Standard Code Library

mobbb

SMU

May 10, 2024

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// ------------------------------------------------------------------------------
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

Fenwick

```cpp
template<class T>
struct Fenwick{
    vector<T> c;
    int n;
    Fenwick(int _n){
        n = _n;
        c.resize(n + 1);
    }
    T sum(int x){
        T res = 0;
        for (; x ; x -= x & (-x)){
            res += c[x];
        }
```

```
        return res;
    }
    void modify(int x,T d){
        for (;x <= n;x += x & (-x)){
            c[x] += d;
        }
    }
  T rangesum(int l,int r){
    return sum(r) - sum(l - 1);
  }
};
```

Segment tree (区间修改)

```
const ll N = 2e5 + 7;

const int P = 571373;

int a[N];

struct tag{
    ll mul,add;
};

struct Node{
    ll val,siz;
    tag t;
}seg[N * 4];

tag operator + (const tag &t1,const tag &t2){
    return {t1.mul * t2.mul % P,(t1.add * t2.mul % P + t2.add) % P};
}

void update(int id){
    seg[id].val = (seg[id * 2].val + seg[id * 2 + 1].val) % P;
}

void settag(int id,tag t){
    seg[id].t = seg[id].t + t;
    seg[id].val = (seg[id].val * t.mul % P + t.add * seg[id].siz % P) % P;
}

void pushdown(int id){
    if (seg[id].t.mul != 1 || seg[id].t.add != 0){
        settag(id * 2,seg[id].t);
        settag(id * 2 + 1,seg[id].t);
        seg[id].t = {1,0};
    }
}

void build(int id,int l,int r){
    seg[id].siz = r - l + 1;
    seg[id].t = {1,0};
    if (l == r){
        seg[id].val = a[l];
        return;
    }
    int mid = l + r >> 1;
    build(id * 2,l,mid);
```

```
    build(id * 2 + 1,mid + 1,r);
    update(id);
}

void modify(int id,int l,int r,int ql,int qr,tag t){
    if (l == ql && r == qr){
        settag(id,t);
        return;
    }
    pushdown(id);
    int mid = l + r >> 1;
    if (qr <= mid){
        modify(id * 2,l,mid,ql,qr,t);
    }else if (ql > mid){
        modify(id * 2 + 1,mid + 1,r,ql,qr,t);
    }else{
        modify(id * 2,l,mid,ql,mid,t);
        modify(id * 2 + 1,mid + 1,r,mid + 1,qr,t);
    }
    update(id);
}

int query(int id,int l,int r,int ql,int qr){
    if (l == ql && r == qr){
        return seg[id].val;
    }
    pushdown(id);
    int mid = l + r >> 1;
    if (qr <= mid){
        return query(id * 2,l,mid,ql,qr);
    }else if (ql > mid){
        return query(id * 2 + 1,mid + 1,r,ql,qr);
    }else{
        return (query(id * 2,l,mid,ql,mid) + query(id * 2 + 1,mid + 1,r,mid + 1,qr)) % P;
    }
}
```

Segment tree(单点修改)

```
const int N = 2e5 + 7;

int a[N];

struct info{
    int acc;
};

info operator + (const info &l,const info &r){
    info b;
    b.acc = max(l.acc , r.acc);
    return b;
}

struct node{
    info s;
}seg[N * 4];

void update(int id){
    seg[id].s = seg[2 * id].s + seg[2 * id + 1].s;
}

```

```
void build(int id,int l,int r){
    if (l == r){
        seg[id].s = {a[l]};
    }else{
        int mid = (l + r) / 2;
        build(id * 2,l,mid);
        build(id * 2 + 1,mid + 1,r);
        update(id);
    }
}

void change(int id,int l,int r,int pos,int val){
    if (l == r){
        seg[id].s.acc = val;
        a[pos] = val;
    }else{
        int mid = (l + r) / 2;
        if (pos <= mid) {
            change(2 * id,l,mid,pos,val);
        }
        else{
            change(2 * id + 1,mid + 1,r,pos,val);
        }
        update(id);
    }
}

info query(int id,int l,int r,int ql,int qr){
    if (l == ql && r == qr){
        return seg[id].s;
    }
    int mid = (l + r) / 2;
    if (qr <= mid){
        return query(id * 2,l,mid,ql,qr);
    }else if (ql > mid){
        return query(id * 2 + 1,mid + 1,r,ql,qr);
    }else{
        return query(id * 2,l,mid,ql,mid) + query(id * 2 + 1,mid + 1,r,mid + 1,qr);
    }
}
```

重链树剖

```
int n;
vector<int> e[N];


int in[N],ou[N],sz[N],hs[N],tot,fa[N],idx[N],top[N],dep[N];
// top 为重链头部节点
void dfs1(int u,int f){
    sz[u] = 1;
    hs[u] = -1;
    fa[u] = f;
    for (auto v : e[u]) if (v != f){
            dep[v] = dep[u] + 1;
            dfs1(v,u);
            sz[u] += sz[v];
            if (hs[u] == -1 || sz[v] > sz[hs[u]]){
                hs[u] = v;
            }
        }
}

void dfs2(int u,int t){
    top[u] = t;
```

```
    in[u] = ++tot;
    idx[tot] = u;
    if (hs[u] != -1){
        dfs2(hs[u],t);
    }
    for (auto v : e[u]) if(v != fa[u] && v != hs[u]){
            dfs2(v,v);
        }
    ou[u] = tot;
}
```

莫队

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n,m;cin >> n >> m;
    vector<int> a(n + 5),c(n + 5);
    vector<ll> res(m + 5),ansf(m + 5);

    for (int i = 1;i <= n;i++){
        cin >> a[i];
    }
    vector<array<int,3>> qury;

    for (int i = 1;i <= m;i++){
        int l,r;cin >> l >> r;
        qury.push_back({l, r, i});
        ansf[i] = 1ll * (r - l) * (r - l + 1) / 2;
    }
    int B = 500;
    sort(qury.begin(),qury.end(),[&](array<int,3> a,array<int,3> b){
        int c = a[0] / B;
        if (a[0] / B != b[0] / B) return a[0] / B < b[0] / B;
        return c % 2 == 0 ? a[1] < b[1] : a[1] > b[1];
    });
    int l = 1,r = 0,ans = 0;

    auto add = [&](int x)->void{
        ans += c[a[x]];
        c[a[x]]++;
    };
    auto del = [&](int x)->void{
        c[a[x]]--;
        ans -= c[a[x]];
    };

    for (int i = 0;i < m;i++){
        while (r < qury[i][1]) r++,add(r);
        while (l > qury[i][0]) l--,add(l);
```

```
        while (r > qury[i][1]) del(r),r--;
        while (l < qury[i][0]) del(l),l++;
        res[qury[i][2]] = ans;
    }
    for (int i = 1;i <= m;i++){
        if (ansf[i] == 0) cout << "0/1\n";
        else
          cout << res[i] / gcd(res[i],ansf[i]) << "/" << ansf[i] / gcd(res[i],ansf[i]) << endl;
    }
    return 0;
}
```

树剖例题

输入初始颜色后，每次 change 将 [l , r] 中的全部改为 c，询问区间内的颜色段数。

线段树 + 树剖

```
#include <bits/stdc++.h>

using namespace std;

#define ll long long

const int N = 1e5 + 7;

int a[N],n,m;
vector<int> e[N];


int in[N],ou[N],sz[N],hs[N],tot,fa[N],idx[N],top[N],dep[N];

void dfs1(int u,int f){
    sz[u] = 1;
    hs[u] = -1;
    fa[u] = f;
    for (auto v : e[u]) if (v != f){
            dep[v] = dep[u] + 1;
            dfs1(v,u);
            sz[u] += sz[v];
            if (hs[u] == -1 || sz[v] > sz[hs[u]]){
                hs[u] = v;
            }
        }
}

void dfs2(int u,int t){
    top[u] = t;
    in[u] = ++tot;
    idx[tot] = u;
    if (hs[u] != -1){
        dfs2(hs[u],t);
    }
    for (auto v : e[u]) if(v != fa[u] && v != hs[u]){
            dfs2(v,v);
        }
    ou[u] = tot;
}
```

```
struct info{
    int lc,rc,seg;
};

struct Node{
    info val;
    int tag;
}seg[N * 4];

info operator +(const info &l,const info &r){
    return (info){l.lc,r.rc,l.seg + r.seg + (l.rc != r.lc)};
}

void update(int id){
    seg[id].val = (seg[id * 2].val + seg[id * 2 + 1].val);
}

void settag(int id,int t){
    seg[id].val = {t,t,0};
    seg[id].tag = t;
}

void pushdown(int id){
    if (seg[id].tag){
        settag(id * 2,seg[id].tag);
        settag(id * 2 + 1,seg[id].tag);
        seg[id].tag = 0;
    }
}

void build(int id,int l,int r){
    seg[id].tag = 0;
    if (l == r){
        seg[id].val = {a[idx[l]],a[idx[l]],0};
        return;
    }
    int mid = l + r >> 1;
    build(id * 2,l,mid);
    build(id * 2 + 1,mid + 1,r);
    update(id);
}

void modify(int id,int l,int r,int ql,int qr,int t){
    if (l == ql && r == qr){
        settag(id,t);
        return;
    }
    pushdown(id);
    int mid = l + r >> 1;
    if (qr <= mid){
        modify(id * 2,l,mid,ql,qr,t);
    }else if (ql > mid){
        modify(id * 2 + 1,mid + 1,r,ql,qr,t);
    }else{
        modify(id * 2,l,mid,ql,mid,t);
```

```
        modify(id * 2 + 1,mid + 1,r,mid + 1,qr,t);
    }
    update(id);
}

info query(int id,int l,int r,int ql,int qr){
    if (l == ql && r == qr){
        return seg[id].val;
    }
    pushdown(id);
    int mid = l + r >> 1;
    if (qr <= mid){
        return query(id * 2,l,mid,ql,qr);
    }else if (ql > mid){
        return query(id * 2 + 1,mid + 1,r,ql,qr);
    }else{
        return (query(id * 2,l,mid,ql,mid) + query(id * 2 + 1,mid + 1,r,mid + 1,qr));
    }
}


int query(int u,int v){
    info ansu{0,0,-1},ansv{0,0,-1};
    while (top[u] != top[v]){
        if (dep[top[u]] < dep[top[v]]) {
            ansv = query(1,1,n,in[top[v]],in[v]) + ansv;
            v = fa[top[v]];
        }
        else{
            ansu = query(1,1,n,in[top[u]],in[u]) + ansu;
            u = fa[top[u]];
        }
    }
    if (dep[u] >= dep[v]) ansu = query(1,1,n,in[v],in[u]) + ansu;
    else
        ansv = query(1,1,n,in[u],in[v]) + ansv;
    return ansu.seg + ansv.seg + (ansu.lc != ansv.lc) + 1;
}


void modify(int u,int v,int w){
    while (top[u] != top[v]){
        if (dep[top[u]] < dep[top[v]]) swap(u,v);
        modify(1,1,n,in[top[u]],in[u],w);
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u,v);
    modify(1,1,n,in[v],in[u],w);
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    cin >> n >> m;
```

```
    for (int i = 1;i <= n;i++){
        cin >> a[i];
    }

    for (int i = 1;i < n;i++){
        int u,v;cin >> u >> v;
        e[u].push_back(v);
        e[v].push_back(u);
    }

    dfs1(1,-1);
    dfs2(1,1);
    build(1,1,n);

    while (m--){
        char op;cin >> op;
        if (op == 'C') {
            int u,v,w;cin >> u >> v >> w;
            modify(u,v,w);
        }else{
            int u,v;cin >> u >> v;
            cout << query(u,v) << endl;
        }
    }

    return 0;
}
```

数学

long * long 整数 mould

```
ll mul(ll x,ll y,ll m){
    x%=m,y%=m;
    ll d = ((long double)x * y / m);
    d = x * y - d * m;
    if(d >= m) d-=m;
    if(d < 0) d +=m;
    return d;
}
```

快速幂

```
ll qkm(ll a,ll b){
    ll x = a , res = 1;
    while (b){
        if (b & 1)
            res *= x , res %= P;
        b >>= 1;
        x *= x , x %= P;
    }
    return res;
}
```

扩展欧几里得

```
int exgcd(int a,int b,int &x,int &y){
    if(b == 0){
        y = 0;
        x = 1;
        return a;
    }
```

```
    int d = exgcd(b,a%b,y,x);
    y -= a/b*x;
    return d;
}
```

线性筛

```
struct Euler{
    vector<int> p,pri;
    Euler (int n){
        p.resize(n + 1);
        pri.resize(n + 1);
        int cnt = 0;
        for (int i = 2;i <= n;i++){
            if (!p[i]) p[i] = i,pri[++cnt] = i;
            for (int j = 1;j <= cnt && i * pri[j] <= n;j++){
                p[i * pri[j]] = pri[j];
                if (i == pri[j])break;
            }
        }
    }
};
```

整数分块

```
void solution(){
    ll n;cin>>n;
    unsigned ll sum = 0;
    for(ll l = 1;l<=n;l++){
        ll d = n/l,r = n/d;
        sum += (r-l+1)*d;
        l = r;
    }
}
```

最大质因数

```
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int t;
long long max_factor, n;

long long gcd(long long a, long long b) {
  if (b == 0) return a;
  return gcd(b, a % b);
}

long long quick_pow(long long x, long long p, long long mod) {  // 快速幂
  long long ans = 1;
  while (p) {
    if (p & 1) ans = (__int128)ans * x % mod;
    x = (__int128)x * x % mod;
    p >>= 1;
  }
  return ans;
}

bool Miller_Rabin(long long p) {  // 判断素数
  if (p < 2) return 0;
  if (p == 2) return 1;
  if (p == 3) return 1;
  long long d = p - 1, r = 0;
  while (!(d & 1)) ++r, d >>= 1;  // 将 d 处理为奇数
  for (long long k = 0; k < 10; ++k) {
    long long a = rand() % (p - 2) + 2;
```

```
    long long x = quick_pow(a, d, p);
    if (x == 1 || x == p - 1) continue;
    for (int i = 0; i < r - 1; ++i) {
      x = (__int128)x * x % p;
      if (x == p - 1) break;
    }
    if (x != p - 1) return 0;
  }
  return 1;
}

long long Pollard_Rho(long long x) {
  long long s = 0, t = 0;
  long long c = (long long)rand() % (x - 1) + 1;
  int step = 0, goal = 1;
  long long val = 1;
  for (goal = 1;; goal *= 2, s = t, val = 1) {  // 倍增优化
    for (step = 1; step <= goal; ++step) {
      t = ((__int128)t * t + c) % x;
      val = (__int128)val * abs(t - s) % x;
      if ((step % 127) == 0) {
        long long d = gcd(val, x);
        if (d > 1) return d;
      }
    }
    long long d = gcd(val, x);
    if (d > 1) return d;
  }
}

void fac(long long x) {
  if (x <= max_factor || x < 2) return;
  if (Miller_Rabin(x)) {              // 如果 x 为质数
    max_factor = max(max_factor, x);  // 更新答案
    return;
  }
  long long p = x;
  while (p >= x) p = Pollard_Rho(x);  // 使用该算法
  while ((x % p) == 0) x /= p;
  fac(x), fac(p);  // 继续向下分解 x 和 p
}

int main() {
  scanf("%d", &t);
  while (t--) {
    srand((unsigned)time(NULL));
    max_factor = 0;
    scanf("%lld", &n);
    fac(n);
    if (max_factor == n)  // 最大的质因数即自己
      printf("Prime\n");
    else
      printf("%lld\n", max_factor);
  }
  return 0;
}

```

逆序对

```
int a[N+1];
int c[N+1];
ll solve(int l,int r){
    if(l == r)return 0;
    int m = (l+r)/2;
    int p1 = l , p2 = m+1,cnt = 0;
    ll res = solve(l , m) + solve(m +1 , r);
    while(p1 <= m && p2 <= r){
        if(a[p1] > a[p2])c[++cnt] = a[p1++],res += r - p2 + 1;
        else
            c[++cnt] = a[p2++];
```

```
    }
    while(p1 <= m)c[++cnt] = a[p1++];
    while(p2 <= r)c[++cnt] = a[p2++];
    for(int i  = l ;i <= r ; i++){
        a[i] = c[i - l + 1];
    }
    return res;
}
```

（ax - by = gcd(a,b)）的最小非负整数解 (x,y)

```
int exgcd(int a,int b,int &x,int &y){
    if(b == 0){
        x = 1,y = 0;
        return a;
    }
    int d = exgcd(b,a%b,y,x);
    y -= a/b *x;
    return d;
}
void solution(){
    int a,b,x,y;cin>>a>>b;
    int d = exgcd(a,b,x,y);
    y = -y;
    while(x < 0 || y < 0)x += b/d,y += a/d;
    while(x >= b/d && y >= a/d)x -= b/d , y -= a/d;
    cout<<x<<" "<<y<<" "<<endl;
}
```

(Ax + by = d) 的非负整数解

```
int exgcd(int a,int b,int &x,int &y){
    if(b == 0){
        x = 1,y = 0;
        return a;
    }
    int d = exgcd(b,a%b,y,x);
    y -= a/b *x;
    return d;
}
void solution(){
    int a,b,x,y; ll m;cin>>a>>b>>m;
    int d = exgcd(a,b,x,y);
    if(m % d){
        cout<<-1<<endl;
        return;
    }
    a/=d,b/=d,m/=d;
    // ax+by = 1;
    ll xx =  m % b * x % b;
    if(xx < 0) xx += b;
    ll yy = (m - a*xx)/b;
    if(yy < 0){
        cout<<-1<<endl;
        return;
    }
    cout<<xx<<" "<<yy<<endl;
}
```

组合数//逆元

```
struct Comb {
    int n;
    vector<ll> _fac;
    vector<ll> _invfac;
    vector<ll> _inv;

    Comb() : n{0}, _fac{1}, _invfac{1}, _inv{0} {}
```

```
    ll qkm(ll a,ll b){
        ll x = a , res = 1;
        while (b){
            if (b & 1)
                res *= x , res %= P;
            b >>= 1;
            x *= x , x %= P;
        }
        return res;
    }

    Comb(int n) : Comb() {
        init(n);
    }

    void init(int m) {
        int n = m;
        _fac.resize(n + 1);
        _invfac.resize(n + 1);
        _inv.resize(n + 1);

        for (int i = 1;i <= n;i++){
            (_fac[i] = _fac[i - 1] * i) %= P;
        }
        _invfac[n] = qkm(_fac[n],P - 2);
        for (int i = n - 1;i >= 0;i--){
            _invfac[i] = _invfac[i + 1] * (i + 1) % P;
        }
        for (int i = 2;i <= n;i++){
            _inv[i] = (P - P / i) * _inv[P % i] % P;
        }
    }

    ll fac(int m) {
        return _fac[m];
    }
    ll invfac(int m) {
        return _invfac[m];
    }
    ll inv(int m) {
        return _inv[m];
    }
    ll binom(int n, int m) {
        if (n < m || m < 0) return 0;
        return fac(n) * invfac(m)  % P * invfac(n - m) % P;
    }
} comb(N);
```

CRT（对于 x 同余 ai 于 mi）（保证有解）

```
ll exgcd(ll o,ll i,ll &x,ll &y){
    if(i == 0){
        x = 1,y = 0;
        return o;
    }
    ll d = exgcd(i,o%i,y,x);
```

```
    y -= o / i * x;
    return d;
}
void merge(ll &a,ll &b,ll c,ll d){
    ll x,y;
    ll g = exgcd(b,d,x,y);
    d/=g;
    ll t = (c-a)/g  * x % d;
    if( t < 0)t += d;
    a = b*t+a;
    b = b*d;
}
void solution(){
    int n;cin>>n;
    ll a = 0,b =1;
    for(int i =1;i<=n;i++){
        ll c,d;cin>>c>>d;
        merge(a,b,c,d);
    }
    cout<<a<<endl;
}
```

CRT 判断有解

```
void solution(){
    int n;cin>>n;
    unordered_map<int,vector<pair<int,int>>> f;
     for(int i =1;i<=n;i++){
         int a,m;cin>>a>>m;
         for(int j =2;j * j <= m;j++)if(m % j == 0){
             int tj = 1;
             while(m%j == 0)m/=j,tj*=j;
             f[j].emplace_back(tj,a%tj);
         }
         if(m != 1){
             f[m].emplace_back(m,a%m);
         }
     }
     for(auto [x,y]:f){
         int vv = max_element(y.begin(),y.end())->second;
         for(auto [u,v]:y){
             if(vv % u != v){
                 cout<<"No\n";
                 return;
             }
         }
     }
     cout<<"Yes\n";
}
```

图论

LCA

- 倍增

```
1 void dfs(int u, int fa) {
```

```
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

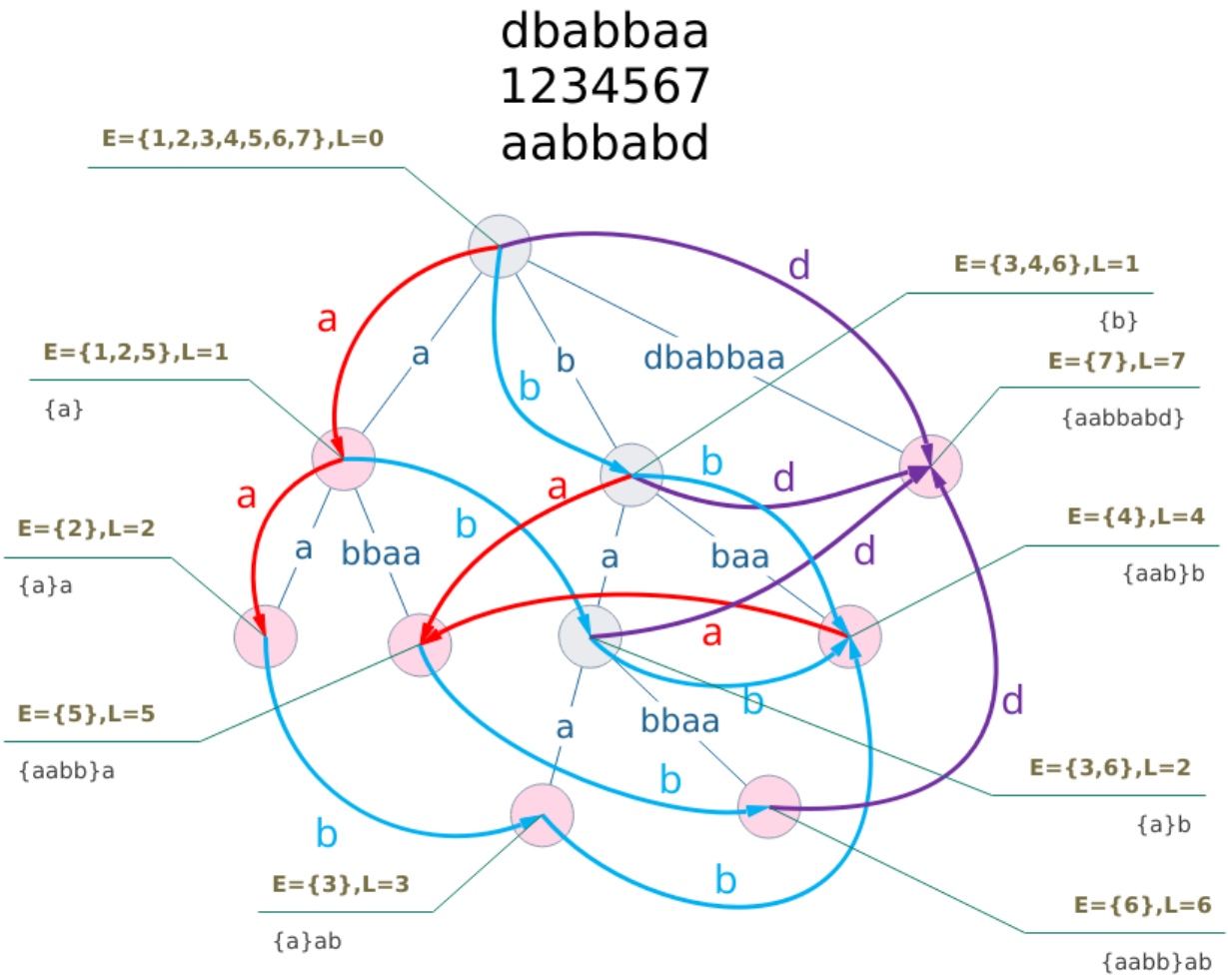
```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y)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

后缀自动机



Manacher

```
1
2   struct manacher{
3       // 1 index
4       string s;
5       vector<int> p,l,r;
6       int res = 0,m;
7       manacher(string t){
8           s = " $";
9           for (auto x : t){
10              s += x;
11              s += "$";
12          }
13          // k.....M......i
14          // M - k = i - M -> k = 2M - i
15          p.resize((int)s.size());
16          int M = 0,R = 0;
17          m = p.size() - 1;
18          for (int i = 1;i <= m;i++){
19              if (i > R)
20                  p[i] = 1;
21              else
22                  p[i] = min(p[2 * M - i],R - i + 1);
23              while (i - p[i] > 0 && i + p[i] <= m && s[i - p[i]] == s[i + p[i]]){
```

```
                p[i]++;
            }
            if (i + p[i] - 1 > R){
                M = i,R = i + p[i] - 1;
            }
            res = max(res,p[i] - 1);
        }
    };
    int get(int pos){
        return p[(pos) << 1] - 1;
    }
    int get(int posl,int por){
        return p[((posl) << 1) | 1] - 1;
    }
    bool isPalindrome(int l, int r){
        if (l > r) swap(l,r);
        int mid = l + r >> 1;
        int len = r - l + 1;
        return len & 1 ? get(mid) >= len : get(mid,mid + 1) >= len;
    }
};
```

字符串哈希

```
template<class T>
struct hashtable{
    vector<unsigned long long> h,p,rh,rp;
    T str; // 1 idx
    int P = 131; //13331
    hashtable(T _str){
        str = _str;
        h.resize(str.size());
        p.resize(str.size());
        rh.resize(str.size() + 1);
        rp.resize(str.size() + 1);
    };
    void pos(){
        auto n = str.size() - 1;
        p[0] = 1;
        for (int i = 1;i <= n;i++){
            h[i] = h[i - 1] * P + str[i];
            p[i] = p[i - 1] * P;
        }
    }
    void suf(){
        auto n = str.size() - 1;
        for (int i = n;i >= 1;i--){
            rh[i] = rh[i + 1] * P + str[i];
        }
    }
    unsigned long long get_pre(int l,int r){
        return h[r] - h[l - 1] * p[r - l + 1];
    }
    unsigned long long get_suf(int l,int r){
        return rh[l] - rh[r + 1] * p[r - l + 1];
    }

    bool isPal(int l,int r){
        return (get_pre(l,r) == get_suf(l,r));
    }

};

```

KMP

```
struct KMP{
    // res存的是匹配成功的最后一位的下标 i ， 第一位为 i - 2 * m;
    vector<int> nxt,ed,bg;
```

```
    KMP(int n,int m,string a,string b){
        b = " " + b,a = a;
        nxt.resize(n + m + 7);
        b += "#" + a;
        int j = 0;
        nxt[1] = 0;
        for (int i = 2;i <= n + m + 1;i++){
            while (j && b[i] != b[j + 1])
                j = nxt[j];
            if (b[i] == b[j + 1])
                j++;
            nxt[i] = j;
        }
        for (int i = m + 2;i <= n + m + 1;i++)
            if (nxt[i] == m)
                ed.push_back(i),bg.push_back(i - 2 * m);
    }
};
```

EXKMP

```
struct EXKMP{
    vector<int> z;
    vector<int> bg;//从第几位开始可以匹配 b
    EXKMP(int n,int m,string a,string b){
        b = " " + b + "#" + a;a = a;
        z.resize(n + m + 2);
        int l = 1,r = 0;
        z[1] = 0;
        for (int i = 2;i <= n + m + 1;i++){
            if (i > r)
                z[i] = 0;
            else{
                int k = i - l + 1;
                z[i] = min(z[k],r - i + 1);
            }
            while (i + z[i] <= n + m + 1 && b[z[i] + 1] == b[z[i] + i]){
                z[i]++;
            }
            if (i + z[i]  - 1 > r)
                l = i,r = i + z[i] - 1;
        }
        for (int i = m + 2;i <= n + m + 1;i++){
            if (z[i] == m)bg.push_back(i - m - 1);
        }
    }
};
```

杂项

高精度

```
struct Bigint {
    // representations and structures
    string a; // 存储数字位数 to store the digits
    int sign; // sign = -1 for negative numbers, sign = 1 otherwise
    // constructors
    Bigint() {} // default constructor
    Bigint(string b) {
        a = b[0] == '-' ? b.substr(1) : b;
        reverse(a.begin(), a.end());
        this->normalize(b[0] == '-' ? -1 : 1);
```

```
}// constructor for string

// some helpful methods
int size() { // 返回数字位数 returns number of digits
    return a.size();
}

Bigint inverseSign() { // changes the sign
    sign *= -1;
    return (*this);
}

Bigint normalize(int newSign) { // removes leading 0, fixes sign
    for (int i = a.size() - 1; i > 0 && a[i] == '0'; i--)
        a.erase(a.begin() + i);
    sign = (a.size() == 1 && a[0] == '0') ? 1 : newSign;
    return (*this);
}

// assignment operator
void operator=(string b) { // assigns a string to Bigint
    a = b[0] == '-' ? b.substr(1) : b;
    reverse(a.begin(), a.end());
    this->normalize(b[0] == '-' ? -1 : 1);
}

void operator=(int b) { //
    string c = to_string(b);
    (*this) = c;
}

// conditional operators
bool operator<(const Bigint &b) const { // less than operator
    if (sign != b.sign) return sign < b.sign;
    if (a.size() != b.a.size())
        return sign == 1 ? a.size() < b.a.size() : a.size() > b.a.size();
    for (int i = a.size() - 1; i >= 0; i--)
        if (a[i] != b.a[i])
            return sign == 1 ? a[i] < b.a[i] : a[i] > b.a[i];
    return false;
}

bool operator<(const int &b) const {
    Bigint c(to_string(b));
    return (*this) < c;
}

bool operator>(const Bigint &b) const {
    return b < (*this);
}

bool operator>(const int &b) const {
    Bigint c(to_string(b));
    return (*this) > c;
}
```

```
bool operator==(const Bigint &b) const { // operator for equality
    return a == b.a && sign == b.sign;
}

bool operator==(const int &b) const {
    Bigint c(to_string(b));
    return (*this) == c;
}

bool operator<=(const Bigint &b) const {
    return (*this) < b or (*this) == b;
}

bool operator<=(const int &b) const {
    Bigint c(to_string(b));
    return (*this) <= c;
}

bool operator>=(const Bigint &b) const {
    return (*this) > b or (*this) == b;
}

bool operator>=(const int &b) const {
    Bigint c(to_string(b));
    return (*this) >= c;
}

// mathematical operators
Bigint operator+(Bigint b) { // addition operator overloading
    if (sign != b.sign) return (*this) - b.inverseSign();
    Bigint c;
    for (int i = 0, carry = 0; i < a.size() || i < b.size() || carry; i++) {
        carry += (i < a.size() ? a[i] - 48 : 0) + (i < b.a.size() ? b.a[i] - 48 : 0);
        c.a += (carry % 10 + 48);
        carry /= 10;
    }
    return c.normalize(sign);
}

Bigint operator+(int b) {
    Bigint c(to_string(b));
    return (*this) + b;
}

Bigint operator-(Bigint b) { // subtraction operator overloading
    if (sign != b.sign) return (*this) + b.inverseSign();
    int s = sign;
    sign = b.sign = 1;
    if ((*this) < b) return ((b - (*this)).inverseSign()).normalize(-s);
    Bigint c;
    for (int i = 0, borrow = 0; i < a.size(); i++) {
        borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);
        c.a += borrow >= 0 ? borrow + 48 : borrow + 58;
        borrow = borrow >= 0 ? 0 : 1;
    }
    return c.normalize(s);
```

```
}

Bigint operator-(int b) {
    Bigint c(to_string(b));
    return (*this) - b;
}

Bigint operator*(Bigint b) { // multiplication operator overloading
    Bigint c("0");
    for (int i = 0, k = a[i] - 48; i < a.size(); i++, k = a[i] - 48) {
        while (k--) c = c + b; // ith digit is k, so, we add k times
        b.a.insert(b.a.begin(), '0'); // multiplied by 10
    }
    return c.normalize(sign * b.sign);
}

Bigint operator*(int b) {
    Bigint c(to_string(b));
    return (*this) * b;
}

Bigint operator/(Bigint b) { // division operator overloading
    if (b.size() == 1 && b.a[0] == '0') b.a[0] /= (b.a[0] - 48);
    Bigint c("0"), d;
    for (int j = 0; j < a.size(); j++) d.a += "0";
    int dSign = sign * b.sign;
    b.sign = 1;
    for (int i = a.size() - 1; i >= 0; i--) {
        c.a.insert(c.a.begin(), '0');
        c = c + a.substr(i, 1);
        while (!(c < b)) c = c - b, d.a[i]++;
    }
    return d.normalize(dSign);
}

Bigint operator/(int b) {
    assert(b != 0);
    Bigint c(to_string(b));
    return (*this) / b;
}

Bigint operator%(Bigint b) { // modulo operator overloading
    if (b.size() == 1 && b.a[0] == '0') b.a[0] /= (b.a[0] - 48);
    Bigint c("0");
    b.sign = 1;
    for (int i = a.size() - 1; i >= 0; i--) {
        c.a.insert(c.a.begin(), '0');
        c = c + a.substr(i, 1);
        while (!(c < b)) c = c - b;
    }
    return c.normalize(sign);
}

Bigint operator%(int b) {
    assert(b != 0);
    Bigint c(to_string(b));
```

```
        return (*this) % b;
    }

    Bigint operator^(Bigint y) {
        Bigint ans("1"), x = (*this);
        while (y > 1) {
            if (y % 2 == 1) ans = ans * x;
            x = x * x, y = y / 2;
        }
        return ans;
    }

    Bigint operator^(int y) {
        Bigint ans("1"), x = (*this);
        while (y > 1) {
            if (y % 2 == 1) ans = ans * x;
            x = x * x, y = y / 2;
        }
        return ans;
    }

    // output method
    void print() {
        if (sign == -1) cout << '-';
        for (int i = a.size() - 1; i >= 0; i--) cout << a[i];
    }
};

istream &operator>>(istream &is, Bigint &x) {
    string y;
    is >> y;
    x = y;
    return is;
}

ostream &operator<<(ostream &os, const Bigint &x) {
    if (x.sign == -1) os << '-';
    for (int i = x.a.size() - 1; i >= 0; i--) os << x.a[i];
    return os;
}
```