University of Tartu Team "Bubblegum Bitset" Notebook (2024-2025)

November 16, 2025

Maxflow Complexity

$\mathcal{O}(V^2E)$ – Dinic

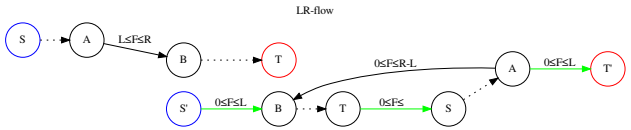
$\Theta(VE \log U)$ – Capacity scaling

$\Theta(\textbf{flow}E)$ – Small flow

$\Theta(\min\{V^{\frac{2}{3}}, E^{\frac{1}{2}}\}E)$ – Unitary capacities

$\Theta(\sqrt{V}E)$ – Each vertex other than S,T has only a single incoming unitary edge or outgoing one (bipartite matching)

$\Theta(\textbf{flow}E \log V)$ – Min-cost-max flow



Min Rotation of string

```
int a=0, N=s.size();
s += s;
ran(b,0,N){
  ran(i,0,N) {
    if (a+i == b || s[a+i] < s[b+i]) {
      b += max(0, i-1);
      break;
    }
    if (s[a+i] > s[b+i]) {
      a = b;
      break;
    }
  }
}
return a;
```

Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots, (-\infty < x < \infty)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots, (-1 < x \leq 1)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \ldots, (-1 \leq x \leq 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots, (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots, (-\infty < x < \infty)$$

Pythagorean Triples

The Pythagorean triples are uniquely generated by

$$a = k \cdot (m^2 - n^2), \;\; b = k \cdot (2mn), \;\; c = k \cdot (m^2 + n^2),$$

with $m > n > 0$, $k > 0$, $m \perp n$, and either m or n even.

Primes

$p = 962592769$ is such that $2^{21} \mid p - 1$, which may be useful. For hashing use 970592641 (31-bit number), 31443539979727 (45-bit), 3006703054056749 (52-bit). There are 78498 primes less than 1 000 000.

Primitive roots exist modulo any prime power p^a, except for $p = 2, a > 2$, and there are $\varphi(\varphi(p^a))$ many. For $p = 2, a > 2$, the group $\mathbb{Z}_{2^a}^\times$ is instead isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_{2^{a-2}}$.

Estimates

$\sum_{d|n} d = O(n \log \log n)$.

The number of divisors of n is at most around 100 for $n < 5e4$, 500 for $n < 1e7$, 2000 for $n < 1e10$, 200 000 for $n < 1e19$.

Möbius inversion

$$\forall n : g(n) = \sum_{d|n} f(d) \iff \forall n : f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right)$$

Derangements

Permutations of a set such that none of the elements appear in their original position.

$$D(n) = (n-1)(D(n-1) + D(n-2)) =$$
$$nD(n-1) + (-1)^n = \left\lfloor \frac{n!}{e} \right\rceil$$

Burnside's lemma

Given a group G of symmetries and a set X, the number of elements of X *up to symmetry* equals

$$\frac{1}{|G|} \sum_{g \in G} |X^g|,$$

where X^g are the elements fixed by g ($g.x = x$).

If $f(n)$ counts "configurations" (of some sort) of length n, we can ignore rotational symmetry using $G = \mathbb{Z}_n$ to get

$$g(n) = \frac{1}{n} \sum_{k=0}^{n-1} f(\gcd(n,k)) = \frac{1}{n} \sum_{k|n} f(k)\varphi(n/k).$$

Partition function

Number of ways of writing n as a sum of positive integers, disregarding the order of the summands.

$$p(0) = 1, p(n) = \sum_{k \in \mathbb{Z} \setminus \{0\}} (-1)^{k+1} p(n - k(3k-1)/2)$$

$$p(n) \sim 0.145/n \cdot \exp\left(2.56\sqrt{n}\right)$$

n	0	1	2	3	4	5	6	7	8	9	20	50	100
$p(n)$	1	1	2	3	5	7	11	15	22	30	627	∼2e5	∼2e8

Stirling numbers of the first kind

Number of permutations on n items with k cycles.

$$c(n,k) = c(n-1,k-1) + (n-1)c(n-1,k),\ c(0,0) = 1$$
$$\textstyle\sum_{k=0}^{n} c(n,k)x^k = x(x+1)\dots(x+n-1)$$

$c(8,k) = 8, 0, 5040, 13068, 13132, 6769, 1960, 322, 28, 1$
$c(n,2) = 0, 0, 1, 3, 11, 50, 274, 1764, 13068, 109584, \dots$

Eulerian numbers

Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$.

$$E(n,k) = (n-k)E(n-1,k-1) + (k+1)E(n-1,k)$$

$$E(n,0) = E(n,n-1) = 1$$

$$E(n,k) = \sum_{j=0}^{k} (-1)^j \binom{n+1}{j} (k+1-j)^n$$

Stirling numbers of the second kind

Partitions of n distinct elements into exactly k groups.

$$S(n,k) = S(n-1,k-1) + kS(n-1,k)$$

$$S(n,1) = S(n,n) = 1$$

$$S(n,k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n$$

Bell numbers

Total number of partitions of n distinct elements. $B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, \dots$. For p prime,

$$B(p^m + n) \equiv mB(n) + B(n+1) \pmod{p}$$

Catalan numbers

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!n!}$$

$$C_0 = 1,\ C_{n+1} = \frac{2(2n+1)}{n+2} C_n,\ C_{n+1} = \sum C_i C_{n-i}$$

$C_n = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, \dots$

- sub-diagonal monotone paths in an $n \times n$ grid.
- strings with n pairs of parenthesis, correctly nested.
- binary trees with with $n+1$ leaves (0 or 2 children).
- ordered trees with $n+1$ vertices.
- ways a convex polygon with $n+2$ sides can be cut into triangles by connecting vertices with straight lines.
- permutations of $[n]$ with no 3-term increasing subseq.

Hall's marriage theorem

Let G be a bipartite graph consisting of sets X and Y. We can match every vertex in X if and only if for every $W \subseteq X$,

$$|W| \leqslant |N(W)|\,.$$

Prüfer sequences

Consider trees with n vertices. Prüfer sequences are sequences of length $n-2$ in bijection with labeled trees.

To convert a tree to a sequence:

- while the tree has more than two vertices:
  - pick the leaf with the smallest label;
  - remove it and append its neighbor to the sequence.

2D geometry

Define $\mathbf{orient}(A,B,C) = \overline{AB} \times \overline{AC}$. CCW iff > 0.
Define $\mathbf{perp}((a,b)) = (-b,a)$. The vectors are orthogonal.
For line $ax + by = c$ def $\overline{v} = (-b,a)$.
Line through P and Q has $\overline{v} = \overline{PQ}$ and $c = \overline{v} \times P$.
$\mathbf{side}_l(P) = \overline{v_l} \times P - c_l$ sign determines which side P is on from l.
$\mathbf{dist}_l(P) = \mathbf{side}_l(P)/\|v_l\|$ squared is integer.
Sorting points along a line: comparator is $\overline{v} \cdot A < \overline{v} \cdot B$.
Translating line by $\overline{t}$: new line has $c' = c + \overline{v} \times \overline{t}$.
Line intersection: is $(c_l\overline{v}_m - c_m\overline{v}_l)/(\overline{v}_l \times \overline{v}_m)$.
Project P onto l: is $P - \mathbf{perp}(v)\,\mathbf{side}_l(P)/\|v\|^2$.
Angle bisectors: $\overline{v} = \overline{v}_l/\|\overline{v}_l\| + \overline{v}_m/\|\overline{v}_m\|$
$c = c_l/\|\overline{v}_l\| + c_m/\|\overline{v}_m\|$.
P is on segment AB iff $\mathbf{orient}(A,B,P) = 0$ and $\overline{PA} \cdot \overline{PB} \leqslant 0$.
Proper intersection of AB and CD exists iff $\mathbf{orient}(C,D,A)$ and $\mathbf{orient}(C,D,B)$ have opp. signs and $\mathbf{orient}(A,B,C)$ and $\mathbf{orient}(A,B,D)$ have opp. signs. Coordinates:

$$\frac{A\,\mathbf{orient}(C,D,B) - B\,\mathbf{orient}(C,D,A)}{\mathbf{orient}(C,D,B) - \mathbf{orient}(C,D,A)}.$$

Circumcircle center:

```
pt circumCenter(pt a, pt b, pt c) {
  b = b-a, c = c-a; // consider coordinates
      relative to A
  assert(cross(b,c) != 0); // no circumcircle if
      A,B,C aligned
  return a + perp(b*sq(c) - c*sq(b))/cross(b,c)
      /2;
```

Circle-line intersect:

```
int circleLine(pt o, double r, line l, pair<pt,pt
    > &out) {
  double h2 = r*r - l.sqDist(o);
  if (h2 >= 0) { // the line touches the circle
    pt p = l.proj(o); // point P
    pt h = l.v*sqrt(h2)/abs(l.v); // vector paral
        to l, of len h
    out = {p-h, p+h};
  }
  return 1 + sgn(h2);
```

Circle-circle intersect:

```
int circleCircle(pt o1, double r1, pt o2, double
    r2, pair<pt,pt> &out) {
  pt d=o2-o1; double d2=sq(d);
  if (d2 == 0) {assert(r1 != r2); return 0;} //
      concentric circles
  double pd = (d2 + r1*r1 - r2*r2)/2; // = |O_1P|
       * d
  double h2 = r1*r1 - pd*pd/d2; // = h^2
  if (h2 >= 0) {
    pt p = o1 + d*pd/d2, h = perp(d)*sqrt(h2/d2);
    out = {p-h, p+h};}
  return 1 + sgn(h2);
```

Tangent lines:

```
int tangents(pt o1, double r1, pt o2, double r2,
    bool inner, vector<pair<pt,pt>> &out) {
  if (inner) r2 = -r2;
  pt d = o2-o1;
  double dr = r1-r2, d2 = sq(d), h2 = d2-dr*dr;
  if (d2 == 0 || h2 < 0) {assert(h2 != 0); return
      0;}
  for (double sign : {-1,1}) {
    pt v = (d*dr + perp(d)*sqrt(h2)*sign)/d2;
    out.push_back({o1 + v*r1, o2 + v*r2});}
  return 1 + (h2 > 0);
```

3D geometry

$\mathbf{orient}(P,Q,R,S) = (\overline{PQ} \times \overline{PR}) \cdot \overline{PS}$.
S above PQR iff > 0.
For plane $ax + by + cz = d$ def $\overline{n} = (a,b,c)$.
Line with normal $\overline{n}$ through point P has $d = \overline{n} \cdot P$.
$\mathbf{side}_\Pi(P) = \overline{n} \cdot P - d$ sign determines side from Π.
$\mathbf{dist}_\Pi(P) = \mathbf{side}_\Pi(P)/\|\overline{n}\|$.
Translating plane by $\overline{t}$ makes $d' = d + \overline{n} \cdot \overline{t}$.
Plane-plane intersection of has direction $\overline{n_1} \times \overline{n_2}$ and goes through $((d_1\overline{n_2} - d_2\overline{n_1}) \times \overline{d})/\|\overline{d}\|^2$.
Line-line distance:

```
double dist(line3d l1, line3d l2) {
  p3 n = l1.d*l2.d;
  if (n == zero) // parallel
    return l1.dist(l2.o);
  return abs((l2.o-l1.o)|n)/abs(n);
```

Spherical to Cartesian:
$(r\cos\varphi\cos\lambda, r\cos\varphi\sin\lambda, r\sin\varphi)$.
Sphere-line intersection:

```
int sphereLine(p3 o, double r, line3d l, pair<p3,
    p3> &out) {
  double h2 = r*r - l.sqDist(o);
  if (h2 < 0) return 0; // the line doesn't touch
      the sphere
  p3 p = l.proj(o); // point P
  p3 h = l.d*sqrt(h2)/abs(l.d); // vector
      parallel to l, of length h
  out = {p-h, p+h};
  return 1 + (h2 > 0);
```

Great-circle distance between points A and B is $r\angle AOB$.
Spherical segment intersection:

```
bool properInter(p3 a, p3 b, p3 c, p3 d, p3 &out)
    {
  p3 ab = a*b, cd = c*d; // normals of planes OAB
      and OCD
  int oa = sgn(cd|a),
    ob = sgn(cd|b),
    oc = sgn(ab|c),
    od = sgn(ab|d);
  out = ab*cd*od; // four multiplications =>
      careful with overflow !
  return (oa != ob && oc != od && oa != oc);
}
bool onSphSegment(p3 a, p3 b, p3 p) {
  p3 n = a*b;
  if (n == zero)
    return a*p == zero && (a|p) > 0;
  return (n|p) == 0 && (n|a*p) >= 0 && (n|b*p) <=
      0;
}
struct directionSet : vector<p3> {
  using vector::vector; // import constructors
  void insert(p3 p) {
    for (p3 q : *this) if (p*q == zero) return;
    push_back(p);
  }
};
directionSet intersSph(p3 a, p3 b, p3 c, p3 d) {
  assert(validSegment(a, b) && validSegment(c, d)
      );
  p3 out;
  if (properInter(a, b, c, d, out)) return {out};
  directionSet s;
  if (onSphSegment(c, d, a)) s.insert(a);
  if (onSphSegment(c, d, b)) s.insert(b);
  if (onSphSegment(a, b, c)) s.insert(c);
  if (onSphSegment(a, b, d)) s.insert(d);
  return s;
}
```

Angle between spherical segments AB and AC is angle between $A \times B$ and $A \times C$.
Oriented angle: subtract from 2π if mixed product is negative.
Area of a spherical polygon:

$$r^2[\text{sum of interior angles} - (n-2)\pi].$$

Radixsort 50M 64 bit integers as single array in 1 sec

```
template <typename T>
void rsort(T *a, T *b, int size, int d = sizeof(T) - 1) {
  int b_s[256]{};
  ran(i, 0, size) { ++b_s[(a[i] >> (d * 8)) & 255]; }
  // ++b_s[*((uchar *)(a + i) + d)];
  T *mem[257];
  mem[0] = b;
  T **l_b = mem + 1;
  l_b[0] = b;
  ran(i, 0, 255) { l_b[i + 1] = l_b[i] + b_s[i]; }
  for (T *it = a; it != a + size; ++it) {
    T id = ((*it) >> (d * 8)) & 255;
    *(l_b[id]++) = *it;
  }
  l_b = mem;
  if (d) {
    T *l_a[256];
    l_a[0] = a;
    ran(i, 0, 255) l_a[i + 1] = l_a[i] + b_s[i];
    ran(i, 0, 256) {
      if (l_b[i + 1] - l_b[i] < 100) {
        sort(l_b[i], l_b[i + 1]);
        if (d & 1) copy(l_b[i], l_b[i + 1], l_a[i]);
      } else {
        rsort(l_b[i], l_a[i], b_s[i], d - 1);
      }
    }
  }
}
const int nmax = 5e7;
ll arr[nmax], tmp[nmax];
int main() {
  for (int i = 0; i < nmax; ++i)
    arr[i] = ((ll)rand() << 32) | rand();
  rsort(arr, tmp, nmax);
  assert(is_sorted(arr, arr + nmax));
}
```

FFT 10-15M length/sec

```
// integer c = a*b is accurate if c_i < 2^49
#pragma GCC optimize ("Ofast") //10% performance
#include <complex.h>
extern "C" __complex__ double __muldc3(
    double a, double b, double c, double d){
  return a*c-b*d+I*(a*d+b*c); // 40% performance
}
#include <bits/stdc++.h>
typedef complex<double> Comp;
void fft_rec(Comp *arr, Comp *root_pow, int len) {
  if (len != 1) {
    fft_rec(arr,       root_pow, len >> 1);
    fft_rec(arr + len, root_pow, len >> 1);
  }
  root_pow += len;
  ran(i, 0, len){
    tie(arr[i], arr[i + len]) = pair<Comp, Comp> {
        arr[i] + root_pow[i] * arr[i + len],
        arr[i] - root_pow[i] * arr[i + len] };
  }
}
void fft(vector<Comp> &arr, int ord, bool invert) {
  assert(arr.size() == 1 << ord);
  static vector<Comp> root_pow(1);
  static int inc_pow = 1;
  static bool is_inv = false;
  if (inc_pow <= ord) {
    int idx = root_pow.size();
    root_pow.resize(1 << ord);
    for (; inc_pow <= ord; ++inc_pow) {
      for (int idx_p = 0; idx_p < 1 << (ord - 1);
           idx_p += 1 << (ord - inc_pow), ++idx) {
        root_pow[idx] = Comp {
           cos(-idx_p * M_PI / (1 << (ord - 1))),
           sin(-idx_p * M_PI / (1 << (ord - 1))) };
        if (is_inv) root_pow[idx] = conj(root_pow[idx]);
      }
    }
  }
  if (invert != is_inv) {
    is_inv = invert;
    for (Comp &cur : root_pow) cur = conj(cur);
  }
  int j = 0;
  ran(i, 1, (1<<ord)){
    int m = 1 << (ord - 1);
    bool cont = true;
    while (cont) {
      cont = j & m;
      j ^= m;
      m >>= 1;
    }
    if (i < j) swap(arr[i], arr[j]);
  }
  fft_rec(arr.data(), root_pow.data(), 1 << (ord - 1));
  if (invert)
    ran(i, 0, 1<<ord) arr[i] /= (1 << ord);
}
void mult_poly_mod(vector<int> &a, vector<int> &b,
  vector<int> &c) { // c += a*b
  static vector<Comp> arr[4];
  // correct upto 0.5-2M elements(mod ~= 1e9)
  if (c.size() < 400) {
    ran(i, 0, (int)a.size())
      ran(j, 0, min((int)b.size(), (int)c.size()-i))
        c[i + j] = ((ll)a[i] * b[j] + c[i + j]) % mod;
  } else {
    int ord = 32 - __builtin_clz((int)c.size()-1);
    if ((int)arr[0].size() != 1 << ord){
      ran(i, 0, 4) arr[i].resize(1 << ord);
    }
    ran(i, 0, 4)
      fill(arr[i].begin(), arr[i].end(), Comp{});
    for (int &cur : a) if (cur < 0) cur += mod;
    for (int &cur : b) if (cur < 0) cur += mod;
    const int shift = 15;
    const int mask = (1 << shift) - 1;
    ran(i, 0, (int)min(a.size(), c.size())){
      arr[0][i] += a[i] & mask;
      arr[1][i] += a[i] >> shift;
    }
    ran(i, 0, (int)min(b.size(), c.size())){
      arr[0][i] += Comp{0, (b[i] & mask)};
      arr[1][i] += Comp{0, (b[i] >> shift)};
    }
    ran(i, 0, 2) fft(arr[i], ord, false);
    ran(i, 0, 2){
      ran(j, 0, 2){
        int tar = 2 + (i + j) / 2;
        Comp mult = {0, -0.25};
        if (i ^ j) mult = {0.25, 0};
        ran(k, 0, 1<<ord){
          int rev_k = ((1 << ord) - k) % (1 << ord);
          Comp ca = arr[i][k] + conj(arr[i][rev_k]);
          Comp cb = arr[j][k] - conj(arr[j][rev_k]);
          arr[tar][k] = arr[tar][k] + mult * ca * cb;
        }
      }
    }
    ran(i, 2, 4){
      fft(arr[i], ord, true);
      ran(k, 0, (int)c.size()){
        c[k] = (c[k] + (((ll)(arr[i][k].real()+0.5)%mod)
                    << (shift * (2 * (i-2) + 0)))) % mod;
        c[k] = (c[k] + (((ll)(arr[i][k].imag()+0.5)%mod)
                    << (shift * (2 * (i-2) + 1)))) % mod;
      }
    }
  }
}
```

Berlekamp-Massey O(LN)

```
template <typename T, T P>
struct intmod {
  intmod() {}
  constexpr intmod(T t) : x((t + P) % P) {}
  T value() const { return x; }
  bool operator!=(const intmod<T, P> i) { return x != i.x; }
  bool operator==(const intmod<T, P> i) { return x == i.x; }
  intmod<T, P> &operator+=(const intmod<T, P> i) {
    x = (x + i.x) % P;
```

```
  return *this;
 }
 intmod<T, P> &operator-=(const intmod<T, P> i) {
   x = (x + P - i.x) % P;
   return *this;
 }
 intmod<T, P> &operator*=(const intmod<T, P> i) {
   x = ((ll)x * i.x) % P;
   return *this;
 }
 intmod<T, P> &operator/=(const intmod<T, P> i) {
   x = ((ll)x * i.inverse().x) % P;
   return *this;
 }
 intmod<T, P> operator+(const intmod<T, P> i) const {
   auto j = *this;
   return j += i;
 }
 intmod<T, P> operator-(const intmod<T, P> i) const {
   auto j = *this;
   return j -= i;
 }
 intmod<T, P> operator*(const intmod<T, P> i) const {
   auto j = *this;
   return j *= i;
 }
 intmod<T, P> operator/(const intmod<T, P> i) const {
   auto j = *this;
   return j /= i;
 }
 intmod<T, P> operator-() const {
   intmod<T, P> n;
   n.x = (P - x) % P;
   return n;
 }
 intmod<T, P> inverse() const {
   if (x == 0) return 0;
   T a = x, b = P;
   T aa = 1, ab = 0;
   T ba = 0, bb = 1;
   while (a) {
     T q = b / a;
     T r = b % a;
     ba -= aa * q;
     bb -= ab * q;
     swap(ba, aa);
     swap(bb, ab);
     b = a;
     a = r;
   }
   intmod<T, P> ix = intmod<T, P>(aa) + intmod<T, P>(ba);
   assert(ix * x == unity);
   return ix;
 }
 static const intmod<T, P> zero;
 static const intmod<T, P> unity;
private:
  T x;
};
template <typename T, T P>
constexpr intmod<T, P> intmod<T, P>::zero = 0;
template <typename T, T P>
constexpr intmod<T, P> intmod<T, P>::unity = 1;
using rem = intmod<char, 2>;
template <typename K>
static vector<K> berlekamp_massey(vector<K> ss) {
  vector<K> ts(ss.size());
  vector<K> cs(ss.size());
  cs[0] = K::unity;
  fill(cs.begin() + 1, cs.end(), K::zero);
  vector<K> bs = cs;
  int l = 0, m = 1;
  K b = K::unity;
  for (int k = 0; k < (int)ss.size(); k++) {
    K d = ss[k];
    assert(l <= k);
    for (int i = 1; i <= l; i++) d += cs[i] * ss[k - i];
    if (d == K::zero) {
      m++;
    } else if (2 * l <= k) {
      K w = d / b;
      ts = cs;
      for (int i = 0; i < (int)cs.size() - m; i++)
        cs[i + m] -= w * bs[i];
      l = k + 1 - l;
      swap(bs, ts);
      b = d;
      m = 1;
    } else {
      K w = d / b;
      for (int i = 0; i < (int)cs.size() - m; i++)
        cs[i + m] -= w * bs[i];
      m++;
    }
  }
  cs.resize(l + 1);
  while (cs.back() == K::zero) cs.pop_back();
  return cs;
}
```

Mo's algorithm

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
vector<int> compress(vector<T>&a){
    int n=a.size();
    set<T>vls;
    for(auto el:a)vls.insert(el);
    map<T,int>mp;
    int ps=0;
    for(auto el:vls)mp[el]=ps++;
    vector<int>b(n);
    for(int i=0;i<n;i++)b[i]=mp[a[i]];
    return b;
}
class Mo{
    int n;
    vector<pair<int,int>>qs;
    public:
    Mo(int n):n(n){}
    void add_q(int l,int r){//[l,r]
        qs.push_back({l,r});
    }
    template<typename AL,typename AR,typename EL,
             typename ER,typename O>
    void build(const AL& al,const AR& ar, const EL &el,
               const ER &er, const O &out){
        int q=qs.size();
        int bs=n/min<int>(n,sqrt(q));
        vector<int>num(q);
        iota(begin(num),end(num),0);
        sort(begin(num),end(num),[&](int a,int b){
            int ab=qs[a].first/bs,bb=qs[b].first/bs;
            if(ab!=bb)return ab<bb;
            return qs[a].second<qs[b].second;
             });
        int l=0,r=-1;
        for(auto ind:num){
            while(r<qs[ind].second)ar(++r);
            while(r>qs[ind].second)er(r--);
            while(l<qs[ind].first)el(l++);
            while(l>qs[ind].first)al(--l);
            out(ind);
        }
    }
    template<typename A,typename E,typename O>
    void build(const A &a,const E &e,const O &out){
        build(a,a,e,e,out);
    }
};
```

LIS

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
vector<int> LIS(vector<T>&a){
  int n=a.size();
  vector<int>p(n),m(n+1);
  m[0]=-1;//undefined so any value
  int L=0;//|LIS| found so far
  for(int i=0;i<n;i++){
    int lo=1;
    int hi=L+1;//bsearch for smallest l<=L: a[m[l]]>x[i]
    while(lo<hi){
      int mid=lo+(hi-lo)/2;//lo<=mid<hi
      if(a[m[mid]]>=a[i])hi=mid;
      else lo=mid+1;
```

```
    }
    int newL=lo;
    p[i]=m[newL-1];//pred of X[i] is new last index of
    m[newL]=i; // ...  subseq of length new-1
    L=max(L,newL);
  }
  vector<int>res(L);
  int k=m[L];
  for(int j=L-1;j>=0;j--){
    res[j]=k;
    k=p[k];
  }
  return res;
}
```

Persistent queue

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
class PersistentQueue{
  struct Tipp{
    T vaartus;
    vector<Tipp*>ulemused;
    int pikkus;
    Tipp(T vaartus,int pikkus):vaartus(vaartus),
                               pikkus(pikkus){}
    Tipp():pikkus(0){}
    void lisaUlemus(Tipp* ul){
      ulemused.push_back(ul);
    }
  };
  vector<Tipp*>versioonid;
public:
  PersistentQueue(){
    versioonid.push_back(new Tipp());
  }//Lisa jrk ul lõppu väärtus, sellest uus järjekord
  void uus(int ul,T &vaartus){
    Tipp *ulemus=versioonid[ul];
    int maht=ulemus->pikkus+1;
    Tipp *uus=new Tipp(vaartus,maht);
    uus->lisaUlemus(ulemus);
    maht-=1;
    int koht=0;
    while(maht>(1<<koht)){
      ulemus=ulemus->ulemused[koht];
      uus->lisaUlemus(ulemus);
      maht-=1<<koht;
      koht++;
    }
    versioonid.push_back(uus);
  }//eemalda järjekorrast i esimene el,tagasta see, uus
  T eemalda(int i){
    Tipp *t=versioonid[i];
    int maht=t->pikkus-1;
    Tipp *eem=KsUlemus(t,maht);
    T vastus=eem->vaartus;
    Tipp *uus=new Tipp(t->vaartus,maht);
    for(auto el:t->ulemused)uus->lisaUlemus(el);
    versioonid.push_back(uus);
    return vastus;
  }
  Tipp *KsUlemus(Tipp *t,int k){//Leia tipu t,k-s ulemus
    int i=0;
    while(k){
      if(k&(1<<i)){
        t=t->ulemused[i];
        k-=1<<i;
      }
      i++;
    }
    return t;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int q;
  cin>>q;
  PersistentQueue<int>T;
  while(q--){
    int op,t,x;
    cin>>op>>t;
    if(op==0){
      cin>>x;
      T.uus(t+1,x);
    }else{
      cout<<T.eemalda(t+1)<<'\n';
    }
  }
  return 0;
}
```

Dynamic N-dimensional Fenwick tree

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int MAXN=1000*1000*1000+5;
template<typename T,typename D,int M>
struct DynamicFenwick{//index type, sum type, dimension
  unordered_map<T,DynamicFenwick<T,D,M-1>>fens;
  void add_point(vector<T>& loc,D &val){
    //location(1-...),value
    T stpos=loc.back();loc.pop_back();
    T pos=stpos;
    while(pos<MAXN){
      fens[pos].add_point(loc,val);
      pos+=pos&(-pos);
    }
    loc.push_back(stpos);
  }
  D get(vector<T>& l,vector<T>&r){//[l,r]
    T posl=l.back();l.pop_back();
    T posr=r.back();r.pop_back();
    T pos=posl-1;
    D res=0;
    while(pos>0){
      if(fens.count(pos))res-=fens[pos].get(l,r);
      pos-=pos&(-pos);
    }
    pos=posr;
    while(pos>0){
      if(fens.count(pos))res+=fens[pos].get(l,r);
      pos-=pos&(-pos);
    }
    l.push_back(posl);
    r.push_back(posr);
    //cout<<posl<<'Q'<<posr<<endl;
    return res;
  }
};
template<typename T,typename D>
struct DynamicFenwick<T,D,1>{
  unordered_map<T,D>dp;
  void add_point(vector<T>& loc,D &val){
    T stpos=loc.back();loc.pop_back();
    T pos=stpos;
    while(pos<MAXN){
      dp[pos]+=val;
      pos+=pos&(-pos);
    }
    loc.push_back(stpos);
  }
  D get(vector<T>& l,vector<T>&r){//[l,r]
    T posl=l.back();l.pop_back();
    T posr=r.back();r.pop_back();
    T pos=posl-1;
    D res=0;
    while(pos>0){
      if(dp.count(pos))res-=dp[pos];
      pos-=pos&(-pos);
    }
    pos=posr;
    while(pos>0){
      if(dp.count(pos))res+=dp[pos];
      pos-=pos&(-pos);
    }
    l.push_back(posl);
    r.push_back(posr);
    //cout<<posl<<'r'<<posr<<'S'<<res<<endl;
    return res;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  DynamicFenwick<int,ll,2>fen;
  int n,q;
  cin>>n>>q;
  while(n--){
    int x,y;
```

```
    ll w;
    cin>>x>>y>>w;
    vector<int>loc={y+1,x+1};
    fen.add_point(loc,w);
  }
  while(q--){
    int op;
    cin>>op;
    if(op==1){
      int l,d,r,u;
      cin>>l>>d>>r>>u;
      vector<int>v={d+1,l+1};
      vector<int>p={u,r};
      cout<<fen.get(v,p)<<'\n';
    }else{
      int x,y;
      ll w;
      cin>>x>>y>>w;
      vector<int>loc={y+1,x+1};
      fen.add_point(loc,w);
    }
  }
  return 0;
}
```

Fenwick

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct F{
    vector<ll>dp;
    int n;
    F(int n){
        this->n=n+1;
        dp.resize(n+1);
    }
    void add(ll pos,ll vl){
        while(pos<n){
            dp[pos]+=vl;
            pos+=pos&(-pos);
        }
    }
    ll get(ll pos){
        ll res=0;
        while(pos>0){
            res+=dp[pos];
            pos-=pos&(-pos);
        }
        return res;
    }
    ll get(ll l,ll r){
        return get(r)-get(l-1);
    }
};
```

Offline Li Chao tree

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
class LiChao{
  struct Joon{
    T m,b;
    Joon(T m,T b):m(m),b(b){}
    T operator()(T x)const{return m*x+b;}
  };
  static constexpr T inf=numeric_limits<T>::max();
  int size;
  vector<T>paringud;
  vector<Joon>tipud;
  void uuenda(int i,int l,int r,Joon joon){
    int m=(l+r)/2;
    bool vasak=joon(paringud[l])<tipud[i](paringud[l]);
    bool kesk=joon(paringud[m])<tipud[i](paringud[m]);
    bool parem=joon(paringud[r])<tipud[i](paringud[r]);
    if(!vasak&&!parem)return;
    if(vasak&&parem){
      tipud[i]=joon;
      return;
    }
    if(kesk)swap(tipud[i],joon);
    if(l==r)return;
    if(vasak!=kesk)uuenda(i*2,l,m,joon);
    else uuenda(i*2+1,m+1,r,joon);
  }
public:
  LiChao()=default;
  LiChao(vector<T>& vec){
    paringud=vec;
    if(paringud.empty())paringud.push_back(inf);
    sort(paringud.begin(),paringud.end());
    paringud.erase(unique(paringud.begin(),
                          paringud.end()),paringud.end());
    size=1;
    while(size<paringud.size())size*=2;
    tipud.resize(2*size,{0,inf});
    while(paringud.size()<size)paringud
              .push_back(paringud.back()+1);
  }
  void add_line(T a,T b){
    Joon j(a,b);
    uuenda(1,0,size-1,j);
  }
  void add_segment(T a,T b, T xl,T xr){//[]
    int l = lower_bound(paringud.begin(),
              paringud.end(), xl) - paringud.begin();
    int r = upper_bound(paringud.begin(),
              paringud.end(), xr) - paringud.begin()-1;
    if(l>r)return;
    if(r<0)return;
    if(l>=size)return;
    Joon joon(a,b);
    add_segment(1,0,size-1,l,r,joon);
  }
  void add_segment(int i,int l,int r,int se,
                   int en,Joon joon){
    if(se<=l&&r<=en){
      uuenda(i,l,r,joon);
      return;
    }
    if(se>r||l>en)return;
    int m=(l+r)/2;
    add_segment(i*2,l,m,se,en,joon);
    add_segment(i*2+1,m+1,r,se,en,joon);
  }
  T saa(T x){
    int k=lower_bound(paringud.begin(),
                      paringud.end(),x)-paringud.begin();
    return saa(1,0,size-1,k,x);
  }
  T saa(int i,int l,int r,int pos,T& x){
    if(l==r)return tipud[i](x);
    int m=(l+r)/2;
    if(pos<=m)return min(tipud[i](x),saa(i*2,l,m,pos,x));
    return min(tipud[i](x),saa(i*2+1,m+1,r,pos,x));
  }
};
```

Wavelet matrix

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct SuccintIndexableDictionary{
  int len,blocks;
  typedef unsigned u;
  vector<u>bit,sum;
  SuccintIndexableDictionary()=default;
  SuccintIndexableDictionary(int len){
    this->len=len;
    blocks=(len+31)>>5;
    bit.assign(blocks,0U);
    sum.assign(blocks,0U);
  }
  //set kth bit
  void set(int k){
    bit[k>>5]|=1U << (k&31);
  }
  void build(){
    sum[0]=0U;
    for(int i=1;i<blocks;i++)sum[i]=sum[i-1]
                +__builtin_popcount(bit[i-1]);
  }
  //get val of kth bit
  bool operator[](int k){
    return (bool((bit[k>>5]>>(k&31))&1));
  }
  //number of 1 bits set <k
  int rank(int k){
```

```
    return (sum[k>>5]+ __builtin_popcount(
        bit[k >> 5] & ((1U << (k & 31)) - 1)));
  }
  int rank(bool val,int k){
    return (val?rank(k):k-rank(k));
  }
};
template<typename T,int MXLOG>
struct WaveletMatrix{
  int len;
  SuccintIndexableDictionary mat[MXLOG];
  int mid[MXLOG];
  WaveletMatrix()=default;
  WaveletMatrix(vector<T>& vec){
    len=vec.size();
    vector<T> l(len),r(len);
    for(int lev=MXLOG-1;lev>=0;lev--){
      mat[lev]=SuccintIndexableDictionary(len+1);
      int lef=0,rig=0;
      for(int i=0;i<len;i++){
        if((vec[i]>>lev)&1){
          mat[lev].set(i);
          r[rig++]=vec[i];
        }else l[lef++]=vec[i];
      }
      mid[lev]=lef;
      mat[lev].build();
      vec.swap(l);
      for(int i=0;i<rig;i++)vec[lef+i]=r[i];
    }
  }
  typedef pair<int,int> pii;
  pii succ(bool f,int l,int r,int lev){
    return {mat[lev].rank(f,l)+mid[lev]*f,
            mat[lev].rank(f,r)+mid[lev]*f};
  }
  //v[k]
  T get_position(int k){
    T vas=0;
    for(int lev=MXLOG-1;lev>=0;lev--){
      bool f=mat[lev][k];
      if(f)vas|=T(1)<<lev;
      k=mat[lev].rank(f,k)+mid[lev]*f;
    }
    return vas;
  }
  T operator[](const int &k){
    return get_position(k);
  }
  //number of i<r:v[i]==k
  int rank(T& k,int r){
    int l=0;
    for(int lev=MXLOG-1;lev>=0;lev--)
      tie(l,r)=succ((k>>lev)&1,l,r,lev);
    return r-l;
  }
  //k,l,r are 0-based
  T kth_smallest(int l,int r,int k){
    assert(0 <= k && k < r - l);
    T vas=0;
    for(int lev=MXLOG-1;lev>=0;lev--){
      int cnt=mat[lev].rank(0,r)-mat[lev].rank(0,l);
      bool f=cnt<=k;
      if(f){
        vas|=T(1)<<lev;
        k-=cnt;
      }
      tie(l,r)=succ(f,l,r,lev);
    }
    return vas;
  }
  T kth_largest(int l,int r,int k){
    return kth_smallest(l,r,r-l-k-1);
  }
  //count i such that l <=i < r and v[i]<up
  int range_freq(int l,int r,T up){
    int vas=0;
    for(int lev=MXLOG-1;lev>=0;lev--){
      bool f=((up>>lev)&1);
      if(f)vas+=mat[lev].rank(0,r)-mat[lev].rank(0,l);
      tie(l,r)=succ(f,l,r,lev);
    }
    return vas;
  }
  //count i that (l<= i < r) and (low <= v[i] < up)
  int range_freq(int l,int r,T low,T up){
    return range_freq(l,r,up)-range_freq(l,r,low);
  }
  //maximal v[i] : v[i]<up
  T prev_val(int l,int r, T up){
    int cnt=range_freq(l,r,up);
    return cnt==0?T(-1):kth_smallest(l,r,cnt-1);
  }
  //minimal v[i] : v[i]>=low
  T next_val(int l,int r,T low){
    int cnt=range_freq(l,r,low);
    return cnt==r-l?T(-1):kth_smallest(l,r,cnt);
  }
};
template<typename T,int MXLOG>
struct CompressedWaveletMatrix{
  WaveletMatrix<int,MXLOG>mat;
  vector<T>vls;
  CompressedWaveletMatrix(vector<T>& vec):vls(vec){
    sort(vls.begin(),vls.end());
    vls.erase(unique(vls.begin(),vls.end()),vls.end());
    vector<int>nw(vec.size());
    for(int i=0;i<vec.size();i++)nw[i]=get(vec[i]);
    mat=WaveletMatrix<int,MXLOG>(nw);
  }
  //x->its compressed int
  int get(T& x){
 return lower_bound(vls.begin(),vls.end(),x)-vls.begin();
  }
  //vec[k]
  T get_position(int k){
    return vls[mat[k]];
  }
  //vec[k]
  T operator[](int &k){
    return get_position(k);
  }
  //number of i<r:v[i]==k
  int rank(T &k,int r){
    auto ind=get(k);
    if(ind==vls.size()||vls[ind]!=k)return 0;
    return mat.rank(ind,r);
  }
  //k,l,r are 0-based
  T kth_smallest(int l,int r,int k){
    return vls[mat.kth_smallest(l,r,k)];
  }
  T kth_largest(int l,int r,int k){
    return vls[mat.kth_largest(l,r,k)];
  }
  //count i such that l <=i < r and v[i]<up
  int range_freq(int l,int r,T up){
    return mat.range_freq(l,r,get(up));
  }
  //count i that (l<= i < r) and (low <= v[i] < up)
  int range_freq(int l,int r,T low,T up){
    return mat.range_freq(l,r,get(low),get(up));
  }
  //maximal v[i] : v[i]<up, or -1
  T prev_val(int l,int r,T up){
    auto vas=mat.prev_val(l,r,get(up));
    return (vas==-1)?T(-1):vls[vas];
  }
  //minimal v[i] : v[i]>=low, or -1
  T next_val(int l,int r,T low){
    auto vas=mat.next_val(l,r,get(low));
    return (vas==-1)?T(-1):vls[vas];
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  const int N=1000*1000;
  vector<int>a(N);
  for(int i=0;i<N;i++)a[i]=i;
  CompressedWaveletMatrix<int,20>w(a);
  for(int i=0;i<N;i++){
    assert(w.get_position(i)==i);
  }
  for(int i=0;i<N;i++){
    assert(w.rank(i,i)==0);
    assert(w.rank(i,i+1)==1);
    assert(w.kth_smallest(0,i+1,i)==i);
    assert(w.kth_largest(0,i+1,0)==i);
    assert(w.range_freq(0,i+1,i)==i);
    assert(w.prev_val(0,N,i)==(i-1));
    assert(w.next_val(0,N,i)==i);
```

```
  }
  return 0;
}
```

Convex hull trick (static)

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
class CHT{
  struct Joon{
    ll m,b;
    Joon(){}
    Joon(ll m,ll b):m(m),b(b){}
    ll operator()(ll x)const{return m*x+b;}
    long double IN(Joon& a){
      return  -((long double) (1.0)*(b-a.b))/(m-a.m);
    }
  };
  int pos;
  vector<Joon>ar;
public:
  CHT(int n){//n kokku jooni maksimaalselt
    pos=0;
    ar.resize(n);
  }
  ll get(ll x){
    if(pos==1)return ar[pos-1](x);
    if(ar[0](x)>=ar[1](x))return ar[0](x);
    if(ar[pos-1](x)>=ar[pos-2](x))return ar[pos-1](x);
    int l=0,r=pos-2,ans=-1;
    while(l<=r){
      int m=(l+r)/2;
      //cout<<ar[m].eval(x)<<' '<<ar[m+1].eval(x)<<'\n';
      if(ar[m](x)>=ar[m+1](x)){
        r=m-1;
        ans=m;
      }else l=m+1;
    }
    return ar[ans](x);
  }
  void lisaViimane(ll m,ll b){//m kasvavas järjekorras
    Joon cur(m,b);
    while(pos>1&&ar[pos-2].IN(cur)>=ar[pos-1].IN(cur))
      pos--;
    ar[pos++]=cur;
  }
};
```

Range tree

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int M=3;
typedef array<int,M> vi;
template<int D>
struct RangeTree{
  vector<array<int,D+1>>vls;
  int n;
  vector<RangeTree<D-1>*>alampuud;
  RangeTree(vector<array<int,D+1>>& v){
    vls=v;
    n=vls.size();
    sort(vls.begin(),vls.end(),[](
     const array<int,D+1>& a, const array<int,D+1>& b){
      return (a[D] < b[D]);});
    alampuud.resize(2*n);
    build(0,0,n-1);
  }
  void build(int i,int l,int r){
    if(l==r)return;
    int m=(l+r)/2;
    build(i*2+1,l,m);
    build(i*2+2,m+1,r);
    vector<array<int,D>> uusVec(r-l+1);
    for(int k=l;k<=r;k++){
      for(int j=0;j<D;j++)uusVec[k-l][j]=vls[k][j];
    }
    alampuud[i]=new RangeTree<D-1>(uusVec);
  }
  ll check_el(int i,vi& lef,vi& rig){
    for(int p=1;p<D;p++){
      if(vls[i][p]<lef[p]||vls[i][p]>rig[p])return 0;
    }
    return vls[i][0];
  }
  ll rangeSum(vi& lef,vi& rig){
    return rangeSum(0,0,n-1,lef,rig);
  }
  ll rangeSum(int i,int l,int r,vi& lef,vi& rig){
    if(vls[l][D]>=lef[D]&&vls[r][D]<=rig[D]){
      if(l==r)return check_el(l,lef,rig);
      return alampuud[i]->rangeSum(lef,rig);
    }
    if(vls[l][D]>rig[D]||vls[r][D]<lef[D])return 0;
    int m=(l+r)/2;
    return rangeSum(i*2+1,l,m,lef,rig)
      + rangeSum(i*2+2,m+1,r,lef,rig);
  }
};
template<> struct RangeTree<1>{
  typedef array<ll,2> pii;
  vector<ll>sum;
  vector<int>vls;
  RangeTree(vector<array<int,2>>& v){
    int n=v.size();
    vls.resize(n);
    sum.resize(n+1);
    sort(v.begin(),v.end(),[](const array<int,2>& a,
         const array<int,2>& b){return (a[1] < b[1]);});
    for(int i=1;i<=n;i++)sum[i]=v[i-1][0]+sum[i-1];
    for(int i=0;i<n;i++)vls[i]=v[i][1];
  }
  ll rangeSum(vi& lef,vi & rig){
    ll l=lef[1],r=rig[1];
    ll parem=upper_bound(vls.begin(),vls.end(),r)
      -vls.begin();
    ll vasak=lower_bound(vls.begin(),vls.end(),l)
      -vls.begin();
    return sum[parem]-sum[vasak];
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,q;
  cin>>n>>q;
  vector<vi>vls(n);
  for(int i=0;i<n;i++){
    cin>>vls[i][2]>>vls[i][1]>>vls[i][0];
  }
  RangeTree<M-1>qs(vls);
  while(q--){
    int l,d,r,u;
    cin>>l>>d>>r>>u;
    vi lef,rig;
    lef[2]=l;
    lef[1]=d;
    rig[2]=r-1;
    rig[1]=u-1;
    cout<<qs.rangeSum(lef,rig)<<'\n';
  }
  return 0;
}
```

LCA with binary lifting

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N=500005,LG2=(int)log2(N)+1;
vector<int>g[N];
class LCA{public:
  vector<int>dep;
  vector<vector<int>>P;
  LCA(int root){
    dep=vector<int>(N,0);
    P=vector<vector<int>>(LG2,vector<int>(N,0));
    dfs(root,-1);
    for(int i=1;i<LG2;i++){
      for(int u=0;u<N;u++)P[i][u]=P[i-1][P[i-1][u]];
    }
  }
  void dfs(int u,int pr){
    for(auto v:g[u]){
      if(v==pr)continue;
      P[0][v]=u;
      dep[v]=dep[u]+1;
      dfs(v,u);
    }
  }
```

```
  int lca(int u,int v){
    if(dep[u]<dep[v])swap(u,v);
    for(int i=LG2-1;i>=0;i--){
      if(dep[u]-(1<<i)>=dep[v]){
        u=P[i][u];
      }
    }
    if(u==v)return u;
    for(int i=LG2-1;i>=0;i--){
      if(P[i][u]!=P[i][v]){
        u=P[i][u];
        v=P[i][v];
      }
    }
    return P[0][u];
  }
};
```

DSU with rollback

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
class DSU{
    struct Kirje{
        int u,v,szu;
        bool check;
        Kirje(int u,int v,int szu):
          u(u),v(v),szu(szu),check(false){}
        Kirje():check(true){}
    };
    vector<int>fa,sz;
    vector<Kirje>kirjed;
    int n,setCnt;
      public:
    DSU(int n):n(n),setCnt(n){
        n++;
        fa.resize(n);
        sz.resize(n,0);
        for(int i=0;i<n;i++)fa[i]=i;
    }
    int F(int u){//Esindaja leidmine
        if(u==fa[u])return u;
        return F(fa[u]);
    }
    void uhenda(int u,int v){
        int fu=F(u);
        int fv=F(v);
        if(fu==fv)return;
        if(sz[fu]<sz[fv])swap(fu,fv);
        kirjed.push_back(Kirje(fu,fv,sz[fu]));
        setCnt--;
        fa[fv]=fu;
        sz[fu]=sz[fu]+(sz[fu]==sz[fv]);
    }
    void persist(){//Kontrollpunkt, alati enne rollbacki
        kirjed.push_back(Kirje());
    }
    void rollback(){//Tagasi viimase kontrollpunktini
        while(!kirjed.back().check){
            auto e=kirjed.back();
            int u=e.u,v=e.v,szu=e.szu;
            fa[v]=v;
            sz[u]=szu;
            kirjed.pop_back();
            setCnt++;
        }
        kirjed.pop_back();
    }
    int getCnt(){//Komponentide arv
        return setCnt;
    }
};
```

Segment tree beats

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct Beats{
  static constexpr ll INF=LLONG_MAX/2.1;
  struct Tipp{
    ll sum=0,mn1=0,mx1=0,mx2=-INF,mn2=INF,add=0;
    int mnc=1,mxc=1;
  };
  vector<Tipp>dp;
  int n;
  Beats(){}
  Beats(int n):Beats(vector<ll>(n)){}
  Beats(const vector<ll>&vec){
    n=vec.size();
    int sz=1;
    while(sz<(int)vec.size())sz*=2;
    dp.resize(2*sz);
    build(1,0,n-1,vec);
  }
  void build(int i,int l,int r,const vector<ll>&vec){
    if(l==r){
      dp[i].sum=dp[i].mn1=dp[i].mx1=vec[l];
      return;
    }
    int m=(l+r)/2;
    build(i*2,l,m,vec);build(i*2+1,m+1,r,vec);
    up(i);
  }
  void range_chmin(int l,int r,ll x){update<1>(l,r,x);}
  void range_chmax(int l,int r,ll x){update<2>(l,r,x);}
  void range_add(int l,int r,ll x){update<3>(l,r,x);}
  void range_update(int l,int r,ll x){update<4>(l,r,x);}
  ll range_min(int l,int r){return query<1>(l,r);}
  ll range_max(int l,int r){return query<2>(l,r);}
  ll range_sum(int l,int r){return query<3>(l,r);}
private:
  void up(int i){
    Tipp& p=dp[i];
    Tipp& l=dp[i*2];
    Tipp& r=dp[i*2+1];
    p.sum=l.sum+r.sum;
    if(l.mx1==r.mx1){
      p.mx1=l.mx1;
      p.mx2=max(l.mx2,r.mx2);
      p.mxc=l.mxc+r.mxc;
    }else{
      bool f=l.mx1>r.mx1;
      p.mx1=f?l.mx1:r.mx1;
      p.mxc=f?l.mxc:r.mxc;
      p.mx2=max(f?r.mx1:l.mx1,f?l.mx2:r.mx2);
    }
    if(l.mn1==r.mn1){
      p.mn1=l.mn1;
      p.mn2=min(l.mn2,r.mn2);
      p.mnc=l.mnc+r.mnc;
    }else{
      bool f=l.mn1<r.mn1;
      p.mn1=f?l.mn1:r.mn1;
      p.mnc=f?l.mnc:r.mnc;
      p.mn2=min(f?r.mn1:l.mn1,f?l.mn2:r.mn2);
    }
  }
  void push_add(int i,int l,int r,ll x){
    Tipp& p=dp[i];
    p.sum+=x*(r-l+1);
    p.mn1+=x;
    p.mx1+=x;
    if(p.mx2!=-INF)p.mx2+=x;
    if(p.mn2!=INF)p.mn2+=x;
    p.add+=x;
  }
  void push_min(int i,int l,int r,ll x){
    Tipp& p=dp[i];
    p.sum+=(x-p.mx1)*p.mxc;
    if(p.mn1==p.mx1)p.mn1=x;
    if(p.mn2==p.mx1)p.mn2=x;
    p.mx1=x;
  }
  void push_max(int i,int l,int r,ll x){
    Tipp& p=dp[i];
    p.sum+=(x-p.mn1)*p.mnc;
    if(p.mx1==p.mn1)p.mx1=x;
    if(p.mx2==p.mn1)p.mx2=x;
    p.mn1=x;
  }
  void push(int i,int l,int r){
    Tipp& p=dp[i];
    int m=(l+r)/2;
    if(p.add!=0){
      push_add(i*2,l,m,p.add);
      push_add(i*2+1,m+1,r,p.add);
      p.add=0;
    }
    if(p.mx1<dp[i*2].mx1)push_min(i*2,l,m,p.mx1);
```

```
  if(p.mn1>dp[i*2].mn1)push_max(i*2,l,m,p.mn1);
  if(p.mx1<dp[i*2+1].mx1)push_min(i*2+1,m+1,r,p.mx1);
  if(p.mn1>dp[i*2+1].mn1)push_max(i*2+1,m+1,r,p.mn1);
}
void subtree_chmin(int i,int l,int r,ll x){
  if(dp[i].mx1<=x)return;
  if(dp[i].mx2<x){
    push_min(i,l,r,x);
    return;
  }
  int m=(l+r)/2;
  push(i,l,r);
  subtree_chmin(i*2,l,m,x);
  subtree_chmin(i*2+1,m+1,r,x);
  up(i);
}
void subtree_chmax(int i,int l,int r,ll x){
  if(x<=dp[i].mn1)return;
  if(x<dp[i].mn2){
    push_max(i,l,r,x);
    return;
  }
  int m=(l+r)/2;
  push(i,l,r);
  subtree_chmax(i*2,l,m,x);
  subtree_chmax(i*2+1,m+1,r,x);
  up(i);
}
template<int CMD>
void _apply(int i,int l,int r,ll x){
  if (CMD==1)subtree_chmin(i,l,r,x);
  if (CMD==2)subtree_chmax(i,l,r,x);
  if (CMD==3)push_add(i,l,r,x);
  if (CMD==4)subtree_chmin(i,l,r,x),
             subtree_chmax(i,l,r,x);
}
template<int CMD>
void update(int l,int r,ll x){
  update<CMD>(1,0,n-1,l,r,x);
}
template<int CMD>
void update(int i,int l,int r,int se,int en,ll x){
  if(se<=l&&r<=en){
    _apply<CMD>(i,l,r,x);
    return;
  }
  if(l>en||r<se)return;
  int m=(l+r)/2;
  push(i,l,r);
  update<CMD>(i*2,l,m,se,en,x);
  update<CMD>(i*2+1,m+1,r,se,en,x);
  up(i);
}
template<int CMD>
ll query(int l,int r){
  return query<CMD>(1,0,n-1,l,r);
}
  template<int CMD>
  ll e(){
    if (CMD==1)return INF;
    if (CMD==2)return -INF;
    if (CMD==3)return 0;
  }
  template<int CMD>
  void op(ll& a,Tipp& b){
    if (CMD==1)a=min(a,b.mn1);
    if (CMD==2)a=max(a,b.mx1);
    if (CMD==3)a+=b.sum;
  }
  template<int CMD>
  ll query(int i,int l,int r,int se,int en){
    ll res=e<CMD>();
    if(se<=l&&r<=en){
      op<CMD>(res,dp[i]);
      return res;
    }
    if(l>en||r<se)return res;
    int m=(l+r)/2;
    push(i,l,r);
    ll nw=query<CMD>(i*2,l,m,se,en);
    if (CMD==1)res=min(res,nw);
    if (CMD==2)res=max(res,nw);
    if (CMD==3)res+=nw;
    nw=query<CMD>(i*2+1,m+1,r,se,en);
    if (CMD==1)res=min(res,nw);
    if (CMD==2)res=max(res,nw);
    if (CMD==3)res+=nw;
    return res;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,q;
  cin>>n>>q;
  vector<ll>a(n);
  for(int i=0;i<n;i++)cin>>a[i];
  Beats t(a);
  while(q--){
    int op,l,r;
    ll b;
    cin>>op>>l>>r;
    r--;
    if(op==3){
      cout<<t.range_sum(l,r)<<'\n';
      continue;
    }
    cin>>b;
    if(op==0){
      t.range_chmin(l,r,b);
    }else if(op==1){
      t.range_chmax(l,r,b);
    }else{
      t.range_add(l,r,b);
    }
  }
  return 0;
}
```

Dominator tree

```
#include <bits/stdc++.h>
//based on https://tanujkhattar.wordpress.com/2016/01/11/
//                  dominator-tree-of-a-directed-graph/
typedef long long ll;
using namespace std;
const int N=200005;
vector<int>g[N],revg[N];
class DominatorTree{
  vector<vector<int>>g,rg,buck;
  int n,S;
  int ind;
//dfs ords,label-vrt with min sdom on part from i to root
  vector<int>nodeToInd,indToNode,label,sdom,dom,dsu,par;
public:
  DominatorTree(int n,int S):n(n),S(S){
    g.resize(n+1);
    rg.resize(n+1);
    buck.resize(n+1);
    nodeToInd.resize(n+1);
    indToNode.resize(n+1);
    label.resize(n+1);
    sdom.resize(n+1);
    dom.resize(n+1,-1);
    dsu.resize(n+1);
    par.resize(n+1);
    ind=0;
  }
  void addEdge(int u,int v){
    g[u].push_back(v);
  }
  void build(){
    dfs(S);
    for(int i=ind;i>=1;i--){
     for(auto j:rg[i])sdom[i]=min(sdom[i],sdom[leia(j)]);
     if(i>1)buck[sdom[i]].push_back(i);
     for(auto w:buck[i]){
       int v=leia(w);
       if(sdom[v]==sdom[w])dom[w]=sdom[w];
       else dom[w]=v;
     }
     if(i>1)uhenda(par[i],i);
    }
    for(int i=2;i<=ind;i++){
      if(dom[i]!=sdom[i])dom[i]=dom[dom[i]];
    }
  }
  void dfs(int u){
    ind++;
    nodeToInd[u]=ind;indToNode[ind]=u;
    label[ind]=ind;sdom[ind]=ind;dsu[ind]=ind;
    for(auto v:g[u]){
```

```
      if(!nodeToInd[v]){
        dfs(v);
        par[nodeToInd[v]]=nodeToInd[u];
      }
      rg[nodeToInd[v]].push_back(nodeToInd[u]);
    }
  }
  int leia(int u,int x=0){
    if(u==dsu[u])return x?-1:u;
    int v=leia(dsu[u],x+1);
    if(v<0)return u;
    if(sdom[label[dsu[u]]]<sdom[label[u]])
      label[u]=label[dsu[u]];
    dsu[u]=v;
    return x?v:label[u];
  }
  void uhenda(int u,int v){
    dsu[v]=u;
  }
  vector<int> vas(){
    vector<int>res(n,-1);
    for(int i=1;i<=ind;i++)
      res[indToNode[i]]=indToNode[dom[i]];
    res[S]=S;
    return res;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,m,S;
  cin>>n>>m>>S;
  DominatorTree t(n,S);
  for(int i=0;i<m;i++){
    int u,v;cin>>u>>v;
    t.addEdge(u,v);
  }
  t.build();
  auto vas=t.vas();
  for(auto el:vas)cout<<el<<' ';
  return 0;
}
```

Offline dynamic connectivity

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct DSU{
  struct Kirje{
    int u,v,szu;
    bool check;
    Kirje(int u,int v,int szu):
      u(u),v(v),szu(szu),check(false){}
    Kirje():check(true){}
  };
  vector<int>fa,sz;
  vector<Kirje>kirjed;
  int n,setCnt;
public:
  DSU(int n):n(n),setCnt(n){
    n++;
    fa.resize(n);
    sz.resize(n,0);
    for(int i=0;i<n;i++)fa[i]=i;
  }
  int F(int u){//Esindaja leidmine
    if(u==fa[u])return u;
    return F(fa[u]);
  }
  void uhenda(int u,int v){
    int fu=F(u);
    int fv=F(v);
    if(fu==fv)return;
    if(sz[fu]<sz[fv])swap(fu,fv);
    kirjed.push_back(Kirje(fu,fv,sz[fu]));
    setCnt--;
    fa[fv]=fu;
    sz[fu]=sz[fu]+(sz[fu]==sz[fv]);
  }
  void persist(){//Kontrollpunkt, alati enne rollbacki
    kirjed.push_back(Kirje());
  }
  void rollback(){//Tagasi viimase kontrollpunktini
    while(!kirjed.back().check){
      auto e=kirjed.back();
      int u=e.u,v=e.v,szu=e.szu;
      fa[v]=v;
      sz[u]=szu;
      kirjed.pop_back();
      setCnt++;
    }
    kirjed.pop_back();
  }
  int getCnt(){//Komponentide arv
    return setCnt;
  }
};
typedef pair<int,int>pii;
struct OP{
  int u,v,ind;
  bool numCmps;
  OP(int u,int v):u(u),v(v),numCmps(false){}
  OP(int ind)://p real pos, ind=num of query
    ind(ind),numCmps(true){}
};
vector<int>res;
struct QueryTree{
  vector<vector<OP>>t;
  DSU dsu;
  int n,m;
  QueryTree(int n,int m):dsu(n),n(n),m(m),t(4*m+4){}
  void addQuery(int i,int l,int r,int se,int en, OP& o){
    if(se<=l&&r<=en){
      t[i].push_back(o);
      return;
    }
    int m=(l+r)/2;
    if(en<=m)addQuery(i*2,l,m,se,en,o);
    else if(m<se)addQuery(i*2+1,m+1,r,se,en,o);
    else{
      addQuery(i*2,l,m,se,en,o);
      addQuery(i*2+1,m+1,r,se,en,o);
    }
  }
  void addQuery(OP o,int l,int r){
    addQuery(1,0,m-1,l,r,o);
  }
  void dfs(int i,int l,int r){
    dsu.persist();
    for(OP &o:t[i]){
      if(!o.numCmps){dsu.uhenda(o.u,o.v);}
    }
    if(l==r){
      for(OP &o:t[i]){
        if(o.numCmps)res[o.ind]=dsu.getCnt();
      }
    }else{
      int m=(l+r)/2;
      dfs(i*2,l,m);
      dfs(i*2+1,m+1,r);
    }
    dsu.rollback();
  }
  void solve(){
    dfs(1,0,m-1);
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,m;
  cin>>n>>m;
  map<pii,int>latest;
  int qcnt=0;
  QueryTree T(n,m+2);
  for(int i=0;i<m;i++){
    char op;
    int u,v;
    cin>>op;
    if(op=='?'){
      T.addQuery(OP(qcnt++),i,i);
    }else if(op=='+'){
      cin>>u>>v;
      if(u>v)swap(u,v);
      u--;v--;
      latest[{u,v}]=i;
    }else{
      cin>>u>>v;
      if(u>v)swap(u,v);
      u--;v--;
      int r=i;
      int l=latest[{u,v}];
```

```
      T.addQuery(OP(u,v),l,r);
      latest.erase({u,v});
    }
  }
  res.resize(qcnt);
  for(auto el:latest)T.addQuery(
    OP(el.first.first,el.first.second),el.second,m);
  T.solve();
  for(int i=0;i<qcnt;i++)cout<<res[i]<<'\n';
  return 0;
}
```

DFS

```
struct DFS{//assume graph G
  //start, end and indexes of all nodes
  vector<int>st,en,ind;
  int pos;
  DFS(int sz,int root):st(sz),en(sz),ind(sz){
    pos=0;
    dfs(root);
  }
  void dfs(int u){
    st[u]=pos;
    ind[pos]=u;
    pos++;
    for(auto v:g[u])dfs(v);
    en[u]=pos-1;
  }
};
```

Mo on trees

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
vector<int> compress(vector<T>&a){
  int n=a.size();
  set<T>vls;
  for(auto el:a)vls.insert(el);
  map<T,int>mp;
  int ps=0;
  for(auto el:vls)mp[el]=ps++;
  vector<int>b(n);
  for(int i=0;i<n;i++)b[i]=mp[a[i]];
  return b;
}
class Mo{
  struct Query{
    int l,r,lca;
  };
  int n,m,curDfsPos=0;
  vector<vector<int>>par;
  vector<int>st,en,dfsOrd,dep;
  vector<Query>qs;
public:
  Mo(vector<vector<int>>&g):n(g.size()){//numbered [0,n)
    m=log2(n)+1;
    par.resize(m,vector<int>(n));
    st.resize(n);
    en.resize(n);
    dep.resize(n);
    dfs(0,-1,g);
    for(int j=1;j<m;j++){
     for(int i=0;i<n;i++)par[j][i]=par[j-1][par[j-1][i]];
    }
//for(int i=0;i<n;i++)cout<<i<<' '<<st[i]<<' '<<en[i]...
  }
  void dfs(int u,int pr,vector<vector<int>>&g){
    dfsOrd.push_back(u);
    st[u]=curDfsPos++;
    for(auto v:g[u]){
      if(v==pr)continue;
      par[0][v]=u;
      dep[v]=dep[u]+1;
      dfs(v,u,g);
    }
    dfsOrd.push_back(u);
    en[u]=curDfsPos++;
  }
  int lca(int u,int v){
    if(dep[u]<dep[v])swap(u,v);
    for(int i=m-1;i>=0;i--){
      if(dep[u]-(1<<i)>=dep[v]){
        u=par[i][u];
      }
    }
    if(u==v)return u;
    for(int i=m-1;i>=0;i--){
      if(par[i][u]!=par[i][v]){
        u=par[i][u];
        v=par[i][v];
      }
    }
    return par[0][u];
  }
  void add_q(int u,int v){
    int lc=lca(u,v);
    if(st[u]>st[v])swap(u,v);
    Query q;
    if(lc==u)q={st[u],st[v],lc};
    else q={en[u],st[v],lc};
    qs.push_back(q);
  }
  template<typename A,typename O>
  void build(const A &a,const O &out){
    int q=qs.size();
    int bs=n/min<int>(n,sqrt(q));
    vector<int>num(q);
    iota(begin(num),end(num),0);
    sort(begin(num),end(num),[&](int a,int b){
      int ab=qs[a].l/bs,bb=qs[b].l/bs;
      if(ab!=bb)return ab<bb;
      return qs[a].r<qs[b].r;
    });
    int l=0,r=-1;
    for(auto ind:num){
      while(l<qs[ind].l)a(dfsOrd[l++]);
      while(l>qs[ind].l)a(dfsOrd[--l]);
      while(r<qs[ind].r)a(dfsOrd[++r]);
      while(r>qs[ind].r)a(dfsOrd[r--]);
      if(st[qs[ind].lca]!=qs[ind].l)a(qs[ind].lca);
      out(ind);
      if(st[qs[ind].lca]!=qs[ind].l)a(qs[ind].lca);
    }
  }
};
```

Parallel binary search

```
void calc(int l,int r,vector<int>&candidates,BIT &b){
    if(l+1==r||candidates.empty()){
        for(auto c:candidates)ans[c]=l;
        return;
    }
    int m=(l+r)/2;
    for(int i=l;i<m;i++){//append updates [l,mid)
        auto [l,r,a]=events[i];
        b.upd(l,r,a);
    }
    //split candidates into 2 parts
    vector<int>ok,nok;
    for(auto c:candidates){
        ll sum=0;
        for(auto land:lands[c])sum+=b.q(land);
        if(sum>=target[c])ok.push_back(c);
        else{
            target[c]-=sum;
            nok.push_back(c);
        }
    }
    //undo bit updates
     for(int i=l;i<m;i++){//append updates [l,mid)
        auto [l,r,a]=events[i];
        b.upd(l,r,-a);
    }
    calc(l,m,ok,b);vector<int>().swap(ok);//clear mem
    calc(m,r,nok,b);vector<int>().swap(nok);
}
```

DSU

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct DSU{
    vector<int>p,sz;
```

```
    DSU(int n){
        p.resize(n);
        sz.resize(n);
        for(int i=0;i<n;i++)p[i]=i;
    }
    int F(int u){
        if(u==p[u])return u;
        return p[u]=F(p[u]);
    }
    void unite(int u,int v){
        u=F(u);
        v=F(v);
        if(u==v)return;
        if(sz[u]>sz[v])swap(u,v);
        if(sz[u]==sz[v])sz[v]++;
        p[u]=v;
    }
    bool same_comp(int u,int v){
        return F(u)==F(v);
    }
};
```

Heavy-light decomposition

```
vector<int>g[N];
class HLD{
    int n,ind;
    vector<int>sz,biggestN,pos,dep,par,head,en;
    F fen;
    void build(int u,int pr){
        sz[u]++;
        int biggestSz=0;
        for(auto v:g[u]){
            if(v==pr)continue;
            dep[v]=dep[u]+1;
            par[v]=u;
            build(v,u);
            sz[u]+=sz[v];
            if(sz[v]>biggestSz){
                biggestSz=sz[v];
                biggestN[u]=v;
            }
        }
    }
    void dfs(int u,int clss){
        pos[u]=ind++;
        head[u]=clss;
        if(biggestN[u]!=-1)dfs(biggestN[u],clss);
        for(auto v:g[u]){
            if(v==par[u]||v==biggestN[u])continue;
            dfs(v,v);
        }
        en[u]=ind-1;
    }
    public:
    HLD(int n_arg):fen(n_arg){
        n=n_arg;
        ind=1;
        sz.resize(n);
        biggestN=vector<int>(n,-1);
        pos.resize(n);
        dep.resize(n);
        par.resize(n);
        head.resize(n);
        en.resize(n);
        build(0,-1);
        dfs(0,0);
    }
    ll query(int a,int b){
        ll ans=0;
        while(head[a]!=head[b]){
            if(dep[head[a]]<dep[head[b]])swap(a,b);
            ll val=fen.get(pos[head[a]],pos[a]);
            ans+=val;
            a=par[head[a]];
        }
        if(pos[a]>pos[b])swap(a,b);
        ll val=fen.get(pos[a],pos[b]);
        ans+=val;
        return ans;
    }
    ll getSubTreeSum(int a){
        return fen.get(pos[a],en[a]);
    }
    void add(int a,int x){
        fen.add(pos[a],x);
    }
};
```

DSU on tree sack

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct Sack{
  vector<vector<int>>g;
  int n,lef,rig;
  F fenwick;
  ll res;
  vector<int>dep,sz;
  vector<vector<int>*>alampuuTipud;
  Sack(vector<vector<int>>&g,int L,int R,int root=0):
    g(g),fenwick(g.size()+1),res(0){
    n=g.size();
    dep.resize(n,1);
    sz.resize(n);
    alampuuTipud.resize(n);
    calcSize(root);
    lef=n-1-R;
    rig=n-1-L;
    dfs(root);
//for(i=0;i<n;i++)cout<<i<<' '<<dep[i]<<' '<<sz[i]<<endl;
  }
  void calcSize(int u,int pr=-1){
    sz[u]++;
    for(auto v:g[u]){
      if(v==pr)continue;
      dep[v]=dep[u]+1;
      calcSize(v,u);
      sz[u]+=sz[v];
    }
  }
  void dfs(int u,int pr=0,bool keep=false){
    int mx=-1,bigChild=-1;
    for(auto v:g[u]){
      if(v!=pr && sz[v]>mx)
        mx=sz[v],bigChild=v;
    }
    for(auto v:g[u]){
      if(v!=pr && v!=bigChild)dfs(v,u,0);
    }
    if(bigChild!=-1)dfs(bigChild,u,1),alampuuTipud[u]=
                                    alampuuTipud[bigChild];
    else alampuuTipud[u]=new vector<int>();
    alampuuTipud[u]->push_back(u);
    res+=fenwick.get(lef+2*dep[u]-dep[u],
                     rig+2*dep[u]-dep[u]);
    fenwick.add(dep[u],1);
    for(auto v:g[u]){
      if(v==pr || v==bigChild)continue;
      for(auto tipp:*alampuuTipud[v]){
        res+=fenwick.get(lef+2*dep[u]-dep[tipp],
                         rig+2*dep[u]-dep[tipp]);
      }
      for(auto tipp:*alampuuTipud[v]){
        fenwick.add(dep[tipp],1);
        alampuuTipud[u]->push_back(tipp); //NB
      }
    }
    if(keep==0){
      for(auto v:*alampuuTipud[u])fenwick.add(dep[v],-1);
    }
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int t;
  cin>>t;
  while(t--){
    int n,l,r;
    cin>>n>>l>>r;
    vector<vector<int>>g(n);
    for(int i=1;i<n;i++){
      int x,y;cin>>x>>y;
      x--;y--;
      g[x].push_back(y);
      g[y].push_back(x);
    }
    Sack S(g,l,r);
    cout<<S.res<<'\n';
  }
```

```
  return 0;
}
```

Treap

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct T{
    T *l=0,*r=0;
    int val=0,prior=0,sz=1;
    ll sum;
    bool rev=0;
    T(){};
    T(int V,int P){
        val=V;prior=P;
        sum=V;
    }
};
int SUM(T* RT){
    if(RT)return RT->sz;
    return 0;
}
ll SSUM(T* RT){
    if(RT)return RT->sum;
    return 0;
}
void upd(T* RT){
    RT->sz=SUM(RT->l)+SUM(RT->r)+1;
    RT->sum=SSUM(RT->l)+SSUM(RT->r)+RT->val;
}
void push(T* RT){
    if(RT->rev){
        swap(RT->l,RT->r);
        if(RT->l)RT->l->rev=1-RT->l->rev;
        if(RT->r)RT->r->rev=1-RT->r->rev;
        RT->rev=0;
    }
}
T* root=0;
typedef pair<T*,T*>PTT;
T* MERGE(T* L,T* R){
    if(!L&&!R)return 0;
    if(!L)return R;
    if(!R)return L;
    push(L);
    push(R);
    if(L->prior>R->prior){
        L->r=MERGE(L->r,R);
        upd(L);
        return L;
    }else{
        R->l=MERGE(L,R->l);
        upd(R);
        return R;
    }
}
PTT SPLIT_BY_CNT(T* RT,int sz){
    if(!RT)return {0,0};
    push(RT);
    if(SUM(RT->l)+1<=sz){
        PTT Q=SPLIT_BY_CNT(RT->r,sz-SUM(RT->l)-1);
        RT->r=Q.first;
        upd(RT);
        return {RT,Q.second};
    }else{
        PTT Q=SPLIT_BY_CNT(RT->l,sz);
        RT->l=Q.second;
        upd(RT);
        return {Q.first,RT};
    }
}
PTT SPLIT_BY_KEY(T* RT,int key){
    if(!RT)return {0,0};
    push(RT);
    if(RT->val<=key){
        PTT Q=SPLIT_BY_KEY(RT->r,key);
        RT->r=Q.first;
        upd(RT);
       return {RT,Q.second};
    }else{
       PTT Q=SPLIT_BY_KEY(RT->l,key);
        RT->l=Q.second;
        upd(RT);
        return {Q.first,RT};
    }
}
T* RT=0;
void INS_LAST(int x){
    //PTT Q=SPLIT(RT,);
    T* N=new T(x,rand());
    RT=MERGE(RT,N);
}
void U2(int l,int r){
    PTT Q1=SPLIT_BY_CNT(RT,r);
    PTT Q2=SPLIT_BY_CNT(Q1.first,l);
    Q2.second->rev^= 1;
    RT=MERGE(Q2.first,MERGE(Q2.second,Q1.second));
}
/*
void U1(int l,int r){
    PTT Q1=SPLIT_BY_CNT(RT,r);
    PTT Q2=SPLIT_BY_CNT(Q1.first,l);
    PTT Q3=SPLIT_BY_CNT(Q2.second,(r-l-1));
    RT=MERGE(Q2.first,MERGE(Q3.second,
                            MERGE(Q3.first,Q1.second)));
}
int get(int pos){
    PTT Q1=SPLIT_BY_CNT(RT,pos);
    PTT Q2=SPLIT_BY_CNT(Q1.first,pos-1);
    int ans=Q2.second->key;
    RT=MERGE(Q2.first,MERGE(Q2.second,Q1.second));
    return ans;
}*/
ll getSum(int l,int r){
    PTT Q1=SPLIT_BY_CNT(RT,r);
    PTT Q2=SPLIT_BY_CNT(Q1.first,l);
    ll res=Q2.second->sum;
    RT=MERGE(Q2.first,MERGE(Q2.second,Q1.second));
    return res;
}
void print(T* RT){
    if(!RT)return;
    push(RT);
    print(RT->l);
    cout<<RT->val<<' ';
    print(RT->r);
}
```

Online Li Chao tree

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T,T low,T high>
struct LiChao{
  struct Joon{
    T m,b;
    Joon(T m,T b):m(m),b(b){}
    ll operator()(T x)const{return m*x+b;}
  };
  static constexpr T inf=numeric_limits<T>::max();
  struct Tipp{
    Joon j;
    Tipp *l,*r;
    Tipp(const Joon &j):j{j},l{nullptr},r{nullptr}{}
  };
  Tipp* juur;
  LiChao():juur{nullptr}{}
  Tipp* lisa_joon(Tipp* t,Joon j,T l,T r){
    if(!t)return new Tipp(j);
    T jl=j(l),jr=j(r);
    T tl=t->j(l),tr=t->j(r);
    if(tl<=jl&&tr<=jr){
      return t;
    }else if(tl>=jl&&tr>=jr){
      t->j=j;
      return t;
    }else{
      T m=(l+r)/2;
      if(m==r)--m;
      T tm=t->j(m),jm=j(m);
      if(tm>jm){
        swap(t->j,j);
        if(jl>=tl)t->l=lisa_joon(t->l,j,l,m);
        else t->r=lisa_joon(t->r,j,m+1,r);
      }else{
        if(tl>=jl)t->l=lisa_joon(t->l,j,l,m);
        else t->r=lisa_joon(t->r,j,m+1,r);
      }
      return t;
```

```
    }
  }
  void lisa_joon(T a,T b){
    Joon j(a,b);
    juur=lisa_joon(juur,j,low,high);
  }
  Tipp* lisa_segment(Tipp *t,Joon j,T se,T en,T l,T r){
    if(r<se||en<l)return t;
    if(se<=l&&r<=en){
      return lisa_joon(t,j,l,r);
    }
    if(t){
      T jl=j(l),jr=j(r);
      T tl=t->j(l),tr=t->j(r);
      if(tl<=jl&&tr<=jr)return t;
    }else{
      t=new Tipp(Joon(0,inf));
    }
    T m=(l+r)/2;
    if(m==r)m--;
    t->l=lisa_segment(t->l,j,se,en,l,m);
    t->r=lisa_segment(t->r,j,se,en,m+1,r);
    return t;
  }
  void lisa_segment(T l,T r,T a,T b){//[l,r] y=a*x+b
    Joon j(a,b);
    juur=lisa_segment(juur,j,l,r,low,high);
  }
  T saa(const Tipp *t,T l,T r,T x) const {
    if(!t)return inf;
    if(l==r)return t->j(x);
    T m=(l+r)/2;
    if(m==r)m--;
    if(x<=m)return min(t->j(x),saa(t->l,l,m,x));
    else return min(t->j(x),saa(t->r,m+1,r,x));
  }
  T saa(const T &x) const {
    return saa(juur,low,high,x);
  }
};
```

Link-cut tree

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
//F on assotsiatiivne op, S on ümberpööramise funktsioon
template<typename T,typename F,typename S>
struct LinkCutTree{
  F f;
  S s;
  struct Tipp{
    Tipp *l,*r,*p;
    T voti,summa;
    bool rev;
    int suurus;
    Tipp(const T &v):voti(v),summa(v),suurus(1),
          rev(false),l(nullptr),r(nullptr),p(nullptr){}
    bool on_juur(){
      return  !p or (p->l!=this && p->r!=this);
    }
  };
  using TP=Tipp*;
  TP uuenda(TP t){
    t->suurus=1;
    t->summa=t->voti;
    if(t->l)t->suurus+=t->l->suurus,t->summa=
              f(t->l->summa,t->summa);
    if(t->r)t->suurus+=t->r->suurus,t->summa=
              f(t->summa,t->r->summa);
    return t;
  }
  void parempoore(TP t){
    TP x=t->p,y=x->p;
    if((x->l=t->r))t->r->p=x;
    t->r=x,x->p=t;
    uuenda(x),uuenda(t);
    if((t->p=y)){
      if(y->l==x)y->l=t;
      if(y->r==x)y->r=t;
      uuenda(y);
    }
  }
  void vasakpoore(TP t){
    TP x=t->p,y=x->p;
    if((x->r=t->l))t->l->p=x;
    t->l=x,x->p=t;
    uuenda(x),uuenda(t);
    if((t->p=y)){
      if(y->l==x)y->l=t;
      if(y->r==x)y->r=t;
      uuenda(y);
    }
  }
  void toggle(TP t){
    swap(t->l,t->r);
    t->summa=s(t->summa);
    t->rev ^= true;
  }
  void alla(TP t){
    if(t->rev){
      if(t->l)toggle(t->l);
      if(t->r)toggle(t->r);
      t->rev=false;
    }
  }
  void splay(TP t){
    alla(t);
    while(!t->on_juur()){
      TP q=t->p;
      if(q->on_juur()){
        alla(q),alla(t);
        if(q->l==t)parempoore(t);
        else vasakpoore(t);
      }else{
        TP r=q->p;
        alla(r),alla(q),alla(t);
        if(r->l==q){
          if(q->l==t)parempoore(q),parempoore(t);
          else vasakpoore(t),parempoore(t);
        }else{
          if(q->r==t)vasakpoore(q),vasakpoore(t);
          else parempoore(t),vasakpoore(t);
        }
      }
    }
  }
  LinkCutTree(const F &f,const S &s):f(f),s(s){}
  TP alloc(const T&v=T()){
    return new Tipp(v);
  }
  vector<TP> build(vector<T>&vec){
    vector<TP>tipud(vec.size());
    for(int i=0;i<vec.size();i++)tipud[i]=alloc(vec[i]);
    return tipud;
  }
  TP expose(TP t){
    TP rp=nullptr;
    for(TP cur=t;cur;cur=cur->p){
      splay(cur);
      cur->r=rp;
      uuenda(cur);
      rp=cur;
    }
    splay(t);
    return rp;
  }
  void evert(TP t){
    expose(t);
    toggle(t);
    alla(t);
  }
  void link(TP child,TP parent){
    if(is_connected(child,parent)){
      throw runtime_error("child and par diff comp");
    }
    if(child->l){
      throw runtime_error("child must be root");
    }
    child->p=parent;
    parent->r=child;
    uuenda(parent);
  }
  void cut(TP child){
    expose(child);
    TP parent=child->l;
    if(!parent)throw runtime_error("child == root??");
    child->l=nullptr;
    parent->p=nullptr;
    uuenda(child);
  }
```

```
  bool is_connected(TP u,TP v){
    expose(u),expose(v);
    return u==v or u->p;
  }
  TP lca(TP u,TP v){
    if(!is_connected(u,v))return nullptr;
    expose(u);
    return expose(v);
  }
  TP get_kth(TP x,int k){
    expose(x);
    while(x){
      alla(x);
      if(x->r&&x->r->suurus>k){
        x=x->r;
      }else{
        if(x->r)k-=x->r->suurus;
        if(k==0){
          splay(x);
          return x;
        }
        k-=1;
        x=x->l;
      }
    }
    return nullptr;
  }
  const T &query(TP u){
    expose(u);
    return u->summa;
  }
  const T &query(TP u,TP v){
    evert(u);
    return query(v);
  }
  void set_key(TP t,T v){
    expose(t);
    t->voti=v;
    uuenda(t);
  }
};
template< typename T, typename F, typename S >
LinkCutTree< T, F, S >
get_link_cut_tree(const F &f, const S &s) {
  return {f, s};
}
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,q;
  cin>>n>>q;
  auto liida=[](ll a,ll b){return a+b;};
  auto poora=[](ll a){return a;};
  auto lct=get_link_cut_tree<ll>(liida,poora);
  vector<ll>a(n);
  for(int i=0;i<n;i++)cin>>a[i];
  auto vs=lct.build(a);
  for(int i=1;i<n;i++){
    int a,b;cin>>a>>b;
    lct.evert(vs[a]);
    lct.link(vs[a],vs[b]);
  }
  while(q--){
    int t;cin>>t;
    if(t==0){
      int u,v,w,x;
      cin>>u>>v>>w>>x;
      lct.evert(vs[u]);
      lct.cut(vs[v]);
      lct.evert(vs[w]);
      lct.link(vs[w],vs[x]);
    }else if(t==1){
      int p,x;
      cin>>p>>x;
      lct.set_key(vs[p],vs[p]->voti+x);
    }else{
      int u,v;
      cin>>u>>v;
      cout<<lct.query(vs[u],vs[v])<<'\n';
    }
  }
  return 0;
}
```

Tetration

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
T POW(T a,T b,T mod){
    T res=1;
    while(b){
        if(b%2)res=(res*a)%mod;
        a=(a*a)%mod;
        b/=2;
    }
    return res%mod;
}
template<typename T>
T PHI(T n){
    T res=n;
    for(T i=2;i*i<=n;i++){
        if(n%i==0){
            res-=res/i;
            while(n%i==0)n/=i;
        }
    }
    if(n>1)res-=res/n;
    return res;
}
template<typename T>
T tetration(T a,T b,T mod){
    if(mod==1)return 0;
    if(a==0)return 1-b%2;
    if(b==0)return 1;
    if(b==1)return a%mod;
    if(b==2)return POW(a,a,mod);
    T phi=PHI(mod);
    T pw=tetration(a,b-1,phi);
    if(pw==0)pw+=phi;
    return POW(a,pw,mod);
}
```

Pollard rho factorization

```
#include <bits/stdc++.h>
//works for q=100,n=10**18, in 4.4s
typedef long long ll;
using namespace std;
struct MR{
  using u64 = uint64_t;
  using u128 = __uint128_t;
  static u64 binpower(u64 base, u64 e, u64 mod) {
    u64 result = 1;
    base %= mod;
    while (e) {
      if (e & 1)
        result = (u128)result * base % mod;
      base = (u128)base * base % mod;
      e >>= 1;
    }
    return result;
  }
 static bool check_composite(u64 n, u64 a, u64 d, int s){
    u64 x = binpower(a, d, n);
    if (x == 1 || x == n - 1)
      return false;
    for (int r = 1; r < s; r++) {
      x = (u128)x * x % n;
      if (x == n - 1)
        return false;
    }
    return true;
  };
  static bool MillerRabin(u64 n) { // is n prime?
    if (n < 2)
      return false;
    int r = 0;
    u64 d = n - 1;
    while ((d & 1) == 0) {
      d >>= 1;
      r++;
    }
    for (int a : {2,3,5,7,11,13,17,19,23,29,31,37}) {
      if (n == a)
        return true;
      if (check_composite(n, a, d, r))
        return false;
    }
    return true;
  }
```

```
};
struct Floyd{
  ll mult(ll a,ll b,ll MOD){
    return (__int128)a*b%MOD;
  }
  ll f(ll x,ll c,ll MOD){
    return (mult(x,x,MOD)+c)%MOD;
  }
  ll single_rho(ll n){
    if(n%2==0)return 2;
    if(MR::MillerRabin(n))return n;
    ll st=0;
    while(true){
      st++;
      ll x=st,y=f(x,st,n);
      while(true){
        ll g=__gcd(y-x+n,n);
        if(g==0||g==n)break;
        if(g!=1)return g;
        x=f(x,st,n);
        y=f(y,st,n);
        y=f(y,st,n);
      }
    }
  }
  vector<ll>rho(ll n){
    if(n==1)return {};
    ll res=single_rho(n);
    if(res==n)return {n};
    auto l=rho(res);
    auto r=rho(n/res);
    l.insert(l.end(),r.begin(),r.end());
    return l;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  Floyd f;
  int q;
  cin>>q;
  while(q--){
    ll a;
    cin>>a;
    vector<ll>facs=f.rho(a);
    sort(facs.begin(),facs.end()),
      cout<<facs.size()<<' ';
    for(auto el:facs)cout<<el<<' ';
    cout<<'\n';
  }
  return 0;
}
```

Python BigInt

```
import sys
input = sys.stdin.read#Use ctrl+d to end
sys.set_int_max_str_digits(0)
import decimal
with decimal.localcontext() as ctx:
    ctx.prec = decimal.MAX_PREC
    ctx.Emax = decimal.MAX_EMAX
    ctx.Emin = decimal.MIN_EMIN
    ctx.traps[decimal.Inexact] = 1
    inp=[decimal.Decimal(_) for _ in input().split()]
    ps=1
    for _ in range(int(inp[0])):
        a=inp[ps]
        b=inp[ps+1]
        ps+=2
        div=a//b
        mod=a-b*div
        print(div,mod)
```

Disjoint sparse table

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T,T op(T,T)>
struct DST{
    vector<vector<T>>table;
    DST(vector<T>& a){
        int n=a.size();
        int hei=0;
        while((1<<hei)<n)hei++;
        hei++;
        table.resize(hei,vector<T>(n));
        for(int i=0;i<n;i++)table[0][i]=a[i];
        for(int i=1;i<hei;i++){
            int ln=1<<i;
            for(int cen=0;cen+ln/2<n;cen+=ln){
                int m=cen+ln/2;
                table[i][m-1]=a[m-1];
                for(int j=m-2;j>=cen;j--)
                  table[i][j]=op(a[j],table[i][j+1]);
                table[i][m]=a[m];
                for(int j=m+1;j<min(cen+ln,n);j++)
                  table[i][j]=op(table[i][j-1],a[j]);
            }
        }
    }
    T get(int l,int r){//0 based [l,r]
        if(l==r)return table[0][l];
        int i=32-__builtin_clz(l xor r);
        return op(table[i][l],table[i][r]);
    }
};
```

SCC

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
class SCC{
    vector<vector<int>>g,rg;
    int n;
public:
    SCC(int n):n(n){//0 based
        g.resize(n);
        rg.resize(n);
    }
    void add_edge(int a,int b){
        g[a].push_back(b);
        rg[b].push_back(a);
    }
    vector<vector<int>>build(){
        vector<int>vis(n),nodes;
        for(int i=0;i<n;i++)dfs(i,g,vis,nodes,false);
        vector<vector<int>>res;
        vector<int>vis2(n);
        while(nodes.size()){
            int u=nodes.back();nodes.pop_back();
            if(!vis2[u]){
                vector<int>tipud;
                dfs(u,rg,vis2,tipud,true);
                reverse(tipud.begin(),tipud.end());
                res.push_back(tipud);
            }
        }
        return res;
    }
    void dfs(int u,vector<vector<int>>&g,vector<int>&vis,
             vector<int>&ns,bool add_first){
        if(vis[u])return;
        vis[u]=1;
        if(add_first)ns.push_back(u);
        for(auto v:g[u])dfs(v,g,vis,ns,add_first);
        if(!add_first)ns.push_back(u);
    }
};
```

Online bridges

```
#include <bits/stdc++.h>
//source
// cp-algorithms.com/graph/bridge-searching-online.html
typedef long long ll;
using namespace std;
struct DSU{
    vector<int>p,sz;
    DSU(int n):p(n),sz(n,1){iota(begin(p),end(p),0);}
    int F(int u){
        if(u==-1)return -1;
        if(u==p[u])return u;
        return p[u]=F(p[u]);
    }
};
class Bridges{
    int n,bridges,lca_it;
    DSU twoEdge,simpleConn;
    vector<int>par,last_vis;
```

```
public:
Bridges(int n):n(n),bridges(0),lca_it(0),twoEdge(n),
               simpleConn(n),par(n,-1),last_vis(n){}
void makeRoot(int u){
    u=twoEdge.F(u);
    int root=u;
    int ch=-1;
    while(u!=-1){
        int p=twoEdge.F(par[u]);
        par[u]=ch;
        simpleConn.p[u]=root;
        ch=u;
        u=p;
    }
    simpleConn.sz[root]=simpleConn.sz[ch];
}
void mergePath(int u,int v){
    lca_it++;
    vector<int>pu,pv;
    int lca=-1;
    while(true){
        if(u!=-1){
            u=twoEdge.F(u);
            pu.push_back(u);
            if(last_vis[u]==lca_it){
                lca=u;
                break;
            }
            last_vis[u]=lca_it;
            u=par[u];
        }
        if(v!=-1){
            v=twoEdge.F(v);
            pv.push_back(v);
            if(last_vis[v]==lca_it){
                lca=v;
                break;
            }
            last_vis[v]=lca_it;
            v=par[v];
        }
    }
    for(auto a:pu){
        twoEdge.p[a]=lca;
        if(a==lca)break;
        bridges--;
    }
    for(auto a:pv){
        twoEdge.p[a]=lca;
        if(a==lca)break;
        bridges--;
    }
}
void addEdge(int u,int v){
    u=twoEdge.F(u);
    v=twoEdge.F(v);
    if(u==v)return;
    int cu=simpleConn.F(u),cv=simpleConn.F(v);
    if(cu!=cv){
        bridges++;
        if(simpleConn.sz[cu]>simpleConn.sz[cv]){
            swap(u,v);
            swap(cu,cv);
        }
        makeRoot(u);
        par[u]=simpleConn.p[u]=v;
        simpleConn.sz[cv]+=simpleConn.sz[u];
    }else mergePath(u,v);
}
int numOfBridges(){
    return bridges;
}
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int t;
    cin>>t;
    while(t--){
        int n,m;
        cin>>n>>m;
        Bridges t(n);
        while(m--){
            int u,v;cin>>u>>v;
            t.addEdge(u,v);
            cout<<t.numOfBridges()<<'\n';
        }
    }
    return 0;
}
```

Manhattan MST

```
#include <bits/stdc++.h>
// source
// ei1333.github.io/library/graph/mst/manhattan-mst.hpp
typedef long long ll;
using namespace std;
struct Edge{
    int u,v;
    ll c;
    Edge(int u,int v,ll c):u(u),v(v),c(c){}
};
vector<Edge> calcNeighs(vector<int>&x,vector<int>&y){
    int n=x.size();
    vector<Edge>edges;
    vector<int>ord(n);
    iota(begin(ord),end(ord),0);
    for(int t1=0;t1<2;t1++){
        for(int t2=0;t2<2;t2++){
            sort(begin(ord),end(ord),[&](int i,int j){
              return x[i]+y[i]<x[j]+y[j];});
            map<int,int>mp;
            for(int i:ord){
                for(auto it=mp.lower_bound(-y[i]);
                    it!=mp.end();it=mp.erase(it)){
                auto j=it->second;
                if(x[i]-x[j]<y[i]-y[j])break;
                edges.push_back({i,j,
                    abs(x[i]-x[j])+abs(y[i]-y[j])});
                }
                mp[-y[i]]=i;
            }
            for(int i=0;i<n;i++)x[i]*=-1;
        }
        swap(x,y);
    }
    return edges;
}
struct DSU{
    vector<int>p,sz;
    DSU(int n){
        p.resize(n);
        sz.resize(n);
        for(int i=0;i<n;i++)p[i]=i;
    }
    int F(int u){
        if(u==p[u])return u;
        return p[u]=F(p[u]);
    }
    bool unite(int u,int v){
        u=F(u);
        v=F(v);
        if(u==v)return false;
        if(sz[u]>sz[v])swap(u,v);
        if(sz[u]==sz[v])sz[v]++;
        p[u]=v;
        return true;
    }
    bool same_comp(int u,int v){
        return F(u)==F(v);
    }
};
pair<ll,vector<pair<int,int>>>
kruskal(int n,vector<Edge>&edges){
    sort(begin(edges),end(edges),[&](Edge& a,Edge& b){
      return a.c<b.c;});
    DSU t(n);
    ll sum=0;
    vector<pair<int,int>>paarid;
    for(Edge& a:edges){
        if(t.unite(a.u,a.v)){
            sum+=a.c;
            paarid.push_back({a.u,a.v});
        }
    }
    return {sum,paarid};
}
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int n;
    cin>>n;
```

```
    vector<int>x(n),y(n);
    for(int i=0;i<n;i++)cin>>x[i]>>y[i];
    vector<Edge>neighs=calcNeighs(x,y);
    auto res=kruskal(n,neighs);
    cout<<res.first<<'\n';
    for(auto el:res.second)
      cout<<el.first<<' '<<el.second<<'\n';
    return 0;
}
```

Directed MST

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
typedef pair<ll,ll> pii;
const int N=200005;
struct Edge{
  int u,v;
  ll c;
  Edge():u(0),v(0),c(0){}
  Edge(int u,int v,ll c):u(u),v(v),c(c){}
};
struct DSU{
  vector<int>p,sz;
  DSU(int n){
    p.resize(n);
    sz.resize(n);
    for(int i=0;i<n;i++)p[i]=i;
  }
  int F(int u){
    if(u==p[u])return u;
    return p[u]=F(p[u]);
  }
  bool unite(int u,int v){
    u=F(u);
    v=F(v);
    if(u==v)return false;
    if(sz[u]>sz[v])swap(u,v);
    if(sz[u]==sz[v])sz[v]++;
    p[u]=v;
    return true;
  }
  void setPar(int u,int v){
    p[u]=v;
  }
  bool same_comp(int u,int v){
    return F(u)==F(v);
  }
};
template<typename T,typename L>//Edge, lazy type
struct SkewHeap{
  struct Tipp{
    Tipp* ch[2]={nullptr,nullptr};
    T voti;
    L lazy;
    Tipp(T voti, L lazy):voti(voti),lazy(lazy){}
  }*juur;
  SkewHeap():juur(nullptr){}
  void lukka(Tipp *t){
    t->voti.c+=t->lazy;
    if(t->ch[0])t->ch[0]->lazy+=t->lazy;
    if(t->ch[1])t->ch[1]->lazy+=t->lazy;
    t->lazy=0;
  }
  Tipp* uhenda(Tipp *a,Tipp *b){
    if(!a)return b;
    if(!b)return a;
    lukka(a);lukka(b);
    if(a->voti.c>b->voti.c)swap(a,b);
    a->ch[1]=uhenda(b,a->ch[1]);
    swap(a->ch[0],a->ch[1]);
    return a;
  }
  void lisa(T voti){
    Tipp *t=new Tipp(voti,0);
    juur=uhenda(juur,t);
  }
  void pop(){
    lukka(juur);
    juur=uhenda(juur->ch[0],juur->ch[1]);
  }
  T top(){
    lukka(juur);
    return juur->voti;
  }
  bool empty(){
    return !juur;
  }
  void lisa(L lazy){
    if(empty())return;
    juur->lazy+=lazy;
  }
  void uhenda(SkewHeap t){
    juur=uhenda(juur,t.juur);
  }
};
//0 indekseeritud
//kahe tipupaari vahel olgu max 1 serv
struct DirectedMST{
  int n;
  DirectedMST(int n):n(n){}
  vector<Edge>servad;
  void lisaServ(int s,int t,int c){
    servad.push_back({s,t,c});
  }
  pair<ll,vector<int>> lahenda(int juur){
    int N=2*n-1;
    DSU dsu(N);
    vector<int>u(N,-1),par(N,-1);
    Edge parEdge[N];
    vector<int>child[N];
    vector<SkewHeap<Edge,ll>>kuhjad(N);
    for(Edge& e:servad)kuhjad[e.v].lisa(e);
    for(int i=0;i<n;i++)kuhjad[(i+1)%n].lisa(
                        {i,(i+1)%n,LLONG_MAX/2});
    int pea=0;
    int esimeneVaba=n;
    while(!kuhjad[pea].empty()){
      auto e=kuhjad[pea].top();kuhjad[pea].pop();
      int nxHead=dsu.F(e.u);
      if(nxHead==pea)continue;
      u[pea]=nxHead;
      parEdge[pea]=e;
      if(u[nxHead]==-1)pea=nxHead;
      else{
        int j=nxHead;
        do{
          kuhjad[j].lisa(-parEdge[j].c);
          kuhjad[esimeneVaba].uhenda(kuhjad[j]);
          child[esimeneVaba].push_back(j);
          j=dsu.F(u[j]);
        }while(j!=nxHead);
        for(auto u:child[esimeneVaba])
          par[u]=esimeneVaba,dsu.setPar(u,esimeneVaba);
        pea=esimeneVaba++;
      }
    }
    vector<int>ulemus(N,-1);
    deque<int>q;q.push_back(juur);
    ll koguKaal=0;
    while(!q.empty()){
      int u=q.front();q.pop_front();
      while(par[u]!=-1){
        for(auto v:child[par[u]]){
          if(v!=u){
            ulemus[parEdge[v].v]=parEdge[v].u;
            q.push_back(parEdge[v].v);
            par[v]=-1;
          }
        }
        u=par[u];
      }
    }
    for(auto &e:servad){
      if(ulemus[e.v]==e.u)koguKaal+=e.c;
    }
    ulemus[juur]=juur;ulemus.resize(n);
    return {koguKaal,ulemus};
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,m,s;
  cin>>n>>m>>s;
  DirectedMST t(n);
  while(m--){
    int a,b,c;
    cin>>a>>b>>c;
    t.lisaServ(a,b,c);
  }
```

```
  auto P=t.lahenda(s);
  cout<<P.first<<'\n';
  for(auto &el:P.second)cout<<el<<' ';
  return 0;
}
```

Bipartite matching (Hopcroft-Karp)

```
#include <bits/stdc++.h>
//source: https://anz1217.tistory.com/50
typedef long long ll;
using namespace std;
struct BMATCH{
  int left,right,flow=0;
  vector<vector<int>>g;
  vector<int>matchFromLeft,matchFromRight,dist;
  BMATCH(int l,int r):left(l),right(r),g(l),
    matchFromLeft(l,-1),matchFromRight(r,-1),dist(l){}
  void add(int u,int v){
    g[u].push_back(v);
  }
  void bfs(){
    queue<int>q;
    for(int i=0;i<left;i++){
      if (!~matchFromLeft[i]){
        q.push(i);
        dist[i]=0;
      }else dist[i]=-1;
    }
    while(!q.empty()){
      int u=q.front();q.pop();
      for(auto v:g[u]){
        if(~matchFromRight[v]&&!~dist[matchFromRight[v]]){
          dist[matchFromRight[v]]=dist[u]+1;
          q.push(matchFromRight[v]);
        }
      }
    }
  }
  bool dfs(int u){
    for(auto v:g[u]){
      if(!~matchFromRight[v]){
        matchFromLeft[u]=v;matchFromRight[v]=u;
        return true;
      }
    }
    for(auto v:g[u]){
      if(dist[matchFromRight[v]]==dist[u]+1
         &&dfs(matchFromRight[v])){
        matchFromLeft[u]=v,matchFromRight[v]=u;
        return true;
      }
    }
    return false;
  }
  int getMaxMatching(){
    while(true){
      bfs();
      int cur=0;
      for(int u=0;u<left;u++){
        if(!~matchFromLeft[u])cur+=dfs(u);
      }
      if(!cur)break;
      flow+=cur;
    }
    return flow;
  }
  pair<vector<int>,vector<int>> minVertexCover(){
    vector<int>L,R;
    for(int i=0;i<left;i++){
      if(!~dist[i])L.push_back(i);
      else if(~matchFromLeft[i])R.push_back(
                                  matchFromLeft[i]);
    }
    return {L,R};
  }
  vector<pair<int,int>> getEdges(){
    vector<pair<int,int>>res;
    for(int i=0;i<left;i++){
      if(matchFromLeft[i]!=-1)res.push_back(
                                {i,matchFromLeft[i]});
    }
    return res;
  }
};
```

Single source shortest path (V*max w + E)

```
#include <bits/stdc++.h>
//source https://blog.naver.com/jinhan814/222511581119
//works well for small integer edge weights
//complexity O(V*maxw+E)
typedef long long ll;
using namespace std;
template<typename Graph>
vector<int>Dial(Graph& g,int s,int maxw){//edges [0,maxw)
    vector<vector<int>>qs(maxw);
    vector<int>dist(g.size(),-1);
    dist[s]=0;
    qs[0].push_back(s);
    for(int d=0,maxd=0;d<=maxd;d++){
        for(auto& q=qs[d%maxw];q.size();){
            auto u=q.back();q.pop_back();
            if(dist[u]!=d)continue;
            for(auto el:g[u]){
                auto v=el.first,w=el.second;
                if(dist[v]!=-1 && dist[v]<=d+w)continue;
                dist[v]=d+w;
                qs[(d+w)%maxw].push_back(v);
                maxd=max(maxd,d+w);
            }
        }
    }
    return dist;
}
```

Two edge connected components

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
typedef pair<int,int> pii;
class TwoEdgeConnectedComponents{
    vector<vector<pii>>g;
    int n;
    int ind=0;
    set<int>bridges;
public:
    TwoEdgeConnectedComponents(int n):n(n){
        g.resize(n);
    }
    void add_edge(int a,int b,int i){
        g[a].push_back({b,i});
        g[b].push_back({a,i});
    }
    vector<vector<int>>build(){
        vector<int>vis(n),pos(n),low(n);
        for(int i=0;i<n;i++){
            dfs(i,-1,vis,pos,low);
        }
        vector<int>vis2(n);
        vector<vector<int>>res;
        for(int i=0;i<n;i++){
            if(!vis2[i]){
                vector<int>ns;
                dfs2(i,ns,vis2);
                res.push_back(ns);
            }
        }
        return res;
    }
    void dfs(int u,int pr,vector<int>&vis,
             vector<int>&pos,vector<int>&low){
        if(vis[u])return;
        vis[u]=1;
        pos[u]=low[u]=ind++;
        bool dup=false;
        for(auto el:g[u]){
            int v=el.first;
            if(v==pr&&!dup){
                dup=true;
                continue;
            }
            if(vis[v]){
                low[u]=min(low[u],pos[v]);
            }
            else{
                dfs(v,u,vis,pos,low);
                low[u]=min(low[u],low[v]);
                if(low[v]>pos[u]){
                    bridges.insert(el.second);
```

```
            }
        }
    }
  }
  void dfs2(int u,vector<int>&ns,vector<int>&vis){
    //add all nodes reachable from this node to ns
    if(vis[u])return;
    ns.push_back(u);
    vis[u]=1;
    for(auto el:g[u]){
        if(bridges.count(el.second))continue;
        dfs2(el.first,ns,vis);
    }
  }
};
```

Offline dynamic MST

```
//source: https://codeforces.com/blog/entry/105192
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
struct DSU{
  vector<int>sz,pr;
  int cmps;
  void reset(int n){
    fill(sz.begin(),sz.begin()+n,1);
    iota(pr.begin(),pr.begin()+n,0);
    cmps=n;
  }
  DSU(int n):sz(n),pr(n){reset(n);}
  bool unite(int u,int v){
    u=F(u),v=F(v);
    if(u==v)return false;
    cmps--;
    if(sz[u]<sz[v])swap(u,v);
    sz[u]+=sz[v];
    pr[v]=u;
    return true;
  }
  int F(int u){
    return u==pr[u]?u:pr[u]=F(pr[u]);
  }
};
template<typename EW>//Edge weight type
struct OfflineDynamicMST{
  struct Edge{
    int l,r;//Times active,[l,r)
    int u,v;//Edge points, undirected
    EW w;//Edge weight
    bool operator<(const Edge& e){return w<e.w;}
  };
  vector<Edge>changes;
  vector<tuple<int,int,EW>>activeEdges;
  vector<int>lastChanged,id;
  vector<EW> ans;
  int totalN,queryNum;
  DSU dsu,dsu2;
  OfflineDynamicMST(vector<tuple<int,int,EW>>
    startEdges,int n):activeEdges(startEdges),
    lastChanged(startEdges.size()),totalN(n),
          dsu(n),dsu2(n),id(n),queryNum(0){}
  void update(int i,EW x){
    queryNum++;
    auto& [u,v,w]=activeEdges[i];
    changes.push_back({lastChanged[i],queryNum,
                       u,v,w});
    lastChanged[i]=queryNum;
    w=x;
  }
  void solve(int l,int r,vector<Edge>es,int n,
             EW cost=0){
    es.erase(stable_partition(begin(es),end(es),
[&](const Edge& e) {return !(e.r<=l or r<=e.l);})
             ,es.end());
    dsu.reset(n);dsu2.reset(n);
    for(auto& e:es){//partially overlapping
      if(l<e.l or e.r<r)dsu.unite(e.u,e.v);
    }
    for(auto& e:es){
      if(e.l<=l and r<=e.r){//fully overlapping
        if(dsu.unite(e.u,e.v)){
          cost+=e.w;
          dsu2.unite(e.u,e.v);
        }
      }
    }
    if(l+1==r){
      ans[l]=cost;
      return;
    }
    int cnt=0;
    for(int i=0;i<n;i++){
      if(dsu2.F(i)==i)id[i]=cnt++;
    }
    dsu.reset(cnt);
    for(auto& e:es){
      e.u=id[dsu2.F(e.u)],e.v=id[dsu2.F(e.v)];
      if(e.l<=l and r<=e.r){
        if(!dsu.unite(e.u,e.v))e.l=INT_MAX,
                    e.r=INT_MIN;//useless edge
      }
    }
    int m=(l+r)/2;
    solve(l,m,es,cnt,cost);
    solve(m,r,es,cnt,cost);
  }
  vector<EW> run(){
    int m=activeEdges.size();
    queryNum++;
    for(int i=0;i<m;i++){
      auto& [u,v,w]=activeEdges[i];
      changes.push_back({lastChanged[i],
                         queryNum,u,v,w});
    }
    sort(begin(changes),end(changes));
    ans.resize(queryNum);
    solve(0,queryNum,changes,totalN);
    return ans;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,m,q;
  cin>>n>>m>>q;
  vector<tuple<int,int,ll>>es(m);
  for(auto& [u,v,w]:es){
    cin>>u>>v>>w;u--;v--;
  }
  OfflineDynamicMST<ll>mst(es,n);
  for(int i=0;i<q;i++){
    int k;
    ll d;
    cin>>k>>d;k--;
    mst.update(k,d);
  }
  auto res=mst.run();
  for(int i=1;i<=q;i++)cout<<res[i]<<'\n';
  return 0;
}
```

Maximum independent set

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
const int N=45;
ll G[N];
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
    int N,M;
    cin>>N>>M;
    for(int i=0;i<M;i++){
        ll u,v;cin>>u>>v;
        G[u]|=(1LL)<<v;
        G[v]|=(1LL)<<u;
    }
    int half=N/2;
    int smalf=(1<<half);
    vector<int>dp(smalf,0);
    vector<int>prev(smalf,0);
    for(int b=0;b<smalf;b++){
        int in=0;
        bool can=true;
        for(int i=0;i<half;i++){
            if(b&(1<<i)){
                if(G[i]&in)can=false;
                in|=1<<i;
            }
        }
        if(can){
```

```
            dp[b]=__builtin_popcount(in);
            prev[b]=b;
        }
    }
    for(int i=0;i<smalf;i++){
        for(int j=0;j<half;j++){
            if(dp[i]==-1)continue;
            if(i&(1<<j))continue;
            int u=i|(1<<j);
            if(dp[i]>dp[u]){
                dp[u]=dp[i];
                prev[u]=i;
            }
        }
    }
    int oth=N-half;
    int bigf=1<<oth;
    int resCnt=0;
    vector<int>vals;
    for(int b=0;b<bigf;b++){
        ll in=0;
        bool can=true;
        //cout<<b<<endl;
        for(int i=0;i<oth;i++){
            if(b&(1<<i)){
  //cout<<i<<' '<<G[i]<<' '<<in<<' '<<(G[i]&in)<<endl;
                if(G[i+half]&in)can=false;
                in|=(1LL)<<(i+half);
            }
        }
        if(!can)continue;
        ll o=smalf-1;
        for(int i=0;i<oth;i++){
            if(b&(1<<i)){
                o&=~G[i+half];
            }
        }
        int cnt=__builtin_popcountll(in);
        cnt+=dp[o];
        //cout<<b<<' '<<in<<endl;
        if(cnt>resCnt){
             resCnt=cnt;
             while(o!=prev[o])o=prev[o];
             vals.clear();
              for(int i=0;i<oth;i++){
                if(b&(1<<i))vals.push_back(i+half);
              }
              for(int i=0;i<half;i++){
                if(o&(1<<i))vals.push_back(i);
              }
        }
    }
    cout<<resCnt<<endl;
    for(auto el:vals)cout<<el<<' ';
    return 0;
}
```

Persistent union find

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T,int LG>//Blocksize
struct PersistentArray{
  static constexpr int BTMSK=(1<<LG)-1;
  struct Tipp{
    T vaartus;
    Tipp* jargmised[1<<LG]={};
    Tipp(){}
    Tipp(T &vaartus):vaartus(vaartus){}
  };
  Tipp* juur;
  PersistentArray():juur(nullptr){}
  T saa(Tipp *t,int koht){
    if(koht==0)return t->vaartus;
    return saa(t->jargmised[koht&BTMSK],koht>>LG);
  }
  T saa(int k){
    return saa(juur,k);
  }
  pair<Tipp*,T*> muteeritav_saa(Tipp *t,int koht){
    t=t?new Tipp(*t):new Tipp();
    if(koht==0)return {t,&t->vaartus};
    auto alumine=muteeritav_saa(
                t->jargmised[koht&BTMSK],koht>>LG);
    t->jargmised[koht&BTMSK]=alumine.first;
    return {t,alumine.second};
  }
  T* muteeritav_saa(int k){
    auto uus=muteeritav_saa(juur,k);
    juur=uus.first;
    return uus.second;
  }
  Tipp* ehita(Tipp* t,T &vaartus,int koht){
    if(!t)t=new Tipp();
    if(koht==0){
      t->vaartus=vaartus;
      return t;
    }
    auto alumine=ehita(t->jargmised[koht&BTMSK],
                       vaartus,koht>>LG);
    t->jargmised[koht&BTMSK]=alumine;
    return t;
  }
  void ehita(vector<T> vec){
    juur=nullptr;
    for(int i=0;i<vec.size();i++)
      juur=ehita(juur,vec[i],i);
  }
};
struct PersistentUnionFind{
  PersistentArray<int,3>massiiv;
  PersistentUnionFind(){}
  PersistentUnionFind(int suurus){//komponentide suurus 1
    massiiv.ehita(vector<int>(suurus,-1));
  }
  int leia(int k){
    int p=massiiv.saa(k);
    if(p>=0)return leia(p);
    return k;
  }
  int suurus(int k){
    return -massiiv.saa(leia(k));
  }
  bool uhenda(int u,int v){
    u=leia(u);
    v=leia(v);
    if(u==v)return false;
    auto i=massiiv.saa(u);
    auto j=massiiv.saa(v);
    if(i>=j){
      swap(u,v);
      swap(i,j);
    }
    auto a=massiiv.muteeritav_saa(u);
    *a+=j;
    auto b=massiiv.muteeritav_saa(v);
    *b=u;
    return true;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,q;
  cin>>n>>q;
  vector<PersistentUnionFind>hulgad(q+1);
  hulgad[0]=PersistentUnionFind(n);
  for(int i=1;i<=q;i++){
    int t,k,u,v;
    cin>>t>>k>>u>>v;
    k++;
    if(t==0){
      hulgad[i]=hulgad[k];
      hulgad[i].uhenda(u,v);
    }else{
      cout<<(hulgad[k].leia(u)==hulgad[k].leia(v))<<'\n';
    }
  }
  return 0;
}
```

Suffix array

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
struct SA{
  vector<int>p,c,l;
  SA(string s){
    int n=s.size();
    p.resize(n);
```

```
    c.resize(n);
    {
      vector<pair<char,int>>a(n);
      for(int i=0;i<n;i++)a[i]={s[i],i};
      sort(a.begin(),a.end());
      for(int i=0;i<n;i++)p[i]=a[i].second;
      c[p[0]]=0;
      for(int i=1;i<n;i++){
        c[p[i]]=c[p[i-1]];
        if(a[i].first!=a[i-1].first)c[p[i]]++;
      }
    }
    vector<pair<pair<int,int>,int>>a(n);
    int k=1;
    while(k<n){
      for(int i=0;i<n;i++)a[i]={{c[i],c[(i+k)%n]},i};
      SORT(a);
      for(int i=0;i<n;i++)p[i]=a[i].second;
      c[p[0]]=0;
      for(int i=1;i<n;i++){
        c[p[i]]=c[p[i-1]];
        if(a[i].first!=a[i-1].first)c[p[i]]++;
      }
      k*=2;
    }
    l.resize(n);
    k=0;
    for(int i=0;i<n-1;i++){
      int pi=c[i];
      int j=p[pi-1];
      while(s[i+k]==s[j+k])k++;
      l[pi]=k;
      k=max(k-1,0);
    }
  }
  void SORT(auto &ar){
    int n=ar.size();
    {
      vector<int>p(n),c(n);
      auto br=ar;
      for(auto el:ar)c[el.first.second]++;
      for(int i=1;i<n;i++)p[i]=p[i-1]+c[i-1];
      for(auto el:ar)br[p[el.first.second]++]=el;
      ar=br;
    }
    {
      vector<int>p(n),c(n);
      auto br=ar;
      for(auto el:ar)c[el.first.first]++;
      for(int i=1;i<n;i++)p[i]=p[i-1]+c[i-1];
      for(auto el:ar)br[p[el.first.first]++]=el;
      ar=br;
    }
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  string s;
  cin>>s;
  s+='$';
  SA t(s);
  for(int i=1;i<s.size();i++)cout<<t.p[i]<<' ';
  return 0;
}
```

Duval

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
vector<int>duval(string &s){
    int n=s.size();
    int i=0;
    vector<int>res;
    while(i<n){
        int j=i+1,k=i;
        while(j<n&&s[k]<=s[j]){
            if(s[k]<s[j])k=i;
            else k++;
            j++;
        }
        while(i<=k){
            res.push_back(i);
            i+=j-k;
        }
    }
    res.push_back(n);
    return res;
}
```

Z-algorithm

```
vector<int>z(string &s){
    int n=s.size();
    vector<int>z(n);
    z[0]=n;
    int l=0,r=0;
    for(int i=1;i<n;i++){
        if(r>=i)z[i]=min(z[i-l],r+1-i);
        while(i+z[i]<n && s[i+z[i]]==s[z[i]])z[i]++;
        if(i+z[i]-1>r){
            r=i+z[i]-1;
            l=i;
        }
    }
    return z;
}
```

Aho-Corasick

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
class AC{
public:
  static const int K = 26;
  struct Vertex {
    int next[K];
    vector<int>vls;
    //bool output = false;
    int p = -1;
    char pch;
    int link = -1;
    int go[K];
    Vertex(int p=-1, char ch='$') : p(p), pch(ch) {
      fill(begin(next), end(next), -1);
      fill(begin(go), end(go), -1);
    }
  };
  vector<Vertex>t;
  AC(){
    t.resize(1);
  }
  void add_string(string const& s,int vl) {
    int v = 0;
    for (char ch : s) {
      int c = ch - 'a';
      if (t[v].next[c] == -1) {
        t[v].next[c] = t.size();
        t.emplace_back(v, ch);
      }
      v = t[v].next[c];
    }
    t[v].vls.push_back(vl);
  }
  int go(int v, char ch);
  int get_link(int v) {
    if (t[v].link == -1) {
      if (v == 0 || t[v].p == 0)
        t[v].link = 0;
      else
        t[v].link = go(get_link(t[v].p), t[v].pch);
    }
    return t[v].link;
  }
};
int AC::go(int v, char ch) {
  int c = ch - 'a';
  if (t[v].go[c] == -1) {
    if (t[v].next[c] != -1)
      t[v].go[c] = t[v].next[c];
    else
      t[v].go[c] = v == 0 ? 0 : go(get_link(v), ch);
  }
  return t[v].go[c];
}
```

Suffix automaton

```
#include <bits/stdc++.h>
typedef long long ll;
```

```
using namespace std;
struct SAM{
  struct Olek{
    int pikkus, link;
    int sonesid;//Mitu sone loppeb selles kohas
    //pole vaja kui ei taha esinemiste arvu
    map<char,int>jargmine;
    Olek(int pikkus=0,int link=-1):pikkus(pikkus),
                            link(link),sonesid(0){}
  };
  vector<Olek>olekud;
  int esimeneVaba,eelmine;
  SAM(){
    esimeneVaba=eelmine=0;
    olekud.push_back(Olek());
    esimeneVaba++;
  }
  SAM(string &s):SAM(){
    for(auto taht:s)lisa_taht(taht);
  }
  void arvutaEsinemisteArv(){
    vector<pair<int,int>>pikkuseJargi;//Kahanevalt
    for(int i=1;i<olekud.size();i++)pikkuseJargi
           .push_back(make_pair(olekud[i].pikkus,i));
    sort(pikkuseJargi.begin(),pikkuseJargi.end());
    reverse(pikkuseJargi.begin(),pikkuseJargi.end());
    for(auto& paar:pikkuseJargi){
      int i=paar.second;
      olekud[olekud[i].link].sonesid+=olekud[i].sonesid;
    }
  }
  int mituKordaEsineb(string &s){
    int koht=0;
    for(auto &taht:s){
      if(!olekud[koht].jargmine.count(taht))return 0;
      koht=olekud[koht].jargmine[taht];
    }
    return olekud[koht].sonesid;
  }
  void lisa_taht(char c){
    int koht=esimeneVaba++;
    olekud.push_back(Olek(olekud[eelmine].pikkus+1));
    olekud[koht].sonesid=1;//1 sõne lõppeb
    int p=eelmine;
    while(p!=-1 && !olekud[p].jargmine.count(c)){
      olekud[p].jargmine[c]=koht;
      p=olekud[p].link;
    }
    if(p==-1)olekud[koht].link=0;
    else{
      int q=olekud[p].jargmine[c];
      if(olekud[p].pikkus+1==olekud[q].pikkus){
        olekud[koht].link=q;
      }else{
        int kloon=esimeneVaba++;
        olekud.push_back(Olek(olekud[p].pikkus+1,
                              olekud[q].link));
        olekud[kloon].jargmine=olekud[q].jargmine;
        while(p!=-1 && olekud[p].jargmine[c]==q){
          olekud[p].jargmine[c]=kloon;
          p=olekud[p].link;
        }
        olekud[q].link=olekud[koht].link=kloon;
      }
    }
    eelmine=koht;
  }
};
int main()
{ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
  int n,q;
  cin>>n>>q;
  string s;
  cin>>s;
  SAM T(s);
  T.arvutaEsinemisteArv();
  while(q--){
    string t;
    cin>>t;
    cout<<T.mituKordaEsineb(t)<<'\n';
  }
  return 0;
}
```

Mo's algorithm with updates

```
//source: https://kodu.ut.ee/~bleurgh/mo_eio_2019/mo.pdf
//MO algoritm uuendustega, keerukus O(QN**2/3+N**5/3)
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
template<typename T>
class Mo{//indeksid alates nullist
  struct Query{int l,r,t;};
  int n;//massiivi pikkus
  vector<T>ar,dp,cur,nx;//massiiv, massiivi koopia,
  //algne element, uus element
  vector<Query>qs;//Küsimuse päringud
  vector<int>ind;//uuenduse indeks
public:
  Mo(vector<T>&ar):ar(ar),dp(ar),n(ar.size()){}
  void add_upt(int i,const T& newVal){
    //Lisab uuenduse kohal i, uus väärtus newVal
    ind.push_back(i);
    cur.push_back(dp[i]);
    nx.push_back(newVal);
    dp[i]=newVal;
  }
  void add_get(int l,int r){//lisab küsimuse päringu
    qs.push_back({l,r,ind.size()});
  }
  template<typename A,typename E,typename O>
  void build(const A &a,const E &e,const O &out){
    int q=qs.size();
    int bs=n/min<int>(n,pow(n,0.333333));
    vector<int>num(q);
    iota(begin(num),end(num),0);
    sort(begin(num),end(num),[&](int a,int b){
      int ab=qs[a].l/bs,bb=qs[b].l/bs;
      if(ab!=bb)return ab<bb;
      int ac=qs[a].r/bs,bc=qs[b].r/bs;
      if(ac!=bc)return ac<bc;
      return qs[a].t<qs[b].t;
    });
    int l=0,r=-1,t=0;
    auto lisaUuendus=[&](int aeg){
      if(l<=ind[aeg]&&ind[aeg]<=r){
        e(cur[aeg]);
        a(nx[aeg]);
      }
      ar[ind[aeg]]=nx[aeg];
    };
    auto eemaldaUuendus=[&](int aeg){
      if(l<=ind[aeg]&&ind[aeg]<=r){
        e(nx[aeg]);
        a(cur[aeg]);
      }
      ar[ind[aeg]]=cur[aeg];
    };
    for(auto ind:num){
      while(r<qs[ind].r)a(ar[++r]);
      while(r>qs[ind].r)e(ar[r--]);
      while(l<qs[ind].l)e(ar[l++]);
      while(l>qs[ind].l)a(ar[--l]);
      while(t<qs[ind].t)lisaUuendus(t++);
      while(t>qs[ind].t)eemaldaUuendus(--t);
      out(ind);
    }
  }
};
```