return e[kth(now, k)].v;
   }
   int pre(int& now, int v) {
      int x, y;
      split(now, x, y, v-1);
      int ans = e[kth(x, e[x].s)].v;
      now = merge(x, y);
      return ans;
   }
   int nxt(int& now, int v) {
      int x, y;
      split(now, x, y, v);
      int ans = e[kth(y, 1)].v;
      now = merge(x, y);
      return ans;
   }
}
using namespace FHQTreap;
int rt;
int main() {
   rt = 0;
   int n = read();
   while(n--) {
      int opt = read(), x = read();
      if(opt == 1) ins(rt, x);
      else if(opt == 2) del(rt, x);
      else if(opt == 3) printf("%d\n", rk(rt, x));
      else if(opt == 4) printf("%d\n", atrank(rt, x));
      else if(opt == 5) printf("%d\n", pre(rt, x));
      else printf("%d\n", nxt(rt, x));
   }
   return 0;
}
```

4.6 左偏树

```
const int MAXN = 1e5 + 5;
int n, m;
int ls[MAXN], rs[MAXN], val[MAXN], dis[MAXN];
int fa[MAXN];//该节点所在的最大的左偏树的树根
int getfa(int x) {return x == fa[x] ? x : fa[x] = getfa(fa[x]);}
int merge(int x, int y) {//合并x,y并把x作为根
    if(!x || !y) return x + y;
    if(val[x] > val[y] || (val[x] == val[y] && x > y)) swap(x, y);
    rs[x] = merge(rs[x], y);//把y和右子树合并
    if(dis[ls[x]] < dis[rs[x]]) swap(ls[x], rs[x]);//左偏树的性质
    fa[x] = fa[ls[x]] = fa[rs[x]] = x;
    dis[x] = dis[rs[x]] + 1;//also性质
    return x;
}
void pop(int x) {
    val[x] = -1;
    fa[ls[x]] = ls[x];
    fa[rs[x]] = rs[x];
    fa[x] = merge(ls[x], rs[x]);
    //在路径压缩之后，必须要在pop后，给pop掉的点一个指针指向新的根（否则就会直接断掉）
    //大意就是有些点的值是直接指向fa[x]并且靠它进行一个传递，所以还是要保留
}
int main() {
    n = read(), m = read();
    dis[0] = -1;//空子树定义为-1，方便运用dis[x] = dis[rs[x]] + 1
    for(int i = 1; i <= n; i++) fa[i] = i, val[i] = read();
    for(int i = 1; i <= m; i++) {
        int opt = read();
        if(opt == 1) {
            int x = read(), y = read();
            if(val[x] == -1 || val[y] == -1) continue;
            int fx = getfa(x), fy = getfa(y);
            if(fx != fy) fa[fx] = fa[fy] = merge(fx, fy);
        } else {
            int x = read();
            if(val[x] == -1) printf("-1\n");
            else {
                int fx = getfa(x);
                printf("%d\n", val[fx]);
                pop(fx);
            }
        }
    }
    return 0;
}
```

4.7 K-D Tree

```
const int MAXN = 100005;
const ll INF = 0x3f3f3f3f3f3f3f3f;
struct Pnt {
    int x, y;
    Pnt(int x = 0, int y = 0) :x(x), y(y) {}
};
bool cmpx(const Pnt& a, const Pnt& b) {return a.x < b.x;}
bool cmpy(const Pnt& a, const Pnt& b) {return a.y < b.y;}
priority_queue<ll, vector<ll>, greater<ll> > pq;
int n, k;
Pnt a[MAXN];
namespace KDTree {
    struct Node {
        int ls, rs, L, R, D, U, val;
    }e[MAXN];
    int tot, rt;
    bool (*Getd(int l, int r))(const Pnt&, const Pnt&) {
        double avx = 0, avy = 0, vax = 0, vay = 0;
        for(int i = l; i <= r; i++) avx += a[i].x, avy += a[i].y;
        avx /= (r-l+1); avy /= (r-l+1);
        for(int i = l; i <= r; i++)
            vax += 1ll * (a[i].x - avx) * (a[i].x - avx),
            vay += 1ll * (a[i].y - avy) * (a[i].y - avy);
        return vax > vay ? cmpx : cmpy;
    }
    void upd(int p) {
        e[p].L = e[p].R = a[e[p].val].x;
        e[p].U = e[p].D = a[e[p].val].y;
        if(e[p].ls) {
            e[p].L = min(e[p].L, e[e[p].ls].L);
            e[p].R = max(e[p].R, e[e[p].ls].R);
            e[p].D = min(e[p].D, e[e[p].ls].D);
            e[p].U = max(e[p].U, e[e[p].ls].U);
        }
        if(e[p].rs) {
            e[p].L = min(e[p].L, e[e[p].rs].L);
            e[p].R = max(e[p].R, e[e[p].rs].R);
            e[p].D = min(e[p].D, e[e[p].rs].D);
            e[p].U = max(e[p].U, e[e[p].rs].U);
        }
    }
    void build(int& p, int l, int r) {
        if(l > r) {p = 0; return;}
        int m = (l + r) >> 1;
        p = ++tot;
        nth_element(a + l, a + m, a + r + 1, Getd(l, r));
        e[p].val = m;
        build(e[p].ls, l, m-1); build(e[p].rs, m+1, r);
        upd(p);
    }
    ll sqr(ll x) {return x * x;}
    ll dist(Pnt a, Pnt b) {return sqr(a.x - b.x) + sqr(a.y - b.y);}
    ll distto(int p, Pnt v) {return max(sqr(v.x - e[p].L), sqr(v.x - e[p].R)) + max(
         sqr(v.y - e[p].D), sqr(v.y - e[p].U));}
    void query(int p, Pnt v) {
        if(!p) return;
        ll d = dist(a[e[p].val], v);
        if(d > pq.top()) pq.pop(), pq.push(d);
        ll distl = e[p].ls ? distto(e[p].ls, v) : -INF;
        ll distr = e[p].rs ? distto(e[p].rs, v) : -INF;
        if(distl > distr) {
            if(distl > pq.top()) query(e[p].ls, v);
            if(distr > pq.top()) query(e[p].rs, v);
        } else {
            if(distr > pq.top()) query(e[p].rs, v);
            if(distl > pq.top()) query(e[p].ls, v);
        }
    }
} using namespace KDTree;
int main() {
    n = read(); k = read(); k *= 2;
    for(int i = 1; i <= k; i++) pq.push(0);
    for(int i = 1; i <= n; i++) a[i].x = read(), a[i].y = read();
    build(rt, 1, n);
    for(int i = 1; i <= n; i++) query(rt, a[i]);
    printf("%lld\n", pq.top());
    return 0;
}
```

4.8 笛卡尔树

```
const int MAXN = 2e5 + 5;
namespace LCT {
#define ls ch[0]
#define rs ch[1]
    struct Node {
        int ch[2], fa;
        int val, sum, rev;
    }e[MAXN];
    void upd(int p) {e[p].sum = e[e[p].ls].sum ^ e[e[p].rs].sum ^ e[p].val;}
    int nrt(int p) {return p == e[e[p].fa].ls || p == e[e[p].fa].rs;}
    int idy(int p) {return p == e[e[p].fa].rs;}
    void psdrev(int p) {if(p) swap(e[p].ls, e[p].rs), e[p].rev ^= 1;}
    void psd(int p) {if(e[p].rev) e[p].rev = 0, psdrev(e[p].ls), psdrev(e[p].rs);}
    void psdall(int p) {if(nrt(p)) psdall(e[p].fa);  psd(p);}
    void rtt(int p) {
        int y = e[p].fa, z = e[y].fa, k = idy(p), s = e[p].ch[k^1];
        e[p].fa = z; if(nrt(y)) e[z].ch[idy(y)] = p;
        e[s].fa = y; e[y].ch[k] = s;
        e[y].fa = p; e[p].ch[k^1] = y;
        upd(y); upd(p);
    }
    void splay(int p) {
        psdall(p);
        for(int y = e[p].fa; nrt(p); rtt(p), y = e[p].fa)
            if(nrt(y)) rtt(idy(p) == idy(y) ? y : p);
    }
    void access(int p) {for(int y = 0; p; y = p, p = e[p].fa) splay(p), e[p].rs = y,
        upd(p);}
    void mkrt(int p) {access(p); splay(p); psdrev(p);}
    int getrt(int p) {access(p); splay(p); while(e[p].ls) psd(p = e[p].ls); splay(p);
        return p;}
    void split(int x, int y) {mkrt(x); access(y); splay(y);}
    void link(int x, int y) {mkrt(x); if(getrt(y) != x) e[x].fa = y;}
    void cut(int x, int y) {mkrt(x); if(getrt(y) == x && e[y].fa == x && e[y].ls == 0)
        e[x].rs = e[y].fa = 0, upd(x);}
#undef ls
#undef rs
} using namespace LCT;
int n, m;
int main() {
    n = read(), m = read();
    for(int i = 1; i <= n; i++) e[i].val = read(), upd(i);
    for(int i = 1; i <= m; i++) {
        int opt = read(), x = read(), y = read();
        if(opt == 0) {
            split(x, y); printf("%d\n", e[y].sum);
        } else if(opt == 1) {
            link(x, y);
        } else if(opt == 2) {
            cut(x, y);
        } else {
            splay(x); e[x].val = y; upd(x);
        }
    }
    return 0;

}
```

5 字符串

5.1 KMP

```
const int MAXL = 1e6 + 5;
char s[MAXL], t[MAXL]; //t is format string
int _s, _t, nxt[MAXL];
int main() {
   scanf("%s%s", s+1, t+1);
   _s = strlen(s+1); _t = strlen(t+1);
   for(int i = 2, j = 0; i <= _t; i++) {
      while(j && t[i] != t[j+1]) j = nxt[j];
      if(t[i] == t[j+1]) j++;
      nxt[i] = j;
   }
   for(int i = 1, j = 0; i <= _s; i++) {
      while(j && s[i] != t[j+1]) j = nxt[j];
      if(s[i] == t[j+1]) j++;
      if(j == _t) {
         printf("%d\n", i - j + 1);
      }
   }
   for(int i = 1; i <= _t; i++) printf("%d ", nxt[i]);
   printf("\n");
   return 0;
}
```

5.2 AC 自动机

```
const int MAXN = 2e5 + 5;
const int MAXL = 2e5 + 5;
const int MAXT = 2e6 + 5;
int n, gt[MAXN];
char s[MAXL], t[MAXT];

int ch[MAXL][26], tot, fail[MAXL], val[MAXL];
int que[MAXL], hd, tl;
int head[MAXL], ver[MAXL], nxt[MAXL], cnt;//fail tree
void ins(char s[], int id) {
   int l = strlen(s + 1), now = 0;
   for(int i = 1; i <= l; i++) {
      int v = s[i] - 'a';
      if(!ch[now][v]) ch[now][v] = ++tot;
      now = ch[now][v];
   }
   gt[id] = now;
}
void addedge(int u, int v) {
   ver[++cnt] = v; nxt[cnt] = head[u]; head[u] = cnt;
}
void build() {
   hd = 1; tl = 0;
   for(int i = 0; i < 26; i++)
      if(ch[0][i]) fail[ch[0][i]] = 0, que[++tl] = ch[0][i];
   while(hd <= tl) {
      int u = que[hd++];
      for(int i = 0; i < 26; i++)
         if(ch[u][i]) fail[ch[u][i]] = ch[fail[u]][i], que[++tl] = ch[u][i];
         else ch[u][i] = ch[fail[u]][i];
   }
   for(int i = 1; i <= tot; i++) addedge(fail[i], i);
}
void qry(char t[]) {
   int l = strlen(t + 1), now = 0;
   for(int i = 1; i <= l; i++) {
      now = ch[now][t[i] - 'a'];
      val[now]++;
   }
}
void dfs(int u) {
    for(int i = head[u]; i; i = nxt[i]) {
        dfs(ver[i]);
        val[u] += val[ver[i]];
    }
}

int main() {
    n = read();
    for(int i = 1; i <= n; i++) {
        scanf("%s", s+1);
        ins(s, i);
    }
    build();
    scanf("%s", t+1);
    qry(t);
    dfs(0);
    for(int i = 1; i <= n; i++) printf("%d\n", val[gt[i]]);
    return 0;
}
```

5.3 SA

```
const int MAXN = 2e5 + 5;
int n;
char s[MAXN];
int sa[MAXN], rk[MAXN], x[MAXN], y[MAXN], ht[MAXN], c[MAXN];
ll ans;
void SuffixSort() {
    int m = 305;
    for(int i = 1; i <= n; i++) c[x[i] = s[i]]++;
    for(int i = 1; i <= m; i++) c[i] += c[i-1];
    for(int i = n; i >= 1; i--) sa[c[x[i]]--] = i;
    for(int k = 1; k <= n; k <<= 1) {
        int num = 0;
        for(int i = n - k + 1; i <= n; i++) y[++num] = i;
        for(int i = 1; i <= n; i++) if(sa[i] > k) y[++num] = sa[i] - k;
        for(int i = 1; i <= m; i++) c[i] = 0;
        for(int i = 1; i <= n; i++) c[x[i]]++;
        for(int i = 1; i <= m; i++) c[i] += c[i-1];
        for(int i = n; i >= 1; i--) sa[c[x[y[i]]]--] = y[i], y[i] = 0;
        for(int i = 1; i <= n; i++) swap(x[i], y[i]);
        x[sa[1]] = num = 1;
        for(int i = 2; i <= n; i++)
            x[sa[i]] = (y[sa[i]] == y[sa[i-1]] && y[sa[i] + k] == y[sa[i-1] + k]) ? num
                 : ++num;
        if(num == n) break;
        m = num;
    }
    for(int i = 1; i <= n; i++) rk[sa[i]] = i;
}
void GetHeight() {
    for(int i = 1, k = 0; i <= n; i++) {
        if(k) k--;
        if(rk[i] == 1) ht[rk[i]] = 0;
        int j = sa[rk[i] - 1];
        while(s[i + k] == s[j + k]) k++;
        ht[rk[i]] = k;
    }
}
int main() {
    n = read();
    scanf("%s", s+1);
    SuffixSort();
    GetHeight();
    for(int i = 1; i <= n; i++) ans += n - i + 1 - ht[rk[i]];
    printf("%lld\n", ans);
    return 0;
}
```

5.4 SAM

```
const int MAXN = 2e6 + 5;//the num of sam is 2*n
namespace SAM {
    struct Node{int ch[26], fa, len;}e[MAXN];
    int lst = 1, tot = 1, sz[MAXN];
    void ins(int c) {
        int p = lst, np = lst = ++tot; e[np].len = e[p].len + 1; sz[np]++;
        for(; p && !e[p].ch[c]; p = e[p].fa) e[p].ch[c] = np;
        if(!p) e[np].fa = 1;
        else {
            int q = e[p].ch[c];
            if(e[q].len == e[p].len + 1) e[np].fa = q;
            else {
                int nq = ++tot; e[nq] = e[q];
                e[nq].len = e[p].len + 1; e[q].fa = e[np].fa = nq;
                for(; p && e[p].ch[c] == q; p = e[p].fa) e[p].ch[c] = nq;
            }
        }
    }
    int head[MAXN], ver[MAXN], nxt[MAXN], cnt;
    void addedge(int u, int v) {
        ver[++cnt] = v; nxt[cnt] = head[u]; head[u] = cnt;
    }
    void build() {
        for(int i = 2; i <= tot; i++) addedge(e[i].fa, i);
    }
    void dfs(int u, ll& ans) {
        for(int i = head[u]; i; i = nxt[i]) {
            dfs(ver[i], ans); sz[u] += sz[ver[i]];
        }
        if(sz[u] > 1) ans = max(ans, 1ll * sz[u] * e[u].len);
    }
}
using namespace SAM;
char s[MAXN];
int n;
ll ans;
int main() {
    scanf("%s", s+1); n = strlen(s+1);
    for(int i = 1; i <= n; i++) ins(s[i] - 'a');
    build();
    dfs(1, ans);
    printf("%lld\n", ans);
    return 0;
}
```

5.5 manacher

```
const int MAXN = 2.2e7+5;
int n, r[MAXN], c;
char s[MAXN];
int main() {
    scanf("%s", s+1); n = strlen(s+1);
    for(int i = n; i >= 1; i--) s[2 * i + 1] = s[i], s[2 * i + 2] = '#';
    s[1] = '@'; s[2] = '#';
    n = 2 * n + 2;
    c = 0;
    for(int i = 1; i <= n; i++) {
        if(i < c + r[c]) r[i] = min(c + r[c] - i, r[2 * c - i]);
        while(i + r[i] <= n && s[i + r[i]] == s[i - r[i]]) r[i]++;
        if(i + r[i] > c + r[c]) c = i;
    }
    int ans = 0;
    for(int i = 1; i <= n; i++) ans = max(ans, r[i] - 1);
    printf("%d\n", ans);
    return 0;
}
```

6 树

6.1 LCA

6.1.1 DFS 序 +RMQ

```
int n, m, s;
struct Edge {int v, nxt;}e[MAXN*2];
int head[MAXN], cnt;
void addedge(int u, int v) {
  e[++cnt] = (Edge){v, head[u]}; head[u] = cnt;
}
int dep[MAXN], fa[MAXN], st[MAXN][19], dfn[MAXN], _dfn, lg2[MAXN];
void dfs(int u, int f) {
  dep[u] = dep[f] + 1; fa[u] = f; dfn[u] = ++_dfn;
  for(int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v; if(v == f) continue;
    dfs(v, u);
  }
}
int getbt(int u, int v) {
  return dep[u] < dep[v] ? u : v;
}
void init() {
  dfs(s, 0);
  lg2[0] = -1;
  for(int i = 1; i <= n; i++) lg2[i] = lg2[i >> 1] + 1;
  for(int i = 1; i <= n; i++) st[dfn[i]][0] = i;
  for(int k = 1; k <= 18; k++)
    for(int i = 1; i + (1 << k) - 1 <= n; i++)
      st[i][k] = getbt(st[i][k-1], st[i+(1<<(k-1))][k-1]);
}
int lca(int u, int v) {
  if(u == v) return u;
  u = dfn[u], v = dfn[v];
  if(u > v) swap(u, v);
  u++;
  int k = lg2[v - u + 1];
  return fa[getbt(st[u][k], st[v - (1 << k) + 1][k])];
}
int main() {
  #ifdef LOCAL
  freopen("main.in", "r", stdin);
  #endif
  scanf("%d%d%d", &n, &m, &s);
  for(int i = 1; i < n; i++) {
    int u, v; scanf("%d%d", &u, &v);
    addedge(u, v); addedge(v, u);
  }
  init();
  for(int i = 1; i <= m; i++) {
    int u, v;
    scanf("%d%d", &u, &v);
    printf("%d\n", lca(u, v));
  }
  return 0;
}
```

6.1.2 树链剖分

```
const int MAXN = 5e5 + 5;
const int MAXM = 5e5 + 5;
int n, m, s;
struct Edge {int v, nxt;}e[MAXN * 2];
int head[MAXN], cnt;
void addedge(int u, int v) {
  e[++cnt] = (Edge){v, head[u]}; head[u] = cnt;
}

int sz[MAXN], son[MAXN], top[MAXN], dep[MAXN], fa[MAXN];
void dfs1(int u, int f) {
  fa[u] = f; dep[u] = dep[f] + 1; sz[u] = 1;
  for(int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v; if(v == f) continue;
    dfs1(v, u);
    sz[u] += sz[v];
    if(sz[v] > sz[son[u]]) son[u] = v;
  }
}
void dfs2(int u, int tprt) {
  top[u] = tprt; if(son[u]) dfs2(son[u], tprt);
  for(int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v; if(v == fa[u] || v == son[u]) continue;
    dfs2(v, v);
  }
}
```

```
int lca(int u, int v) {
  while(top[u] != top[v]) {
    if(dep[top[u]] < dep[top[v]]) swap(u, v);
    u = fa[top[u]];
  }
  return dep[u] < dep[v] ? u : v;
}
int main() {
  #ifdef LOCAL
  freopen("main.in", "r", stdin);
  #endif
  scanf("%d%d%d", &n, &m, &s);
  for(int i = 1; i < n; i++) {
    int u, v; scanf("%d%d", &u, &v);
    addedge(u, v); addedge(v, u);
  }
  dfs1(s, 0);
  dfs2(s, s);
  for(int i = 1; i <= m; i++) {
    int u, v; scanf("%d%d", &u, &v);
    printf("%d\n", lca(u, v));
  }
  return 0;
}
```

6.2 树链剖分

```
int n, m, rt;
ll w[MAXN];
struct Edge {
   int v, nxt;
}e[MAXN << 1];
int head[MAXN], cnt;
void addedge(int u, int v) {
   e[++cnt].v = v; e[cnt].nxt = head[u]; head[u] = cnt;
}


//Treecut
int fa[MAXN], son[MAXN], sz[MAXN], dep[MAXN];
void dfs1(int u, int f) {
   fa[u] = f; son[u] = 0; sz[u] = 1; dep[u] = dep[f] + 1; int mxsz = -1;
   for(int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v; if(v == f) continue;
      dfs1(v, u);
      sz[u] += sz[v];
      if(sz[v] > mxsz) mxsz = sz[v], son[u] = v;
   }
}
int top[MAXN], dfn[MAXN], tim;
ll wd[MAXN];
void dfs2(int u, int tprt) {
   top[u] = tprt; dfn[u] = ++tim; wd[tim] = w[u];
   if(son[u]) dfs2(son[u], tprt);
   for(int i = head[u]; i; i = e[i].nxt) {
      int v = e[i].v; if(v == fa[u] || v == son[u]) continue;
      dfs2(v, v);
   }
}

//segment_Tree

#define ls o << 1
#define rs o << 1 | 1
ll val[MAXN << 2], la[MAXN << 2];
void pushup(int o) {val[o] = pls(val[ls], val[rs]);}
void build(int o, int l, int r, ll a[]) {
   la[o] = 0; if(l == r) {val[o] = a[l]; return ;}
   int m = (l + r) >> 1; build(ls, l, m, a); build(rs, m+1, r, a);
   pushup(o);
}
void addpoint(int o, int l, int r, ll k) {
   val[o] = pls(val[o], mul(r-l+1, k)); la[o] = pls(la[o], k);
}
void pushdown(int o, int l, int r) {
   if(!la[o]) return ; int m = (l + r) >> 1;
   addpoint(ls, l, m, la[o]); addpoint(rs, m+1, r, la[o]);
   la[o] = 0;
}
void addrange(int o, int l, int r, int x, int y, ll k) {
   if(x <= l && r <= y) {addpoint(o, l, r, k); return ;}
   pushdown(o, l, r); int m = (l + r) >> 1;
   if(x <= m) addrange(ls, l, m, x, y, k);
   if(y > m) addrange(rs, m+1, r, x, y, k);
   pushup(o);
}
ll query(int o, int l, int r, int x, int y) {
   if(x <= l && r <= y) return val[o];
   pushdown(o, l, r); int m = (l + r) >> 1;
   if(y <= m) return query(ls, l, m, x, y);
   else if(x > m) return query(rs, m+1, r, x, y);
```

```
    else return pls(query(ls, l, m, x, y), query(rs, m+1, r, x, y));
}
#undef ls
#undef rs
int main() {
    n = read(); m = read(); rt = read(); P = read();
    for(int i = 1; i <= n; i++) w[i] = read();
    for(int i = 1; i < n; i++) {
        int u = read(), v = read();
        addedge(u, v); addedge(v, u);
    }
    dfs1(rt, 0); dfs2(rt, rt); build(1, 1, n, wd);
    while(m--) {
        int opt = read();
        if(opt == 1) {
            int x = read(), y = read(); ll z = read();
            while(top[x] != top[y]) {
                if(dep[top[x]] < dep[top[y]]) swap(x, y);
                addrange(1, 1, n, dfn[top[x]], dfn[x], z);
                x = fa[top[x]];
            }
            if(dep[x] < dep[y]) swap(x, y);
            addrange(1, 1, n, dfn[y], dfn[x], z);
        } else if(opt == 2) {
            int x = read(), y = read(); ll ans = 0;
            while(top[x] != top[y]) {
                if(dep[top[x]] < dep[top[y]]) swap(x, y);
                ans = pls(ans, query(1, 1, n, dfn[top[x]], dfn[x]));
                x = fa[top[x]];
            }
            if(dep[x] < dep[y]) swap(x, y);
            ans = pls(ans, query(1, 1, n, dfn[y], dfn[x]));
            printf("%lld\n", ans);
        } else if(opt == 3) {
            int x = read(); ll z = read();
            addrange(1, 1, n, dfn[x], dfn[x] + sz[x] - 1, z);
        } else {
            int x = read();
            printf("%lld\n", query(1, 1, n, dfn[x], dfn[x] + sz[x] - 1));
        }
    }

    return 0;
}
```

6.3 长链剖分

```
#include<algorithm>
#include<cstdio>
#include<queue>
using namespace std;
typedef long long ll;
typedef double db;
ll read() {
   ll x = 0, f = 1; char ch = getchar();
   for(; ch < '0' || ch > '9'; ch = getchar()) if(ch == '-') f = -1;
   for(; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + int(ch - '0');
   return x * f;
}
const int MAXN = 5005, MAXM = 4e5 + 5;
const db inf = 1e40, eps = 1e-8;
int dcmp(db x) {return x < -eps ? -1 : (x > eps ? 1 : 0);}
int n, m;
db E;
namespace Graph {
   struct Edge {
      int v, nxt;
      db w;
   }e[MAXM];
   int head[MAXN], cnt = 1; // 0 is positive, 1 is negative (mod 2)
   void addedge(int u, int v, db w) {
      e[++cnt] = (Edge){v, head[u], w}; head[u] = cnt;
      e[++cnt] = (Edge){u, head[v], w}; head[v] = cnt;
   }
} using namespace Graph;
namespace SP {
   struct QNode {
      db d; int u;
   };
   bool operator < (const QNode& a, const QNode& b) {return a.d > b.d;}
   priority_queue<QNode> que;
   db dist[MAXN];
   int vis[MAXN], pre[MAXN];
   void dijkstra() {
      for(int i = 1; i <= n; i++) dist[i] = inf, vis[i] = pre[i] = 0;
      dist[n] = 0; que.push((QNode){0.0, n});
      while(que.size()) {
         int u = que.top().u; que.pop();
         if(vis[u]) continue;
         vis[u] = 1;
         for(int i = head[u]; i; i = e[i].nxt) if(i & 1) {//neg
```

```
                int v = e[i].v;
                if(dcmp(dist[v] - (dist[u] + e[i].w)) > 0) {
                    dist[v] = dist[u] + e[i].w; pre[v] = i; que.push((QNode){dist[v], v})
                        ;
                }
            }
        }
    }
} using namespace SP;
namespace LeftyTree {
    struct Node {
        int ls, rs, dist, v; db val;
    }h[MAXM << 5];
    int tot;
    int crenode(int v, db val) {
        h[++tot] = (Node){0, 0, 1, v, val}; return tot;
    }
    int merge(int x, int y) {
        if(!x || !y) return x + y;
        if(dcmp(h[x].val - h[y].val) > 0) swap(x, y);
        int p = ++tot; h[p] = h[x];
        h[p].rs = merge(h[x].rs, y);
        if(h[h[p].ls].dist < h[h[p].rs].dist) swap(h[p].ls, h[p].rs);
        h[p].dist = h[h[p].rs].dist + 1;
        return p;
    }
    void ins(int& p, int v, db val) {
        p = merge(p, crenode(v, val));
    }
} using namespace LeftyTree;
int seq[MAXN], rt[MAXN];
bool cmp(int a, int b) {return dist[a] < dist[b];}
struct QN {
    db d; int u;
};
bool operator < (const QN& a, const QN& b) {return a.d > b.d;}
priority_queue<QN> pq;
int main() {
    n = read(), m = read(); scanf("%lf", &E);
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read(); db w; scanf("%lf", &w);
        if(u == n) continue;
        addedge(u, v, w);
    }
    dijkstra();
    for(int i = 1; i <= n; i++) seq[i] = i;
    sort(seq + 1, seq + 1 + n, cmp);
    for(int k = 1; k <= n; k++) {
        int u = seq[k];
        if(pre[u]) rt[u] = rt[e[pre[u]^1].v];
        for(int i = head[u]; i; i = e[i].nxt) if(~i & 1 && i != (pre[u]^1)) {
            int v = e[i].v; db delta = e[i].w - (dist[u] - dist[v]);
            ins(rt[u], v, delta);
        }
    }
    int ans = 0;
    if(dcmp(E-dist[1]) >= 0) ans++, E -= dist[1];
    if(rt[1]) pq.push((QN){h[rt[1]].val, rt[1]});
    while(pq.size()) {
        int u = pq.top().u; db ld = pq.top().d; pq.pop();
        //now sigma = dist[1] - ld
        if(dcmp(E-dist[1]-ld) < 0) break;
        E -= dist[1] + ld; ans++;
        int nxt = rt[h[u].v];
        if(nxt) pq.push((QN){ld + h[nxt].val, nxt});
        if(h[u].ls) pq.push((QN){ld - h[u].val + h[h[u].ls].val, h[u].ls});
        if(h[u].rs) pq.push((QN){ld - h[u].val + h[h[u].rs].val, h[u].rs});
    }
    printf("%d\n", ans);
    return 0;
}
```

6.4 点分治

```
const int MAXN = 1e4 + 5;
const int MAXM = 105;
const int MAXV = 1e7 + 5;
const int INF = 0x3f3f3f3f;
int n, m, head[MAXN], ver[MAXN << 1], nxt[MAXN << 1], edg[MAXN << 1], cnt, mxk, ques[
     MAXM], ans[MAXM], tsz, sz[MAXN], mxsz[MAXN], rt, vis[MAXN], judge[MAXV], tmp[MAXN
     ], tnum, dis[MAXN];
void addedge(int u, int v, int w) {
    ver[++cnt] = v; nxt[cnt] = head[u]; edg[cnt] = w; head[u] = cnt;
}
void getrt(int u, int f) {
    sz[u] = 1; mxsz[u] = 0;
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(vis[v] || v == f) continue;
        getrt(v, u); sz[u] += sz[v]; mxsz[u] = max(mxsz[u], sz[v]);
    }
    mxsz[u] = max(mxsz[u], tsz - sz[u]);
    if(mxsz[u] < mxsz[rt]) rt = u;
}
void getdis(int u, int f) {
    if(dis[u] <= mxk) tmp[++tnum] = dis[u];
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(v == f || vis[v]) continue;
        dis[v] = dis[u] + edg[i];
        getdis(v, u);
    }
}
void calc(int u) {
    static int que[MAXN];
    judge[0] = 1;
    int tl = 0;
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(vis[v]) continue;
        tnum = 0; dis[v] = edg[i];
        getdis(v, u);
        for(int j = 1; j <= m; j++)
            for(int k = 1; k <= tnum; k++)
                if(ques[j] >= tmp[k]) ans[j] |= judge[ques[j] - tmp[k]];
        for(int j = 1; j <= tnum; j++)
            que[++tl] = tmp[j], judge[tmp[j]] = 1;
    }
    for(int i = 1; i <= tl; i++) judge[que[i]] = 0;
}
void solve(int u) {
    vis[u] = 1; calc(u);
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(vis[v]) continue;
        tsz = sz[v]; mxsz[rt = 0] = INF;
        getrt(v, u); solve(rt);
    }
}
int main() {
    //freopen("code.in", "r", stdin);
    //freopen("code.out", "w", stdout);
    n = read(), m = read();
    for(int i = 1; i < n; i++) {
        int u = read(), v = read(), w = read();
        addedge(u, v, w); addedge(v, u, w);
    }
    for(int i = 1; i <= m; i++) ques[i] = read(), mxk = max(mxk, ques[i]);
    tsz = n; mxsz[rt = 0] = INF;
    getrt(1, 0); solve(rt);
    for(int i = 1; i <= m; i++) printf(ans[i] ? "AYE\n" : "NAY\n");
    return 0;
}
```

6.5 点分树

```
const int MAXN = 2e5 + 5, MAXV = 1e5 + 5;
int n, m, val[MAXN], head[MAXN], ver[MAXN], nxt[MAXN], cnt, tim, st[MAXN][18], bg[MAXN
     ], lg2[MAXN], dep[MAXN], vis[MAXN], rt, tsz, sz[MAXN], mxsz[MAXN], anc[MAXN], sgt
     [2][MAXN], lans;
namespace Sgt {
    const int MAXNODE = MAXN << 5;
    int tot;
    struct Node{int ls, rs, sum;}e[MAXNODE];
    void upd(int p) {e[p].sum = e[e[p].ls].sum + e[e[p].rs].sum;}
    void addpos(int& p, int l, int r, int x, int v) {
        if(!p) p = ++tot;
        if(l == r) {e[p].sum += v; return ;}
        int m = (l + r) >> 1;
        if(x <= m) addpos(e[p].ls, l, m, x, v);
        else addpos(e[p].rs, m+1, r, x, v);
        upd(p);
    }
    int qryrange(int p, int l, int r, int x, int y) {
        if(!p) return 0;
        if(x <= l && r <= y) return e[p].sum;
        int m = (l + r) >> 1;
        if(y <= m) return qryrange(e[p].ls, l, m, x, y);
        else if(x > m) return qryrange(e[p].rs, m+1, r, x, y);
        else return qryrange(e[p].ls, l, m, x, m) + qryrange(e[p].rs, m+1, r, m+1, y);
    }
}
void addedge(int u, int v) {
    ver[++cnt] = v; nxt[cnt] = head[u]; head[u] = cnt;
}
void dfs1(int u, int f) {
    dep[u] = dep[f] + 1; st[++tim][0] = u; bg[u] = tim;
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(v == f) continue;
        dfs1(v, u);
        st[++tim][0] = u;
    }
}
int NDmax(int u, int v) {return dep[u] < dep[v] ? u : v;}
void getst() {
    for(int k = 1; (1 << k) <= tim; k++)
```

```
        for(int i = 1; i + (1 << k) - 1 <= tim; i++)
            st[i][k] = NDmax(st[i][k-1], st[i + (1 << (k-1))][k-1]);
    lg2[0] = -1;
    for(int i = 1; i <= tim; i++) lg2[i] = lg2[i >> 1] + 1;
}
int Lca(int u, int v) {
    int a = bg[u], b = bg[v]; if(a > b) swap(a, b);
    int k = lg2[b - a + 1];
    return NDmax(st[a][k], st[b - (1 << k) + 1][k]);
}
int Dist(int u, int v) {return dep[u] + dep[v] - 2 * dep[Lca(u, v)];}
void getrt(int u, int f) {
    sz[u] = 1; mxsz[u] = 0;
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(v == f || vis[v]) continue;
        getrt(v, u); sz[u] += sz[v]; mxsz[u] = max(mxsz[u], sz[v]);
    }
    mxsz[u] = max(mxsz[u], tsz - sz[u]);
    if(mxsz[u] < mxsz[rt]) rt = u;
}
void divide(int u) {
    vis[u] = 1;
    for(int i = head[u]; i; i = nxt[i]) {
        int v = ver[i]; if(vis[v]) continue;
        tsz = sz[v]; mxsz[rt = 0] = MAXN;
        getrt(v, 0);
        sz[rt] = sz[v]; anc[rt] = u;
        divide(rt);
    }
}
void modify(int u, int w) {
    for(int i = u; i; i = anc[i]) Sgt::addpos(sgt[0][i], 0, sz[i], Dist(i, u), w);
    for(int i = u; anc[i]; i = anc[i]) Sgt::addpos(sgt[1][i], 0, sz[anc[i]], Dist(anc[
         i], u), w);
}
int query(int u, int k) {
    int ans = 0;
    ans += Sgt::qryrange(sgt[0][u], 0, sz[u], 0, min(sz[u], k));
    for(int i = u; anc[i]; i = anc[i]) {
        int d = Dist(anc[i], u);
        if(k >= d) ans += Sgt::qryrange(sgt[0][anc[i]], 0, sz[anc[i]], 0, min(k - d, sz
             [anc[i]]))
                      - Sgt::qryrange(sgt[1][i], 0, sz[anc[i]], 0, min(k - d, sz[anc[i
                           ]]));
    }
    return ans;
```

```
}
int main() {
   n = read(), m = read();
   for(int i = 1; i <= n; i++) val[i] = read();
   for(int i = 1; i < n; i++) {
      int u = read(), v = read();
      addedge(u, v); addedge(v, u);
   }
   dfs1(1, 0); getst();
   tsz = n; mxsz[rt = 0] = MAXN;
   getrt(1, 0);
   sz[rt] = sz[1]; divide(rt);
   for(int i = 1; i <= n; i++) modify(i, val[i]);
   lans = 0;
   for(int i = 1; i <= m; i++) {
      int opt = read(), x = read() ^ lans, y = read() ^ lans;
      if(opt == 0) {
         printf("%d\n", lans = query(x, y));
      } else {
         modify(x, y - val[x]); val[x] = y;
      }
   }
   return 0;
}
```

6.6 LCT

```
const int MAXN = 2e5 + 5;
namespace LCT {
#define ls ch[0]
#define rs ch[1]
   struct Node {
      int ch[2], fa;
      int val, sum, rev;
   }e[MAXN];
   void upd(int p) {e[p].sum = e[e[p].ls].sum ^ e[e[p].rs].sum ^ e[p].val;}
   int nrt(int p) {return p == e[e[p].fa].ls || p == e[e[p].fa].rs;}
   int idy(int p) {return p == e[e[p].fa].rs;}
   void psdrev(int p) {if(p) swap(e[p].ls, e[p].rs), e[p].rev ^= 1;}
   void psd(int p) {if(e[p].rev) e[p].rev = 0, psdrev(e[p].ls), psdrev(e[p].rs);}
   void psdall(int p) {if(nrt(p)) psdall(e[p].fa);  psd(p);}
   void rtt(int p) {
      int y = e[p].fa, z = e[y].fa, k = idy(p), s = e[p].ch[k^1];
      e[p].fa = z; if(nrt(y)) e[z].ch[idy(y)] = p;
      e[s].fa = y; e[y].ch[k] = s;
```

```
        e[y].fa = p; e[p].ch[k^1] = y;
        upd(y); upd(p);
    }
    void splay(int p) {
        psdall(p);
        for(int y = e[p].fa; nrt(p); rtt(p), y = e[p].fa)
            if(nrt(y)) rtt(idy(p) == idy(y) ? y : p);
    }
    void access(int p) {for(int y = 0; p; y = p, p = e[p].fa) splay(p), e[p].rs = y,
        upd(p);}
    void mkrt(int p) {access(p); splay(p); psdrev(p);}
    int getrt(int p) {access(p); splay(p); while(e[p].ls) psd(p = e[p].ls); splay(p);
        return p;}
    void split(int x, int y) {mkrt(x); access(y); splay(y);}
    void link(int x, int y) {mkrt(x); if(getrt(y) != x) e[x].fa = y;}
    void cut(int x, int y) {mkrt(x); if(getrt(y) == x && e[y].fa == x && e[y].ls == 0)
         e[x].rs = e[y].fa = 0, upd(x);}
#undef ls
#undef rs
} using namespace LCT;
int n, m;
int main() {
    n = read(), m = read();
    for(int i = 1; i <= n; i++) e[i].val = read(), upd(i);
    for(int i = 1; i <= m; i++) {
        int opt = read(), x = read(), y = read();
        if(opt == 0) {
            split(x, y); printf("%d\n", e[y].sum);
        } else if(opt == 1) {
            link(x, y);
        } else if(opt == 2) {
            cut(x, y);
        } else {
            splay(x); e[x].val = y; upd(x);
        }
    }
    return 0;

}
```

6.7 虚树

```
#include<algorithm>
#include<cstdio>
#include<queue>
```

```
using namespace std;
typedef long long ll;
typedef double db;
ll read() {
   ll x = 0, f = 1; char ch = getchar();
   for(; ch < '0' || ch > '9'; ch = getchar()) if(ch == '-') f = -1;
   for(; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + int(ch - '0');
   return x * f;
}
const int MAXN = 5005, MAXM = 4e5 + 5;
const db inf = 1e40, eps = 1e-8;
int dcmp(db x) {return x < -eps ? -1 : (x > eps ? 1 : 0);}
int n, m;
db E;
namespace Graph {
   struct Edge {
      int v, nxt;
      db w;
   }e[MAXM];
   int head[MAXN], cnt = 1; // 0 is positive, 1 is negative (mod 2)
   void addedge(int u, int v, db w) {
      e[++cnt] = (Edge){v, head[u], w}; head[u] = cnt;
      e[++cnt] = (Edge){u, head[v], w}; head[v] = cnt;
   }
} using namespace Graph;
namespace SP {
   struct QNode {
      db d; int u;
   };
   bool operator < (const QNode& a, const QNode& b) {return a.d > b.d;}
   priority_queue<QNode> que;
   db dist[MAXN];
   int vis[MAXN], pre[MAXN];
   void dijkstra() {
      for(int i = 1; i <= n; i++) dist[i] = inf, vis[i] = pre[i] = 0;
      dist[n] = 0; que.push((QNode){0.0, n});
      while(que.size()) {
         int u = que.top().u; que.pop();
         if(vis[u]) continue;
         vis[u] = 1;
         for(int i = head[u]; i; i = e[i].nxt) if(i & 1) {//neg
            int v = e[i].v;
            if(dcmp(dist[v] - (dist[u] + e[i].w)) > 0) {
               dist[v] = dist[u] + e[i].w; pre[v] = i; que.push((QNode){dist[v], v})
                   ;
            }
            }
        }
    }
} using namespace SP;
namespace LeftyTree {
    struct Node {
        int ls, rs, dist, v; db val;
    }h[MAXM << 5];
    int tot;
    int crenode(int v, db val) {
        h[++tot] = (Node){0, 0, 1, v, val}; return tot;
    }
    int merge(int x, int y) {
        if(!x || !y) return x + y;
        if(dcmp(h[x].val - h[y].val) > 0) swap(x, y);
        int p = ++tot; h[p] = h[x];
        h[p].rs = merge(h[x].rs, y);
        if(h[h[p].ls].dist < h[h[p].rs].dist) swap(h[p].ls, h[p].rs);
        h[p].dist = h[h[p].rs].dist + 1;
        return p;
    }
    void ins(int& p, int v, db val) {
        p = merge(p, crenode(v, val));
    }
} using namespace LeftyTree;
int seq[MAXN], rt[MAXN];
bool cmp(int a, int b) {return dist[a] < dist[b];}
struct QN {
    db d; int u;
};
bool operator < (const QN& a, const QN& b) {return a.d > b.d;}
priority_queue<QN> pq;
int main() {
    n = read(), m = read(); scanf("%lf", &E);
    for(int i = 1; i <= m; i++) {
        int u = read(), v = read(); db w; scanf("%lf", &w);
        if(u == n) continue;
        addedge(u, v, w);
    }
    dijkstra();
    for(int i = 1; i <= n; i++) seq[i] = i;
    sort(seq + 1, seq + 1 + n, cmp);
    for(int k = 1; k <= n; k++) {
        int u = seq[k];
        if(pre[u]) rt[u] = rt[e[pre[u]^1].v];
        for(int i = head[u]; i; i = e[i].nxt) if(~i & 1 && i != (pre[u]^1)) {
            int v = e[i].v; db delta = e[i].w - (dist[u] - dist[v]);
            ins(rt[u], v, delta);
         }
      }
   int ans = 0;
   if(dcmp(E-dist[1]) >= 0) ans++, E -= dist[1];
   if(rt[1]) pq.push((QN){h[rt[1]].val, rt[1]});
   while(pq.size()) {
      int u = pq.top().u; db ld = pq.top().d; pq.pop();
      //now sigma = dist[1] - ld
      if(dcmp(E-dist[1]-ld) < 0) break;
      E -= dist[1] + ld; ans++;
      int nxt = rt[h[u].v];
      if(nxt) pq.push((QN){ld + h[nxt].val, nxt});
      if(h[u].ls) pq.push((QN){ld - h[u].val + h[h[u].ls].val, h[u].ls});
      if(h[u].rs) pq.push((QN){ld - h[u].val + h[h[u].rs].val, h[u].rs});
   }
   printf("%d\n", ans);
   return 0;
}
```

7 计算几何

7.1 基础模板

```
const db eps = 1e-8, inf = 1e10;
int dcmp(db x) {return x < -eps ? -1 : (x > eps ? 1 : 0);}
db Abs(db x) {return x * dcmp(x);}
struct Pnt {
   db x, y;
   Pnt(db x = 0, db y = 0) : x(x), y(y) {}
};
typedef Pnt Vec;
db Dot(const Vec& a, const Vec& b) {return a.x * b.x + a.y * b.y;}
db Cro(const Vec& a, const Vec& b) {return a.x * b.y - a.y * b.x;}
db Len(const Vec& a) {return sqrt(Dot(a, a));}
Vec operator + (const Vec& a, const Vec& b) {return Vec(a.x + b.x, a.y + b.y);}
Vec operator - (const Vec& a, const Vec& b) {return Vec(a.x - b.x, a.y - b.y);}
Vec operator * (const Vec& a, const db& b) {return Vec(a.x * b, a.y * b);}
bool operator == (const Pnt& a, const Pnt& b) {return !dcmp(a.x - b.x) && !dcmp(a.y -
     b.y);}
db Angle(const Vec& a, const Vec& b) {return acos(Dot(a, b) / Len(a) / Len(b));}
Pnt Turn(const Pnt& p, const db& rad) {return Pnt(p.x * cos(rad) - p.y * sin(rad), p.x
      * sin(rad) + p.y * cos(rad));}
```

```
typedef pair<Pnt, Pnt> Line;
typedef pair<Pnt, Pnt> Seg;
bool isPointonLine(const Pnt& p, const Pnt& a, const Pnt& b) {
    return !dcmp(Cro(p-a, b-a));
}
Pnt FootPoint(const Pnt& p, const Pnt& a, const Pnt& b) {
    Vec ab = b-a, ap = p-a;
    return a + ab * (Dot(ap, ab) / Dot(ab, ab));
}
db DistPointLine(const Pnt& p, const Pnt& a, const Pnt& b) {
    return Cro(p-a, p-b) / Len(b-a);
}
Pnt Reflect(const Pnt& p, const Pnt& a, const Pnt& b) {
    return p + (FootPoint(p, a, b)-p) * 2;
}
bool isPointonSeg(const Pnt& p, const Pnt& a, const Pnt& b) {
    return !dcmp(Cro(p-a, b-a)) && dcmp(Dot(p-a,p-b)) <= 0;
}
db DistPointSeg(const Pnt& a, const Pnt& b) {
    if(a == b) return a;
    Vec ap = p-a, bp = p-b, ab = b-a;
    if(dcmp(Dot(ap, ab)) <= 0) return Len(ap);
    if(dcmp(Dot(bp, ab)) >= 0) return Len(bp);
    return Abs(Cro(ap, bp) / Len(ab));
}
Pnt CrossPoint(const Pnt& a, const Pnt& b, const Pnt& c, const Pnt& d) {
    Vec ab = b-a, cd = d-c, ca = a-c;
    return a + ab * (Cro(cd, ca) / Cro(ab, cd));
}
bool isCrossLineSeg(const Pnt& a, const Pnt& b, const Pnt& c, const Pnt& d) {
    Pnt p = CrossPoint(a, b, c, d);
    return isPointonSeg(p, c, d);
}
bool isCrossSegSeg(const Pnt& a, const Pnt& b, const Pnt& c, const Pnt& d) {
    if(max(a.x, b.x) < min(c.x, d.x) || max(c.x, d.x) < min(a.x, b.x) || max(a.y, b.y)
          < min(c.y, d.y) || max(c.y, d.y) < min(a.y, b.y)) return 0;
    if(dcmp(Cro(c-b, a-b)) * dcmp(Cro(d-b, a-b)) > 0 || dcmp(Cro(a-c, d-c) * dcmp(b-c,
          d-c)) > 0) return 0;
    return 1;
}
typedef vector<Pnt> Poly;
int PIP(const Pnt& p, const Poly& G) {
    int cnt = 0;
    int n = G.size();
    int n = G.size();
    if(n == 1) return p == G[0] ? 2 : 0;
    else if(n == 2) return isPointonSeg(p, G[0], G[1]) ? 2 : 0;
    for(int i = 0; i < n; i++) {
        int j = (i+1) % n;
        if(isPointonSeg(p, G[i], G[j])) return 2;
        if(p.y >= min(G[i].y, G[j].y) && p.y < max(G[i].y, G[j].y)) {
            db tmp = G[i].x + (p.y - G[i].y) / (G[j].y - G[i].y) * (G[j].x - G[i].x);
            if(dcmp(tmp - p.x) > 0) cnt++;
        }
    }
    return cnt & 1;
}
/*
返回 1: 在多边形内
返回 2: 在多边形上
返回 3: 在多边形外。
*/
int PIP(const Pnt& p, const Poly& G) {//顺
    int n = G.size();
    if(n == 1) return p == G[0] ? 2 : 0;
    else if(n == 2) return isPointonSeg(p, G[0], G[1]) ? 2 : 0;
    if(Cro(p-G[0], G[1]-G[0]) < 0 || Cro(p-G[0], G[n-1]-G[0]) > 0) return 0;
    if(isPointonSeg(p, G[0], G[1]) || isPointonSeg(p, G[0], G[n-1])) return 2;
    int l = 1, r = n-2, t = -1;
    while(l <= r) {
        int m = (l + r) >> 1;
        if(Cro(p-G[0], G[m]-G[0]) > 0) t = m, l = m + 1;
        else r = m - 1;
    }
    if(t == -1) return 0;
    if(Cro(p-G[t], G[t+1]-G[t]) < 0) return 0;
    if(isPointonSeg(p, G[t], G[t+1])) return 2;
    return 1;
}
db Circum(const Poly& G) {
    db ans = 0;
    int n = G.size();
    for(int i = 0; i < n; i++) {
        int j = (i+1) % n;
        ans += Len(G[j]-G[i]);
    }
    return ans;
}
db PolyArea(const Poly& G) {
    db ans = 0;
    int n = G.size();
    for(int i = 0; i < n; i++) {
        int j = (i+1) % n;
        ans += Cro(G[i], G[j]);
    }
    return Abs(ans / 2);
}
```

7.2 凸包

```
//Graham 扫描法
bool cmp(const Pnt& a, const Pnt& b) {return dcmp(a.x - b.x) == 0 ? a.y < b.y : a.x <
     b.x;}
Poly ConvexHull(Poly G) {//顺
    static int st[MAXN];
    int _st = 0;
    sort(G.begin(), G.end(), cmp);
    unique(G.begin(), G.end());
    if(G.size() <= 2) return G;
    int n = G.size();
    st[++_st] = 0; st[++_st] = 1;
    for(int i = 2; i < n; i++) {
        while(_st > 1 && Cro(G[st[_st]]-G[st[_st-1]], G[i]-G[st[_st]]) >= 0) _st--;
        st[++_st] = i;
    }
    int t = _st; st[++_st] = n-2;
    for(int i = n-3; i >= 0; i--) {
        while(_st > t && Cro(G[st[_st]]-G[st[_st-1]], G[i]-G[st[_st]]) >= 0) _st--;
        st[++_st] = i;
    }
    Poly T;
    for(int i = 1; i < _st; i++) T.push_back(G[st[i]]);
    return T;
}
```

7.3 旋转卡壳

```
db Getmax(const Poly& G) {//逆
    int n = G.size();
    if(n == 1) return 0;
    else if(n == 2) return Len(G[1]-G[0]);
    db ans = Len(G[1]-G[0]);
    for(int i = 0, j = 2; i < n; i++) {
        int i2 = (i+1) % n, j2 = (j+1) % n;
        while(dcmp(Cro(G[i2]-G[i], G[j]-G[i]) - Cro(G[i2]-G[i], G[j2]-G[i])) < 0) j =
             j2, j2 = (j2+1) % n;
        ans = max(ans, max(Len(G[j]-G[i]), Len(G[j]-G[i2])));
    }
    return ans;
}
```

7.4 半平面交

```
int dir(const Pnt& p, const Pnt& a, const Pnt& b) {return dcmp(Cro(p - a, b - a));}
struct Line {
    Pnt a, b;
    db k;
    Line(Pnt A = Pnt(0, 0), Pnt B = Pnt(0, 0)) {
        a = A; b = B; k = atan2(b.y - a.y, b.x - a.x);
    }
};
int dir(const Pnt& p, const Line& l) {return dir(p, l.a, l.b);}
bool operator < (const Line& p, const Line& q) {
    return dcmp(p.k - q.k) == 0 ? dir(p.a, q) == -1 : p.k < q.k;
}
Pnt CrossPoint (const Pnt& a, const Pnt& b, const Pnt& c, const Pnt& d) {
    Vec ab = b-a, cd = d-c, ca = a-c;
    return a + ab * (Cro(cd, ca) / Cro(ab, cd));
}
Pnt CrossPoint(const Line& p, const Line& q) {return CrossPoint(p.a, p.b, q.a, q.b);}
Poly Halfplanecut(vector<Line> G) {
    sort(G.begin(), G.end());
    int n = 0;
    for(int i = 1; i < (int)G.size(); i++)
        if(dcmp(G[i].k - G[i-1].k)) G[++n] = G[i];
    n++; G.resize(n);
    static int que[MAXN];
    int hd = 1, tl = 0;
    for(int i = 0; i < n; i++) {
        while(hd < tl && dir(CrossPoint(G[que[tl]], G[que[tl-1]]), G[i]) > 0) --tl;
        while(hd < tl && dir(CrossPoint(G[que[hd]], G[que[hd+1]]), G[i]) > 0) ++hd;
        que[++tl] = i;
    }
    while(hd < tl && dir(CrossPoint(G[que[tl]], G[que[tl-1]]), G[que[hd]]) > 0) --tl;
    while(hd < tl && dir(CrossPoint(G[que[hd]], G[que[hd+1]]), G[que[tl]]) > 0) ++hd;
    Poly ans;
    for(int i = hd; i <= tl; i++) {
        int j = i == tl ? hd : i + 1;
        ans.push_back(CrossPoint(G[que[i]], G[que[j]]));
    }
    return ans;//逆
}
```