lincolncpp ICPC notebook

Contents

1 data structure

1.1 ordered set

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <functional>

using namespace std;
using namespace __gnu_pbds;

template<typename T>
using ordered_set = tree<T, null_type, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>;

int main(){

    ordered_set<int>s;
    s.insert(10);
    s.insert(9);
    s.insert(50);
    s.insert(1);

    cout << s.order_of_key(50) << endl;
    cout << *s.find_by_order(0) << endl;

    return 0;
}
```

1.2 bit2D

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

using namespace std;

const int maxn = 1e3+3;
int bit[maxn+11][maxn+11];

void updatej(int i, int j, int x){
    while(j <= maxn){
        bit[i][j] += x;
        j += j&(-j);
    }
}

void update(int i, int j, int x){
    while(i <= maxn){
        updatej(i, j, x);
        i += i&(-i);
    }
}

int queryj(int i, int j){
    int res = 0;

    while(j > 0){
        res += bit[i][j];
        j -= j&(-j);
    }

    return res;
}

int query(int i, int j){
    int res = 0;

    while(i > 0){
        res += queryj(i, j);
        i -= i&(-i);
    }

    return res;
}

int query(int ai, int aj, int bi, int bj){
    return query(bi, bj) - query(bi, aj-1) - query(ai-1, bj) + query(ai
        -1, aj-1);
}

int main(){


    return 0;
}
```

1.3 dsu rollback

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

using namespace std;

const int maxn = 2e5+123;

struct operation{
    int a, b;
    int rnka, rnkb;
    operation(int a_, int b_, int rnka_, int rnkb_) : a(a_), b(b_),
        rnka(rnka_), rnkb(rnkb_) {}
};

int link[maxn];
int rnk[maxn];
int comp = 0;
stack<operation>hist;

int find(int x){
    while(x != link[x]) x = link[x];
    return x;
}

void unite(int a, int b){
    a = find(a);
    b = find(b);
    if (a == b) return;
    if (rnk[b] > rnk[a]) swap(a, b);
    hist.push(operation(a, b, rnk[a], rnk[b]));
    comp--;
    link[b] = a;
    if (rnk[a] == rnk[b]) rnk[a]++;
}

void rollback(){
    if (hist.empty()) return;

    operation op = hist.top();
    hist.pop();

    link[op.a] = op.a;
    link[op.b] = op.b;
    rnk[op.a] = op.rnka;
    rnk[op.b] = op.rnkb;
    comp++;
}

int main(){

    int n = 10;
    comp = n;
        for(int i = 1;i <= n;i++) link[i] = i;

    unite(1, 2);
    cout << comp << endl;
    unite(3, 4);
    cout << comp << endl;
    unite(1, 3);
    cout << comp << endl;

    rollback();
    rollback();
    rollback();

    cout << comp << endl;

    return 0;
}
```

1.4 segtree lazy

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

using namespace std;

const int maxn = 1e5;
int tree[4*maxn+7] = {};
int lazy[4*maxn+7] = {};
int n;

void build(const vector<int>&v, int node = 1, int tl = 0, int tr = n-1)
    {
    if (tl == tr) return void(tree[node] = v[tl]);
    int mid = (tl+tr)/2;
    build(v, node*2, tl, mid);
    build(v, node*2+1, mid+1, tr);
    tree[node] = tree[node*2]+tree[node*2+1];
}

void push(int node, int tl, int tr){
    if (tl == tr) tree[node] += lazy[node];
    else{
        tree[node] += lazy[node]*(tr-tl+1);
        lazy[node*2] += lazy[node];
        lazy[node*2+1] += lazy[node];
    }
    lazy[node] = 0;
}

int update(int l, int r, int val, int node = 1, int tl = 0, int tr = n
    -1){
    push(node, tl, tr);
    if (l > r) return 0;
    if (tl == l && tr == r){
        lazy[node] += val;
        push(node, tl, tr);
        return tree[node];
    }
    int mid = (tl+tr)/2;
    int left = update(l, min(r, mid), val, node*2, tl, mid);
    int right = update(max(mid+1, l), r, val, node*2+1, mid+1, tr);
    tree[node] = tree[node*2]+tree[node*2+1];
    return left+right;
}

int query(int l, int r){
    return update(l, r, 0);
}

int main(){
    vector<int>v = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    n = v.size();

    build(v);

    update(0, n-1, 10);
    cout << query(0, n-1) << endl;

    return 0;
}
```

1.5 min queue

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

using namespace std;

#define INF (1<<29)

struct minqueue{
private:
    int add = 0;
    stack<pair<int, int>>sl, sr;

    void push(int x, stack<pair<int, int>>&s){
        if (s.size() > 0) s.push({x, std::min(x, s.top().second)});
        else s.push({x, x});
    }
    void move(){
        if (sl.size() == 0){
            while(sr.size()) {
                push(sr.top().first, sl);
                sr.pop();
            }
        }
    }

public:
    void push(int x){
        push(x-add, sr);
    }

    void pop(){
        move();
        if (sl.size()) sl.pop();
    }

    int top(){
        move();
        if (sl.size()) return sl.top().first+add;
        return 0;
    }

    int min(){
        move();
        int res = INF;
        if (sl.size()) res = std::min(res, sl.top().second+add);
        if (sr.size()) res = std::min(res, sr.top().second+add);
        return res;
    }

    int size(){
        return sl.size()+sr.size();
    }
    void increase(int x){
        add += x;
    }

};

int main(){

    minqueue q;
    q.push(1);
    q.push(2);
    q.push(3);
    q.increase(100);
    q.push(7);

    while(q.size()){
        cout << q.top() << " " << q.min() << endl;
        q.pop();
    }

    return 0;
}
```

1.6 implicit treap

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

using namespace std;

/*
    This structure is 0-based !!!

    Operations O(logN):
    - Insert
    - Erase
    - Update
    - Erase interval
    - Add on the interval
    - Query value
    - Query sum on the interval
    - Reverse on the interval
*/

struct implicit_treap{
    struct node{
        int value, prior;
        node *l = nullptr, *r = nullptr;
        int cnt = 1;
        int sum = 0;
        int lazy = 0;
        bool rev = false;

        node(int k) : value(k), sum(k), prior(rand()) {}
    };

    node *t = nullptr;

    void push(node *n){
        if (n){
            n->value += n->lazy;
            n->sum += cnt(n)*n->lazy;
            if (n->l) n->l->lazy += n->lazy;
            if (n->r) n->r->lazy += n->lazy;
            n->lazy = 0;

            if (n->rev){
                n->rev = false;
                swap(n->l, n->r);
                if (n->l) n->l->rev ^= true;
                if (n->r) n->r->rev ^= true;
            }
        }
    }

int cnt(node *n){return n?n->cnt:0;}
void upd_cnt(node *n){if (n) n->cnt = 1+cnt(n->l)+cnt(n->r);}

int sum(node *n){return n?n->sum:0;}
void upd_sum(node *n){if (n) n->sum = n->value+sum(n->l)+sum(n->r)
    ;}

void merge(node *&root, node *l, node *r){
    push(l);
    push(r);

    if (!r || !l) root = l?l:r;
    else if (l->prior > r->prior){
        root = l;
        merge(l->r, l->r, r);
    }
    else{
        root = r;
        merge(r->l, l, r->l);
    }

    upd_cnt(root);
    upd_sum(root);
}

void split(node *root, int key, node *&l, node *&r, int add = 0){
    if (!root) return void(l = r = nullptr);
    push(root);

    int curr_key = add + cnt(root->l);
    if (key <= curr_key){
        r = root;
        split(root->l, key, l, root->l, add);
    }
    else{
        l = root;
        split(root->r, key, root->r, r, add + 1 + cnt(root->l));
    }

    upd_cnt(root);
    upd_sum(root);
}

void insert(int key, int value){
    node *element = new node(value);
    node *tl = nullptr, *tr = nullptr, *aux = nullptr;

    split(t, key, tl, tr);
    merge(aux, tl, element);
    merge(t, aux, tr);
}

void erase(node *&root, int key, int add = 0){
    if (!root) return;
    push(root);

    int curr_key = add + cnt(root->l);
    if (curr_key == key) {
        node *aux = root;
        merge(root, root->l, root->r);
```

```
            delete aux;
        }
        else{
            if (key < curr_key) erase(root->l, key, add);
            else erase(root->r, key, add + 1 + cnt(root->l));
        }

        upd_cnt(root);
        upd_sum(root);
    }
    void erase(int key){erase(t, key);}

    void erase(int key_l, int key_r){
        node *tl = nullptr, *tr = nullptr, *aux = nullptr;
        split(t, key_l, tl, aux);
        split(aux, key_r-key_l+1, aux, tr);
        merge(t, tl, tr);
    }

    int get(node *root, int key, int add = 0){
        if (!root) return 0;
        push(root);
        int curr_key = add + cnt(root->l);
        if (curr_key == key) return root->value;
        if (key < curr_key) return get(root->l, key, add);
        return get(root->r, key, add + 1 + cnt(root->l));
    }
    int get(int key){return get(t, key);}

    int query(int key_l, int key_r){
        node *tl = nullptr, *tr = nullptr, *aux = nullptr;
        int res = 0;

        split(t, key_l, tl, aux);
        split(aux, key_r-key_l+1, aux, tr);

        push(aux);
        upd_sum(aux);
        res = sum(aux);

        merge(t, tl, aux);
        merge(t, t, tr);

        return res;
    }

    void update(node *&root, int key, int value, int add = 0){
        if (!root) return;
        push(root);

        int curr_key = add + cnt(root->l);
        if (curr_key == key) root->value = value;
        else{
            if (key < curr_key) update(root->l, key, value, add);
            else update(root->r, key, value, add + 1 + cnt(root->l));
        }

        upd_cnt(root);
        upd_sum(root);
    }
    void update(int key, int value){update(t, key, value);}

    void add(int key_l, int key_r, int value){
        node *tl = nullptr, *tr = nullptr, *aux = nullptr;

        split(t, key_l, tl, aux);
        split(aux, key_r-key_l+1, aux, tr);
        aux->lazy += value;
        merge(t, tl, aux);
        merge(t, t, tr);
    }

    void reverse(int key_l, int key_r){
        node *tl = nullptr, *tr = nullptr, *aux = nullptr;

        split(t, key_l, tl, aux);
        split(aux, key_r-key_l+1, aux, tr);
        aux->rev ^= true;
        merge(t, tl, aux);
        merge(t, t, tr);
    }

    void del(node *n){
        if (!n) return;
        del(n->l);
        del(n->r);
        delete n;
    }

    void print(node *n){
        if (!n) return;
        push(n);
        print(n->l);
        cout << n->value << " ";
        print(n->r);
    }

    void print(){
        print(t);
        cout << endl;
    }

    int size(){return cnt(t);}
    ~implicit_treap(){del(t);}
};

implicit_treap t;

int main(){

    t.insert(0, 1);
    t.insert(0, 2);
    t.insert(0, 3);
    t.insert(0, 4);
    t.insert(0, 5);
    t.insert(0, 6);
    t.insert(0, 7);
    t.insert(0, 8);
    t.insert(0, 9);
    t.print();

    t.erase(1);
    t.update(t.size()-1, 1000);
    t.print();

    t.erase(4, 6);
    t.print();

    t.add(-1e9, 1e9, -5);
    t.print();

    cout << "sum(1, 3) = " << t.query(1, 3) << endl;

    t.reverse(-1e9, 1e9);
    t.print();

    cout << "get(0) = " << t.get(0) << endl;
    return 0;
}
```

1.7 sparse table

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

using namespace std;

/*
    Build: O(nlogn)
    Query: O(1)
*/

#define lg2(x) 31-__builtin_clz(x)

const int maxn = 1e5;
const int logmaxn = lg2(maxn);

int st[maxn+7][logmaxn+3] = {};

void build(const vector<int>&v){
    int n = (int)v.size();

    for(int i = 0;i < n;i++) st[i][0] = v[i];

    for(int j = 1;j <= logmaxn;j++){
        for(int i = 0;i+(1<<j) <= maxn+1;i++){
            st[i][j] = max(st[i][j-1], st[i+(1<<(j-1))][j-1]);
        }
    }
}

// Range maximum query
int query(int l, int r){
    int j = lg2(r-l+1);
    return max(st[l][j], st[r-(1<<j)+1][j]);
}

int main(){

    vector<int>v = {10, 2, 3, 4, 5, 6};
    build(v);

    cout << query(1, 4) << endl;

    return 0;
}
```

1.8 persistent segtree

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

using namespace std;

int n;

struct node{
    node *l, *r;
    int sum;

    node(int x){
        l = nullptr;
        r = nullptr;
        sum = x;
    }
    node(node* left, node* right){
        l = left;
        r = right;
        sum = 0;
        if (l != nullptr) sum += l->sum;
        if (r != nullptr) sum += r->sum;
    }
};

node* build(const vector<int> &v, int tl = 0, int tr = n-1){
    if (tl == tr) return new node(v[tl]);
    int mid = (tl+tr)/2;
    return new node(build(v, tl, mid), build(v, mid+1, tr));
}

int query(node *seg, int l, int r, int tl = 0, int tr = n-1){
    if (l > r || seg == nullptr) return 0;
    if (tl == l && tr == r) return seg->sum;
    int mid = (tl+tr)/2;
    return  query(seg->l, l, min(r, mid), tl, mid)+
            query(seg->r, max(l, mid+1), r, mid+1, tr);
}

node *update(node *seg, int i, int x, int tl = 0, int tr = n-1){
    if (tl == tr) return new node(x);
    if (seg == nullptr) seg = new node(0);
    int mid = (tl+tr)/2;
    if (i <= mid) return new node(update(seg->l, i, x, tl, mid), seg->r
        );
    else return new node(seg->l, update(seg->r, i, x, mid+1, tr));
}

int main(){

    n = 1e9+111;
    node *seg = nullptr;
    node *seg2 = update(seg, 1e9, 10);
    node *seg3 = update(seg2, 2, 10);
    cout << query(seg, 0, 1e9) << endl;
    cout << query(seg2, 0, 1e9) << endl;
    cout << query(seg3, 0, 1e9) << endl;

    return 0;
}
```

1.9 dsu

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

using namespace std;

const int maxn = 1e5;
int link[maxn];
int siz[maxn];

int find(int x){
    while(x != link[x]){
        link[x] = link[link[x]];
        x = link[x];
    }
    return x;
}
```

```
void unite(int a, int b){
    a = find(a);
    b = find(b);
    if (a == b) return;
    if (siz[a] < siz[b]) swap(a, b);
    link[b] = a;
    siz[a] += siz[b];
}

int main(){

    for(int i = 0;i < maxn;i++){
        link[i] = i;
        siz[i] = 1;
    }

    return 0;
}
```

1.10 bit

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

using namespace std;

#define N (1<<10)

int BIT[N+1] = {0};

void add(int i, int x){
    while(i <= N){
        BIT[i] += x;
        i += i&-i;
    }
}

int query(int i){
    int sum = 0;
    while(i > 0){
        sum += BIT[i];
        i -= i&-i;
    }
    return sum;
}

int main(){

    add(1, 1);
    add(2, 2);
    add(999, 999);

    cout << query(1000) << endl;

    return 0;
}
```

1.11 merge sort tree

```
#include <bits/stdc++.h>
using namespace std;
#define all(x) x.begin(), x.end()

const int maxn = 1e5;
vector<int> tree[4*maxn];
int n;

void build(vector<int>&v, int node = 1, int l = 0, int r = n-1){
    if (l == r) return void(tree[node].push_back(v[l]));
    int mid = (l+r)/2;
    build(v, node*2, l, mid);
    build(v, node*2+1, mid+1, r);
    merge(all(tree[node*2]), all(tree[node*2+1]), back_inserter(tree[
        node]));
}

// Number of elements greater than x
int query(int l, int r, int x, int node = 1, int tl = 0, int tr = n-1){
    if (l > r) return 0;
    if (tl == l && tr == r) return tree[node].end() - lower_bound(all(
        tree[node]), x+1);
    int mid = (tl+tr)/2;
    int a = query(l, min(mid, r), x, node*2, tl, mid);
    int b = query(max(l, mid+1), r, x, node*2+1, mid+1, tr);
    return a+b;
}

int main(){

    vector<int>v = {9, 9, 5, 1, 6, 3, 4, 8, 0, 6, 1, 5, 2};
    n = v.size();
    build(v);

    cout << query(4, 9, 4) << endl;

    return 0;
}
```

1.12 treap

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

using namespace std;

struct treap{
private:
    struct node{
        int key, prior;
        node *l = nullptr, *r = nullptr;
        int cnt = 1;

        node(int k) : key(k), prior(rand()) {}
    };

    node *t = nullptr;

    int cnt(node *n){
        return n?n->cnt:0;
    }

    void upd_cnt(node *n){
        if (n) n->cnt = 1+cnt(n->l)+cnt(n->r);
    }

    void split(node *root, int key, node *&l, node *&r){
        if (!root) l = r = nullptr;
        else if (key <= root->key){
            r = root;
            split(root->l, key, l, root->l);
        }
        else{
            l = root;
            split(root->r, key, root->r, r);
        }

        upd_cnt(l);
        upd_cnt(r);
    }

    void insert(node *&root, node *element){
        if (!root) return void(root = element);

        if (element->prior > root->prior){
            split(root, element->key, element->l, element->r);
            root = element;
        }
        else{
            if (element->key < root->key) insert(root->l, element);
            else insert(root->r, element);
        }

        upd_cnt(root);
    }

    void merge(node *&root, node *l, node *r){
        if (!r || !l) root = l?l:r;
        else if (l->prior > r->prior){
            root = l;
            merge(l->r, l->r, r);
        }
        else{
            root = r;
            merge(r->l, l, r->l);
        }

        upd_cnt(root);
    }

    void erase(node *&root, int key){
        if (!root) return;
        if (root->key == key) {
            node *aux = root;
            merge(root, root->l, root->r);
            delete aux;
        }
        else{
            if (key < root->key) erase(root->l, key);
            else erase(root->r, key);
        }

        upd_cnt(root);
    }

    void del(node *n){
        if (!n) return;
        del(n->l);
        del(n->r);
        delete n;
    }

    /* ==================== EXTRA ==================== */
    int kth(node *root, int pos){ // O(log(N))
        if (!root) return 0;

        int p = 1+cnt(root->l);
        if (p == pos) return root->key;

        if (p > pos) return kth(root->l, pos);
        else return kth(root->r, pos-p);
    }

    node *unite(node *l, node *r){ // O(M*log(N/M))
        if (!l || !r) return l?l:r;
        if (l->prior < r->prior) swap(l, r);
        node *tl;
        node *tr;
        split(r, l->key, tl, tr);
        l->l = unite(l->l, tl);
        l->r = unite(l->r, tr);

        upd_cnt(l);
        return l;
    }

    void print(node *n){
        if (!n) return;
        print(n->l);
        cout << n->key << " ";
        print(n->r);
    }

public:
    void insert(int key){
        insert(t, new node(key));
    }

    void erase(int key){
        erase(t, key);
    }

    ~treap(){
        del(t);
    }

    /* ==================== EXTRA ==================== */
    bool find(int key){
        node *aux = t;
        while(aux && aux->key != key){
            if (key < aux->key) aux = aux->l;
            else aux = aux->r;
        }
        return aux != nullptr;
    }

    int order_of_key(int key){
        int res = 0;

        node *l = nullptr, *r = nullptr;
        split(t, key, l, r);
        res = cnt(l);

        merge(t, l, r);
        return res;
    }

    void erase_range(int l, int r){
        node *tl = nullptr, *tr = nullptr, *aux = nullptr;
        split(t, l, tl, aux);
        split(aux, r+1, aux, tr);
        del(aux);
        merge(t, tl, tr);
    }

    int kth(int pos){
        return kth(t, pos+1);
    }
    void unite(treap &b){
        t = unite(t, b.t);
        b.t = nullptr;
    }

    int size(){
        return cnt(t);
    }

    void print(){
        print(t);
        cout << endl;
    }
};

int main(){

    treap k;
    k.insert(-5);
    k.insert(20);

    treap t;
    t.insert(11);
    t.insert(200);
    t.insert(156);
    t.insert(7);
    t.insert(10);
    t.insert(12);

    cout << "Treap t" << endl;
    t.print();
    cout << endl;

    cout << "Treap k" << endl;
    k.print();
    cout << endl;

    cout << "Treap t + k" << endl;
    t.unite(k);
    t.print();
    cout << endl;

    cout << "Erase [10, 100]" << endl;
    t.erase_range(10, 100);
    t.print();
    cout << endl;

    cout << "3-th element" << endl;
    cout << t.kth(3) << endl;
    cout << endl;

    cout << "order of key 156" << endl;
    cout << t.order_of_key(156) << endl;
    cout << endl;

    cout << "find 11" << endl;
    cout << t.find(11) << endl;

    return 0;
}
```

1.13 segtree

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

using namespace std;

const int maxn = 1e5;
int tree[4*maxn+11] = {};
int n;

void update(int i, int val, int node = 1, int tl = 0, int tr = n-1){
    if (tl > i || tr < i) return;
    if (tl == tr && tl == i) return void(tree[node] = val);

    int mid = (tl+tr)/2;
    update(i, val, node*2, tl, mid);
    update(i, val, node*2+1, mid+1, tr);
    tree[node] = min(tree[node*2], tree[node*2+1]);
}

// Range min query
int query(int l, int r, int node = 1, int tl = 0, int tr = n-1){
    if (l > r) return 0x7fffffff;
    if (tl == l && tr == r) return tree[node];

    int mid = (tl+tr)/2;
    int left = query(l, min(r, mid), node*2, tl, mid);
    int right = query(max(l, mid+1), r, node*2+1, mid+1, tr);
    return min(left, right);
}

int main(){

    vector<int>v = {-555, 70, 4201, -956, 30};
    n = v.size();

    for(int i = 0;i < n;i++) update(i, v[i]);
    cout << query(0, n-1) << endl;

    return 0;
}
```

2 graph

2.1 boruvka

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

using namespace std;

const int maxn = 2e5+123;
int link[maxn];
int siz[maxn];

int find(int x){
    while(link[x] != x){
        link[x] = link[link[x]];
        x = link[x];
    }
    return x;
}

void unite(int a, int b){
    a = find(a);
    b = find(b);
    if (a == b) return;
    if (siz[a] < siz[b]) swap(a, b);
    link[b] = a;
    siz[a] += siz[b];
}
```

```
int boruvka(int n, const vector<tuple<int, int, int>> &edges){
    for(int i = 1;i <= n;i++){
        link[i] = i;
        siz[i] = 1;
    }

    int total_trees = n;
    int weight = 0;

    while(total_trees > 1){
        vector<int>smallest_edge(n+1, -1);

        for(int i = 0;i < edges.size();i++){
            int a, b, c;
            tie(a, b, c) = edges[i];

            a = find(a);
            b = find(b);
            if (a == b) continue;

            if (smallest_edge[a] != -1 && c < get<2>(edges[
                smallest_edge[a]])) smallest_edge[a] = i;
            if (smallest_edge[a] == -1) smallest_edge[a] = i;
            if (smallest_edge[b] != -1 && c < get<2>(edges[
                smallest_edge[b]])) smallest_edge[b] = i;
            if (smallest_edge[b] == -1) smallest_edge[b] = i;
        }

        for(int i = 1;i <= n;i++){
            if (smallest_edge[i] == -1) continue;

            int a, b, c;
            tie(a, b, c) = edges[smallest_edge[i]];

            if (find(a) == find(b)) continue;
            unite(a, b);
            weight += c;
            total_trees--;
        }
    }

    return weight;
}

int main(){

    int n = 4;

    vector<tuple<int, int, int>>edges;
    edges.push_back(make_tuple(1, 2, 100));
    edges.push_back(make_tuple(2, 3, 10));
    edges.push_back(make_tuple(3, 4, 1));
    edges.push_back(make_tuple(4, 1, 10));
    edges.push_back(make_tuple(4, 2, 20));

    cout << boruvka(n, edges) << endl;

    return 0;
}
```

2.2 dijkstra

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

using namespace std;

const int inf = 1<<29;

const int maxn = 1e5+13;

vector<pair<int, int>>adj[maxn+11];
int dist[maxn+11] = {};
bool vis[maxn+13] = {};

int main(){

    int n = 10;
    adj[1].push_back(make_pair(7, 20));
    adj[1].push_back(make_pair(2, 5));
    adj[1].push_back(make_pair(3, 1));
    adj[3].push_back(make_pair(2, 10));
    adj[3].push_back(make_pair(10, 1));
    adj[10].push_back(make_pair(7, 16));

    for(int i = 1;i <= n;i++) dist[i] = inf;
    dist[1] = 0;

    priority_queue<pair<int, int>>pq;
    pq.push({0, 1});
    while(!pq.empty()){
        int a = pq.top().second;
        pq.pop();

        if (vis[a]) continue;
        vis[a] = true;

        for(auto &pr:adj[a]){
            int b = pr.first;
            int w = pr.second;
            if (dist[a]+w < dist[b]){
                dist[b] = dist[a]+w;
                pq.push(make_pair(-dist[b], b));
            }
        }
    }

    for(int i = 1;i <= n;i++){
        cout << "dist from " << 1 << " to " << i << ": " << dist[i] <<
            endl;
    }

    return 0;
}
```

2.3 kosaraju

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

using namespace std;

/*
    Runtime: O(n+m)
*/

const int maxn = 1e5;
vector<int>adj[maxn+11];
vector<int>radj[maxn+11];
bool vis[maxn+11] = {};
vector<int>order, comp;

void dfs1(int a){
    if (vis[a]) return;
    vis[a] = true;
    for(int b:adj[a]) dfs1(b);
    order.push_back(a);
}

void dfs2(int a){
    if (vis[a]) return;
    vis[a] = true;
    comp.push_back(a);
    for(int b:radj[a]) dfs2(b);
}

int main(){

    int n, m;cin>>n>>m;
    for(int i = 0;i < m;i++){
        int a, b;cin>>a>>b;
        adj[a].push_back(b);
        radj[b].push_back(a);
    }

    for(int i = 1;i <= n;i++) dfs1(i);
    for(int i = 1;i <= n;i++) vis[i] = false;

    cout << "Components:" << endl;

    reverse(order.begin(), order.end());

    // It iterates over vertices in topological sort order
    for(int a:order){
        if (vis[a]) continue;

        dfs2(a);
        for(int b:comp) cout << b << " ";
        cout << endl;

        comp.clear();
    }

    /* sample
        5 5
        1 2
        2 3
        3 1
        1 4
        1 5
    */

    return 0;
}
```

2.4 kuhn

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

using namespace std;

/*
    Runtime: O(nm)
    n - vertices
    m - edges
*/

const int maxn = 1e3;

int vis[maxn+123];
int mt[maxn+123];
vector<int> g[maxn+123];

bool kuhn(int v, int t){
    if (vis[v] == t) return false;
    vis[v] = t;

    for(int u:g[v]){
        if (mt[u] == -1 || kuhn(mt[u], t)){
            mt[u] = v;
            return true;
        }
    }
    return false;
}


int main(){

    for(int i = 0;i < maxn;i++) {
        mt[i] = -1;
        vis[i] = -1;
    }

    g[1].push_back(4);
    g[1].push_back(5);
    g[1].push_back(6);
    g[2].push_back(5);
    g[2].push_back(6);
    g[3].push_back(5);

    for(int i = 1;i <= 3;i++) kuhn(i, i); // first part vertex
    for(int i = 4;i <= 6;i++){ // second part vertex
        if (mt[i] != -1){
            cout << mt[i] << " matches " << i << endl;
        }
    }

    return 0;
}
```

2.5 topological sort

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

using namespace std;

const int maxn = 1e5;

int n, m;
int status[maxn] = {};
vector<int>adj[maxn];
vector<int>vec;
bool has_cycle = false;

void dfs(int a){
    if (status[a] == 1) return void(has_cycle = true);
    status[a] = 1;

    for(int b:adj[a]){
        if (status[b] == 2) continue;
        dfs(b);
    }

    vec.push_back(a);
    status[a] = 2;
}

void topological_sort(){
    for(int i = 1;i <= n;i++){
        if (status[i] == 2) continue;
        dfs(i);
    }
    reverse(vec.begin(), vec.end());
}

int main(){

    cin>>n>>m;
    for(int i = 0;i < m;i++){
        int a, b;cin>>a>>b;
        adj[a].push_back(b);
    }

    topological_sort();

    if (has_cycle) cout << "the given graph has cycle." << endl;
    else for(int e:vec) cout << e << " ";

    return 0;
}
```

2.6 eulerian path

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

using namespace std;

// Time complexity O(M*log(M) + N)

int const maxn = 1e5;

set<int>adj[maxn+7];
int degree[maxn+7] = {};
vector<int>path;

void dfs(int a){
    while(!adj[a].empty()){
        int b = *adj[a].begin();
        adj[a].erase(adj[a].begin());
        adj[b].erase(a);
        dfs(b);
    }

    path.push_back(a);
}

int main(){

    int n, m;cin>>n>>m;
    for(int i = 0;i < m;i++){
        int a, b;cin>>a>>b;
        adj[a].insert(b);
        adj[b].insert(a);

        degree[a]++;
        degree[b]++;
    }

    // Checking if the given graph has an eulerian path.
    int cnt_odd = 0;
    int start = 1;
    for(int i = 1;i <= n;i++){
        if (degree[i]%2 == 1){
            cnt_odd++;
            start = i;
        }
    }

    if (cnt_odd == 0 || cnt_odd == 2){
        dfs(start);

        cout << "Eulerian path: ";
        for(int a:path) cout << a << " ";
        cout << endl;
    }
    else{
        cout << "The given graph does not have an eulerian path." <<
            endl;
    }

    return 0;
}
```

2.7 kruskal

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

using namespace std;

#define N (int)1e5

int link[N];
int siz[N];

int find(int x){
    while(x != link[x]) x = link[x];
    return x;
}

void unite(int a, int b){
    a = find(a);
    b = find(b);
    if (siz[a] < siz[b]) swap(a, b);
    link[b] = a;
    siz[a] += siz[b];
}

bool same(int a, int b){
    return find(a) == find(b);
}

int main(){

    for(int i = 0;i < N;i++){
        link[i] = i;
        siz[i] = 1;
    }

    vector<tuple<int, int, int>>edges;
    edges.push_back(make_tuple(1, 2, 100));
    edges.push_back(make_tuple(2, 3, 10));
    edges.push_back(make_tuple(3, 4, 1));
    edges.push_back(make_tuple(4, 1, 10));
    edges.push_back(make_tuple(4, 2, 20));

    sort(edges.begin(), edges.end(), [](tuple<int, int, int>a, tuple<
        int, int, int>b){
        return get<2>(a) < get<2>(b);
    });

    int weight = 0;
    for(auto e:edges){
        int a = get<0>(e);
        int b = get<1>(e);
        int w = get<2>(e);
        if (!same(a, b)){
            weight += w;
            unite(a, b);
        }
    }

    cout << weight << endl;

    return 0;
}
```

2.8 bellmanford

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

using namespace std;

int n, m;
vector<tuple<int, int, int>>edges;

// max can find positive cycle
// min can find negative cycle
void bellmanford(vector<int>&d){
    for(int i = 1;i <= n-1;i++){
        for(auto &e:edges){
            int a, b, w;
            tie(a, b, w) = e;
            d[b] = max(d[b], d[a]+w);
        }
    }
}

int main(){

    cin>>n>>m;
    for(int i = 0;i < m;i++){
        int a, b, w;cin>>a>>b>>w;
        edges.push_back({a, b, w});
    }

    vector<int>d1(n+11, -1e9);
    d1[1] = 0;
    bellmanford(d1);

    vector<int>d2 = d1;
    bellmanford(d2);

    for(int i = 1;i <= n;i++){
        if (d2[i] > d1[i]){
            cout << "the node " << i << " is part of a positive cycle"
                << endl;
        }
    }

    /* input
        3 3
        1 2 1
        2 3 1
        3 1 100
    */

    return 0;
```

```
}
```

2.9 edmonds karp

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

using namespace std;

const int inf = 0x7fffffff;
const int maxn = 1e3;

vector<int>adj[maxn+7];
int parent[maxn+7] = {};
int capacity[maxn+7][maxn+7] = {};
int n;

int bfs(int s, int t){
    fill(parent, parent+n+1, 0);
    parent[s] = -1;

    queue<pair<int, int>>q;
    q.push({s, inf});
    while(!q.empty()){
        int a = q.front().first;
        int flow = q.front().second;
        q.pop();

        for(int b:adj[a]){
            if (parent[b] == 0 && capacity[a][b]){
                parent[b] = a;

                int nflow = min(flow, capacity[a][b]);
                if (b == t) return nflow;

                q.push({b, nflow});
            }
        }
    }

    return 0;
}

int max_flow(int s, int t){
    int flow = 0;

    while(int add = bfs(s, t)){
        flow += add;

        int a = t;
        while(a != s){
            int b = parent[a];
            capacity[a][b] += add;
            capacity[b][a] -= add;
            a = b;
        }
    }

    return flow;
}

void add_edge(int a, int b, int c){
    adj[a].push_back(b);
    adj[b].push_back(a);
    capacity[a][b] = c;
}
```

```cpp
int main(){
    // Sample graph
    //
    //       2  -    3 -
    //     /       /     \
    //    1 - 4 - 5 - 6 - 7

    n = 7;
    add_edge(1, 2, 20);
    add_edge(2, 3, 20);
    add_edge(3, 7, 11);
    add_edge(1, 4, 9);
    add_edge(4, 5, 9);
    add_edge(5, 3, 20);
    add_edge(5, 6, 100);
    add_edge(6, 7, 100);

    cout << max_flow(1, 7) << endl;

    return 0;
}
```

2.10 floyd

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

using namespace std;

#define INF (1<<29)
#define N 100

int main(){

    int edge[N][N] = {0};
    edge[0][1] = 10;
    edge[1][2] = 25;
    edge[2][3] = 1;
    edge[0][2] = 200;

    int dist[N][N];
    for(int i = 0;i < N;i++){
        for(int j = 0;j < N;j++){
            if (i == j) dist[i][j] = 0;
            else if (edge[i][j]) dist[i][j] = edge[i][j];
            else dist[i][j] = INF;
        }
    }

    for(int k = 0;k < N;k++){
        for(int i = 0;i < N;i++){
            for(int j = 0;j < N;j++){
                dist[i][j] = min(dist[i][j], dist[i][k]+dist[k][j]);
            }
        }
    }

    cout << dist[0][2] << endl;

    return 0;
}
```

2.11 bruijn

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

using namespace std;

unordered_map<string, bool>vis;
string curr;

void dfs(string node, string &A){
    for(int i = 0;i < A.size();i++){
        string edge = node+A[i];
        if (vis[edge]) continue;
        vis[edge] = true;

        dfs(edge.substr(1), A);
        curr += A[i];
    }
}

string bruijn(int n, string A){
    string t(n-1, A[0]);

    vis.clear();
    curr = "";
    dfs(t, A);
    reverse(curr.begin(), curr.end());

    return t+curr;
}

int main(){

    // O(k^n), k = #(A)
    cout << bruijn(3, "ABC") << endl;

    return 0;
}
```

2.12 hld

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

using namespace std;

/*
    Build: O(n)
    Query: O(T*logn), where T is the tree_query complexity
*/

const int maxn = 1e5;
vector<int>adj[maxn+7];
int parent[maxn+7] = {};
int depth[maxn+7] = {};
int heavy[maxn+7] = {};
int head[maxn+7] = {};
int pos[maxn+7] = {};
int curr_pos = 0;

int tree_query(int l, int r){
    return 0;
}

int dfs(int a){
    int size = 1;
    int max_csize = 0;

    for(int b:adj[a]){
        if (b == parent[a]) continue;
        parent[b] = a;
        depth[b] = depth[a]+1;

        int csize = dfs(b);
        size += csize;
        if (csize > max_csize){
            max_csize = csize;
            heavy[a] = b;
        }
    }

    return size;
}

void decompose(int a, int h){
    head[a] = h;
    pos[a] = curr_pos++;

    if (heavy[a] != 0) decompose(heavy[a], h);
    for(int b:adj[a]){
        if (b == heavy[a] || b == parent[a]) continue;
        decompose(b, b);
    }
}

// Max query
int query(int a, int b){
    int res = 0;

    while(head[a] != head[b]){
        if (depth[head[a]] < depth[head[b]]) swap(a, b);
        res = max(res, tree_query(pos[head[a]], pos[a]));
        a = parent[head[a]];
    }

    if (pos[a] > pos[b]) swap(a, b);
    res = max(res, tree_query(pos[a], pos[b]));

    return res;
}

int main(){

    dfs(1);
    decompose(1, 1);

    return 0;
}
```

2.13 lca binary lifting

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

using namespace std;

const int maxn = 1e5;
const int maxl = 20;
int up[maxn+13][maxl+3] = {};
int dep[maxn+13] = {};
int timein[maxn+13] = {};
int timeout[maxn+13] = {};
vector<int>adj[maxn+13];
int curr_time = 0;

void dfs(int a, int p){
    timein[a] = curr_time++;
    dep[a] = dep[p]+1;

    up[a][0] = p;
    for(int i = 1;i <= maxl;i++){
        up[a][i] = up[up[a][i-1]][i-1];
    }

    for(int b:adj[a]){
        if (b == p) continue;
        dfs(b, a);
    }

    timeout[a] = curr_time++;
}

// Check if a is ancestor of b
bool is_ancestor(int a, int b){
    return timein[a] < timein[b] && timeout[a] > timeout[b];
}

int lca(int a, int b){
    if (a == b) return a;
    if (is_ancestor(a, b)) return a;
    if (is_ancestor(b, a)) return b;

    for(int i = maxl;i >= 0;i--){
        if (!is_ancestor(up[a][i], b)){
            a = up[a][i];
        }
    }

    return up[a][0];
}

int distance(int a, int b){
    return dep[a]+dep[b]-2*dep[lca(a, b)];
}


int main(){

    // Sample tree
    //
    //          1
    //        /   \
    //       2     3
    //      / \   / \
    //     4   5 6   7
    adj[1].push_back(2);
    adj[1].push_back(3);
    adj[2].push_back(4);
    adj[2].push_back(5);
    adj[3].push_back(6);
    adj[3].push_back(7);

    dfs(1, 1);

    cout << lca(4, 3) << endl;
    cout << distance(4, 3) << endl;

    return 0;
}
```

2.14 dinic

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

```
using namespace std;

const int inf = 1<<29;
const int maxn = 1e3;

struct flow_edge{
    int to, cap, flow;
    flow_edge(int b, int c) : to(b), cap(c), flow(0){}
};
vector<flow_edge>edge;
vector<vector<int>>adj(maxn+7);
vector<int>level(maxn+7);
vector<int>ptr(maxn+7);

void add_edge(int a, int b, int c){
    int n = (int)edge.size();
    edge.emplace_back(b, c);
    edge.emplace_back(a, 0);
    adj[a].push_back(n);
    adj[b].push_back(n+1);
}

bool bfs(int s, int t){
    queue<int>q;
    q.push(s);

    while(!q.empty()){
        int a = q.front();
        q.pop();

        for(int e:adj[a]){
            if (edge[e].cap-edge[e].flow == 0) continue;
            if (level[edge[e].to] != -1) continue;

            level[edge[e].to] = level[a]+1;
            q.push(edge[e].to);
        }
    }

    return level[t] != -1;
}

int dfs(int a, int t, int pushed){
    if (pushed == 0) return 0;
    if (a == t) return pushed;

    for(int &i = ptr[a];i < (int)adj[a].size();i++){
        int e = adj[a][i];
        if (level[edge[e].to] != level[a]+1) continue;
        if (edge[e].cap-edge[e].flow == 0) continue;

        int f = dfs(edge[e].to, t, min(edge[e].cap-edge[e].flow, pushed
            ));
        if (f == 0) continue;

        edge[e].flow += f;
        edge[e^1].flow -= f;
        return f;
    }
    return 0;
}

int max_flow(int s, int t){
    int flow = 0;

    while(true){
        fill(level.begin(), level.end(), -1);
        level[s] = 0;
        if (!bfs(s, t)) break;

        fill(ptr.begin(), ptr.end(), 0);
        while(int add = dfs(s, t, inf)){
            flow += add;
        }
    }

    return flow;
}

int main(){
    // Sample graph
    //
    //       2  -    3 -
    //     /       /     \
    //    1 - 4 - 5 - 6 - 7

    add_edge(1, 2, 20);
    add_edge(2, 3, 20);
    add_edge(3, 7, 11);
    add_edge(1, 4, 9);
    add_edge(4, 5, 9);
    add_edge(5, 3, 20);
    add_edge(5, 6, 100);
    add_edge(6, 7, 100);

    cout << max_flow(1, 7) << endl;

    return 0;
}
```

2.15 centroid decomp

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

using namespace std;

const int maxn = 1e5;

int n;
vector<int>adj[maxn+13];
int size[maxn+13] = {};
int dad[maxn+13] = {};
bool removed[maxn+13] = {};

void dfs(int a, int p){
    size[a] = 1;
    for(int b:adj[a]){
        if (b == p || removed[b]) continue;
        dfs(b, a);
        size[a] += size[b];
    }
}

int centroid(int a, int p, int m){
    for(int b:adj[a]){
        if (b == p || removed[b]) continue;
        if (size[b]*2 > m) return centroid(b, a, m);
    }
    return a;
}

void build(int a, int p){
    dfs(a, -1);
    a = centroid(a, -1, size[a]);
    dad[a] = p;
    removed[a] = true;

    for(int b:adj[a]){
        if (removed[b]) continue;
        build(b, a);
    }
}

int main(){

    cin>>n;
    for(int i = 0;i < n-1;i++){
        int a, b;cin>>a>>b;
        adj[a].push_back(b);
        adj[b].push_back(a);
    }

    build(1, -1);

    return 0;
}
```

2.16 2-sat

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

using namespace std;

const int maxn = 3 * 2;

vector<int>adj[maxn+11];
vector<int>radj[maxn+11];
int vis[maxn+11] = {};
int comp[maxn+11] = {};
vector<int>order;

void dfs1(int a){
    if (vis[a]) return;
    vis[a] = true;
    for(int b:adj[a]) dfs1(b);
    order.push_back(a);
}

void dfs2(int a, int component){
    if (vis[a]) return;
    vis[a] = true;
    comp[a] = component;
    for(int b:radj[a]) dfs2(b, component);
}

int neg(int a){
    if (a&1) return a+1;
    else return a-1;
}

int main(){

    // sample expression
    // (1^2)v(-3^-1)

    // (1^-2)
    // not 1 implies 2
    adj[neg(1*2)].push_back(2*2);
    radj[2*2].push_back(neg(1*2));
    // not 2 implies 1
    adj[neg(2*2)].push_back(1*2);
    radj[1*2].push_back(neg(2*2));

    // (-3^-1)
    // 3 implies not 1
    adj[3*2].push_back(neg(1*2));
    radj[neg(1*2)].push_back(3*2);
    // 1 implies not 3
    adj[1*2].push_back(neg(3*2));
    radj[neg(3*2)].push_back(1*2);

    for(int i = 1;i <= maxn;i++) dfs1(i);
    for(int i = 1;i <= maxn;i++) vis[i] = false;

    reverse(order.begin(), order.end());
    int curr = 1;
    for(int a:order){
        if (vis[a]) continue;
        dfs2(a, curr);
        curr++;
    }

    bool solution = true;
    bool ans[maxn+11] = {};
    for(int i = 1;i <= maxn/2;i++){
        int a = i*2;
        int a_ = i*2-1;

        if (comp[a] == comp[a_]) solution = false;
        ans[i] = comp[a] > comp[a_];
    }

    if (!solution) cout << "There is no solution" << endl;
    else{
        for(int i = 1;i <= maxn/2;i++){
            if (ans[i]) cout << i << ": true" << endl;
            else cout << i << ": false" << endl;
        }
    }

    return 0;
}
```

3 others

3.1 mo

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

using namespace std;

/*
    Runtime: O((n+q)*sqrt(n))
    Runtime w/ hilbert: O(n*sqrt(q))
*/

const int maxn = 2e5+123;
const int logmaxn = 20;

// https://codeforces.com/blog/entry/61203
inline int64_t hilbertOrder(int x, int y, int pow, int rotate) {
        if (pow == 0) {
                return 0;
        }
        int hpow = 1 << (pow-1);
        int seg = (x < hpow) ? (
                (y < hpow) ? 0 : 3
        ) : (
                (y < hpow) ? 1 : 2
        );
        seg = (seg + rotate) & 3;
        const int rotateDelta[4] = {3, 0, 0, 1};
        int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
        int nrot = (rotate + rotateDelta[seg]) & 3;
        int64_t subSquareSize = int64_t(1) << (2*pow - 2);
        int64_t res = seg * subSquareSize;
        int64_t add = hilbertOrder(nx, ny, pow-1, nrot);
        res += (seg == 1 || seg == 2) ? add : (subSquareSize - add - 1)
            ;
        return res;
}

struct Query{
    int l, r, id, block;
    long long ord;

    bool operator<(const Query &b){

        // hilbert curve
        return ord < b.ord;

        // sqrt decomposition
        // if (block == b.block){
        //     if (block%2 == 0) return r < b.r;
        //     else return r > b.r;
        // }
        // else return block < b.block;
    }
};

int a[maxn];
Query qry[maxn];
int curr = 0;
int ans[maxn];

void add(int pos){
    curr += a[pos];
}

void remove(int pos){
    curr -= a[pos];
}

void process(int q){
    int currentl = 0;
    int currentr = -1;

    for(int i = 0;i < q;i++){
        int l = qry[i].l;
        int r = qry[i].r;

        while (currentl < l) remove(currentl++);
        while (currentl > l) add(--currentl);
        while (currentr < r) add(++currentr);
        while (currentr > r) remove(currentr--);

        ans[qry[i].id] = curr;
    }
}

int main(){
    int n, q;cin>>n>>q;
    for(int i = 0;i < n;i++) cin>>a[i];

    int len = (int)sqrt(n)+1;

    for(int i = 0;i < q;i++){
        int l, r;cin>>l>>r;
        l--;r--;

        qry[i].l = l;
        qry[i].r = r;
        qry[i].id = i;
        qry[i].block = l/len;
        qry[i].ord = hilbertOrder(qry[i].l, qry[i].r, logmaxn, 0);
    }

    sort(qry, qry+q);
    process(q);

    for(int i = 0;i < q;i++) cout << ans[i] << endl;

    return 0;
}
```

3.2 alg1

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

using namespace std;

int main(){

    vector<int>v = {1, 2, 3, 4, 3, 2, 1};
    int n = (int)v.size();

    vector<int>left(n);
    for(int i = 0;i < n;i++){
        left[i] = i-1;
        while(left[i] >= 0 && v[i] >= v[left[i]]) left[i] = left[left[i
            ]];
    }

    vector<int>right(n);
    for(int i = n-1;i >= 0;i--){
        right[i] = i+1;
        while(right[i] < n && v[i] >= v[right[i]]) right[i] = right[
            right[i]];
    }

    for(int x:v) cout << x << " ";
    cout << endl;

    cout << "left:" << endl;
    for(int l:left) cout << l << " " ;
    cout << endl;

    cout << "right" << endl;
    for(int r:right) cout << r << " ";
    cout << endl;

    return 0;
}
```

3.3 ancestor

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

using namespace std;

const int maxn = 1e5;
vector<int>adj[maxn+7];
int timein[maxn+7] = {};
int timeout[maxn+7] = {};
int curr_time = 0;

void dfs(int a, int parent){
    timein[a] = curr_time++;

    for(int b:adj[a]){
        if (b == parent) continue;
        dfs(b, a);
    }

    timeout[a] = curr_time++;
}

// Check if node a is ancestor of node b
bool is_ancestor(int a, int b){
    if (timein[b] > timein[a] && timeout[b] < timeout[a]) return true;
    return false;
}

int main(){
    // Sample tree
    //
    //          1
    //        /   \
    //       2     3
    //      / \   / \
    //     4   5 6   7
    adj[1].push_back(2);
    adj[1].push_back(3);
    adj[2].push_back(4);
    adj[2].push_back(5);
    adj[3].push_back(6);
    adj[3].push_back(7);

    dfs(1, 0);

    for(int i = 1;i <= 7;i++){
        for(int j = i+1;j <= 7;j++){
            if (i == j) continue;

            cout << i << " is ancestor of " << j << ": " << (
                is_ancestor(i, j)?"YES":"NO") << endl;
        }
    }

    return 0;
}
```

4 math

4.1 primitive root

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

using namespace std;

#define ll long long

int powmod(int x, int n, int MOD){
    if (n == 0) return 1;
    if (n%2 == 0){
        int y = powmod(x, n/2, MOD);
        return (y*(ll)y)%MOD;
    }
    return (powmod(x, n-1, MOD)*(ll)x)%MOD;
}

// p must be prime
int proot(int p){
    int phi = p-1;
    int n = phi;

    vector<int>fact;
    for(int i = 2;i*i <= n;i++){
        if (n%i == 0){
            fact.push_back(i);
            while(n%i == 0) n /= i;
        }
    }
    if (n > 1) fact.push_back(n);

    for(int a = 2;a <= p;a++){
        bool ok = true;
        for(int i = 0;i < fact.size() && ok;i++){
            ok &= powmod(a, phi/fact[i], p) != 1;
        }
        if (ok) return a;
    }

    return -1;
}

int main(){

    // 90441961
    int x;cin>>x;
    cout << proot(x) << endl;

    return 0;
}
```

4.2 ext gcd

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

using namespace std;

int ext_gcd(int a, int b, int &x, int &y){
    if (a == 0){
        x = 0;
        y = 1;
        return b;
    }
    int x1, y1;
    int gcd = ext_gcd(b%a, a, x1, y1);
    x = y1 - (b/a) * x1;
    y = x1;
    return gcd;
}
```

```
int main(){
    int a = 2011;
    int b = 4201;
    int i, j;
    ext_gcd(a, b, i, j);
    cout << "("<<i<<")*" << a << " + ("<<j<<")*" << b << " = " << __gcd
        (a, b) << endl;
    return 0;
}
```

4.3 pollard rho 128bit

```
#include <bits/stdc++.h>
using namespace std;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
namespace u128{
    typedef __uint128_t type;
    type mul(type a, type b, type MOD){
        if (a == 0) return 0;
        if (a%2 == 0) return (mul(a/2, b, MOD)*2)%MOD;
        return (mul(a-1, b, MOD)+b)%MOD;
    }
    type pow(type x, type n, type MOD){
        if (n == 0) return 1;
        if (n%2 == 0){
            type y = pow(x, n/2, MOD);
            return mul(y, y, MOD);
        }
        return mul(pow(x, n-1, MOD), x, MOD);
    }
    type abs(type x){
        return (x>0)?x:-x;
    }
    string u128tos(type x){
        if (x == 0) return "0";
        string s;
        while(x){
            s += char('0'+(x%10));
            x /= 10;
        }
        reverse(s.begin(), s.end());
        return s;
    }
    type stou128(string s){
        type x = 0;
        type b = 1;
        int i = (int)s.size();
        while(i--){
            x += (s[i]-'0')*b;
            b *= 10;
        }
        return x;
    }
}

u128::type primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41,
    43, 47};
bool rabin_miller(u128::type x){
    if (x <= 1) return false;
    for(u128::type p:primes) if (x == p) return true;
    for(u128::type p:primes) if (x % p == 0) return false;
    u128::type k = 1;
    while(!(((x-1)>>k)&1)) k++;
    u128::type q = (x-1)>>k;
    for(u128::type p:primes){
        u128::type a = u128::pow(p, q, x);
        if (a == 1 || a == x-1) continue;
        for(u128::type j = 1;j <= k;j++){
            a = u128::mul(a, a, x);
            if (a == x-1) break;
            if (a == 1) return false;
            if (j == k && a != 1) return false;
        }
    }
    return true;
}
u128::type f(u128::type x, u128::type c, u128::type n){
    return (u128::mul(x, x, n)+c)%n;
}
u128::type pollard_rho(u128::type n, u128::type x0, u128::type c){
    u128::type g = 1;
    u128::type x = x0;
    u128::type y = x0;
    while(g == 1){
        x = f(x, c, n);
        y = f(f(y, c, n), c, n);
        g = __gcd(u128::abs(x-y), n);
    }
    if (g == n) return 0;
    return g;
}
u128::type factor(u128::type x){
    if (x%2 == 0) return 2;
    if (x%3 == 0) return 3;
    int i = 1;
    u128::type f = 0;
    while(f == 0){
        long long x0 = uniform_int_distribution<long long>(0, (1LL<<60)
            )(rng);
        f = pollard_rho(x, x0, i++);
    }
    return f;
}
void factorization(vector<pair<u128::type, int>> &factors, u128::type x
    ){
    if (x == 1) return;
    if (rabin_miller(x)) {
        for(auto &p:factors){
            if (p.first == x) return void(p.second++);
        }
        factors.push_back({x, 1});
    }
    else{
        u128::type f = factor(x);
        factorization(factors, f);
        factorization(factors, x/f);
    }
}

int main(){

    string s;cin>>s;
    u128::type x = u128::stou128(s);

    vector<pair<u128::type, int>>factors;
    factorization(factors, x);

    for(auto f:factors) cout << u128::u128tos(f.first) << "^" << f.
        second << " ";
    cout << endl;

    return 0;
}
```

4.4 rabin miller

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

using namespace std;

#define ll long long

ll primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};

ll mul(ll a, ll b, ll MOD){
    if (a == 0) return 0;
    if (a%2 == 0) return (mul(a/2, b, MOD)*2)%MOD;
    else return (mul(a-1, b, MOD)+b)%MOD;
}

ll fast_exp(ll x, ll n, ll m){
    if (n == 0) return 1;
    if (n%2 == 0){
        ll y = fast_exp(x, n/2, m);
        return mul(y, y, m);
    }
    return mul(fast_exp(x, n-1, m), x, m);
}

bool rabin_miller(ll x){
    if (x <= 1) return false;
    for(ll p:primes) if (x == p) return true;
    for(ll p:primes) if (x % p == 0) return false;

    ll k = __builtin_ctzll(x-1);
    ll q = (x-1)>>k;

    for(ll p:primes){
        ll a = fast_exp(p, q, x);
        if (a == 1 || a == x-1) continue;

        for(int j = 1;j <= k;j++){
            a = mul(a, a, x);
            if (a == x-1) break;
            if (a == 1) return false;
            if (j == k && a != 1) return false;
        }
    }

    return true;
}

int main(){

    cout << rabin_miller(1000000007) << endl;

    return 0;
}
```

4.5 gaussian elimination

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

using namespace std;
const double eps = 1e-9;

// 0 - no solution
// 1 - one solution
// 2 - inf solutions
int gauss(vector<vector<double>> &a, vector<double> &ans){
    int n = a.size();
    int m = a[0].size()-1;

    vector<int>where(m, -1);

    int row = 0;
    for(int col = 0;col < m && row < n;col++){
        int sel = row;
        for(int i = row;i < n;i++){
            if (abs(a[i][col]) > abs(a[sel][col])) sel = i;
        }

        // x_col is an independent variable
        if (abs(a[sel][col]) < eps) continue;

        where[col] = row;
        for(int j = col;j <= m;j++) swap(a[sel][j], a[row][j]);
        for(int i = 0;i < n;i++){
            if (i == row) continue;
            double c = a[i][col] / a[row][col];
            for(int j = col;j <= m;j++){
                a[i][j] -= a[row][j] * c;
            }
        }

        row++;
    }

    ans.assign(m, 0);
    for(int col = 0;col < m;col++){
        if (where[col] == -1) continue;
        ans[col] = a[where[col]][m] / a[where[col]][col];
    }

    for(int row = 0;row < n;row++){
        double sum = 0;
        for(int col = 0;col < m;col++){
            if (where[col] == -1) continue;
            sum += a[row][col] * ans[col];
        }
        if (abs(sum - a[row][m]) > eps) return 0;
    }

    for(int col = 0;col < m;col++){
        if (where[col] == -1) return 2;
    }
    return 1;
}
int main(){

    /*
    SLAE
    1x + 1y + 1z = 3
    1x - 2y + 1z = 0
    2x + 2y - 1z = 5

    Augmented matrix
    1  1  1  3
    1 -2  1  0
    2  2 -1  5
    */

    vector<vector<double>>a = {
        {1, 1, 1, 3},
        {1, -2, 1, 0},
        {2, 2, -1, 5}
    };
    vector<double>ans;
    gauss(a, ans);

    for(int i = 0;i < a.size();i++){
        for(int j = 0;j < a[0].size();j++) cout << a[i][j] << " " ;
        cout << endl;
    }

    cout << endl;
    cout << "x = " << ans[0] << endl;
    cout << "y = " << ans[1] << endl;
    cout << "z = " << ans[2] << endl;

    return 0;
}
```

4.6 lpf

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

using namespace std;

/*
    Build: O(nlogn)
    Factor: O(logn)
*/

const int maxn = 1e6;

int lpf[maxn+123] = {};

void build(){
    for(int i = 1;i <= maxn;i++) lpf[i] = i;
    for(int i = 2;i <= maxn;i++){
        if (lpf[i] == i){
            for(int j = 2;i*j <= maxn;j++){
                int x = i*j;
                if (lpf[x] == x) lpf[x] = i;
            }
        }
    }
}

vector<int> factor(int x){
    vector<int>primes;
    while(x != 1){
        primes.push_back(lpf[x]);
        x /= lpf[x];
    }
    return primes;
}

int main(){

    build();

    int x = 123456;
    cout << "Factors of " << x << ": " << endl;
    for(auto p:factor(x)) cout << p << " ";
    cout << endl;

    return 0;
}
```

4.7 matrix expo

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

using namespace std;

#define ll long long
const ll mod = 1e9+7;

vector<vector<ll>> multiply(const vector<vector<ll>>&a, const vector<
    vector<ll>>&b){
    vector<vector<ll>>res((int)a.size(), vector<ll>((int)a.size()));
    for(int i = 0;i < (int)a.size();i++){
        for(int j = 0;j < (int)a.size();j++){
            for(int k = 0;k < (int)a.size();k++){
                res[i][j] += a[i][k]*b[k][j];
                res[i][j] %= mod;
            }
        }
    }
    return res;
}

vector<vector<ll>> power(vector<vector<ll>>a, ll n){
    vector<vector<ll>>res((int)a.size(), vector<ll>((int)a.size()));
    for(int i = 0;i < (int)a.size();i++) res[i][i] = 1;

    while(n > 0){
        if (n&1) res = multiply(res, a);
        a = multiply(a, a);
        n /= 2;
    }

    return res;
}

int main(){

    vector<vector<ll>>r = {
        {10, 1},
        {1, 5}
    };
    r = power(r, 2);

    for(int i = 0;i < (int)r.size();i++){
        for(int j = 0;j < (int)r.size();j++){
            cout << r[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

4.8 linear div cnt

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

using namespace std;

/*
    Build: O(n)
*/

const int maxn = 1e7;

int dcnt[maxn+123];
pair<int, int> lp[maxn+123];

void sieve(){
    for(int i = 1;i <= maxn;i++) {
        dcnt[i] = 2;
        lp[i] = {i, 1};
    }
    dcnt[1] = 1;

    vector<int>primes;
    for(int i = 2;i <= maxn;i++){
        if (dcnt[i] == 2) primes.push_back(i);
        for(auto p:primes){
            if (i*p > maxn) break;
            if (i%p == 0) {
                lp[i*p] = lp[i];
                lp[i*p].first *= p;
                lp[i*p].second++;
                dcnt[i*p] = dcnt[i/lp[i].first]*(lp[i*p].second+1);
            }
            else {
                dcnt[i*p] = dcnt[i]*dcnt[p];
                lp[i*p] = {p, 1};
            }
        }
    }
}

int main(){

    sieve();

    cout << dcnt[21613] << endl;

    return 0;
}
```

4.9 linear sieve

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

using namespace std;

#define MAXN (int)1e6

vector<int>primes;
bool is_composite[MAXN+1] = {};

void sieve(){
    for(int i = 2;i <= MAXN;i++){
        if (!is_composite[i]) primes.push_back(i);
        for(auto p:primes){
            if (p*i > MAXN) break;
            is_composite[p*i] = true;
            if (i % p == 0) break;
        }
    }
}

int main(){

    sieve();

    return 0;
}
```

4.10 discrete log

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

using namespace std;

#define ll long long

int dlog(int a, int b, int m){
    int n = (int)sqrt(m)+1;

    int an = 1;
    for(int i = 0;i < n;i++){
        an = (an*(ll)a)%m;
    }

    unordered_map<int, int>values;
    for(int p = 1, cur = an;p <= n;p++){
        if (cur == 0) break;
        if (values.count(cur) == 0) values[cur] = p;
        cur = (cur*(ll)an)%m;
    }

    int c = b;
    for(int q = 0;q <= n;q++){
        if (values.count(c)){
            int x = values[c]*n-q;
            return x;
        }
        c = (c*(ll)a)%m;
    }

    return -1;
}

int main(){

    cout << dlog(3, 4, 11) << endl;

    return 0;
}
```

4.11 pollard rho

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

using namespace std;

#define ll long long

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

ll mul(ll a, ll b, ll MOD){
    if (a == 0) return 0;
    if (a%2 == 0) return (mul(a/2, b, MOD)*2)%MOD;
    return (mul(a-1, b, MOD)+b)%MOD;
}

ll pow(ll x, ll n, ll MOD){
    if (n == 0) return 1;
    if (n%2 == 0){
        ll y = pow(x, n/2, MOD);
        return mul(y, y, MOD);
    }
    return mul(pow(x, n-1, MOD), x, MOD);
}

ll primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};

bool rabin_miller(ll x){
    if (x <= 1) return false;
    for(ll p:primes) if (x == p) return true;
    for(ll p:primes) if (x % p == 0) return false;

    ll k = 1;
    while(!(((x-1)>>k)&1)) k++;
    ll q = (x-1)>>k;

    for(ll p:primes){
        ll a = pow(p, q, x);
        if (a == 1 || a == x-1) continue;

        for(ll j = 1;j <= k;j++){
            a = mul(a, a, x);
            if (a == x-1) break;
            if (a == 1) return false;
            if (j == k && a != 1) return false;
        }
    }

    return true;
}

ll f(ll x, ll c, ll n){
    return (mul(x, x, n)+c)%n;
}

ll pollard_rho(ll n, ll x0, ll c){
    ll g = 1;
    ll x = x0;
    ll y = x0;

    while(g == 1){
        x = f(x, c, n);
        y = f(f(y, c, n), c, n);
        g = __gcd(abs(x-y), n);
    }
    if (g == n) return 0;
    return g;
}

ll factor(ll x){
    if (x%2 == 0) return 2;
    if (x%3 == 0) return 3;

    int i = 1;
    ll f = 0;
    while(f == 0){
        ll x0 = uniform_int_distribution<ll>(0, (1LL<<60))(rng);
        f = pollard_rho(x, x0, i++);
    }

    return f;
}

void factorization(vector<pair<ll, int>> &factors, ll x){
    if (x == 1) return;
    if (rabin_miller(x)) {
        for(auto &p:factors){
            if (p.first == x) return void(p.second++);
        }
        factors.push_back({x, 1});
    }
    else{
        ll f = factor(x);
        factorization(factors, f);
        factorization(factors, x/f);
    }
}

int main(){

    ll x;cin>>x;

    vector<pair<ll, int>>factors;
    factorization(factors, x);

    for(auto f:factors) cout << f.first << "^" << f.second << " ";
    cout << endl;

    return 0;
}
```

4.12 segmented sieve

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

using namespace std;

#define ll long long

vector<ll> sieve(int n){
    vector<bool>is_composite(n+1, false);
    vector<ll>primes;

    for(int i = 2;i <= n;i++){
        if (!is_composite[i]) primes.push_back(i);

        for(auto p:primes){
            if (i*p > n) break;
            is_composite[i*p] = true;
```

```
                if (i % p == 0) break;
            }
        }

        return primes;
    }

    vector<bool> ranged_sieve(ll l, ll r){
        vector<ll>primes = sieve((int)sqrt(r)+1);
        vector<bool>mark(r-l+1, true);

        for(auto p:primes){
            ll a = (l/p)*p;
            if (a < l || a == 0) a += p;
            if (a == p) a += p;
            for(;a <= r;a += p) mark[a-l] = false;
        }

        return mark;
    }

    int main(){

        ll l = 0;
        ll r = 100;

        cout << "Primes between " << l << " and " << r << endl;
        vector<bool>mark = ranged_sieve(l, r);
        for(int i = 0;i < (int)mark.size();i++){
            if (l+i <= 1) continue;
            if (mark[i]) cout << l+i << endl;
        }

        return 0;
    }
```

4.13 gaussian elimination xor

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

using namespace std;
const int maxn = 4;

// 0 - no solution
// 1 - one solution
// 2 - inf solutions
int gauss(vector<bitset<maxn>> &a){
    int n = a.size();
    int m = maxn-1;

    vector<int>where(m, -1);

    int row = 0;
    for(int col = 0;col < m && row < n;col++){
        for(int i = row;i < n;i++){
            if (a[i][col]){
                swap(a[i], a[row]);
                break;
            }
        }

        // x_col is an independent variable
        if (a[row][col] == 0) continue;

        where[col] = row;
        for(int i = 0;i < n;i++){
            if (i != row && a[i][col]) a[i] ^= a[row];
        }
        row++;
    }

    for(int row = 0;row < n;row++){
        bool value = false;
        for(int col = 0;col < m;col++){
            if (where[col] == -1) continue;
            value ^= a[where[col]][m];
        }
        if (a[row][m] != value) return 0;
    }

    for(int col = 0;col < m;col++){
        if (where[col] == -1) return 2;
    }
    return 1;
}

int main(){
    /*
    SLAE (mod 2)
    1x ^ 1y ^ 1z = 1
    1x ^ 0y ^ 1z = 0
    0x ^ 0y ^ 1z = 1

    Augmented matrix
    1 1 1 1      rev      1 1 1 1
    1 0 1 0    ----->     0 1 0 1
    0 0 1 1               1 1 0 0
    */

    vector<bitset<maxn>>a = {
        bitset<maxn>("1111"),
        bitset<maxn>("0101"),
        bitset<maxn>("1100")
    };
    int sols = gauss(a);

    cout << "solutions: " << sols << endl;
    for(int i = 0;i < a.size();i++){
        for(int j = 0;j < a[0].size();j++) cout << a[i][j] << " " ;
        cout << endl;
    }

    // cout << endl;
    // cout << "x = " << ans[0] << endl;
    // cout << "y = " << ans[1] << endl;
    // cout << "z = " << ans[2] << endl;

    return 0;
}
```

4.14 ternary search

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

using namespace std;

int f(int x){
    return (x-5)*(x-5);
}

// Discrete algorithm
```

```
int ternary_search(int l, int r){
    while(r-l >= 3){
        int m1 = l+(r-l)/3;
        int m2 = r-(r-l)/3;

        if (f(m1) > f(m2)) l = m1;
        else r = m2;
    }

    int res = 0x7fffffff;
    for(;l <= r;l++) res = min(res, f(l));
    return res;
}

double g(double x){
    return x*exp(x);
}

// Real algorithm
double ternary_search(double l, double r){
    const double eps = 1e-9;

    while(r-l > eps){
        double m1 = l+(r-l)/3;
        double m2 = r-(r-l)/3;

        if (g(m1) > g(m2)) l = m1;
        else r = m2;
    }

    return g(l);
}

int main(){

    cout << "Discrete function f(x) = (x-5)^2" << endl;
    cout << "Minimum: " << ternary_search(-20, 20) << endl << endl;

    cout << "Real function g(x) = x*e^x" << endl;
    cout << "Minimum: " << ternary_search(-1e3, 1e3) << endl;

    return 0;
}
```

4.15 modmul 64bit

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

using namespace std;

#define ll long long

const uint64_t mod = 2305843009213693951;
uint64_t modmul(uint64_t a, uint64_t b){
        uint64_t l1 = (uint32_t)a, h1 = a>>32, l2 = (uint32_t)b, h2 = b
            >>32;
        uint64_t l = l1*l2, m = l1*h2 + l2*h1, h = h1*h2;
        uint64_t ret = (l&mod) + (l>>61) + (h << 3) + (m >> 29) + (m <<
             35 >> 3) + 1;
        ret = (ret & mod) + (ret>>61);
        ret = (ret & mod) + (ret>>61);
        return ret-1;
}

int main(){

    long long a = 125462315651256000;
    long long b = 665232002331569800;

    cout << modmul(a, b) << endl;

    return 0;
}
```

4.16 linear div sum

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

using namespace std;

/*
    Build: O(n)
*/

#define ll long long
const int maxn = 1e7;

ll dsum[maxn+123];
ll lp[maxn+123];

void sieve(){
    for(int i = 1;i <= maxn;i++) {
        dsum[i] = i+1;
        lp[i] = i;
    }
    dsum[1] = 1;

    vector<ll>primes;
    for(int i = 2;i <= maxn;i++){
        if (dsum[i] == i+1) primes.push_back(i);
        for(ll p:primes){
            if (i*p > maxn) break;
            if (i % p == 0){
                ll n = i/lp[i];
                lp[i*p] = lp[i]*p;

                // use long long to avoid overflow here
                dsum[i*p] = dsum[n]*(lp[i*p]*p-1)/(p-1);
                break;
            }
            else {
                dsum[i*p] = dsum[i]*dsum[p];
                lp[i*p] = p;
            }
        }
    }
}

int main(){

    sieve();

    cout <<  dsum[21613] << endl;

    return 0;
}
```

4.17 linear phi

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

```
using namespace std;

#define N (int)1e6

int phi[N+1];

void sieve(){
    for(int i = 1;i <= N;i++) phi[i] = i-1;
    phi[1] = 1;

    vector<int>primes;
    for(int i = 2;i <= N;i++){
        if (phi[i] == i-1) primes.push_back(i);
        for(auto p:primes){
            if (i*p > N) break;
            if (i%p == 0){
                phi[i*p] = phi[i]*p;
                break;
            }
            phi[i*p] = phi[i]*phi[p];
        }
    }
}

int main(){

    sieve();

    int x;cin>>x;
    cout << phi[x] << endl;

    return 0;
}
```

4.18 nCk lucas

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

using namespace std;
#define ll long long

const ll mod = 1000003;
ll factorial[mod+13] = {};

vector<ll>expand(ll x, ll base){
    vector<ll>res;

    while(x){
        res.push_back(x%base);
        x /= base;
    }

    return res;
}

ll modpow(ll x, ll n){
    if (n == 0) return 1;
    ll y = modpow(x, n/2);
    if (n&1) return y*y % mod * x % mod;
    return y*y % mod;
}

ll inverse(ll x){
    return modpow(x, mod-2);
}

ll nCk(ll n, ll k){
    vector<ll>nmod = expand(n, mod);
    vector<ll>kmod = expand(k, mod);

    int size = max(nmod.size(), kmod.size());
    nmod.resize(size);
    kmod.resize(size);

    ll res = 1;
    for(int i = 0;i < size;i++){
        if (nmod[i] < kmod[i]) {
            res = 0;
            break;
        }

        res *= factorial[nmod[i]];
        res %= mod;
        res *= inverse(factorial[kmod[i]]);
        res %= mod;
        res *= inverse(factorial[nmod[i]-kmod[i]]);
        res %= mod;
    }

    return res;
}

void build(){
    factorial[0] = 1;
    for(int i = 1;i <= mod;i++) factorial[i] = factorial[i-1]*i % mod;
}

int main(){

    // O(MOD)
    build();

    // O(max(logMOD(N), logMOD(K)) * log2(MOD))
    cout << nCk(1234567895121314313, 92345612131579713) << endl;

    return 0;
}
```

4.19 phi

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

using namespace std;

int phi(int n){
    int res = n;
    for(int i = 2;i*i <= n;i++){
        if (n%i == 0){
            while(n%i == 0) n /= i;
            res -= res/i;
        }
    }
    if (n > 1) res -= res/n;
    return res;
}

int main(){

    int x;cin>>x;
    cout << phi(x) << endl;

    return 0;
}
```

4.20 discrete root

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

using namespace std;

#define ll long long

int powmod(int x, int n, int MOD){
    if (n == 0) return 1;
    if (n%2 == 0){
        int y = powmod(x, n/2, MOD);
        return (y*(ll)y)%MOD;
    }
    return (powmod(x, n-1, MOD)*(ll)x)%MOD;
}

int proot(int p){
    int phi = p-1;
    int n = phi;

    vector<int>fact;
    for(int i = 2;i*i <= n;i++){
        if (n%i == 0){
            fact.push_back(i);
            while(n%i == 0) n /= i;
        }
    }
    if (n > 1) fact.push_back(n);

    for(int a = 2;a <= p;a++){
        bool ok = true;
        for(int i = 0;i < fact.size() && ok;i++){
            ok &= powmod(a, phi/fact[i], p) != 1;
        }
        if (ok) return a;
    }

    return -1;
}

int dlog(int a, int b, int m){
    int n = (int)sqrt(m)+1;

    int an = 1;
    for(int i = 0;i < n;i++){
        an = (an*(ll)a)%m;
    }

    unordered_map<int, int>values;
    for(int p = 1, cur = an;p <= n;p++){
        if (cur == 0) break;
        if (values.count(cur) == 0) values[cur] = p;
        cur = (cur*(ll)an)%m;
    }

    int c = b;
    for(int q = 0;q <= n;q++){
        if (values.count(c)){
            int x = values[c]*n-q;
            return x;
        }
        c = (c*(ll)a)%m;
    }

    return -1;
}
int main(){

    // Find x such that x^k = a (mod n)
    // n must be prime
    int k, a, n;cin>>k>>a>>n;

    // Primitive root of n
    int g = proot(n);

    // Finding one solution with discrete log
    int y = dlog(powmod(g, k, n), a, n);
    if (y == -1) return ((cout << "No solution" << endl), 0);

    // Finding all solutions
    vector<int>solutions;
    int d = (n-1)/__gcd(n-1, k);

    for(int cur = y % d;cur < n-1;cur += d){
        solutions.push_back(powmod(g, cur, n));
    }
    sort(solutions.begin(), solutions.end());

    cout << "List of x such that x^" << k << " = " << a << " (mod "<< n
        << ")" << endl;
    for(auto s:solutions) cout << s << endl;

    return 0;
}
```

5 dp

5.1 lcs

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

using namespace std;

const int maxn = 1e3;

int main(){

    string s1 = "abacaba";
    string s2 = "abacate";
    int n = (int)s1.size();
    int m = (int)s2.size();

    int dp[maxn][maxn] = {};
    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= m;j++){
            if (s1[i-1] == s2[j-1]) dp[i][j] = dp[i-1][j-1]+1;
            else dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
        }
    }

    cout << dp[n][m] << endl;

    return 0;
}
```

5.2 lis nlog2

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

using namespace std;

int main(){

    vector<int>v(1e5);
    generate(v.begin(), v.end(), rand);

    vector<int>dp;
    for(int x:v){
        auto it = lower_bound(dp.begin(), dp.end(), x);
        if (it == dp.end()) dp.push_back(x);
        else *it = x;
    }

    cout << dp.size() << endl;

    return 0;
}
```

5.3 lps

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

using namespace std;

const int maxn = 1e3;

int main(){

    string s = "iabacaxi";
    int n = (int)s.size();

    int dp[maxn][maxn] = {};
    for(int i = 0;i < n;i++) dp[i][i] = 1;

    for(int i = n-1;i >= 0;i--){
        for(int j = i+1;j < n;j++){
            if (s[i] == s[j]) dp[i][j] = 2+dp[i+1][j-1];
            else dp[i][j] = max(dp[i][j-1], dp[i+1][j]);
        }
    }

    int l = 0;
    int r = n-1;
    cout << "Length of the largest palindrome from " << l << " to " <<
        r << endl;
    cout << dp[l][r] << endl;

    return 0;
}
```

5.4 lis

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

using namespace std;

const int maxn = 1e3;

int main(){

    vector<int>vec = {10, 22, 9, 33, 21, 50, 41, 60};
    int n = (int)vec.size();

    int dp[maxn] = {};

    for(int i = 0;i < n;i++){
        dp[i] = 1;
        for(int j = 0;j < i;j++){
            if (vec[i] > vec[j]) dp[i] = max(dp[i], 1+dp[j]);
        }
    }

    for(int i = 0;i < n;i++) cout << dp[i] << " ";

    return 0;
}
```

5.5 lis nlog

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

using namespace std;

const int maxn = 1e5;
int bit[maxn+7] = {};

void update(int i, int val){
    while(i <= maxn){
        bit[i] = max(bit[i], val);
        i += i&(-i);
    }
}

int query(int i){
    int res = 0;
    while(i > 0){
        res = max(res, bit[i]);
        i -= i&(-i);
    }
    return res;
}

int main(){

    vector<int>v(maxn);
    generate(v.begin(), v.end(), rand);

    map<int, int>f;
    int curr = 1;
    vector<int>vsorted = v;
    sort(vsorted.begin(), vsorted.end());
    for(int x:vsorted){
        if (f[x] == 0) f[x] = curr++;
    }

    for(int x:v) update(f[x], 1+query(f[x]-1));

    cout << "Length of the longest increasing subsequence" << endl;
    cout << query(curr) << endl;

    return 0;
}
```

5.6 tsp

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

using namespace std;

const int inf = 1<<29;
const int maxn = 20;

int n;
int dis[maxn+3][maxn+3] = {};
int dp[(1<<maxn)+3][maxn+11] = {};

int solve(int s, int last){
    if (dp[s][last] == 0){
        dp[s][last] = inf;

        if (__builtin_popcount(s) == 1) dp[s][last] = dis[0][last];
        else{
            for(int i = 1;i < n;i++){
                if (i == last) continue;
                if (((1<<i)&s) == 0) continue;
                dp[s][last] = min(dp[s][last], solve(s^(1<<last), i)+
                    dis[i][last]);
            }
        }
    }
    return dp[s][last];
}

int main(){

    cin>>n;
    for(int i = 0;i < n;i++){
        for(int j = 0;j < n;j++){
            cin>>dis[i][j];
        }
    }

    cout << solve((1<<n)-1, 0) << endl;

    return 0;
}

/* sample input
    4
    0 20 35 42
    20 0 34 30
    35 34 0 12
    42 30 12 0
*/
```

6 string

6.1 lcp

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

using namespace std;

/*
    Build: O(nlogn)
    Query: O(1)
*/

#define lg2(x) 31-__builtin_clz(x)
```

```
const int maxn = 1e5;
const int logmaxn = lg2(maxn);

int len;
int pos[maxn+123];
int st[maxn+7][logmaxn+3] = {};

void build_sparse_table(const vector<int>&v){
    int n = (int)v.size();

    for(int i = 0;i < n;i++) st[i][0] = v[i];

    for(int j = 1;j <= logmaxn;j++){
        for(int i = 0;i+(1<<j) <= maxn+1;i++){
            st[i][j] = min(st[i][j-1], st[i+(1<<(j-1))][j-1]);
        }
    }
}

int query(int l, int r){
    if (l == r) return len-l;
    l = pos[l];
    r = pos[r];
    if (l > r) swap(l, r);
    r--;

    int j = lg2(r-l+1);
    return min(st[l][j], st[r-(1<<j)+1][j]);
}

vector<int> sort_cyclic(const string &s){
    int n = (int)s.size();
    int alpha = 256;

    vector<int>p(n), c(n), cnt(max(alpha, n));

    for(int i = 0;i < n;i++) cnt[s[i]]++;
    for(int i = 1;i < alpha;i++) cnt[i] += cnt[i-1];
    for(int i = n-1;i >= 0;i--) p[--cnt[s[i]]] = i;

    int classes = 1;
    c[p[0]] = 0;
    for(int i = 1;i < n;i++){
        if (s[p[i]] != s[p[i-1]]) classes++;
        c[p[i]] = classes-1;
    }

    vector<int>pn(n), cn(n);
    for(int k = 0;(1<<k) < n;k++){
        fill(cnt.begin(), cnt.begin()+classes, 0);

        for(int i = 0;i < n;i++) pn[i] = (p[i]-(1<<k)+n)%n;
        for(int i = 0;i < n;i++) cnt[c[pn[i]]]++;
        for(int i = 1;i < classes;i++) cnt[i] += cnt[i-1];
        for(int i = n-1;i >= 0;i--) p[--cnt[c[pn[i]]]] = pn[i];

        classes = 1;
        cn[p[0]] = 0;
        for(int i = 1;i < n;i++){
            pair<int, int> a = {c[p[i]], c[(p[i]+(1<<k))%n]};
            pair<int, int> b = {c[p[i-1]], c[(p[i-1]+(1<<k))%n]};
            if (a != b) classes++;
            cn[p[i]] = classes-1;
        }
        swap(c, cn);
    }

    return p;
}

vector<int> suffix_array(string s){
    s += '$';
    vector<int> sorted = sort_cyclic(s);
    sorted.erase(sorted.begin());
    return sorted;
}

vector<int> lcp_array(const string &s, const vector<int> &sa){
    int n = (int)s.size();

    vector<int> pi(n);
    for(int i = 0;i < n;i++) pi[sa[i]] = i;

    vector<int>lcp(n);
    int k = 0;
    for(int i = 0;i < n;i++){
        if (pi[i]+1 == n){
            k = 0;
            continue;
        }

        int j = sa[pi[i]+1];
        while(max(i+k, j+k) < n && s[i+k] == s[j+k]) k++;

        lcp[pi[i]] = k;
        if (k > 0) k--;
    }

    return lcp;
}

void build(string s){
    len = s.size();
    vector<int>sa = suffix_array(s);
    vector<int>lcp = lcp_array(s, sa);

    for(int i = 0;i < s.size();i++) pos[sa[i]] = i;

    build_sparse_table(lcp);
}

int main(){

    string s = "abab";
    build(s);

    cout << s << endl;
    for(int i = 0;i < s.size();i++){
        for(int j = i;j < s.size();j++){
            cout << "lcp(" << s.substr(i, 10) << ", " << s.substr(j,
                10) << ") = " << query(i, j) << endl;
        }
    }

    return 0;
}
```

6.2 prefix function

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

using namespace std;

/*
    Build: O(n)
*/

vector<int> prefix(const string &s){
    int n = (int)s.size();
    vector<int>p(n);

    for(int i = 1;i < n;i++){
        p[i] = p[i-1];
        while(p[i] > 0 && s[p[i]] != s[i]) p[i] = p[p[i]-1];
        if (s[p[i]] == s[i]) p[i]++;
    }

    return p;
}

int main(){

    vector<int>p = prefix("aabxaabxay");

    for(int i = 0;i < p.size();i++) cout << p[i] << " ";
    cout << endl;

    return 0;
}
```

6.3 suffix automaton

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

using namespace std;

/*
    With map<char, int> next
        Build: O(nlogk)
        Memory: O(n)

    With int next[k]
        Build: O(n)
        Memory: O(nk)
*/

const int maxn = 1e5+123;

struct state{
    int link, len;
    map<char, int>next;
};

state st[2*maxn];
int aut_sz, last;

void init(){
    aut_sz = 1;
    last = 0;
    st[0].link = -1;
    st[0].len = 0;
}

void append(char c){
    int curr = aut_sz++;
    st[curr].len = st[last].len+1;

    int p = last;
    while(p != -1 && !st[p].next.count(c)){
        st[p].next[c] = curr;
        p = st[p].link;
    }
```

```
        if (p == -1) {
            st[curr].link = 0;
        }
        else{
            int q = st[p].next[c];
            if (st[q].len == st[p].len+1){
                st[curr].link = q;
            }
            else{
                int clone = aut_sz++;
                st[clone].len = st[p].len+1;
                st[clone].link = st[q].link;
                st[clone].next = st[q].next;
                while(p != -1 && st[p].next[c] == q){
                    st[p].next[c] = clone;
                    p = st[p].link;
                }
                st[q].link = clone;
                st[curr].link = clone;
            }
        }
        last = curr;
    }

    void build(const string &s){
        init();
        for(int i = 0;i < s.size();i++) append(s[i]);
    }

    int main(){

        build("abb");

        for(int i = 0;i < aut_sz;i++){
            for(auto &pr:st[i].next) cout << i << " --" << pr.first << "-->
                 " << pr.second << endl;
        }

        return 0;
    }
```

6.4 trie

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

    using namespace std;

    struct node{
        node *child[30] = {};
        bool is_end = false;

        void add(const string &s){
            node *curr = this;

            for(int i = 0;i < (int)s.size();i++){
                int x = s[i]-'a';

                if (curr->child[x] == nullptr) curr->child[x] = new node();
                curr = curr->child[x];
            }
            curr->is_end = true;
        }

        void remove(const string &s){
            node *curr = this;

            for(int i = 0;i < (int)s.size();i++){
                int x = s[i]-'a';

                if (curr->child[x] == nullptr) return;
                curr = curr->child[x];
            }
            curr->is_end = false;
        }

        bool query(const string &s){
            node *curr = this;

            for(int i = 0;i < (int)s.size();i++){
                int x = s[i]-'a';

                if (curr->child[x] == nullptr) return false;
                curr = curr->child[x];
            }
            return curr->is_end;
        }
    };

    node *trie = new node();

    int main(){

        trie->add("aba");
        trie->add("abacaba");
        cout << trie->query("aba") << endl;
        cout << trie->query("abacaba") << endl;

        trie->remove("aba");
        cout << trie->query("aba") << endl;
        cout << trie->query("abacaba") << endl;

        return 0;
    }
```

6.5 duval

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

    using namespace std;

    /*
        Build: O(n)
    */

    vector<string> duval(string s){
        vector<string>res;

        int n = (int)s.size();
        int i = 0; // s1 => [0...i-1] processed
        int j = 0; // s2 => pre-simple string
        int k = 1; // s3 => [k...n] unprocessed
        while(i < n){
            while (k < n && s[j] <= s[k]){
                if (s[j] == s[k]) j++;
                else j = i; // new simple created
                k++;
            }

            // getting all simple string from s2
            while(i <= j){
                res.push_back(s.substr(i, k-j));
                i += k-j;
        }
        j = i;
        k = i+1;
    }
    return res;
}

int main(){
    string s = "aababbaaab";
    for(string &t:duval(s))
        cout << t << endl;
    return 0;
}
```

6.6 lp substring

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    string s = "kkabacabaxx";
    int n = (int)s.size();
    int best_len = 0;
    int best_l = 0;
    int best_r = 0;
    for(int i = 0;i < n;i++){
        // even length
        int l = i;
        int r = i+1;
        while(l >= 0 && r < n && s[l] == s[r]){
            l--;
            r++;
        }
        int len = r-l-2;
        if (len > best_len){
            best_len = len;
            best_l = l+1;
            best_r = r-1;
        }
        //odd length
        l = i-1;
        r = i+1;
        while(l >= 0 && r < n && s[l] == s[r]){
            l--;
            r++;
        }
        len = r-l-2;
        if (len > best_len){
            best_len = len;
            best_l = l+1;
            best_r = r-1;
        }
    }
    cout << s.substr(best_l, best_r-best_l+1) << endl;
    return 0;
}
```

6.7 kmp

```
#include <bits/stdc++.h>
using namespace std;
vector<int> prefix(const string &s){
    int n = (int)s.size();
    vector<int>p(n);
    for(int i = 1;i < n;i++){
        p[i] = p[i-1];
        while(p[i] > 0 && s[p[i]] != s[i]) p[i] = p[p[i]-1];
        if (s[p[i]] == s[i]) p[i]++;
    }
    return p;
}
vector<int> kmp(const string &s, const string &p, const vector<int>&pre
    ){
    int n = (int)s.size();
    int m = (int)p.size();
    vector<int>res;
    int j = 0;
    for(int i = 0;i < n;i++){
        while(j > 0 && p[j] != s[i]) j = pre[j-1];
        if (p[j] == s[i]) j++;
        if (j == m) res.push_back(i-m+1);
    }
    return res;
}
int main(){
    string s = "yxabacabaxy";
    string p = "aba";
    vector<int>pre = prefix(p);
    vector<int>k = kmp(s, p, pre);
    for(int x:k) cout << x << endl;
    return 0;
}
```

6.8 hashing update

```
#include <bits/stdc++.h>
using namespace std;
/*
    Build: O(nlogn)
    Substring hash: O(logn)
    Check palindrome: O(logn)
    Update: O(logn)
*/
#define ll long long
#define hash apdmfoiwahofqjenfoj
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
```

```
ll rnd(ll a, ll b){return uniform_int_distribution<ll>(a, b)(rng);}

const ll mod = 1e9+7;
const int maxn = 2e5+123;

ll modpow(ll x, ll n){
    if (n == 0) return 1;
    ll y = modpow(x, n/2);
    if (n&1) return y*y % mod * x % mod;
    return y*y % mod;
}

// 0-based
struct fenwick{
    ll bit[maxn+123];

    void update(int i, int x){
        for(i++;i < maxn;i += i&-i) (bit[i] += x) %= mod;
    }

    ll query(int i){
        ll res = 0;
        for(i++;i > 0;i -= i&-i) (res += bit[i]) %= mod;
        return res;
    }
};

struct hash{
    int n;
    string s;
    fenwick pref, suff;
    vector<ll>power, inverse;

    hash(const string &t, ll p){
        s = t;
        n = (int)s.size();

        power.assign(n, 0);
        inverse.assign(n, 0);

        p %= mod;
        ll ip = modpow(p, mod-2); // inverse of p
        power[0] = inverse[0] = 1;
        for(int i = 1;i < n;i++) power[i] = power[i-1]*p % mod;
        for(int i = 1;i < n;i++) inverse[i] = inverse[i-1]*ip % mod;
        for(int i = 0;i < n;i++) pref.update(i, s[i]*power[i] % mod);
        for(int i = 0;i < n;i++) suff.update(i, s[n-1-i]*power[i] % mod
            );
    }

    ll substr(int i, int j){
        if (i == 0) return pref.query(j);
        return (pref.query(j)-pref.query(i-1)+mod) % mod *inverse[i] %
            mod;
    }

    ll rsubstr(int i, int j){
        if (j == n-1) return suff.query(n-1-i);
        return (suff.query(n-1-i)-suff.query(n-1-(j+1))+mod) % mod *
            inverse[n-1-j] % mod;
    }

    bool is_palindrome(int i, int j){
        if (i == j) return true;
        int len = j-i+1;
        int k = len/2;
        return substr(i, i+k-1) == rsubstr(j-k+1, j);
    }

    void update(int i, char c){
        pref.update(i, -(s[i]*power[i] % mod)+mod);
        suff.update(n-1-i, -(s[i]*power[n-1-i] % mod)+mod);
        s[i] = c;
        pref.update(i, s[i]*power[i] % mod);
        suff.update(n-1-i, s[i]*power[n-1-i] % mod);
    }
};

int main(){

    string s = "zbaoabk";
    int n = (int)s.size();

    hash h(s, rnd(100, mod-2));

    h.update(0, 'x');
    h.update(3, 'a');
    h.update(n-1, 'x');
    cout << h.s << endl;

    cout << h.substr(0, n-1) << endl;
    cout << h.rsubstr(0, n-1) << endl;
    cout << h.is_palindrome(0, n-1) << endl;

    return 0;
}
```

6.9 hashing

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

using namespace std;

/*
    Build: O(n)
    Substring hash: O(1)
    Check palindrome: O(1)
*/

#define ll long long
#define hash apdmfoiwahofqjenfoj

mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());
ll rnd(ll a, ll b){return uniform_int_distribution<ll>(a, b)(rng);}

const ll mod = 1e9+7;

ll modpow(ll x, ll n){
    if (n == 0) return 1;
    ll y = modpow(x, n/2);
    if (n&1) return y*y % mod * x % mod;
    return y*y % mod;
}

struct hash{
    int n;
    vector<ll>prefix, suffix;
    vector<ll>power, inverse;

    hash(const string &s, ll p){
        n = (int)s.size();

        power.assign(n, 0);
        inverse.assign(n, 0);
        p %= mod;
        ll ip = modpow(p, mod-2); // inverse of p
        power[0] = inverse[0] = 1;
        for(int i = 1;i < n;i++) power[i] = power[i-1]*p % mod;
        for(int i = 1;i < n;i++) inverse[i] = inverse[i-1]*ip % mod;

        prefix.assign(n, 0);
        suffix.assign(n, 0);

        prefix[0] = s[0];
        suffix[n-1] = s[n-1];
        for(int i = 1;i < n;i++) prefix[i] = (prefix[i-1] + s[i]*power[
            i] % mod) % mod;
        for(int i = n-2;i >= 0;i--) suffix[i] = (suffix[i+1] + s[i]*
            power[n-1-i] % mod) % mod;
    }

    ll substr(int i, int j){
        if (i == 0) return prefix[j];
        return (prefix[j]-prefix[i-1]+mod) % mod *inverse[i] % mod;
    }

    ll rsubstr(int i, int j){
        if (j == n-1) return suffix[i];
        return (suffix[i]-suffix[j+1]+mod) % mod *inverse[n-1-j] % mod;
    }

    bool is_palindrome(int i, int j){
        if (i == j) return true;
        int len = j-i+1;
        int k = len/2;
        return substr(i, i+k-1) == rsubstr(j-k+1, j);
    }
};

int main(){

    string s = "abacaba";
    int n = (int)s.size();

    hash h(s, rnd(100, mod-2));

    cout << h.substr(0, n-1) << endl;
    cout << h.rsubstr(0, n-1) << endl;
    cout << h.is_palindrome(0, n-1) << endl;

    return 0;
}
```

6.10 prefix automaton

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

using namespace std;

vector<int> prefix(const string &s){
    int n = (int)s.size();
    vector<int>p(n);

    for(int i = 1;i < n;i++){
        p[i] = p[i-1];
        while(p[i] > 0 && s[p[i]] != s[i]) p[i] = p[p[i]-1];
        if (s[p[i]] == s[i]) p[i]++;
    }

    return p;
}

vector<vector<int>> automaton(string s){
    vector<vector<int>>res;

    s += '#';
    vector<int>pi = prefix(s);

    int n = (int)s.size();
    res.assign(n, vector<int>(26));
    for(int i = 0;i < n;i++){
        for(int j = 0;j < 26;j++){
            if (i > 0 && s[i] != 'a'+j) res[i][j] = res[pi[i-1]][j];
            else res[i][j] = i+('a'+j == s[i]);
        }
    }

    return res;
}

int main(){

    string p = "abacaba";
    vector<vector<int>>aut = automaton(p);

    string s = "ababxabacaba";
    int n = (int)s.size();
    int state = 0;
    for(int i = 0;i < n;i++){
        state = aut[state][s[i]-'a'];
        cout << state << " ";
    }
    cout << endl;

    return 0;
}
```

6.11 rabin karp

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

using namespace std;

#define ll long long

// s = text
// t = pattern
vector<int> rabin_karp(const string &s, const string &t){
    const int p = 53;
    const int mod = 1e9+7;
    int n = (int)s.size();
    int m = (int)t.size();

    vector<ll>power(max(n, m));
    ll pwr = 1;
    for(int i = 0;i < max(n, m);i++){
        power[i] = pwr;
        (pwr *= p) %= mod;
    }

    vector<ll>prefix(n+1, 0);
    for(int i = 1;i <= n;i++)
        prefix[i] = (prefix[i-1] + (s[i-1]*power[i-1]) % mod) % mod;

    ll hash_t = 0;
    for(int i = 0;i < m;i++)
```

```
            (hash_t += t[i]*power[i] % mod) %= mod;

        vector<int>res;
        for(int i = 0;i+m-1 < n;i++){
            ll a = ((prefix[i+m]-prefix[i]+mod)%mod);
            ll b = (hash_t*power[i]) % mod;

            if (a == b) res.push_back(i);
        }
        return res;
    }

    int main(){

        vector<int>occurences = rabin_karp("abababab", "ab");
        for(int i:occurences) cout << i << endl;

        return 0;
    }
```

6.12 z

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

using namespace std;

/*
    Build: O(n)
*/

vector<int>z(const string &s){
    int n = (int)s.size();
    vector<int>v(n);

    int l, r;
    l = r = 0;
    for(int i = 1;i < n;i++){
        if (i <= r) v[i] = min(v[i-l], r-i+1);
        while(i+v[i] < n && s[i+v[i]] == s[v[i]]) v[i]++;
        if (i+v[i]-1 > r){
            l = i;
            r = i+v[i]-1;
        }
    }

    return v;
}

int main(){

    vector<int>v = z("aabxaabxay");

    for(int i = 0;i < v.size();i++) cout << v[i] << " ";
    cout << endl;

    return 0;
}
```

6.13 suffix array

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

using namespace std;

/*
    Build: O(nlogn)
*/

vector<int> sort_cyclic(const string &s){
    int n = (int)s.size();
    int alpha = 256;

    vector<int>p(n), c(n), cnt(max(alpha, n));

    for(int i = 0;i < n;i++) cnt[s[i]]++;
    for(int i = 1;i < alpha;i++) cnt[i] += cnt[i-1];
    for(int i = n-1;i >= 0;i--) p[--cnt[s[i]]] = i;

    int classes = 1;
    c[p[0]] = 0;
    for(int i = 1;i < n;i++){
        if (s[p[i]] != s[p[i-1]]) classes++;
        c[p[i]] = classes-1;
    }

    vector<int>pn(n), cn(n);
    for(int k = 0;(1<<k) < n;k++){
        fill(cnt.begin(), cnt.begin()+classes, 0);

        for(int i = 0;i < n;i++) pn[i] = (p[i]-(1<<k)+n)%n;
        for(int i = 0;i < n;i++) cnt[c[pn[i]]]++;
        for(int i = 1;i < classes;i++) cnt[i] += cnt[i-1];
        for(int i = n-1;i >= 0;i--) p[--cnt[c[pn[i]]]] = pn[i];

        classes = 1;
        cn[p[0]] = 0;
        for(int i = 1;i < n;i++){
            pair<int, int> a = {c[p[i]], c[(p[i]+(1<<k))%n]};
            pair<int, int> b = {c[p[i-1]], c[(p[i-1]+(1<<k))%n]};
            if (a != b) classes++;
            cn[p[i]] = classes-1;
        }
        swap(c, cn);
    }

    return p;
}

vector<int> suffix_array(string s){
    s += '$';
    vector<int> sorted = sort_cyclic(s);
    sorted.erase(sorted.begin());
    return sorted;
}

int main(){

    for(int i:suffix_array("caule")) cout << i << " ";
    cout << endl;

    return 0;
}
```

6.14 hashing2d

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

using namespace std;

#define ll long long
```

```cpp
mt19937 rng(chrono::steady_clock::now().time_since_epoch().count());

const ll mod = 1e9+7;

struct hash2d{
    vector<ll>power[2];
    vector<vector<ll>>h;

    hash2d(const vector<vector<ll>> &matrix, ll p, ll q){
        int n = matrix.size();
        int m = matrix[0].size();

        power[0].assign(n, 0);
        power[1].assign(m, 0);

        power[0][0] = power[1][0] = 1;
        for(int i = 1;i < n;i++) power[0][i] = power[0][i-1]*p % mod;
        for(int i = 1;i < m;i++) power[1][i] = power[1][i-1]*q % mod;

        h.assign(n, vector<ll>(m, 0));
        h[0][0] = matrix[0][0];

        for(int i = 1;i < n;i++) h[i][0] = (h[i-1][0]*p % mod + matrix[
            i][0]) % mod;
        for(int j = 1;j < m;j++) h[0][j] = (h[0][j-1]*q % mod + matrix
            [0][j]) % mod;
        for(int i = 1;i < n;i++){
            for(int j = 1;j < m;j++){
                (h[i][j] += h[i-1][j]*p) %= mod;
                (h[i][j] += h[i][j-1]*q) %= mod;
                (h[i][j] += mod - h[i-1][j-1]*q % mod *p % mod) %= mod;
                (h[i][j] += matrix[i][j]) %= mod;
            }
        }
    }

    ll submatrix(int a, int b, int x, int y){
        ll res = h[x][y];
        if (a-1 >= 0) (res += mod - h[a-1][y]*power[0][x-a+1] % mod) %=
            mod;
        if (b-1 >= 0) (res += mod - h[x][b-1]*power[1][y-b+1] % mod) %=
            mod;
        if (a*b > 0) (res += h[a-1][b-1]*power[0][x-a+1] % mod * power
            [1][y-b+1] % mod) %= mod;
        return res;
    }
};

int main(){

    vector<vector<ll>>v = {
        {1, 2, 0},
        {3, 1, 2},
        {0, 3, 1}
    };

    hash2d hash(v,
        uniform_int_distribution<ll>(100, mod-1)(rng), // base 1
        uniform_int_distribution<ll>(100, mod-1)(rng)  // base 2
    );

    cout << hash.submatrix(1, 1, 2, 2) << endl;
    cout << hash.submatrix(0, 0, 1, 1) << endl;

    return 0;
}
```

7 Math Extra

7.1 Combinatorial formulas

$\sum_{k=0}^{n} k^2 = n(n+1)(2n+1)/6$
$\sum_{k=0}^{n} k^3 = n^2(n+1)^2/4$
$\sum_{k=0}^{n} k^4 = (6n^5 + 15n^4 + 10n^3 - n)/30$
$\sum_{k=0}^{n} k^5 = (2n^6 + 6n^5 + 5n^4 - n^2)/12$
$\sum_{k=0}^{n} x^k = (x^{n+1} - 1)/(x - 1)$
$\sum_{k=0}^{n} kx^k = (x - (n+1)x^{n+1} + nx^{n+2})/(x-1)^2$
$\binom{n}{k} = \frac{n!}{(n-k)!k!}$
$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$
$\binom{n}{k} = \frac{n}{n-k}\binom{n-1}{k}$
$\binom{n}{k} = \frac{n-k+1}{k}\binom{n}{k-1}$
$\binom{n+1}{k} = \frac{n+1}{n-k+1}\binom{n}{k}$
$\binom{n}{k+1} = \frac{n-k}{k+1}\binom{n}{k}$
$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$
$\sum_{k=1}^{n} k^2\binom{n}{k} = (n+n^2)2^{n-2}$
$\binom{m+n}{r} = \sum_{k=0}^{r}\binom{m}{k}\binom{n}{r-k}$
$\binom{n}{k} = \prod_{i=1}^{k}\frac{n-k+i}{i}$

7.2 Number theory identities

Lucas' Theorem: For non-negative integers m and n and a prime p,

$$\binom{m}{n} \equiv \prod_{i=0}^{k}\binom{m_i}{n_i} \pmod{p},$$

where

$$m = m_k p^k + m_{k-1}p^{k-1} + \cdots + m_1 p + m_0$$

is the base p representation of m, and similarly for n.

7.3 Stirling Numbers of the second kind

Number of ways to partition a set of n numbers into k non-empty subsets.

$$\left\{ {n \atop k} \right\} = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{(k-j)}\binom{k}{j}j^n$$

Recurrence relation:

$$\left\{ {0 \atop 0} \right\} = 1$$

$$\left\{ {n \atop 0} \right\} = \left\{ {0 \atop n} \right\} = 1$$
$$\left\{ {n+1 \atop k} \right\} = k\left\{ {n \atop k} \right\} + \left\{ {n \atop k-1} \right\}$$

7.4 Burnside's Lemma

Let G be a finite group that acts on a set X. For each g in G let X^g denote the set of elements in X that are fixed by g, which means $X^g = \{x \in X | g(x) = x\}$. Burnside's lemma assers the following formula for the number of orbits, denoted $|X/G|$:

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

7.5 Numerical integration

RK4: to integrate $\dot{y} = f(t, y)$ with $y_0 = y(t_0)$, compute

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\ k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\ k_4 &= f(t_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

	S	R	X	Assunto	Descricao	Diff
A						
B						
C						
D						
E						
F						
G						
H						
I						
J						
K						
L						