Standard Code Library

Floze3

FZOI

May 17, 2025

Contents

一切的开始

宏定义

- 需要 C++20

```cpp
#include <bits/stdc++.h>
#define mp(x, y) std::make_pair(x, y)
#define mt std::make_tuple
#define eb emplace_back
#define fi first
#define se second
#define all(s) s.begin(), s.end()
#define rall(s) s.rbegin(), s.rend()
#define file(name)                 \
  freopen(name ".in", "r", stdin); \
  freopen(name ".out", "w", stdout);
#define fs(x) std::fixed << std::setprecision(x)
#define il inline
#define multitask    \
  int _; rd >> _; \
  while (_--)
using std::cerr;

using i32 = int32_t;
using i64 = long long;
using u64 = unsigned long long;
using u32 = uint32_t;
using i128 = __int128_t;
using u128 = __uint128_t;
using pii = std::pair<i32, i32>;
using pi64 = std::pair<i64, i64>;
using vi = std::vector<i32>;
using vu = std::vector<u32>;
using vi64 = std::vector<i64>;
using vu64 = std::vector<u64>;
using vpii = std::vector<pii>;
using vpi64 = std::vector<pi64>;
using str = std::string;
using vstr = std::vector<str>;
using f64 = long double;
template <typename T> using vc = std::vector<T>;
template <typename FI, typename SE> using pr = std::pair<FI, SE>;

namespace basic_algorithm {
template <typename T> il T abs(T a) { return a >= 0 ? a : -a; }
template <typename T> il void chmin(T &a, T b) { if (a > b) a = b; }
template <typename T> il void chmax(T &a, T b) { if (a < b) a = b; }
template <typename T> il T lowbit(T x) { return (x & (-x)); }
il i32 pct(i32 x) { return __builtin_popcount(x); }
il i32 pct(u32 x) { return __builtin_popcount(x); }
il i32 pct(i64 x) { return __builtin_popcountll(x); }
il i32 pct(u64 x) { return __builtin_popcountll(x); }
}  // namespace basic_algorithm

using namespace basic_algorithm;

std::istream &rd = std::cin;
std::ostream &wt = std::cout;

constexpr i32 N = 1e5 + 5;
constexpr i32 mod = 1e9 + 7;
constexpr i32 inf = 0x3f3f3f3f;
constexpr i64 inf64 = 0x3f3f3f3f3f3f3f3fll;
// constexpr f64 pi = std::numbers::pi_v<f64>;

// std::mt19937 rng(RANDOM_SEED);

signed main() {
  rd.tie(nullptr) -> sync_with_stdio(false);
  return 0;
}
// ----------------------------------------------------------------------------
```

数据结构

ST 表

```cpp
void build_st() {
  for (int i = 1; i <= n; ++i) st[i][0] = a[i];
  for (int j = 1; j <= lg[n]; ++j) {
    for (int i = 1; i + (1 << j) - 1 <= n; ++i) {
      st[i][j] = std::max(st[i][j - 1], st[i + (1 << j - 1)][j - 1]);
    }
  }
  return ;
}

il int query(int l, int r) {
  int k = lg[r - l + 1];
  return std::max(st[l][k], st[r - (1 << k) + 1][k]);
}
```

并查集

```cpp
struct DSU {
  vi fa, siz;

  DSU() {}
  DSU(int n) { init(n); }

  void init(int n) {
    fa.resize(n + 1), siz.resize(n + 1);
    for (int i = 1; i <= n; ++i) fa[i] = i, siz[i] = 1;
    return ;
  }

  int find(int x) { return x == fa[x] ? x : fa[x] = find(fa[x]); }

  bool same(int x, int y) { return find(x) == find(y); }

  bool merge(int x, int y) {
    x = find(x), y = find(y);
    if (x == y) return false;
    if (siz[x] < siz[y]) std::swap(x, y);
    fa[y] = x, siz[x] += siz[y];
    return true;
  }

  il int size(int x) { return siz[find(x)]; }
};
```

树状数组

```cpp
template <typename T>
struct FenwickTree {
  int n;
  std::vector<T> bit;

  FenwickTree(int n): n(n), bit(n + 1) {}

  void add(int p, T val) {
    for (; p <= n; p += p & -p) bit[p] += val;
    return ;
  }

  T query(int p) {
    T res = 0;
    for (; p; p -= p & -p) res += bit[p];
    return res;
  }

  T rquery(int l, int r) {
    return query(r) - query(l - 1);
  }
}; // FenwickTree<int> t(n);
```

二维线段树

```cpp
// P3437
struct ST {
  int cnt;
  struct node {
    int ls, rs, val, tag;

    #define ls(x) t[x].ls
    #define rs(x) t[x].rs
    #define val(x) t[x].val
    #define tag(x) t[x].tag
    #define mid (l + r >> 1)
  } t[N << 11];

  il int update(int p, int l, int r, int ql, int qr, int v) {
    if (!p) p = ++cnt;
    val(p) = max(val(p), v);
    if (l >= ql && r <= qr) {
      tag(p) = max(tag(p), v);
      return p;
    }
    if (ql <= mid) ls(p) = update(ls(p), l, mid, ql, qr, v);
    if (qr > mid) rs(p) = update(rs(p), mid + 1, r, ql, qr, v);
    return p;
  }

  il int query(int p, int l, int r, int ql, int qr) {
    if (!p) return p;
    if (l >= ql && r <= qr) return max(tag(p), val(p));
    int ans = tag(p);
    if (ql <= mid) ans = max(ans, query(ls(p), l, mid, ql, qr));
    if (qr > mid) ans = max(ans, query(rs(p), mid + 1, r, ql, qr));
    return ans;
  }
} st;

int tag[N << 2], val[N << 2], n, D, S;

il void update(int l, int r, int ql, int qr, int u, int d, int x, int v) {
  val[x] = st.update(val[x], 1, S, u, d, v);
  if (l >= ql && r <= qr) {
    tag[x] = st.update(tag[x], 1, S, u, d, v);
    return ;
  }
  if (ql <= mid) update(l, mid, ql, qr, u, d, x << 1, v);
  if (qr > mid) update(mid + 1, r, ql, qr, u, d, x << 1 | 1, v);
  return ;
}

il int query(int p, int l, int r, int ql, int qr, int u, int d) {
  int ans = st.query(tag[p], 1, S, u, d);
  if (l >= ql && r <= qr) return max(ans, st.query(val[p], 1, S, u, d));
  if (ql <= mid) ans = max(ans, query(p << 1, l, mid, ql, qr, u, d));
  if (qr > mid) ans = max(ans, query(p << 1 | 1, mid + 1, r, ql, qr, u, d));
  return ans;
}
```

左偏树

```cpp
template <typename T>
struct LeftTree {
  int n;
  vi lc, rc, dis, rt;
  vc<bool> del;
  vc<T> ltt;

  LeftTree(int n): n(n), lc(n + 1), rc(n + 1), dis(n + 1), rt(n + 1), del(n + 1), ltt(n + 1) {}

  int find(int x) { return x == rt[x] ? x : rt[x] = find(rt[x]); }

  int merge(int x, int y) {
    if (!x || !y) return x | y;
    if (ltt[y] < ltt[x]) std::swap(x, y);
    rc[x] = merge(rc[x], y);
    if (dis[lc[x]] < dis[rc[x]]) std::swap(lc[x], rc[x]);
    dis[x] = dis[rc[x]] + 1;
    return x;
  }

  void Merge(int x, int y) {
    if (del[x] || del[y]) return ;
    x = find(x), y = find(y);
    if (x != y) rt[x] = rt[y] = merge(x, y);
    return ;
  }

  int Del(int x) {
    if (del[x]) return -1;
    x = find(x);
    int res = ltt[x].v;
    del[x] = true;
    rt[lc[x]] = rt[rc[x]] = rt[x] = merge(lc[x], rc[x]);
    lc[x] = rc[x] = dis[x] = 0;
    return res;
  }
};
```

平衡树

- 普通平衡树

```
i32 n, m, a[N], idx, rt, lst, ans;

struct node {
  i32 ls, rs, val, pri, siz;

  #define ls(p) t[p].ls
  #define rs(p) t[p].rs
  #define val(p) t[p].val
  #define pri(p) t[p].pri
  #define siz(p) t[p].siz
} t[N + M];

il i32 create(i32 x) {
  val(++idx) = x, siz(idx) = 1;
  ls(idx) = rs(idx) = 0, pri(idx) = (i32)rng();
  return idx;
}

il void pushup(int p) { return void(siz(p) = siz(ls(p)) + siz(rs(p)) + 1); }

il void split(i32 u, i32 x, i32 &L, i32 &R) {
  if (!u) return void(L = R = 0);
  if (val(u) <= x) L = u, split(rs(u), x, rs(u), R);
  else R = u, split(ls(u), x, L, ls(u));
  return pushup(u);
}

il i32 merge(i32 L, i32 R) {
  if (!L || !R) return L | R;
  if (pri(L) < pri(R)) {
    rs(L) = merge(rs(L), R);
```

```cpp
    return pushup(L), L;
  }
  else {
    ls(R) = merge(L, ls(R));
    return pushup(R), R;
  }
}

il void insert(i32 x) {
  i32 L, R; split(rt, x, L, R);
  rt = merge(merge(L, create(x)), R);
  return ;
}

il void del(i32 x) {
  i32 L, R, A;
  split(rt, x, L, R);
  split(L, x - 1, L, A);
  rt = merge(merge(L, merge(ls(A), rs(A))), R);
  return ;
}

il i32 getrnk(i32 x) {
  i32 L, R; split(rt, x - 1, L, R);
  i32 res = siz(L) + 1;
  rt = merge(L, R);
  return res;
}

il i32 getkth(i32 u, i32 k) {
  if (siz(ls(u)) + 1 == k) return val(u);
  if (siz(ls(u)) >= k) return getkth(ls(u), k);
  return getkth(rs(u), k - siz(ls(u)) - 1);
}

il i32 getpre(i32 x) {
  i32 L, R; split(rt, x - 1, L, R);
  i32 res = getkth(L, siz(L));
  rt = merge(L, R);
  return res;
}

il i32 getnxt(i32 x) {
  i32 L, R; split(rt, x, L, R);
  i32 res = getkth(R, 1);
  rt = merge(L, R);
  return res;
}
```

- 文艺平衡树

```cpp
i32 n, m, rt, idx;

struct node {
  i32 ls, rs, siz, pri, tag, val;

  #define ls(p) t[p].ls
  #define rs(p) t[p].rs
  #define siz(p) t[p].siz
  #define pri(p) t[p].pri
  #define tag(p) t[p].tag
  #define val(p) t[p].val
} t[N];

il int create(i32 x) {
  val(++idx) = x, pri(idx) = (i32)rng(), siz(idx) = 1;
  return idx;
}

il void pushdown(i32 p) {
  if (tag(p)) {
    tag(ls(p)) ^= 1, tag(rs(p)) ^= 1;
    std::swap(ls(p), rs(p));
    tag(p) = 0;
  }
  return ;
}

il void pushup(i32 p) { return void(siz(p) = siz(ls(p)) + siz(rs(p)) + 1); }

il void split(i32 u, i32 x, i32 &L, i32 &R) {
  if (!u) return void(L = R = 0);
  pushdown(u);
  if (siz(ls(u)) < x) L = u, split(rs(u), x - siz(ls(u)) - 1, rs(u), R);
  else R = u, split(ls(u), x, L, ls(u));
  return pushup(u);
}

il int merge(i32 x, i32 y) {
  if (!x || !y) return x | y;
  if (pri(x) < pri(y)) {
    pushdown(x);
    rs(x) = merge(rs(x), y);
    pushup(x);
    return x;
  }
  else {
    pushdown(y);
    ls(y) = merge(x, ls(y));
    pushup(y);
    return y;
  }
}

il void print(i32 u) {
  if (!u) return ;
  pushdown(u);
  print(ls(u));
  wt << val(u) << ' ';
  print(rs(u));
  return ;
}
```

笛卡尔树

```
for (i32 i = 1; i <= n; st[++top] = i++) {
  while (top && p[st[top]] > p[i]) ls[i] = st[top--];
  if (top) rs[st[top]] = i;
}
```

数学

类欧几里得

- $m = \lfloor \frac{an+b}{c} \rfloor$.
- $f(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$f(a,b,c,n) = (\frac{a}{c})n(n+1)/2 + (\frac{b}{c})(n+1) + f(a \bmod c, b \bmod c, c, n)$；否则 $f(a,b,c,n) = nm - f(c, c-b-1, a, m-1)$。
- $g(a,b,c,n) = \sum_{i=0}^{n} i \lfloor \frac{ai+b}{c} \rfloor$: 当 $a \ge c$ or $b \ge c$ 时，$g(a,b,c,n) = (\frac{a}{c})n(n+1)(2n+1)/6 + (\frac{b}{c})n(n+1)/2 + g(a \bmod c, b \bmod c, c, n)$；否则 $g(a,b,c,n) = \frac{1}{2}(n(n+1)m - f(c, c-b-1, a, m-1) - h(c, c-b-1, a, m-1))$。
- $h(a,b,c,n) = \sum_{i=0}^{n} \lfloor \frac{ai+b}{c} \rfloor^2$: 当 $a \ge c$ or $b \ge c$ 时，$h(a,b,c,n) = (\frac{a}{c})^2 n(n+1)(2n+1)/6 + (\frac{b}{c})^2(n+1) + (\frac{a}{c})(\frac{b}{c})n(n+1) + h(a \bmod c, b \bmod c, c, n) + 2(\frac{a}{c})g(a \bmod c, b \bmod c, c, n) + 2(\frac{b}{c})f(a \bmod c, b \bmod c, c, n)$；否则 $h(a,b,c,n) = nm(m+1) - 2g(c, c-b-1, a, m-1) - 2f(c, c-b-1, a, m-1) - f(a,b,c,n)$。

埃筛

```
std::bitset<N> ispri;
vi pri;

```

```
il void getpri(int n) {
  ispri.set(), ispri[1] = false;
  for (int i = 2; i <= n; ++i) {
    if (!ispri[i]) continue;
    pri.eb(i);
    for (i64 j = 1ll * i * i; j <= n; j += i) ispri[j] = false;
  }
  return ;
}
```

欧拉筛

```
bool isprime[N];

il void GetPrime() {
  memset(isprime, 1, sizeof(isprime));
  isprime[1] = 0;
  for (i32 i = 2; i <= n; ++i) {
    if (isprime[i]) prime[++tot] = i;
    for (i32 j = 1; j <= tot && i * prime[j] <= n; ++j) {
      isprime[i * prime[j]] = 0;
      if (i % prime[j] == 0) break;
    }
  }
  return ;
}
```

高斯消元

```
std::cin >> n;
for (int i = 1; i <= n; ++i)
  for (int j = 1; j <= n + 1; ++j)
    std::cin >> a[i][j];
for (int i = 1; i <= n; ++i) { // 考虑第 i 个未知数
  int id = i; double mx = a[i][i];
  for (int j = i + 1; j <= n; ++j) {
    if (fabs(a[j][i]) > fabs(mx)) mx = a[j][i], id = j;
  }
  if (fabs(mx) < eps) return io.write("No Solution"), 0;
  for (int j = 1; j <= n + 1; ++j) std::swap(a[i][j], a[id][j]);
  for (int j = 1; j <= n; ++j) {
    if (j == i) continue;
    double xs = a[j][i] / a[i][i];
    for (int k = i + 1; k <= n + 1; ++k) a[j][k] -= xs * a[i][k];
  }
}
for (int i = 1; i <= n; ++i) std::cout << fs(2) << a[i][n + 1] / a[i][i] << '\n';
```

modint

```
template<int MOD> class ModInt {
  int value;

  static constexpr int norm(int x) { return x >= MOD ? x - MOD : x; }

public:
  constexpr ModInt(i64 v = 0): value(static_cast<int>(v % MOD)) {
    if (value < 0) value += MOD;
  }

  constexpr int val() const { return value; }

  friend constexpr ModInt operator+(const ModInt &a, const ModInt &b) {
    return ModInt(norm(a.value + b.value));
  }

  constexpr ModInt &operator+=(const ModInt &other) {
    value = norm(value + other.value);
    return *this;
```

```cpp
  }

  friend constexpr ModInt operator-(const ModInt &a, const ModInt &b) {
    return ModInt(norm(a.value - b.value + MOD));
  }

  constexpr ModInt &operator-=(const ModInt &other) {
    value = norm(value - other.value + MOD);
    return *this;
  }

  friend constexpr ModInt operator*(const ModInt &a, const ModInt &b) {
    return ModInt(1LL * a.value * b.value % MOD);
  }

  constexpr ModInt &operator*=(const ModInt &other) {
    value = static_cast<int>(1LL * value * other.value % MOD);
    return *this;
  }

  constexpr ModInt pow(i64 exp) const {
    ModInt result(1), base(value);
    for (; exp; exp >>= 1, base *= base)
      if (exp & 1) result *= base;
    return result;
  }

  friend constexpr ModInt operator/(const ModInt &a, const ModInt &b) {
    return a * b.inv();
  }

  constexpr ModInt &operator/=(const ModInt& other) {
    return *this *= other.inv();
  }

  constexpr ModInt inv() const { return pow(MOD - 2); }

  constexpr ModInt &operator++() {
    value = norm(value + 1);
    return *this;
  }

  constexpr ModInt operator++(int) {
    ModInt temp = *this;
    ++(*this);
    return temp;
  }

  constexpr ModInt &operator--() {
    value = norm(value - 1 + MOD);
    return *this;
  }

  constexpr ModInt operator--(int) {
    ModInt temp = *this;
    --(*this);
    return temp;
  }

  constexpr ModInt operator-() const {
    return ModInt(-value);
  }

  friend constexpr bool operator==(const ModInt &a, const ModInt &b) {
    return a.value == b.value;
  }

  friend constexpr bool operator!=(const ModInt &a, const ModInt &b) {
    return a.value != b.value;
  }

  template<int M> friend Scanner &operator>>(Scanner &in, ModInt<M> &x) {
    i64 v;
    in >> v;
    x = ModInt<M>(v);
    return in;
  }

  template<int M> friend Printer &operator<<(Printer &out, const ModInt<M> &x) {
    return out << x.val();
  }

  friend std::ostream &operator<<(std::ostream &os, const ModInt &m) {
    return os << m.value;
  }
};

using mint = ModInt<mod>;

il void print(mint x) { std::cerr << x; }
```

卢卡斯定理

$\binom{n}{k} = \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{k}{p} \rfloor}\binom{n \bmod p}{b \bmod p}$

exgcd

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

BSGS

```cpp
il i64 BSGS() {
  t = sqrt(p) + 1;
  // cerr << t << '\n';
  i64 tmp = b;
  for (int i = 0; i < t; ++i) {
    // cerr << tmp << '\n';
    cnt[tmp] = i;
    (tmp *= a) %= p;
  }
  a = qpow(a, t, p);
  if (!a) return b ? -1 : 1;
  tmp = 1;
  for (int i = 0; i <= t; ++i) {
    if (cnt.count(tmp)) {
      int j = cnt[tmp];
      if (i * t - j >= 0) return i * t - j;
    }
    (tmp *= a) %= p;
  }
  return -1;
}
```

图论

2-SAT

```cpp
i32 n, m, scc_id[N << 1], scc_cnt, dfn[N << 1], low[N << 1], dfc, st[N << 1], top; // 1 ~ n 真 n + 1 ~ 2 * n 假
bool in_sta[N << 1];

vi g[N << 1];

void tarjan(i32 u) {
  dfn[u] = low[u] = ++dfc, st[++top] = u, in_sta[u] = true;
  for (i32 v : g[u]) {
    if (!dfn[v]) tarjan(v), chmin(low[u], low[v]);
    else if (in_sta[v]) chmin(low[u], dfn[v]);
  }
  if (low[u] == dfn[u]) {
    ++scc_cnt;
    do {
      scc_id[st[top]] = scc_cnt;
      in_sta[st[top]] = false;
    } while (st[top--] != u);
  }
  return ;
}

bool med;

signed main() {
  rd >> n >> m;
  for (i32 i = 1, a, b, c, d; i <= m; ++i) {
    rd >> a >> b >> c >> d;
    g[a + b * n].eb(c + (d ^ 1) * n);
    g[c + d * n].eb(a + (b ^ 1) * n);
  }
  for (i32 i = 1; i <= n * 2; ++i)
    if (!dfn[i]) tarjan(i);
  for (i32 i = 1; i <= n; ++i)
    if (scc_id[i] == scc_id[i + n]) return wt << "IMPOSSIBLE", 0;
  wt << "POSSIBLE\n";
  for (i32 i = 1; i <= n; ++i)
    wt << (scc_id[i] < scc_id[i + n]) << ' ';
  Avada_Kedavra;
}
```

无向图缩点

```
i32 n, m, a[N], in[N], scc_cnt, dfn[N], low[N], dfc, st[N], top, scc_id[N], val[N], f[N];
bool in_sta[N];

vi g[N], e[N];

void tarjan(i32 u) {
  in_sta[u] = true, dfn[u] = low[u] = ++dfc, st[++top] = u;
  for (i32 v : g[u]) {
    if (!dfn[v]) tarjan(v), chmin(low[u], low[v]);
    else if (in_sta[v]) chmin(low[u], dfn[v]);
  }
  if (low[u] == dfn[u]) {
    ++scc_cnt;
    do {
      scc_id[st[top]] = scc_cnt;
      val[scc_cnt] += a[st[top]];
      in_sta[st[top]] = false;
    } while (st[top--] != u);
  }
  return ;
}

bool med;

signed main() {
  rd >> n >> m;
  for (i32 i = 1; i <= n; ++i) rd >> a[i];
  for (i32 i = 1, u, v; i <= m; ++i)
    rd >> u >> v, g[u].eb(v);
  for (i32 i = 1; i <= n; ++i)
    if (!dfn[i]) tarjan(i);
  for (i32 u = 1; u <= n; ++u) {
    for (i32 v : g[u]) {
      if (scc_id[v] != scc_id[u]) e[scc_id[u]].eb(scc_id[v]);
    }
```

```cpp
  }
  i32 ans = 0;
  for (i32 i = scc_cnt; i; --i) {
    f[i] += val[i];
    chmax(ans, f[i]);
    for (i32 it : e[i]) chmax(f[it], f[i]);
  }
  wt << ans;
  Avada_Kedavra;
}
```

边双连通分量

```cpp
i32 n, m, head[N], cnt = 1, dfn[N], low[N], dfc, top, st[N];

vc<vi> ans;

struct Edge {
  i32 v, nxt;
} e[M << 1];

il void add(i32 u, i32 v) {
  e[++cnt] = {v, head[u]}, head[u] = cnt;
  return ;
}

il vi form(i32 x) {
  vi v; i32 y;
  do v.eb(y = st[top--]); while (y != x);
  return v;
}

void tarjan(i32 u, i32 in) {
  dfn[u] = low[u] = ++dfc, st[++top] = u;
  for (i32 i = head[u]; i; i = e[i].nxt) {
    i32 v = e[i].v;
    if (!dfn[v]) {
      tarjan(v, i), chmin(low[u], low[v]);
      if (low[v] > dfn[u]) ans.eb(form(v));
    }
    else if (i != (in ^ 1)) chmin(low[u], dfn[v]);
  }
  return ;
}

for (i32 i = 1; i <= n; ++i) {
  if (!dfn[i]) tarjan(i, 0), ans.eb(form(i));
}
```

割点

```cpp
int n, m, head[N], cnt, res, rt;
int dfn[N], low[N], dfc;
bool buc[N];

struct Edge {
  int v, nxt;
} e[M << 1];

il void add(int u, int v) {
  e[++cnt] = {v, head[u]}, head[u] = cnt;
  return ;
}

void tarjan(int u) {
  dfn[u] = low[u] = ++dfc;
  int son = 0;
  for (int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v;
    if (!dfn[v]) {
      ++son;
      tarjan(v), chmin(low[u], low[v]);
      if (rt != u && low[v] >= dfn[u]) {
        res += !buc[u], buc[u] = true;
      }
    }
    else chmin(low[u], dfn[v]);
  }
  if (son > 1 && rt == u) res += !buc[u], buc[u] = true;
  return ;
}

for (int i = 1; i <= n; ++i) {
  if (!dfn[i]) rt = i, tarjan(i);
}
```

点双连通分量

```
int n, m, head[N], cnt, dfn[N], low[N], dfc, st[N], top, Rt;

std::vector<vi> ans;

struct Edge {
  int v, nxt;
} e[M << 1];

il void add(int u, int v) {
  e[++cnt] = {v, head[u]}, head[u] = cnt;
  return ;
}

il void form(int x, int f) {
  vi res; int y;
  do res.eb(y = st[top--]); while (y != x);
  res.eb(f);
  ans.eb(res);
  return ;
}

void tarjan(int u) {
  dfn[u] = low[u] = ++dfc;
  if (Rt == u && !head[u]) {
    vi v = {u};
    ans.eb(v);
    return ;
  }
  st[++top] = u;
  for (int i = head[u]; i; i = e[i].nxt) {
    int v = e[i].v;
    if (!dfn[v]) {
      tarjan(v), low[u] = std::min(low[u], low[v]);
      if (low[v] >= dfn[u]) form(v, u);
    }
    else low[u] = std::min(low[u], dfn[v]);
  }
  return ;
}

for (int i = 1; i <= n; ++i) {
  if (!dfn[i]) Rt = i, tarjan(i);
}
```

最大流

```
i32 n, m, s, t, cur[N], head[N], dep[N], cnt = 1;

struct Edge {
  i32 v, nxt;
  i64 w;
} e[M << 1];
```

```cpp

il void add(i32 u, i32 v, i64 w) {
  e[++cnt] = {v, head[u], w}, head[u] = cnt;
  return ;
}

il bool bfs() {
  memset(dep, 0, sizeof(dep)), dep[s] = 1;
  memcpy(cur, head, sizeof(head));
  qi q; q.push(s);
  while (!q.empty()) {
    i32 u = q.front(); q.pop();
    for (i32 i = head[u]; i; i = e[i].nxt) {
      i32 v = e[i].v, w = e[i].w;
      if (w && !dep[v]) {
        dep[v] = dep[u] + 1, q.push(v);
        if (v == t) return true;
      }
    }
  }
  return false;
}

i64 dfs(i32 p, i64 flow) {
  if (p == t) return flow;
  i64 tmp = flow;
  for (i32 i = cur[p]; i && tmp; i = e[i].nxt) {
    cur[p] = i;
    i32 v = e[i].v; i64 w = e[i].w;
    if (w && dep[v] == dep[p] + 1) {
      i64 c = dfs(v, std::min(tmp, w));
      tmp -= c, e[i].w -= c, e[i ^ 1].w += c;
    }
  }
  return flow - tmp;
}

il i64 dinic() {
  i64 res = 0;
  while (bfs()) res += dfs(s, inf64);
  return res;
}

for (i32 i = 1, u, v, w; i <= m; ++i) {
  rd >> u >> v >> w;
  add(u, v, w), add(v, u, 0);
}
```

最小费用最大流

```cpp
i32 n, m, s, t, head[N], dis[N], cur[N], maxflow, mincost, cnt = 1;

std::bitset<N> vis;
qi q;

struct Edge {
  i32 v, w, c, nxt;
} e[M << 1];

il void add(i32 u, i32 v, i32 w, i32 c) {
  e[++cnt] = {v, w, c, head[u]}, head[u] = cnt;
  e[++cnt] = {u, 0, -c, head[v]}, head[v] = cnt;
  return ;
}

il bool spfa() {
  memset(dis, 0x3f, sizeof(dis)), dis[s] = 0;
  memcpy(cur, head, sizeof(cur));
  // while (!q.empty()) q.pop();
  q.push(s);
  while (!q.empty()) {
    i32 u = q.front(); q.pop();
    vis[u] = false;
    for (i32 i = head[u]; i; i = e[i].nxt) {
      i32 v = e[i].v, w = e[i].w, c = e[i].c;
      if (w && dis[v] > dis[u] + c) {
        dis[v] = dis[u] + c;
        if (!vis[v]) {
          vis[v] = true;
          q.push(v);
        }
      }
    }
  }
  return dis[t] != inf;
}

i32 dfs(i32 u, i32 flow) {
  if (u == t) return flow;
  vis[u] = true;
  i32 tmp = flow;
  for (i32 i = cur[u]; i && tmp; i = e[i].nxt) {
    cur[u] = i;
    i32 v = e[i].v, w = e[i].w, c = e[i].c;
    if (!vis[v] && w && dis[v] == dis[u] + c) {
      i32 tt = dfs(v, std::min(tmp, w));
      tmp -= tt, e[i].w -= tt, e[i ^ 1].w += tt;
    }
  }
  vis[u] = false;
  return flow - tmp;
}

il void mcmf() {
  while (spfa()) {
    i32 flow = dfs(s, inf);
    maxflow += flow, mincost += flow * dis[t];
  }
  return ;
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
```

```cpp
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// --------------------------------------------
```

字符串

KMP

```cpp
for (int i = 2, j = 0; i <= m; ++i) {
  while (j && b[j + 1] != b[i]) j = nxt[j];
  if (b[j + 1] == b[i]) ++j;
  nxt[i] = j;
}
for (int i = 1, j = 0; i <= n; ++i) {
  while (j && b[j + 1] != a[i]) j = nxt[j];
  if (b[j + 1] == a[i]) ++j;
  if (j == m) {
    io.write(i - m + 1, '\n');
    j = nxt[j];
  }
}
```

AC 自动机

```cpp
int cnt = 1, in[N], vis[N], n, rev[N];

std::string s;

struct node {
  int son[26], fail, flag, ans;
} trie[N];

qi q;

il void insert(std::string s, int num) {
  int p = 1;
  _FRO(i, 0, s.length()) {
    int v = s[i] - 'a';
    if (!trie[p].son[v]) trie[p].son[v] = ++cnt;
    p = trie[p].son[v];
  }
  if(!trie[p].flag) trie[p].flag = num;
  rev[num] = trie[p].flag;
  return ;
}

il void getFail() {
  _FRO(i, 0, 26) trie[0].son[i] = 1;
  trie[1].fail = 0;
  q.push(1);
  while (!q.empty()) {
    int u = q.front(); q.pop();
    int Fail = trie[u].fail;
    _FRO(i, 0, 26) {
      int v = trie[u].son[i];
      if (!v) {
        trie[u].son[i] = trie[Fail].son[i];
        continue;
      }
      trie[v].fail = trie[Fail].son[i];
      ++in[trie[Fail].son[i]];
      q.push(v);
    }
  }
  return ;
}

il void query(std::string s) {
  int p = 1;
  _FRO(i, 0, s.length()) {
    p = trie[p].son[s[i] - 'a'];
    ++trie[p].ans;
  }
  return ;
}

il void topo() {
  _FOR(i, 1, cnt) if (!in[i]) q.push(i);
  while (!q.empty()) {
    int u = q.front(); q.pop();
    vis[trie[u].flag] = trie[u].ans;
    int v = trie[u].fail;
    trie[v].ans += trie[u].ans;
    if (!(--in[v])) q.push(v);
  }
  return ;
}

// insert -> getfail -> query -> topo
```

Manacher

```cpp
t[0] = '!', t[m = 1] = '@';
for (int i = 1; i <= n; ++i) t[++m] = s[i], t[++m] = '@';
t[++m] = '#';
for (int i = 1, c = 0, r = 0; i <= m; ++i) {
  R[i] = r < i ? 1 : std::min(r - i + 1, R[c * 2 - i]);
  while (t[i - R[i]] == t[i + R[i]]) ++R[i];
  chmax(ans, R[i] - 1);
  if (i + R[i] - 1 > r) r = i + R[i] - 1, c = i;
}
```

Z 函数

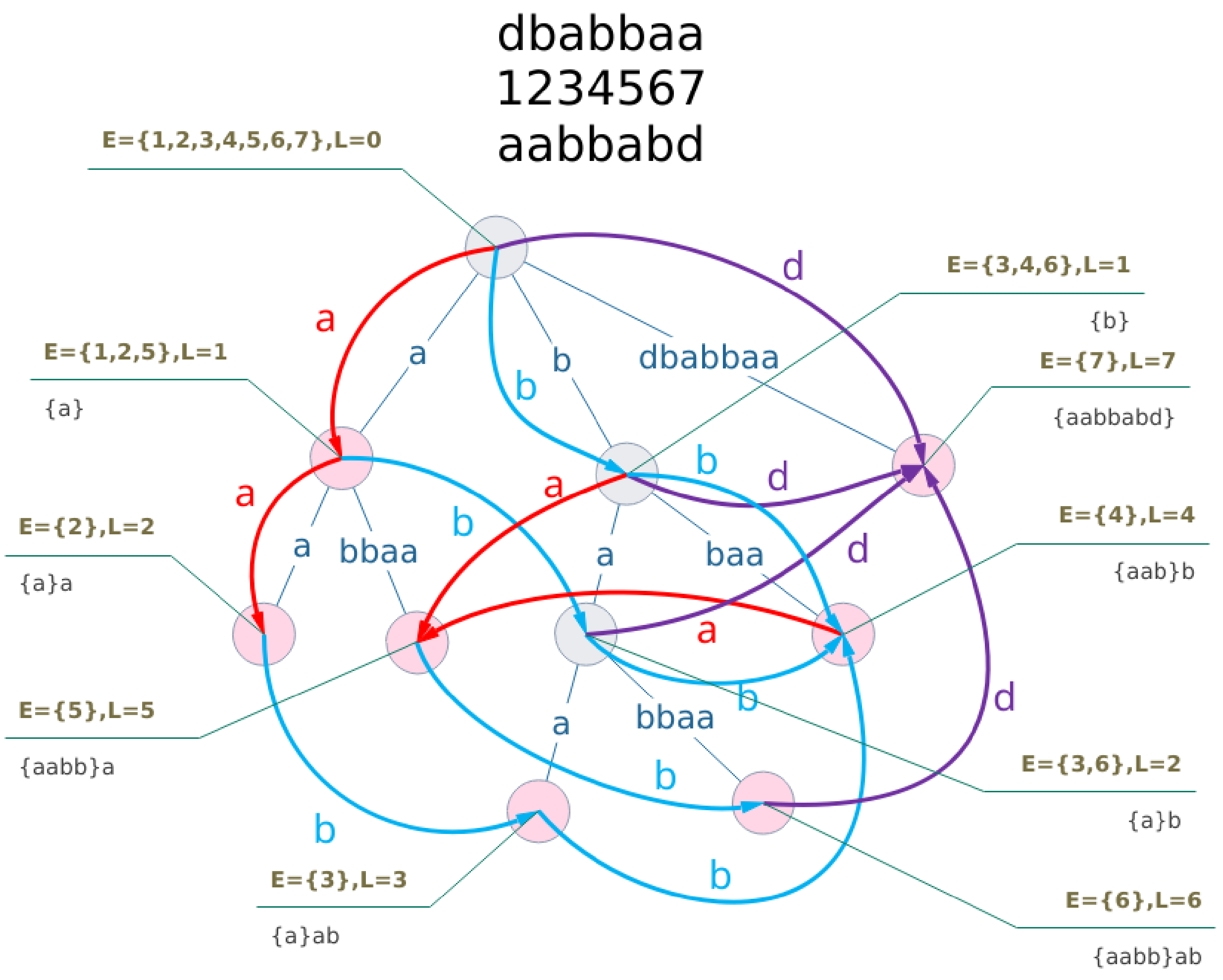
```cpp
n = strlen(a + 1), m = strlen(b + 1);
  z[1] = m;
  for (i32 i = 2, l = 0, r = 0; i <= m; ++i) {
    z[i] = i > r ? 0 : std::min(r - i + 1, z[i - l + 1]);
    while (b[z[i] + 1] == b[i + z[i]]) ++z[i];
    if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
  }
  // for (i32 i = 1; i <= m; ++i) cerr << z[i] << ' ';
  for (i32 i = 1, l = 0, r = 0; i <= n; ++i) {
    p[i] = i > r ? 0 : std::min(r - i + 1, z[i - l + 1]);
    while (p[i] < m && b[p[i] + 1] == a[p[i] + i]) ++p[i];
    if (i + p[i] - 1 > r) l = i, r = i + p[i] - 1;
  }
```

后缀数组

```cpp
for (int i = 1; i <= n; ++i) ++cnt[rk[i] = s[i]];
for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
for (int i = n; i; --i) sa[cnt[rk[i]]--] = i;
for (int w = 1, p, cur; p != n; w <<= 1, m = p) {
  cur = p = 0;
  for (int i = n - w + 1; i <= n; ++i) id[++cur] = i;
  for (int i = 1; i <= n; ++i)
    if (sa[i] > w) id[++cur] = sa[i] - w;
  memset(cnt, 0, (m + 1) * sizeof(int));
  for (int i = 1; i <= n; ++i) ++cnt[rk[i]];
  for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
  for (int i = n; i; --i) sa[cnt[rk[id[i]]]--] = id[i];
  memcpy(oldrk + 1, rk + 1, n * sizeof(int));
  for (int i = 1; i <= n; ++i)
    rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
}

// 求 height 数组
for (i32 i = 1, k = 0; i <= n; ++i) {
  if (k) --k;
  while (s[i + k] == s[sa[rk[i] - 1] + k]) ++k;
  ht[rk[i]] = k;
}
```

后缀自动机



杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```

线性基

```cpp
il void insert(i64 v) {
  for (int i = 49; ~i; --i) {
    if (v >> i) {
      if (!p[i]) {
        p[i] = v;
        return ;
      }
      v ^= p[i];
    }
  }
  return ;
}

il i64 query() {
  i64 res = 0;
  for (int i = 49; ~i; --i) {
    if ((res ^ p[i]) > res) res ^= p[i];
  }
  return res;
}
```

差分约束

```cpp
il bool spfa() {
  memset(dis, 0x3f, sizeof(dis));
  dis[0] = 0;
  std::queue<int> q; q.push(0);
  while (!q.empty()) {
    int u = q.front(); q.pop();
    vis[u] = false;
    for (auto [v, w] : g[u]) {
      if (dis[v] > dis[u] + w) {
        dis[v] = dis[u] + w;
        len[v] = len[u] + 1;
        if (len[v] > n) return false;
        if (!vis[v]) q.push(v), vis[v] = true;
      }
    }
  }
  return true;
}
```