1 Setup

1.1 Setup

```
compile.sh
    #!/bin/bash
    g++ $1 && ./a.out < ipt.in > opt.out

sudo chmod u+x compile.sh

vim ~/.bashrc
    alias run='~/compile.sh'
```

1.2 template

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
#define All(x) x.begin(), x.end()
#define endl "\n"
#define sort_unique(x) sort(All(x)); x.erase(unique(All(x)),
    x.end());
#define REP(i,n) for(int i = 0; i < n; i++)
#define print1d(a) {for(auto v: a) cout << v << " "; cout <<
    endl;}
#define eb emplace_back
#define pb push_back
#define ne nth_element
#define lb lower_bound
#define ub upper_bound
#define int long long
typedef pair<int, int> pii;
typedef vector<int>   vi;
using namespace std;

#include <bits/extc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define multiset tree<ll, null_type,less_equal<ll>,
    rb_tree_tag,tree_order_statistics_node_update>
order_of_key(k) : nums strictly smaller than k
find_by_order(k): index from 0
/*
| Tag                 | push  | pop    | join | modify |
| paring_heap_tag     | O(1)  | O(lgN) | O(1) | O(lgN) |
| thin_heap_tag       | O(lgN)| O(lgN) |  慢  |   慢   |
| binomial_heap_tag   | O(1)  | O(lgN) | O(1) | O(lgN) |
| rc_binomial_heap_tag| O(1)  | O(lgN) | O(1) | O(lgN) |
| binary_heap_tag     | O(1)  | O(lgN) |  慢  | O(lgN) |
*/
typedef __gnu_pbds::priority_queue<pair<int,ii>,less<pair<int
    ,ii>>,rc_binomial_heap_tag> heap;


int32_t main(){
    IO;

    return 0;
}
```

2 IO

2.1 IO

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
#define All(x) x.begin(), x.end()
#define sort_unique(x) sort(All(x)); x.erase(unique(All(x)),
    x.end());
#define ne nth_element
using namespace std;
#include <bits/extc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define multiset tree<ll, null_type,less_equal<ll>,
    rb_tree_tag,tree_order_statistics_node_update>
order_of_key(k) : nums strictly smaller than k
find_by_order(k): index from 0
/*
| Tag                 | push  | pop    | join | modify |
| paring_heap_tag     | O(1)  | O(lgN) | O(1) | O(lgN) |
| thin_heap_tag       | O(lgN)| O(lgN) |  慢  |   慢   |
| binomial_heap_tag   | O(1)  | O(lgN) | O(1) | O(lgN) |
| rc_binomial_heap_tag| O(1)  | O(lgN) | O(1) | O(lgN) |
| binary_heap_tag     | O(1)  | O(lgN) |  慢  | O(lgN) |
*/
typedef __gnu_pbds::priority_queue<pair<int,ii>,less<pair<int
    ,ii>>,rc_binomial_heap_tag> heap;
```

3 Data_Structure

3.1 線段樹

```
ll v[1e5] , seg[4e5+7] ;
void StructSEG(int l , int r , ll node){
    if(l==r) seg[node] = v[r] ;
    else{
        int mid = (l+r) >> 1 , tmp = node << 1;
        StructSEG(l,mid,tmp) ;
        StructSEG(mid+1,r,tmp+1) ;
        seg[node] = seg[tmp] + seg[tmp+1] ;
    }
}
int query(int tL , int tR , int nL , int nR , int node){
    int mid = (nL+nR)>>1 , ans = 0 ;
    if(tL <= nL && nR <= tR) return seg[node] ;
    if(tL <= mid) ans += query(tL,tR,nL   ,mid,(node<<1)  );
    if(tR  > mid) ans += query(tL,tR,mid+1,nR ,(node<<1)+1);
    return ans ;
}
void updateNode(int idx , int val , int l , int r , int node)
    {
    if(l == r) seg[node] = val , v[idx] = val ;
    else{
        int m = (l + r) >> 1 ;
        int leftNode  = (node << 1) ;
        int rightNode = (node << 1) + 1 ;
        if(idx <= m && idx >= l) updateNode(idx , val , l , m
            , leftNode ) ;
        else updateNode(idx , val , m+1,r , rightNode ) ;
        seg[node] = seg[leftNode] + seg[rightNode] ;
    }
}

int main(){
    IO;
    int n, q, l, r;
    cin >> n >> q;
    for(int i = 1 ; i <= n ; i++) cin >> v[i] ;
    StructSEG(1,n,1) ;
    while(q--){
        cin >> l >> r ;
        cout << query(l,r,1,n,1) << '\n' ;
    }
}
```

3.2 區間修改線段樹

```
#define Ls(x) x << 1
#define Rs(x) x << 1 | 1
int N = 1e5+5;
vector<int> segTree(4*N), lazy(4*N) , arr(N);
void build(int l, int r, int idx = 1){
    if(l == r){
        segTree[idx] = arr[l];
        return;
    }
    int mid = (l+r) >> 1;
    build(l, mid, Ls(idx));
    build(mid+1, r, Rs(idx));
    segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
}

void pushDown(int l, int r, int idx){
    if(lazy[idx] == 0) return;
    segTree[idx] += (r-l+1)*lazy[idx];
    if(l != r) lazy[Ls(idx)] += lazy[idx],lazy[Rs(idx)] +=
        lazy[idx];
    lazy[idx] = 0;
}

void update(int l, int r, int ql, int qr, int val, int idx =
    1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return;
    if(l >= ql && r <= qr){
        segTree[idx] += (r-l+1)*val;
        if(l != r) lazy[Ls(idx)] += val, lazy[Rs(idx)] += val
            ;
        return;
    }
    int mid = (l+r) >> 1;
    update(l, mid, ql, qr, val, Ls(idx));
    update(mid+1, r, ql, qr, val, Rs(idx));
    segTree[idx] = segTree[Ls(idx)]+ segTree[Rs(idx)];
}

int query(int l, int r, int ql, int qr, int idx = 1){
    pushDown(l, r, idx);
```

```
    if(l > qr || r < ql) return 0;
    if(l >= ql && r <= qr) return segTree[idx];
    int mid = (l+r) >> 1, sum = 0;
    if(ql <= mid) sum += query(l, mid, ql, qr, Ls(idx));
    if(qr > mid) sum += query(mid+1, r, ql, qr, Rs(idx));
    return sum;
}
int main(){
    int n, t;
    cin >> n >> t;
    for(int i = 1; i <= n; i++) cin >> arr[i];
    build(1,n,1);
    for(int i = 0; i < t; i++){
        int j, l, r, v;
        cin >> j;
        if(j == 1){
            cin >> l >> r >> v;
            update(1,n,l,r,v);
        }else{
            cin >> r;
            cout << query(r,1,n) << endl;
        }
    }
}




// prefix sum
#pragma GCC optimize("O3","unroll-loops")
#include <bits/stdc++.h>
#define IO ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0)
#define endl '\n'
#define Ls(x) x << 1
#define Rs(x) (x << 1) | 1
using namespace std;
typedef long long ll;

int N = 2e5 + 5;
vector<ll> segTree(4 * N, -2e18), lazy(4 * N), arr(N), tmp(N);

void build(int l, int r, int idx = 1){
    if(l == r){
        segTree[idx] = arr[l];
        return;
    }
    int mid = (l+r) >> 1;
    build(l, mid, Ls(idx));
    build(mid+1, r, Rs(idx));
    segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
}

void pushDown(int l, int r, int idx, int x = 0){
    if(lazy[idx] == 0) return;
    segTree[idx] += lazy[idx];
    int mid = (l + r) / 2;
    if(l != r) lazy[Ls(idx)] += lazy[idx],lazy[Rs(idx)] += lazy[idx];
    lazy[idx] = 0;
}

void update(int l, int r, int ql, int qr, ll val, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return;
    if(l >= ql && r <= qr){
        /* segTree[idx] += val; */
        lazy[idx] += val;
        pushDown(l, r, idx);
        return;
    }
    int mid = (l+r) >> 1;
    update(l, mid, ql, qr, val, Ls(idx));
    update(mid+1, r, ql, qr, val, Rs(idx));
    segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
}


ll query(int l, int r, int ql, int qr, int idx = 1){
    pushDown(l, r, idx);
    if(l > qr || r < ql) return -2e18;
    if(l >= ql && r <= qr) return segTree[idx];
    ll mid = (l+r) >> 1, ans = -2e18;
    ans = max(ans, query(l, mid, ql, qr, Ls(idx)));
    ans = max(ans, query(mid+1, r, ql, qr, Rs(idx)));
    segTree[idx] = max(segTree[Ls(idx)], segTree[Rs(idx)]);
    return ans;
}

int main(){
    IO;
    int n, q;
    cin >> n >> q;
    for(int i = 1; i <= n; i++) cin >> tmp[i];
    arr = tmp;
    for(int i = 2; i <= n; i++) arr[i] += arr[i-1];
    build(1, n, 1);

    while(q--){
        int j, k, u;
        cin >> j >> k >> u;
        if(j == 1) update(1, n, k, n, u - tmp[k] ), tmp[k] = u;
        else cout << max(0ll, query(1, n, k, u) - (k!=1?query(1, n, k-1, k-1):0)) << endl;
    }
}
```

3.3 Kurskal

```
struct Edge {
    int src;
    int dest;
    int weight;
    Edge(int s, int d, int w) : src(s), dest(d), weight(w) {}
};
int findParent(const vector<int>& parent, int i) {
    if (parent[i] == i) return i;
    return findParent(parent, parent[i]);
}
vector<Edge> kruskalMST(const vector<vector<int>>& adjacencyMatrix) {
    int size = adjacencyMatrix.size();
    vector<Edge> mst, edges;
    for (int i = 0; i < size; ++i) {
        for (int j = i + 1; j < size; ++j) {
            if (adjacencyMatrix[i][j] > 0) edges.push_back(Edge(i, j, adjacencyMatrix[i][j]));
        }
    }
    sort(edges.begin(), edges.end(), [](const Edge& e1, const Edge& e2) {
        return e1.weight < e2.weight;
    });
    vector<int> parent(size);
    for (int i = 0; i < size; ++i) parent[i] = i;
    int count = 0;
    for (const Edge& edge : edges) {
        if (count == size - 1) break;
        int srcParent = findParent(parent, edge.src) , destParent = findParent(parent, edge.dest);
        if (srcParent != destParent) {
            mst.push_back(edge);
            ++count;
            parent[srcParent] = destParent;
        }
    }
    return mst;
}
```

3.4 Prim

```
vector<Edge> primMST(const vector<vector<int>>& adjacencyMatrix) {
    int size = adjacencyMatrix.size();
    vector<bool> visited(size, false);
    vector<Edge> mst;
    priority_queue<Edge, vector<Edge>, function<bool(Edge, Edge)>> pq(
        [](const Edge& e1, const Edge& e2) { return e1.weight > e2.weight; }
    );
    visited[0] = true;
    for (int i = 1; i < size; ++i) {
        if (adjacencyMatrix[0][i] > 0) {
            pq.push(Edge(0, i, adjacencyMatrix[0][i]));
        }
    }
    while (!pq.empty()) {
        Edge minEdge = pq.top();
        pq.pop();
        int u = minEdge.dest;
        if (visited[u]) continue;
        visited[u] = true;
        mst.push_back(minEdge);
        for (int v = 0; v < size; ++v) {
            if (adjacencyMatrix[u][v] > 0 && !visited[v]) pq.push(Edge(u, v, adjacencyMatrix[u][v]));
        }
    }
    return mst;
}
```

3.5 DAG

```cpp
void DFS(map<int,queue<int>> &graph , map<int,bool> &visited , int place , stack<int> &ans){
    if(visited[place] == true) return;
    visited[place]=true;
    while(!graph[place].empty())
        DFS(graph, visited, graph[place].front() , ans) , graph[place].pop() ;
    ans.push(place);
}
for(auto&n:graph) DFS(graph , visited , n.first , ans) ;
while(!ans.empty()) cout << ans.top() << " " , ans.pop();
```

3.6 Unionfind

```cpp
int Find(int*a,int n){
    if(a[n]==n)return n;
    return a[n]=Find(a,a[n]);
}
void Union(int*a,int n,int m){
    a[Find(a,n)]=Find(a,m);
}
```

3.7 CDQ

```cpp
#include <bits/stdc++.h>
using namespace std;

#define fastio ios_base::sync_with_stdio(false); cin.tie(0);
#define endl '\n'
#define _ << ' ' <<

const int INF = 2e9;

struct info {
    int t, p;
    long long x;
    // 當 x 界在 -1e9 ~ 1e9，代表這是修改
    // 當 x >= INF，代表這是詢問
    // 並且詢問的編號為 (x - INF)
    // 例：第 2 次詢問，那 x 就設為 INF + 2
    // 當 x <= -INF，一樣代表詢問，只不過是倒扣的詢問
};

int n, q;
vector<info> v;
vector<long long> ans;

void cdq(int l, int r) {
    if (l == r) return;

    int mid = (l+r) / 2; cdq(l, mid); cdq(mid+1, r);

    vector<info> temp;
    long long add_sum = 0;
    int pl = l, pr = mid+1;

    while (pl <= mid && pr <= r) {
        if (v[pl].p <= v[pr].p) {
            temp.emplace_back(v[pl]);
            if (abs(v[pl].x) <= 1e9) {
                add_sum += v[pl].x;
            }
            pl++;
        }
        else {
            temp.emplace_back(v[pr]);
            if (v[pr].x >= INF) {
                ans[v[pr].x - INF] += add_sum;
            } else if (v[pr].x <= -INF) {
                ans[-(v[pr].x + INF)] -= add_sum;
            }
            pr++;
        }
    }
    while (pl <= mid) {
        temp.emplace_back(v[pl]);
        // 小小優化，註解掉的這段一樣可以不寫
        // if (abs(v[pl].x) <= 1e9) {
        //     add_sum += v[pl].x;
        // }
        pl++;
    }
    while (pr <= r) {
        temp.emplace_back(v[pr]);
        if (v[pr].x >= INF) {
            ans[v[pr].x - INF] += add_sum;
        } else if (v[pr].x <= -INF) {
            ans[-(v[pr].x + INF)] -= add_sum;
        }
        pr++;
    }

    for (int i = l, j = 0; i <= r; i++, j++) {
        v[i] = temp[j];
    }
}


int main() {
    fastio

    cin >> n >> q;
    v.clear(); ans.clear();

    int time = 0;
    vector<long long> arr(n+1);

    for (int p = 1; p <= n; p++) {
        long long x; cin >> x;
        v.emplace_back(info{++time, p, x});
        arr[p] = x;
    }

    int ask_id = 0;
    for (int i = 0; i < q; i++) {
        int op; cin >> op;
        if (op == 1) {
            int p; long long x;
            cin >> p >> x;
            v.emplace_back(info{++time, p, x - arr[p]});
            arr[p] = x;
        }
        else {
            int l, r; cin >> l >> r;
            v.emplace_back(info{++time, r, INF + ask_id});
            if (l > 1) v.emplace_back(info{time, l-1, -INF - ask_id});
            ask_id++;
            ans.emplace_back(0);
        }
    }

    cdq(0, (int)v.size()-1);

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

    return 0;
}

struct info {
    int x, y, id; // id 是 數 對 在 原 本 陣 列 的 位 置
};
int N;
vector<info> v;
vector<int> ans;

void cdq(int L, int R) {
    if (L == R) return;

    int mid = (L + R) / 2;
    cdq(L, mid);
    cdq(mid + 1, R);

    vector<info> temp;
    int pl = L, pr = mid + 1;

    while (pl <= mid && pr <= R) {
        if (v[pl].y < v[pr].y) {
            temp.emplace_back(v[pl]);
            pl++;
        } else {
            temp.emplace_back(v[pr]);
            ans[v[pr].id] += (pl - L);
            pr++;
        }
    }

    while (pl <= mid) {
        temp.emplace_back(v[pl]);
        pl++;
    }
    while (pr <= R) {
        temp.emplace_back(v[pr]);
        ans[v[pr].id] += (pl - L);
        pr++;
    }

    for (int pt = 0, pv = L; pv <= R; pt++, pv++) {
        v[pv] = temp[pt];
    }
}

int main() {

    // 把 所 有 數 對 按 照 x 排 序;
    cdq(0, N - 1);
    // 輸 出 答 案;

```

```
    return 0;
}


struct info {
    int x, y, z, id; // id 是 數 對 在 原 本 陣 列 的 位 置
};
int N;
vector<info> v;
vector<int> ans;

void cdq(int L, int R) {
    if (L == R) return;
    int mid = (L + R) / 2;
    cdq(L, mid);
    cdq(mid + 1, R);

    vector<info> temp;
    vector<pair<int, int>> BIT_op; // BIT_op: 紀 錄 你 在 BIT
          的 操 作, 包 含 修 改 位 置、 修 改 的 值

    int pl = L, pr = mid + 1;
    while (pl <= mid && pr <= R) {
        if (v[pl].y < v[pr].y) {
            temp.emplace_back(v[pl]);
            BIT.upd(v[pl].z, 1);
            BIT_op.emplace_back(make_pair(v[pl].z, 1));
            pl++;
        } else {
            temp.emplace_back(v[pr]);
            ans[v[pr].id] += BIT.qry(v[pr].z - 1);
            pr++;
        }
    }

    while (pl <= mid) {
        temp.emplace_back(v[pl]);
        pl++;
    }
    while (pr <= R) {
        temp.emplace_back(v[pr]);
        ans[v[pr].id] += BIT.qry(v[pr].z - 1);
        pr++;
    }

    for (int pt = 0, pv = L; pv <= R; pt++, pv++) {
        v[pv] = temp[pt];
    }

    // 撤 銷 所 有 操 作, 讓 BIT 變 回 初 始 狀 態
    for (auto& op : BIT_op) {
        BIT.upd(op.first, -op.second);
    }
}

int main() {
    // 把 所 有 數 對 按 照 x 排 序;
    cdq(0, N - 1);
    // 輸 出 答 案;

    return 0;
}
```

3.8 DSU

```
#include <bits/stdc++.h>
using namespace std;

class UnionFind {
private:
    vector<int> parent;
    vector<int> rank;

public:
    UnionFind(int size) : parent(size), rank(size, 1) {
        for(int i = 0; i < size; ++i) {
            parent[i] = i;
        }
    }

    int getSize() const {
        return parent.size();
    }

    int find(int p) {
        if(p < 0 || p >= parent.size())
            throw out_of_range("Out of bound.");

        while(p != parent[p]) {
            parent[p] = parent[parent[p]];
            p = parent[p];
        }
        return p;
    }

    int findR(int p) {
        if(p < 0 || p >= parent.size())
            throw out_of_range("Out of bound.");

        if(p != parent[p])
            parent[p] = findR(parent[p]);
        return parent[p];
    }

    bool isConnected(int p, int q) {
        return find(p) == find(q);
    }

    void unionElements(int p, int q) {
        int pRoot = find(p);
        int qRoot = find(q);

        if(pRoot == qRoot)
            return;

        if(rank[pRoot] < rank[qRoot]) {
            parent[pRoot] = qRoot;
        } else if(rank[qRoot] < rank[pRoot]) {
            parent[qRoot] = pRoot;
        } else {
            parent[pRoot] = qRoot;
            rank[qRoot]++;
        }
    }
    int countConnectedNodes(int p) {
        int root = find(p);
        int count = 0;
        for (int i = 0; i < parent.size(); i++) {
            if (find(i) == root) {
                count++;
            }
        }
        return count;
    }
};

int main() {
    UnionFind uf(10);

    uf.unionElements(1, 2);
    uf.unionElements(2, 3);
    uf.unionElements(4, 5);
    uf.unionElements(3, 4);

    cout << "Is 1 connected to 3? " << (uf.isConnected(1, 3)
         ? "Yes" : "No") << endl;
    cout << "Is 1 connected to 5? " << (uf.isConnected(1, 5)
         ? "Yes" : "No") << endl;
    cout << "Number of nodes connected to 3: " << uf.
         countConnectedNodes(4) << endl;

    return 0;
}
```

4 DP

4.1 LIS

```
void LIS(vector<int> &arr){
    vector<int> lis;
    for(int i = 0; i < arr.size(); i++){
        auto it = lower_bound(lis.begin(), lis.end(), arr[i])
              ;
        if(it == lis.end()) lis.push_back(arr[i]);
        else *it = arr[i];
    }
    cout << lis.size() << endl;
}
```

4.2 LCS2LIS

```
int LCS2LIS(string &a, string &b){
    vector<int> List;
    for(int i = 0; i < b.size(); i++)
        for(int j = 0; j < a.size(); j++)
            if(b[i] == a[j]) List.push_back(j);
    if(List.size() == 0) return 0;
    vector<int> dp(List.size(), 1);
    for(int i = 1; i < List.size(); i++){
        if(List[i] > dp.back()) dp.push_back(List[i]);
        else *lower_bound(dp.begin(), dp.end(), List[i]) =
             List[i];
    }
    return dp.size();
}
```

4.3 LCS3D

```cpp
int main(){
    string x,y,z ; cin >> x >> y >> z ;
    cout << lcsOf3(x,y,z , x.size() , y.size() , z.size()) << endl;
}

int lcsOf3( string X, string Y, string Z, int m, int n, int o ){
    int L[m+1][n+1][o+1];
    for (int i=0; i<=m; i++){
        for (int j=0; j<=n; j++){
            for (int k=0; k<=o; k++){
                if (i == 0 || j == 0||k==0) L[i][j][k] = 0;
                else if (X[i-1] == Y[j-1] && X[i-1]==Z[k-1])
                    L[i][j][k] = L[i-1][j-1][k-1] + 1;
                else L[i][j][k] = max(max(L[i-1][j][k],L[i][j-1][k]),L[i][j][k-1]);
            }
        }
    }
    return L[m][n][o];
}
```

4.4 LCS

```
若一樣：左上+1
else max(左,上)
```

4.5 countingTours

```cpp
#include <bits/stdc++.h>
#define IO cin.tie(0), ios::sync_with_stdio(0)
#define All(x) x.begin(), x.end()
#define sort_unique(x) sort(All(x)); x.erase(unique(All(x)), x.end());
#define ne nth_element
#define INF 1e9
#define MOD 1000000007
#define Q 1000000
typedef long long ll;
using namespace std;


vector<ll> a(Q+1), b(Q+1);


int main(){
    IO;
    ll t;
    cin >> t;
    a[0] = b[0] = 1;

    for(ll i = 1; i <= Q; i++){
        a[i] = a[i-1] + b[i-1] + a[i-1];
        b[i] = a[i-1] + b[i-1] + b[i-1]*3;
        a[i] %= MOD;
        b[i] %= MOD;
    }
    while(t--){
        ll q;
        cin >> q;
        cout << (a[q-1] + b[q-1])%MOD << endl;
    }
}
```

4.6 retCutting

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int dp[505][505];

ll f(int x, int y){
    if(x==y) return 0 ;
    if(dp[x][y] != 0) return dp[x][y];
    ll ans = 2e18;
    for(int i = 1; i < x; i++){
        ans = min(ans,f(i,y) + f(x-i,y) + 1);
    }
    for(int i = 1; i < y; i++){
        ans = min(ans,f(i,x) + f(y-i,x) + 1);
    }
    return dp[x][y] = ans;
}

int main(){

    int x,y;
    cin >> x >> y;
    cout << f(x,y);
}
```

5 Graph

5.1 Dijkstra

```cpp
/** 問某點到所有圖上的點的最短距離。0/1-based 都安全。 edge 要
 * 是 {cost，dest} 格式。回傳的陣列若含有 -1 表示 src 到該位置
 * 不連通 **/
typedef pair<ll, int> pii;
vector<ll> dijkstra(int src, vector<vector<pii>>& edge) {
    vector<ll> sum(edge.size(), -1);
    priority_queue<pii, vector<pii>, greater<pii>> q;
    q.emplace(0, src);
    while (q.size()) {
        int v = q.top().second; ll d = q.top().first;
        q.pop();
        if (sum[v] != -1) continue;
        sum[v] = d;
        for (auto& e : edge[v])
            if (sum[e.second] == -1)
                q.emplace(d + e.first, e.second);
    } return sum;
}
int main(){
    int n,m,s; cin>>n>>m>>s;
    vector<vector<pii>> G(n+1);
    while(m--) {
        int u,v,w; cin>>u>>v>>w;
        G[u].emplace_back(w,v);
    }
    vector<ll> dis = dijkstra(s,G);
    for(int i=1; i<=n; i++) cout<<dis[i]<<" \n"[i==n];
}
```

5.2 BellmanFord

```cpp
typedef pair<int,int> pii;
const int maxn = 1e5+5;
const int INF = 0x3f3f3f3f;
vector<pii> G[maxn];
int dis[maxn];
bool BellmanFord(int n,int s) {
    for(int i=1; i<=n; i++) dis[i] = INF;
    dis[s] = 0;
    bool relax;
    for(int r=1; r<=n; r++) { //O(VE)
        relax = false;
        for(int i=1; i<=n; i++)
            for(pii e:G[i])
                if( dis[i] + e.second < dis[e.first] )
                    dis[e.first] = dis[i] + e.second, relax = true;
    }
    return relax; //有負環
}
int main(){
    int n,m,s; cin>>n>>m>>s;
    while(m--) {
        int u,v,w; cin>>u>>v>>w;
        G[u].emplace_back(v,w);
    }
    if(BellmanFord(n,s)) cout<<"negative cycle\n";
    else for(int i=1; i<=n; i++) cout<<dis[i]<<" \n"[i==n];
}
```

5.3 SPFA

```cpp
typedef pair<int,int> pii;
const int maxn = 1e5+5;
const int INF = 0x3f3f3f3f;
vector<pii> G[maxn]; int dis[maxn];
void SPFA(int n,int s) {
    for(int i=1; i<=n; i++) dis[i] = INF;
    dis[s] = 0;
    queue<int> q; q.push(s);
    bool inque[maxn] = {};
    while(!q.empty()) {
        int u = q.front(); q.pop();
```

```
        inque[u] = false;
        for(pii e:G[u]) {
            int v = e.first , w = e.second;
            if( dis[u] + w < dis[v]) {
                if(!inque[v]) q.push(v), inque[v] = true;
                dis[v] = dis[u] + w;
            }
        }
    }
}
int main(){
    int n,m,s; cin>>n>>m>>s;
    while(m--) {
        int u,v,w; cin>>u>>v>>w;
        G[u].emplace_back(v,w);
    }
    SPFA(n,s);
    for(int i=1; i<=n; i++) cout<<dis[i]<<" \n"[i==n];
}
```

5.4 MaxFlow

```
struct edge{
    int from, to, cap, flow;
    edge(int u, int v, int c, int f) : from(u), to(v), cap(c)
         , flow(f) {}
};

struct Dinic{
    vector<edge> edges;
    vector<vector<int>> adj;
    vector<int> level, ptr;
    int n, m = 0, s, t;
    Dinic(int n, int s, int t) : n(n), s(s), t(t){
        adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }
    void addEdge(int u, int v, int c){
        edges.emplace_back(u, v, c, 0);
        edges.emplace_back(v, u, 0, 0);
        adj[u].push_back(m);
        adj[v].push_back(m+1);
        m += 2;
    }
    bool bfs(){
        fill(level.begin(), level.end(), -1);
        queue<int> q;
        q.push(s);
        level[s] = 0;
        while(!q.empty()){
            int u = q.front();
            q.pop();
            for(int id : adj[u]){
                if(edges[id].cap - edges[id].flow < 1)
                     continue;
                int v = edges[id].to;
                if(level[v] != -1) continue;
                level[v] = level[u] + 1;
                q.push(v);
            }
        }
        return level[t] != -1;
    }
    int dfs(int u, int pushed){
        if(pushed == 0) return 0;
        if(u == t) return pushed;
        for(int& cid = ptr[u]; cid < (int)adj[u].size(); cid
             ++){
            int id = adj[u][cid];
            int v = edges[id].to;
            if(level[u] + 1 != level[v] || edges[id].cap -
                 edges[id].flow < 1) continue;
            int tr = dfs(v, min(pushed, edges[id].cap - edges
                 [id].flow));
            if(tr == 0) continue;
            edges[id].flow += tr;
            edges[id^1].flow -= tr;
            return tr;
        }
        return 0;
    }
    int flow(){
        int f = 0;
        while(true){
            if(!bfs()) break;
            fill(ptr.begin(), ptr.end(), 0);
            while(int pushed = dfs(s, INT_MAX)){
                f += pushed;
            }
        }
        return f;
    }
};
int main(){
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    Dinic dinic(n, s, t);
    for(int i = 0; i < m; i++){
        int u, v, c;
        cin >> u >> v >> c;
        dinic.addEdge(u, v, c);
    }
    cout << dinic.flow() << endl;
}
```

5.5 EdmondsKarp

```
int main(){
    int n,m,b,e,s,t,w,q=1;
    while(cin>>n,n){
        int ans=0,go=1,Min,a[n+1][n+1]={},last[n+1]={},can[n
             +1]={};cin>>b>>e>>m;
        while(m--){cin>>s>>t>>w;a[s][t]=a[t][s]+=w;}
        while(go){go=0;
            queue<int> Q;Q.push(b);Min=99999999;
            while(Q.size()){
                int top=Q.front();Q.pop();can[top]=1;
                for(int k=1;k<=n;k++)if(a[top][k]&&!can[k]){
                     last[k]=top;if(k==e){go=1;break;}Q.push(
                     k);}
                if(go){
                    int S=s=e;
                    while(last[s])Min=min(Min,a[last[s]][s]),
                         s=last[s];
                    while(last[S])a[last[S]][S]-=Min,a[S][
                         last[S]]+=Min,S=last[S];
                    ans+=Min;
                    fill_n(last,n+1,0);fill_n(can,n+1,0);
                    break;
                }
            }
        }
        printf("Network %d\nThe bandwidth is %d.\n\n",q++,ans
             );
    }
}
```

6 Math

6.1 Bignumber

```
// a + b
string add(string a, string b) {
    string ans;
    int carry = 0;
    while (a.size() < b.size()) a = '0' + a;
    while (a.size() > b.size()) b = '0' + b;
    for (int i = a.size() - 1; i >= 0; i--) {
        int tmp = a[i] - '0' + b[i] - '0' + carry;
        carry = tmp / 10;
        ans = char(tmp % 10 + '0') + ans;
    }
    if (carry) ans = '1' + ans;
    return ans;
}

// a - b
string sub(string a, string b) {
    string ans;
    int carry = 0;
    while (a.size() < b.size()) a = '0' + a;
    while (a.size() > b.size()) b = '0' + b;
    for (int i = a.size() - 1; i >= 0; i--) {
        int tmp = a[i] - b[i] - carry;
        if (tmp < 0) {
            tmp += 10;
            carry = 1;
        } else carry = 0;
        ans = char(tmp + '0') + ans;
    }
    while (ans.size() > 1 && ans[0] == '0') ans.erase(0, 1);
    return ans;
}

// a * b
string mul(string a, string b) {
    string ans;
    int carry = 0;
    for (int i = a.size() - 1; i >= 0; i--) {
        string tmp;
        for (int j = b.size() - 1; j >= 0; j--) {
            int t = (a[i] - '0') * (b[j] - '0') + carry;
            carry = t / 10;
            tmp = char(t % 10 + '0') + tmp;
        }
```

```
        if (carry) tmp = char(carry + '0') + tmp;
        for (int j = 0; j < a.size() - 1 - i; j++) tmp += '0';
        ans = add(ans, tmp);
        carry = 0;
    }
    return ans;
}

// a / b
string div(string a, string b) {
    string ans;
    string tmp;
    for (int i = 0; i < a.size(); i++) {
        tmp += a[i];
        int cnt = 0;
        while (tmp.size() >= b.size()) {
            if (tmp.size() > b.size() || tmp >= b) {
                tmp = sub(tmp, b);
                cnt++;
            } else break;
        }
        ans += char(cnt + '0');
    }
    while (ans.size() > 1 && ans[0] == '0') ans.erase(0, 1);
    return ans;
}

// a % b
string mod(string a, string b) {
    string ans;
    string tmp;
    for (int i = 0; i < a.size(); i++) {
        tmp += a[i];
        int cnt = 0;
        while (tmp.size() >= b.size()) {
            if (tmp.size() > b.size() || tmp >= b) {
                tmp = sub(tmp, b);
                cnt++;
            } else break;
        }
        ans += char(cnt + '0');
    }
    return tmp;
}

// a ^ b
string pow(string a, string b) {
    string ans = "1";
    while (b != "0") {
        if ((b[b.size() - 1] - '0') % 2) ans = mul(ans, a);
        a = mul(a, a);
        b = div(b, "2");
    }
    return ans;
}

// a + b (可为负数)
string add(string a, string b) {
    if (a[0] == '-' && b[0] == '-') return '-' + add(a.erase(0, 1), b.erase(0, 1));
    if (a[0] == '-') return sub(b, a.erase(0, 1));
    if (b[0] == '-') return sub(a, b.erase(0, 1));
    return add(a, b);
}

// A - B
// B - A
// A + B

string add(string a, string b){
    // center = '.'
    // decimal part
    string ans;
    int carry = 0;
    int center = 0;
    while (a.size() < b.size()) a = '0' + a;
    while (a.size() > b.size()) b = '0' + b;
    for (int i = a.size() - 1; i >= 0; i--) {
        if (a[i] == '.') {
            center = 1;
            ans = '.' + ans;
            continue;
        }
        int tmp = a[i] - '0' + b[i] - '0' + carry;
        carry = tmp / 10;
        ans = char(tmp % 10 + '0') + ans;
    }
    if (carry) ans = '1' + ans;
    if (center) ans = '.' + ans;
    return ans;
}
```

6.2 BignumberDec

```
#include <iostream>
#include <string.h>
#include <set>
#include <sstream>
using namespace std;
typedef long long LL;
class Frac{public:
    LL mi,mn,md;
    Frac(LL i=0,LL n=0,LL d=1){
        i+=n/d;n%=d;
        while((i<0&&n>0)||(i>0&&n<0)){
            if(n<0)n+=d,i-=1;
            else n-=d,i+=1;
        }
        mi=i,mn=n,md=d;
        while(n%=d)swap(n,d);if(d<0)d=-d;
        mn/=d;md/=d;
    }
    Frac operator+(Frac b){return Frac(mi+b.mi,mn*b.md+md*b.mn,md*b.md);}
};
istream&operator>>(istream&xin,Frac&a){string s;xin>>s;int sn=s[0]=='-';if(sn)s=s.substr(1);
    int pos=s.find('.');LL i,n,d;
    if(pos==-1){istringstream(s)>>i;n=0;d=1;
    }else{        istringstream(s.substr(0,pos))>>i;
                istringstream(s.substr(pos+1))>>n;
                istringstream(string(s.size()-1-pos,'9'))>>d;
    }
    if(sn)i=-i,n=-n;
    a=Frac(i,n,d);return xin;
}
ostream&operator<<(ostream&xout,Frac&a){
    LL i=a.mi,n=a.mn,d=a.md;
    if(i<0||n<0){xout<<'-';i=-i,n=-n;}
    xout<<i;if(n)xout<<".";
    set<LL>ns;ns.insert(0);
    while(ns.find(n)==ns.end()){ns.insert(n);n*=10;xout<<n/d;n%=d;}
    return xout;
}
int main(){
    int cs;cin>>cs;
    Frac a,b;
    while(cin>>a>>b){
        auto ans=a+b;
        cout<<ans<<endl;
    }
    return 0;
}

string operator+(string a,string b){
    string ans = "";
    if(a.size() > b.size()) b = string(a.size()-b.size(),'0').append(b);
    else                    a = string(b.size()-a.size(),'0').append(a);
    int c=0,tt=0;
    for(int i=a.size()-1;i>=0;i--){
        if(a[i] == '.') {ans = string(".").append(ans);continue;}
        int x = a[i]-'0' , y = b[i]-'0';
        x = x+y+c;
        c=x/10;
        if(c==1 && i-1>=0 && a[i-1] == '.' ) tt=1;
        ans = to_string(x%10).append(ans);
    }

    string temppp = string("0.").append(ans.size()-2-ans.find('.'),'0').append("1");
    if(tt) ans = ans + temppp;
    if(c) ans = string("1").append(ans);
    return ans;
}


string operator-(string a,string b){
    string ans="";

    if(a.size() > b.size()) b = string(a.size()-b.size(),'0').append(b);
    else                    a = string(b.size()-a.size(),'0').append(a);


    int c=0,tt=0,xx=0,ii=0;
    while(ii<b.size() &&a[ii] == b[ii]) ii++;
    if(ii==b.size());
    else if(a[ii] < b[ii]) {tt=1;swap(a,b);}

    for(int i=a.size()-1;i>=0;i--){
        if(a[i] == '.') {ans = string(".").append(ans);continue;}
        int x=a[i]-'0',y=b[i]-'0';
        x=x-c-y;
        if(x<0){c=1;x+=10;}
        else c=0;
        if(c==1 && i-1>=0 && a[i-1]=='.') xx=1;
        ans = to_string(x%10).append(ans);
```

```
    }

    string temppp = string("0.").append(ans.size()-2-ans.find
        ('.'),'0').append("1");
    if(xx) ans = ans - temppp;

    while(ans[0]=='0' && ans.size()>=2 && ans[1] != '.') {
        ans = ans.substr(1,ans.size()-1);
    }

    if(tt || c==1) ans = string("-").append(ans);
    return ans;
}
```

6.3 線篩

```
vector<bool>p(n,1);
vector<int>prime;
for(int i=2;i<n;i++){
    if(p[i])prime.push_back(i);
    for(auto&k:prime){
        if(i*k>=n)break;
        p[i*k]=0;
        if(i%k==0)break;
    }
}
```

6.4 Fib

```
1/sqrt(5)*(pow((1+sqrt(5))/2,n)-pow((1-sqrt(5))/2,n))
```

6.5 鞋帶

```
vector<pair<int,int>> v(n);
for(auto&n:v) cin >> n.first >> n.second;
int area = 0;
for(int i = 0; i < v.size(); i++){
    area += v[i].first * v[(i+1)%v.size()].second;
    area -= v[i].second * v[(i+1)%v.size()].first;
}
cout << abs(area)/2 << endl;
```

6.6 SG

```
Anti Nim (取走最後一個石子者敗) :
先手必勝 if and only if
1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。
-----------------------------------------------------
Anti-SG (決策集合為空的遊戲者贏) :
定義 SG 值為 0 時，遊戲結束，
則先手必勝 if and only if
1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0
    。
-----------------------------------------------------
Sprague-Grundy :
1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝
不為 0：先手(N)必勝
int mex(set S) {
  // find the min number >= 0 that not in the S
  // e.g. S = {0, 1, 3, 4} mex(S) = 2
}
state = []
int SG(A) {
  if (A not in state) {
    S = sub_states(A)
    if( len(S) > 1 ) state[A] = reduce(operator.xor, [SG(B)
        for B in S])
    else state[A] = mex(set(SG(B) for B in next_states(A)))
  } return state[A]
}
```

6.7 擴展歐幾里德

```
// 給 a,b ，解 ax+by=gcd(a,b)
typedef pair<ll, ll> pii;
pii extgcd(ll a, ll b) {
    if (b == 0) return {1, 0};
    ll k = a / b;
    pii p = extgcd(b, a - k * b);
    return {p.second, p.first - k * p.second};
}
```

6.8 FPow

```
// 問 a ^ p
ll fastpow(ll a, int p) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret *= a;
        a *= a, p >>= 1;
    } return ret;
}
// 問 (a ^ p) mod m
ll fastpow(ll a, ll p, ll m) {
    ll ret = 1;
    while (p) {
        if (p & 1) ret = ret * a % m;
        a = a * a % m, p >>= 1;
    } return ret;
}
```

6.9 質因數分解

```
LL func(const LL n,const LL mod,const int c) {
  return (LLmul(n,n,mod)+c+mod)%mod;
}
LL pollorrho(const LL n, const int c) {//循環節長度
  LL a=1, b=1;
  a=func(a,n,c)%n;
  b=func(b,n,c)%n; b=func(b,n,c)%n;
  while(gcd(abs(a-b),n)==1) {
    a=func(a,n,c)%n;
    b=func(b,n,c)%n; b=func(b,n,c)%n;
  }
  return gcd(abs(a-b),n);
}
void prefactor(LL &n, vector<LL> &v) {
  for(int i=0;i<12;++i) {
    while(n%prime[i]==0) {
      v.push_back(prime[i]);
      n/=prime[i];
    }
  }
}
void smallfactor(LL n, vector<LL> &v) {
  if(n<MAXPRIME) {
    while(isp[(int)n]) {
      v.push_back(isp[(int)n]);
      n/=isp[(int)n];
    }
    v.push_back(n);
  } else {
    for(int i=0;i<primecnt&&prime[i]*prime[i]<=n;++i) {
      while(n%prime[i]==0) {
        v.push_back(prime[i]);
        n/=prime[i];
      }
    }
    if(n!=1) v.push_back(n);
  }
}
void comfactor(const LL &n, vector<LL> &v) {
  if(n<1e9) {
    smallfactor(n,v);
    return;
  }
  if(Isprime(n)) {
    v.push_back(n);
    return;
  }
  LL d;
  for(int c=3;;++c) {
    d = pollorrho(n,c);
    if(d!=n) break;
  }
  comfactor(d,v);
  comfactor(n/d,v);
}
void Factor(const LL &x, vector<LL> &v) {
  LL n = x;
  if(n==1) { puts("Factor 1"); return; }
  prefactor(n,v);
  if(n==1) return;
  comfactor(n,v);
  sort(v.begin(),v.end());
}
```

```cpp
void AllFactor(const LL &n,vector<LL> &v) {
  vector<LL> tmp;
  Factor(n,tmp);
  v.clear();
  v.push_back(1);
  int len;
  LL now=1;
  for(int i=0;i<tmp.size();++i) {
    if(i==0 || tmp[i]!=tmp[i-1]) {
      len = v.size();
      now = 1;
    }
    now*=tmp[i];
    for(int j=0;j<len;++j)
      v.push_back(v[j]*now);
  }
}
```

6.10 Expression

```cpp
/**
 * 支援處理四則運算的工具。給四則運算的字串，檢查格式並計算其值。如果
 * 格式不合法，會丟出錯誤。複雜度 O(字串長度) 。支援的符號有四則運算
 * 和求餘數，先乘除後加減。可以使用括號、或前置正負號。數字開頭可以為
 * 零或禁止為零。可以兼容或禁止多重前置號 (例如 --1 視為 1 、+-+-1
 * 視為 -1) 。空字串視為不合法。運算範圍限於 long long 。如果試圖除
 * 以零或對零求餘也會丟出錯誤。
 */
void req(bool b) { if (!b) throw ""; }
const int B = 2; // 可以調整成 B 進位
class Expr {
   private:
    deque<char> src;
    Expr(const string& s) : src(s.begin(), s.end()) {}
    inline char top() {
        return src.empty() ? '\0' : src.front();
    }
    inline char pop() {
        char c = src.front(); src.pop_front(); return c;
    }
    ll n() {
        ll ret = pop() - '0';
        // 若要禁止數字以 0 開頭，加上這行
        // req(ret || !isdigit(top()));
        while (isdigit(top())) ret = B * ret + pop() - '0';
        return ret;
    }
    ll fac() {
        if (isdigit(top())) return n();
        if (top() == '-') { pop(); return -fac(); }
        if (top() == '(') {
            pop();
            ll ret = expr(1);
            req(pop() == ')');
            return ret;
        }
        // 若要允許前置正號，加上這行
        // if(top() == '+') { pop(); return fac(); }
        throw "";
    }
    ll term() {
        ll ret = fac(); char c = top();
        while (c == '*' || c == '/' || c == '%') {
            pop();
            if (c == '*') ret *= fac();
            else {
                ll t = fac(); req(t);
                if (c == '/') ret /= t; else ret %= t;
            }
            c = top();
        } return ret;
    }
    ll expr(bool k) {
        ll ret = term();
        while (top() == '+' || top() == '-')
            if (pop() == '+') ret += term();
            else ret -= term();
        req(top() == (k ? ')' : '\0'));
        return ret;
    }
   public:
    // 給定數學運算的字串，求其值。若格式不合法，丟出錯誤。
    static ll eval(const string& s) {
        // 若要禁止多重前置號，加上這四行
        // req(s.find("--") == -1); // 禁止多重負號
        // req(s.find("-+") == -1);
        // req(s.find("+-") == -1);
        // req(s.find("++") == -1);
        return Expr(s).expr(0);
    }
};
```

6.11 JosephusProblem

```cpp
// O(n) space and O(nlogn) time
#pragma GCC optimize("O3","unroll-loops")
#include <bits/extc++.h>
#include <bits/stdc++.h>
#define IO cin.tie(0), ios::sync_with_stdio(0)
#define ll long long
#define INF 0x3f3f3f3f
#define MAXN 100000
#define orderset tree<ll,null_type,less<ll>,rb_tree_tag,
    tree_order_statistics_node_update>
using namespace __gnu_pbds;
using namespace std;

int32_t main(){
    IO;
    int n, k ;
    cin >> n >> k;
    orderset s;
    for(int i=1;i<=n;i++)s.insert(i);
    int pos = 0;
    while(s.size()){
        pos = (pos + k) % s.size();
        auto it = s.find_by_order(pos);
        cout << *it << " ";
        s.erase(it);
    }
}

// O(1) space and O(n) time
#include <iostream>
using namespace std;
int Josephus(int N, int k){
    // Initialize variables i and ans with 1 and 0
    // respectively.
    int i = 1, ans = 0;
    // Run a while loop till i <= N
    while (i <= N) {
        // Update the Value of ans and Increment i by 1
        ans = (ans + k) % i;
        i++;
    }

    // Return required answer
    return ans + 1;
}

// main function
int main(){
    int N = 14, k = 2;
    cout << Josephus(N, k) << endl;
    return 0;
}



// O(n) space and O(n) time
#include <bits/stdc++.h>
using namespace std;

// Recursive function to implement the Josephus problem
int josephus(int n, int k)
{
    if (n == 1)
        return 1;
    else
        // The position returned by josephus(n - 1, k)
        // is adjusted because the recursive call
        // josephus(n - 1, k) considers the
        // original position k % n + 1 as position 1
        return (josephus(n - 1, k) + k - 1) % n + 1;
}

// Driver code
int main()
{
    int n = 14;
    int k = 2;
    cout << "The chosen place is " << josephus(n, k);
    return 0;
}


// asking last one O(klog(n))
//編號從1開始，結果要加1
int josephus(int n, int k) {
  if (k == 1) return n - 1;
  int ans = 0;
  for (int i = 2; i <= n; ) {
    if (ans + k >= i) {
```

```
      ans = (ans + k) % i;
      i++;
      continue;
    }
    int step = (i - 1 - ans - 1) / (k - 1); //向下取整
    if (i + step > n) {
      ans += (n - (i - 1)) * k;
      break;
    }
    i += step;
    ans += step * k;
  }
  return ans;
}

int main() {
    int n, k;
  while (scanf("%d%d", &n, &k) == 2)
    printf("%d\n", josephus(n, k) % n + 1);
  return 0;
}
```

7 String

7.1 Trie

```
class Trie {
private:
    struct Node {
        int cnt = 0, sum = 0;
        Node *tr[128] = {};
        ~Node() {
            for (int i = 0; i < 128; i++)
                if (tr[i]) delete tr[i];
        }
    };
    Node *root;
public:
    void insert(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) ptr->tr[*s] = new Node();
            ptr = ptr->tr[*s];
            ptr->sum++;
        }
        ptr->cnt++;
    }
    inline int count(char *s) {
        Node *ptr = find(s);
        return ptr ? ptr->cnt : 0;
    }
    Node *find(char *s) {
        Node *ptr = root;
        for (; *s; s++) {
            if (!ptr->tr[*s]) return 0;
            ptr = ptr->tr[*s];
        } return ptr;
    }
    bool erase(char *s) {
        Node *ptr = find(s);
        if (!ptr) return false;
        int num = ptr->cnt;
        if (!num) return false;
        ptr = root;
        for (; *s; s++) {
            Node *tmp = ptr;
            ptr = ptr->tr[*s];
            ptr->sum -= num;
            if (!ptr->sum) {
                delete ptr;
                tmp->tr[*s] = 0;
                return true;
            }
        }
    }
    Trie() { root = new Node(); }
    ~Trie() { delete root; }
};
```

7.2 AC 自動機

```
struct _AC{
  _AC *child[26];
  _AC *Fail;
  vector<pair<int,int>> out;
  _AC(){
    Fail = NULL;
    memset(child,0,sizeof(child));
  }
}*root;
void Insert_AC(string s){
  int n;
  _AC *p = root;
  for(auto&k:s){
    n = k - 'a';
    if(!p->child[n]) p->child[n] = new _AC();
    p = p->child[n];
  }
  p->out.push_back({s.size(),0});
}
void Construct_AC(){
  queue<_AC*> Q;
  for(int k=0;k<26;k++){
    if(root->child[k]){
      root->child[k]->Fail = root;
      Q.push(root->child[k]);
    }
    else root->child[k] = root;
  }
  _AC *p;
  while(!Q.empty()){
    p = Q.front();
    Q.pop();
    for(int k=0;k<26;k++){
      if(!p->child[k])p->child[k] = p->Fail->child[k];
      else {
        p->child[k]->Fail = p->Fail->child[k];
        for(auto&i:p->Fail->child[k]->out)p->child[k]->out.
            push_back({i.first,1});
        Q.push(p->child[k]);
      }
    }
  }
}
```

```
void Match_AC(string t){
  int n;
  _AC *p = root;
  for(int k=0;k<t.size();k++){
    n = t[k] - 'a';
    p = p->child[n];
    _AC *fail = p;
    while(fail != root && fail->out.size()){
      for(auto&i:fail->out)if(!i.second||fail->Fail->out.size
          ())cout<<t.substr(k-i.first+1,i.first)<<'\n';
      fail->out.clear();
      fail->Fail->out.clear();
      fail = fail->Fail;
    }
  }
}

int main(){
  int n,m;string p,t;cin>>n;
  while(cin>>n>>m){
    root = new _AC();
    while(n--){
      cin>>p;
      Insert_AC(p);
    }
    Construct_AC();
    cin>>n>>m;
    cin>>t;
    Match_AC(t);
  }
}
```

7.3 KMP

```
// KMP fail function.
int* kmp_fail(string& s) {
    int* f = new int[s.size()]; int p = f[0] = -1;
    for (int i = 1; s[i]; i++) {
        while (p != -1 && s[p + 1] != s[i]) p = f[p];
        if (s[p + 1] == s[i]) p++;
        f[i] = p;
    }
    return f;
}
// 問 sub 在 str 中出現幾次。
int kmp_count(string& str, string& sub) {
    int* fail = kmp_fail(sub); int p = -1, ret = 0;
    for (int i = 0; i < str.size(); i++) {
        while (p != -1 && sub[p + 1] != str[i]) p = fail[p];
        if (sub[p + 1] == str[i]) p++;
        if (p == sub.size() - 1) p = fail[p], ret++;
    }
    delete[] fail; return ret;
}
// 問 sub 在 str 第一次出現的開頭 index ，-1 表示找不到。
int kmp(string& str, string& sub) {
    int* fail = kmp_fail(sub);
    int i, j = 0;
    while (i < str.size() && j < sub.size()) {
        if (sub[j] == str[i]) i++, j++;
        else if (j == 0) i++;
        else j = fail[j - 1] + 1;
```

```
    }
    delete[] fail;
    return j == sub.size() ? (i - j) : -1;
}
```

7.4 LPS

```
int lps(string s){
    int N=2*s.size()+1;
    vector<int> dp(N);
    string s2="*";
    for(auto&c:s){
        s2.push_back(c);
        s2.push_back('*');
    }
    int C=0,R=0;
    for(int i = 1;i < N;i++){
        if(i>R)C=R=i;
        else{
            int mirrorI=C-(i-C);
            dp[i]=min(dp[mirrorI],R-i);
        }
        int j=dp[i]+1;
        while((i-j>=0)&&(i+j<N)&&(s2[i-j]==s2[i+j]))j++;
        dp[i]=j-1;
        if(i+dp[i]>R){
            C=i;
            R=i+dp[i];
        }
    }
    return *max_element(dp.begin(),dp.end());
}

string lps(string s){
    int N=2*s.size()+1;
    vector<int> dp(N);
    string s2="*";
    for(auto&c:s){
        s2.push_back(c);
        s2.push_back('*');
    }
    int C=0,R=0;
    for(int i = 1;i < N;i++){
        if(i>R)C=R=i;
        else{
            int mirrorI=C-(i-C);
            dp[i]=min(dp[mirrorI],R-i);
        }
        int j=dp[i]+1;
        while((i-j>=0)&&(i+j<N)&&(s2[i-j]==s2[i+j]))j++;
        dp[i]=j-1;
        if(i+dp[i]>R){
            C=i;
            R=i+dp[i];
        }
    }
    auto it=max_element(dp.begin(),dp.end());
    int maxLen=*it;
    int index=it-dp.begin();
    return s.substr((index-maxLen)/2,maxLen);
}
```

7.5 EditDistance

```
// 問從 src 到 dst 的最小 edit distance
// ins 插入一個字元的成本
// del 刪除一個字元的成本
// sst 替換一個字元的成本
ll edd(string& src, string& dst, ll ins, ll del, ll sst) {
    ll dp[src.size() + 1][dst.size() + 1]; // 不用初始化
    for (int i = 0; i <= src.size(); i++) {
        for (int j = 0; j <= dst.size(); j++) {
            if (i == 0) dp[i][j] = ins * j;
            else if (j == 0) dp[i][j] = del * i;
            else if (src[i - 1] == dst[j - 1])
                dp[i][j] = dp[i - 1][j - 1];
            else
                dp[i][j] = min(dp[i][j - 1] + ins,
                               min(dp[i - 1][j] + del,
                                dp[i - 1][j - 1] + sst));
        }
    }
    return dp[src.size()][dst.size()];
}
```

8 Geometry

8.1 angle

```
// 有序的點集合，計算每個內角角度
// 逆時針方向，從第一個點開始
// 0 <= angle <= 2*PI
vector<double> angle(const vector<pair<int,int>>& v){
    vector<double> ret;
    for(int i = 0; i < v.size(); i++){
        int x1 = v[(i-1+v.size())%v.size()].first - v[i].
             first;
        int y1 = v[(i-1+v.size())%v.size()].second - v[i].
             second;
        int x2 = v[(i+1)%v.size()].first - v[i].first;
        int y2 = v[(i+1)%v.size()].second - v[i].second;
        double a = atan2(x1, y1) - atan2(x2, y2);
        if(a < 0) a += 2*M_PI;
        ret.push_back(a);
        // ret = ret*180/M_PI // convert to degree
    }
    return ret;
}

// 計算三個點的內角角度
// 0 <= angle <= 2*PI
double angle(const pair<int,int>& a, const pair<int,int>& b,
     const pair<int,int>& c){
    int x1 = a.first - b.first;
    int y1 = a.second - b.second;
    int x2 = c.first - b.first;
    int y2 = c.second - b.second;
    double ret = atan2(x1, y1) - atan2(x2, y2);
    if(ret < 0) ret += 2*M_PI;
    return ret;
    // ret = ret*180/M_PI // convert to degree
}
```

```
}

// 求兩線交點(兩線必相交)
// 0 <= angle <= 2*PI
pair<double,double> intersection(const pair<double,double>&
     p1, const pair<double,double>& p2, const pair<double,
     double>& q1, const pair<double,double>& q2){
    double a1 = p2.second - p1.second;
    double b1 = p1.first - p2.first;
    double c1 = a1 * p1.first + b1 * p1.second;
    double a2 = q2.second - q1.second;
    double b2 = q1.first - q2.first;
    double c2 = a2 * q1.first + b2 * q1.second;
    double det = a1 * b2 - a2 * b1;
    return make_pair((b2 * c1 - b1 * c2) / det, (a1 * c2 - a2
         * c1) / det);
}
```

8.2 ConvexHull

```
void Dhull(vector<pii> points , vector<pii> &hull){
    hull.push_back(points[0]) , hull.push_back(points[1]);
    for(int i = 2 ; i < points.size() ; i++){
        while(hull.size() >= 2){
            pair<int,int> p1 = hull[hull.size()-2] , p2 =
                 hull[hull.size()-1] , p3 = points[i];
            int x1 = p2.first - p1.first , y1 = p2.second -
                 p1.second;
            int x2 = p3.first - p2.first , y2 = p3.second -
                 p2.second;
            if(x1*y2 - x2*y1 <= 0) break;
            hull.pop_back();
        }
        hull.push_back(points[i]);
    }
}
int main(){
    IO ;
    vector<pair<int,int>> points,hull;
    int n , x , y;
    cin >> n ;
    for(int i = 0 ; i < n ; i++) cin>>x>>y , points.pb({x,y})
         ;
    sort(points.begin() , points.end());
    Dhull(points,hull) ;
    reverse(points.begin(),points.end());
    Dhull(points,hull) ;
    for(auto p : hull) cout << p.first << " " << p.second <<
         endl;
}
```

8.3 旋轉卡尺

```
### 問題：最遠點對

考慮一個問題，你有一個凸多邊形，你想找到多邊形上距離最遠的兩
    個點。這可以通過選轉卡尺方法來解決。

### 方法：

```

1. **初始化**：選擇凸多邊形的一個頂點作為起點，將一條與多邊形的邊平行的輔助線放在該頂點上。

2. **旋轉**：開始將這條輔助線旋轉。對於多邊形的每一條邊，都會有一個時刻該邊成為輔助線上的最遠邊。這意味著在輔助線上，與這條邊的兩個端點的距離是最遠的。

3. **更新**：在每次旋轉時，記錄與起始頂點距離最遠的點。繼續旋轉輔助線，直到它再次與初始邊平行。

4. **換邊**：選擇下一個頂點，並重複上述步驟，直到所有頂點都被考慮過。

經過這些步驟，你就可以找到多邊形上距離最遠的兩個點。

分析：

因為是凸多邊形，所以當你旋轉輔助線時，與輔助線距離最遠的點只會在某個方向上增加。這使得你可以在O(n)的時間複雜度內解決這個問題，其中n是多邊形的頂點數。對於每個頂點，只需要考慮一次旋轉，並且對於每次旋轉，只需要O(1)的時間來更新最遠的點。

這只是使用選轉卡尺方法的一個例子。該方法還可以用於解決其他與凸多邊形相關的問題，如計算最大面積的內接矩形等。

```cpp
typedef pair<ll, ll> pii;
#define x first
#define y second
#define ii (i + 1) % n  // 打字加速！
inline pii operator-(const pii& a, const pii& b) {
    return {a.x - b.x, a.y - b.y};
}  // const 不可省略
inline ll operator*(const pii& a, const pii& b) {
    return a.x * b.y - a.y * b.x;
}
inline ll crzf(const pii& o, const pii& a, const pii& b) {
    return (a - o) * (b - o)
}
inline ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
// 給平面上任意個點，求其凸包。返回順序為逆時針。此方法會移除重複點。
#define jud \
    crzf(ret[ret.size() - 2], ret.back(), pp[i]) <= 0
vector<pii> makepoly(vector<pii>& pp) {
    int n = pp.size();
    sort(pp.begin(), pp.end());
    pp.erase(unique(pp.begin(), pp.end()), pp.end());
    vector<pii> ret;
    for (int i = 0; i < n; i++) {
        while (ret.size() >= 2 && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    for (int i = n - 2, t = ret.size() + 1; i >= 0; i--) {
        while (ret.size() >= t && jud) ret.pop_back();
        ret.push_back(pp[i]);
    }
    if (n >= 2) ret.pop_back();
    return ret;
}
// (shoelace formula)
// 給凸包，問其面積「的兩倍」。若凸包少於三個點，回傳零。
ll area(vector<pii>& poly) {
    int n = poly.size();
    ll ret = 0;
    for (int i = 0; i < n; i++)
        ret += (poly[i].x * poly[ii].y);
    for (int i = 0; i < n; i++)
        ret -= (poly[i].y * poly[ii].x);
    return ret;
}
// 給凸包，問其兩點最遠距離「的平方」。若要問平面上任意個點的兩點最遠
// 距離，請先轉成凸包。若凸包少於兩個點，回傳零。
#define kk (k + 1) % n
ll maxdist(vector<pii>& poly) {
    int k = 1, n = poly.size();
    if (n < 2) return 0;
    if (n == 2) return dd(poly[0], poly[1]);
    ll ret = 0;
    for (int i = 0; i < n; i++) {
        while (abs(crzf(poly[kk], poly[i], poly[ii])) >=
               abs(crzf(poly[k], poly[i], poly[ii])))
            k = kk;
        ret = max(ret, max(dd(poly[i], poly[k]),
                           dd(poly[ii], poly[k])));
    }
    return ret;
}
```

建立凸包 (makepoly 函數)
計算多邊形面積 (area 函數)
計算凸包上兩點之間的最遠距離 (maxdist 函數)
以下是對各部分的簡單解析：

1. 建立凸包 (makepoly 函數):
此函數的目的是為了找出一組點的凸包。算法首先對點進行排序，再進行上下凸殼的建立。在建立過程中，它利用叉積(crzf 函數)來確定三點之間的相對位置，並決定該點是否應該包含在凸殼內。

2. 計算多邊形面積 (area 函數):
此函數使用 shoelace formula (鞋帶公式) 來計算凸包的面積。公式基本上是計算所有連續點之間的叉積並將它們加在一起。正值和負值會相互抵銷，留下的就是多邊形的面積。

3. 計算凸包上兩點之間的最遠距離 (maxdist 函數):
這是用選轉卡尺技巧來解決的。對於凸包上的每一點，找到與該點和它的下一個點形成的直線最遠的點。由於凸性，最遠的點會沿著凸包的邊移動，所以可以使用選轉卡尺技巧來有效地找到最遠的點。

8.4 closestPair

最近點對問題是計算幾何中的另一個經典問題，目的是在平面上的給定點集中找到距離最近的一對點。

常見的算法是基於分治策略，具有O(n log n)的時間複雜度。以下是算法的基本思路：

1. 將點按照x坐標排序。
2. 將點集分成兩半。
3. 遞迴地找出每一半中的最近點對。令這兩個距離中的最小值為d。
4. 考慮中間縱帶的寬度為2d的所有點。這些點在y坐標上必須排序。
5. 對於帶中的每一點，檢查與其相鄰的點，直到y距離大於d。這部分的點對不會超過6個。
6. 如果在中間帶中找到距離小於d的點對，則更新d。

以下是該算法的簡化版本：

```cpp
struct Point {
    double x, y;

    double distanceTo(const Point& other) const {
        return sqrt((x - other.x) * (x - other.x) + (y -
            other.y) * (y - other.y));
    }
};

bool compareX(const Point &a, const Point &b) {
    return a.x < b.x;
}

bool compareY(const Point &a, const Point &b) {
    return a.y < b.y;
}

double closestPair(Point px[], Point py[], int n) {
    if (n <= 3) {
        double minDist = DBL_MAX;
        for (int i = 0; i < n; ++i)
            for (int j = i + 1; j < n; ++j)
                minDist = min(minDist, px[i].distanceTo(px[j
                    ]));
        return minDist;
    }

    int mid = n / 2;
    Point midPoint = px[mid];

    Point pyl[mid], pyr[n - mid];
    int li = 0, ri = 0;
    for (int i = 0; i < n; i++) {
        if (py[i].x < midPoint.x) pyl[li++] = py[i];
        else pyr[ri++] = py[i];
    }

    double dl = closestPair(px, pyl, mid);
    double dr = closestPair(px + mid, pyr, n - mid);
    double d = min(dl, dr);

    Point strip[n];
    int j = 0;
    for (int i = 0; i < n; i++)
        if (abs(py[i].x - midPoint.x) < d)
            strip[j++] = py[i];

    double minStrip = d;
    for (int i = 0; i < j; ++i)
        for (int k = i + 1; k < j && (strip[k].y - strip[i].y
            ) < minStrip; ++k)
```

```
            minStrip = min(minStrip, strip[i].distanceTo(
                strip[k]));

    return min(d, minStrip);
}

double closestPair(Point points[], int n) {
    Point px[n], py[n];
    for (int i = 0; i < n; i++) {
        px[i] = points[i];
        py[i] = points[i];
    }

    sort(px, px + n, compareX);
    sort(py, py + n, compareY);

    return closestPair(px, py, n);
}




typedef pair<ll, ll> pii;
#define x first
#define y second
ll dd(const pii& a, const pii& b) {
    ll dx = a.x - b.x, dy = a.y - b.y;
    return dx * dx + dy * dy;
}
const ll inf = 1e18;
ll dac(vector<pii>& p, int l, int r) {
    if (l >= r) return inf;
    int m = (l + r) / 2;
    ll d = min(dac(p, l, m), dac(p, m + 1, r));
    vector<pii> t;
    for (int i = m; i >= l && p[m].x - p[i].x < d; i--)
        t.push_back(p[i]);
    for (int i = m + 1; i <= r && p[i].x - p[m].x < d; i++)
        t.push_back(p[i]);
    sort(t.begin(), t.end(),
         [](pii& a, pii& b) { return a.y < b.y; });
    int n = t.size();
    for (int i = 0; i < n - 1; i++)
        for (int j = 1; j < 4 && i + j < n; j++)
            // 這裡可以知道是哪兩點是最小點對
            d = min(d, dd(t[i], t[i + j]));
    return d;
}
// 給一堆點，求最近點對的距離「的平方」。
ll closest_pair(vector<pii>& pp) {
    sort(pp.begin(), pp.end());
    return dac(pp, 0, pp.size() - 1);
}
```

首先，定義了一個pair來表示點，其中x和y分別代表x和y坐標。
dd函數是用來計算兩個點之間的距離的平方。
dac是主要的分治函數。給定一個點集p和兩個索引l和r，這個函數會返回這些點中距離最小的點對的距離平方。
如果l大於等於r，則返回inf，表示在這個區間沒有有效的點。
m是中點索引。
這個函數遞迴地計算左半部分和右半部分的最小距離。
t是一個臨時vector，它將存儲距中點在x軸上距離小於d的所有點。
根據x坐標從中點開始，左右兩邊選擇與中點在x軸上距離小於d的點。
接著，根據y坐標對t中的點進行排序。
然後對t中的每個點，只需要檢查接下來的三個點（如果它們存在），因為根據算法的性質，不可能有超過三個點在y軸上與當前點的距離小於d。
返回最小距離的平方。
closest_pair函數根據x坐標對所有點進行排序，然後調用dac函數。
這個程式碼的優點是它避免了計算所有點對之間的距離，從而將時間複雜度從O(n^2)降低到O(n log n)。

8.5 MinCircle

```
using PT = point<T>;
using CPT = const PT;
PT circumcenter(CPT &a, CPT &b, CPT &c) {
  PT u = b-a, v =c-a;
  T c1 = u.abs2()/2, c2 = v.abs2()/2;
  T d = u.cross(v);
  return PT(a.x+(v.y*c1-u.y*c2)/d, a.y+(u.x*c2-v.x*c1)/d);
}
void solve(PT p[], int n, PT &c, T &r2){
  random_shuffle(p,p+n);
  c = p[0]; r2 = 0; // c,r2 = 圓心,半徑平方
  for(int i=1; i<n; i++)
    if( (p[i]-c).abs2() > r2) {
      c=p[i]; r2=0;
      for(int j=0; j<i; j++)
        if( (p[j]-c).abs2() > r2) {
          c.x = (p[i].x+p[j].x)/2;
          c.y = (p[i].y+p[j].y)/2;
          r2 = (p[j]-c).abs2();
          for(int k=0; k<j; k++)
            if( (p[k]-c).abs2() > r2) {
              c = circumcenter(p[i], p[j], p[k]);
              r2 = (p[i]-c).abs2();
            }
        }
    }
}
```

最小覆蓋圓是指一個最小的圓，其中包含給定的所有點。這是一個經典的計算幾何問題。

解決這個問題的一種方法是 Welzl 的算法，這是一個遞迴算法，對於隨機化的輸入，其平均時間複雜度是 O(n)。然而，最壞情況下，該算法的運行時間為 \(O(n^4)\)。

以下是算法的基本想法和步驟：

1. **基本情況**：如果輸入點的集合是空的或已有 3 個點在邊界集合中，則問題可以直接求解。

2. **遞迴步驟**：從點集中移除一個點 `p`，遞迴地找到剩餘點的最小覆蓋圓。

3. 如果 `p` 已在圓內，那麼這個圓也是整個點集的最小覆蓋圓。

4. 如果 `p` 在圓外，那麼我們知道 `p` 必須在新的最小覆蓋圓的邊界上。這意味著最小覆蓋圓的邊界上至少有一個點是 `p`。我們可以遞迴地使用點 `p` 和邊界點集來解決這個問題。

要注意的是，Welzl 的算法通常在實踐中運行得非常快，但由於其遞迴的性質，它可能不是最適合大型實例的解決方案。

以下是該算法的精簡描述：

```cpp
Circle welzl(vector<Point> & P, list<Point> R = list<Point>()
    ) {
    if (P.empty() || R.size() == 3) {
        return trivial(R);
    }

    int idx = rand() % P.size();
    Point p = P[idx];
    P.erase(P.begin() + idx);

    Circle d = welzl(P, R);

    if (inside(d, p)) {
        return d;
    }

    R.push_back(p);
    return welzl(P, R);
}
```

在上述函數中，`Circle` 和 `Point` 分別表示圓和點的資料結構，`trivial` 是一個輔助函數，用於基本情況（例如當邊界集合中有 0、1、2 或 3 個點時），`inside` 是一個檢查點是否在圓內的函數。

這只是算法的高級描述。在實際實施時，還需要考慮如何表示圓和點，以及如何執行諸如計算圓或檢查點是否在圓內等基本操作。

用於計算給定點集中所有點的最小覆蓋圓。它主要基於增量算法，其中每次增加一個新點，我們都檢查這個點是否在當前的圓中。如果不在，我們就更新圓。

以下是程式碼的詳細解析：

點的定義：

PT 是點的資料結構。根據給定的片段，我們不知道 point 的完整定義，但我們可以推斷它有 x 和 y 屬性，以及一些函數方法，如 abs2（可能是點到原點的距離平方）。
外接圓的中心 (circumcenter 函數)：

這個函數計算三個點 a, b, 和 c 的外接圓的中心。給定三點，外接圓的中心可以用線性方程組解出。這裡用到了叉積來求解。
計算最小覆蓋圓 (solve 函數)：

算法從一個隨機點開始，初始化圓心為這個點，半徑為 0。
然後，對於每一個點，檢查它是否在當前的圓中。
如果點不在圓內，我們知道該點必須在新圓的邊界上。我們接著檢查所有之前的點。
如果之前的點也不在新圓內，則我們知道新圓的邊界上至少有這兩個點。此時，新圓的中心是這兩個點的中點。
然後，再次檢查所有之前的點。如果還有點不在新圓內，那麼我們知道新圓的邊界上有這三個點。在這種情況下，新圓的中心是這三個點的外接圓的中心。

該算法基於以下事實：最小覆蓋圓的邊界上有 1、2 或 3 個點。因此，當我們找到一個不在當前圓內的點時，我們知道它必須在新圓的邊界上。使用這個事實，我們可以增量地找到新的最小覆蓋圓。

8.6 MaximumBipartiteMatching

```cpp

// M is number of applicants
// and N is number of jobs
#define M 6
#define N 6

// A DFS based recursive function
// that returns true if a matching
// for vertex u is possible
bool bpm(bool bpGraph[M][N], int u,bool seen[], int matchR[])
    {
  // Try every job one by one
  for (int v = 0; v < N; v++){
    // If applicant u is interested in
    // job v and v is not visited
    if (bpGraph[u][v] && !seen[v]){
      // Mark v as visited
      seen[v] = true;
      // If job 'v' is not assigned to an
      // applicant OR previously assigned
      // applicant for job v (which is matchR[v])
      // has an alternate job available.
      // Since v is marked as visited in
      // the above line, matchR[v] in the following
      // recursive call will not get job 'v' again
      if (matchR[v] < 0 || bpm(bpGraph, matchR[v],seen,
          matchR)){
        matchR[v] = u;
        return true;
      }
    }
  }
  return false;
}

// Returns maximum number
// of matching from M to N
int maxBPM(bool bpGraph[M][N]){
  // An array to keep track of the
  // applicants assigned to jobs.
  // The value of matchR[i] is the
  // applicant number assigned to job i,
  // the value -1 indicates nobody is
  // assigned.
  int matchR[N];
  // Initially all jobs are available
  memset(matchR, -1, sizeof(matchR));
  // Count of jobs assigned to applicants
  int result = 0;
  for (int u = 0; u < M; u++){
    // Mark all jobs as not seen
    // for next applicant.
    bool seen[N];
    memset(seen, 0, sizeof(seen));
    // Find if the applicant 'u' can get a job
    if (bpm(bpGraph, u, seen, matchR)) result++;
  }
  return result;
}

int main(){
  bool bpGraph[M][N] = {{0, 1, 1, 0, 0, 0},
            {1, 0, 0, 1, 0, 0},
            {0, 0, 1, 0, 0, 0},
            {0, 0, 1, 1, 0, 0},
            {0, 0, 0, 0, 0, 0},
            {0, 0, 0, 0, 0, 1}};

  cout << "Maximum number of applicants that can get job is "
       << maxBPM(bpGraph);
}
```

8.7 Hungarian

```cpp
// Time: O(VE)
const int INF = 2e9;
const int N = ? ;       // 男女總人數；女 id: 0 ~ p，男 id: p+1 ~ N-1
int vis[N], rnd, m[N];  // 跑完匈牙利後配對結果儲存於此， -1 表示人醜
vector<int> g[N];       // 關係表
int dfs(int s) {
    for (int x : g[s]) {
        if (vis[x]) continue;
        vis[x] = 1;
        if (m[x] == -1 || dfs(m[x])) {
            m[x] = s, m[s] = x;
            return 1;
        }
    } return 0;
}
int hungarian(int p) {  // p : 女性人數
    memset(m, -1, sizeof(m));
    int c = 0;
    for (int i = 0; i < p; i++) {
        if (m[i] == -1) {
            memset(vis, 0, sizeof(vis));
            c += dfs(i);
        }
    } return c; // 成功結婚對數
}


// 匈牙利算法的優化，二分圖最大匹配 O(EVV)
int n, m, vis[maxn], level[maxn], pr[maxn], pr2[maxn];
vector<int> edge[maxn];  // for Left
bool dfs(int u) {
    vis[u] = true;
    for (vector<int>::iterator it = edge[u].begin();
         it != edge[u].end(); ++it) {
        int v = pr2[*it];
        if (v == -1 ||
            (!vis[v] && level[u] < level[v] && dfs(v))) {
            pr[u] = *it, pr2[*it] = u;
            return true;
        }
    } return false;
}
int hopcroftKarp() {
    memset(pr, -1, sizeof(pr));
    memset(pr2, -1, sizeof(pr2));
    for (int match = 0;;) {
        queue<int> Q;
        for (int i = 1; i <= n; ++i) {
            if (pr[i] == -1) level[i] = 0, Q.push(i);
            else level[i] = -1;
        }
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            for (vector<int>::iterator it = edge[u].begin();
                 it != edge[u].end(); ++it) {
                int v = pr2[*it];
                if (v != -1 && level[v] < 0)
                    level[v] = level[u] + 1, Q.push(v);
            }
        }
        for (int i = 1; i <= n; ++i) vis[i] = false;
        int d = 0;
        for (int i = 1; i <= n; ++i)
            if (pr[i] == -1 && dfs(i)) ++d;
        if (d == 0) return match;
        match += d;
    }
}
```

8.8 EulerCircuit

無向圖：如果恰有兩點的度數為奇數，則存在歐拉路徑，此二點分別為起終點；如果 全部的點度數都是偶數，則存在歐拉迴路。
2. 有向圖：如果恰有一點的出度等於入度+1、另有一點的入度等於出度+1，其餘皆入度 等於出度，則存在歐拉路徑，此二點分別為起終點；如果全部的點入度等於出度，則 存在歐拉迴路。

9 sortingAndSearching

9.1 NestedRangeCheck

```cpp
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
using namespace std;
#define int long long
typedef pair<int,int> pii;

int32_t main(){
    IO;
    int t, r;
    cin >> t;
    vector<int> a(t), b(t), aa(t), bb(t);
    vector<pii> v(t);
    map<pii,int> mp;

```

```
    for(auto &[l, r]: v){
        cin >> l >> r;
        r *= -1;
        mp[{l,-r}] = mp.size();
    }
    sort(v.begin(),v.end());

    for(auto &n: v) n.second *= -1 ;

    r = v[0].second;
    for(int i = 1; i < t; i++){
        if(v[i].second <= r) a[i] = 1;
        else if(v[i].second > r) a[i] = 0 , r = v[i].second;
        if(v[i-1] == v[i]) a[i-1] = a[i] = 1;
    }

    r = v.back().second;
    for(int i = v.size() - 2 ; i > -1 ; i--){
        if(v[i].second >= r) b[i] = 1;
        else if(v[i].second < r) b[i] = 0 , r = v[i].second;
        if(v[i] == v[i+1]) b[i+1] = b[i] = 1;
    }

    for(int i = 0; i < t; i++){
        aa[mp[v[i]]] = a[i];
        bb[mp[v[i]]] = b[i];
    }
    for(int i = 0; i < t; i++) cout << bb[i] << " " ;
    cout << endl ;
    for(int i = 0; i < t; i++) cout << aa[i] << " " ;
    cout << endl ;
}
```

9.2 NestedRangesCount

```
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define multiset tree<int, null_type,less_equal<int>,
     rb_tree_tag,tree_order_statistics_node_update>
/**
order_of_key(k) : nums strictly smaller than k
find_by_order(k): index from 0
**/
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
using namespace std;
#define int long long
typedef pair<int,int> pii;

int32_t main(){
    IO;
    int t, r;
    cin >> t;
    vector<int> a(t), b(t), aa(t), bb(t);
    vector<pii> v(t);
    map<pii,int> mp;

    for(auto &[l, r]: v){
        cin >> l >> r;
        r *= -1;
        mp[{l,-r}] = mp.size();
    }
    sort(v.begin(),v.end());

    for(auto &n: v) n.second *= -1 ;

    multiset cnt;
    r = v[0].second;
    for(int i = 0; i < t; i++){
        cnt.insert(v[i].second);
        a[i] = cnt.size() - cnt.order_of_key(v[i].second)-1;
    }

    cnt.clear();
    r = v.back().second;
    for(int i = v.size() - 1 ; i > -1 ; i--){
        cnt.insert(v[i].second);
        b[i] = cnt.order_of_key(v[i].second+1)-1;
    }

    for(int i = 0; i < t; i++){
        aa[mp[v[i]]] = a[i];
        bb[mp[v[i]]] = b[i];
    }
    for(int i = 0; i < t; i++) cout << bb[i] << " " ;
    cout << endl ;
    for(int i = 0; i < t; i++) cout << aa[i] << " " ;
    cout << endl ;
}
```

9.3 SubarrayDistinctVal

```
Given an array of n integers, your task is to calculate the
     number of subarrays that have at most k distinct values.

Input
The first input line has two integers n and k.

Output
Print one integer: the number of subarrays.

Input:
5 2
1 2 3 1 1
Output:
10
```

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
using namespace std;
#define int long long

int32_t main(){
    int n, t, l, c = 0, ans = 0; cin >> n >> t;
    vector<int> v(n);
    map<int,int> mp;
    for(auto&n:v)cin>>n;
    l = 0;
    for(int i = 0 ; i < n ;i++){
        if(!mp[v[i]]) c ++;
        mp[v[i]] ++;
        while(c > t){
            mp[v[l]] -- ;
            if(!mp[v[l]]) c--;
            l++ ;
        }
        ans += i - l + 1;
    }
    cout << ans << endl ;
}
```

9.4 missingCoinSum

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
using namespace std;
typedef long long ll;

int main(){
    IO;
    ll n ; cin >> n ;
    vector<ll> v(n) ;
    for(auto&x:v) cin >> x ;
    sort(v.begin(),v.end());
    ll i=v[0] , j = 1 ;
    if(i>1){
        cout << 1 << endl ;
        return 0 ;
    }
    for(;j<n;j++){
        if(v[j]>i+1) break;
        i+=v[j];
    }
    cout << (ll)(i+1) << endl ;
}
```

9.5 sumOfFourValues

```
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
using namespace std;
#define int long long
#define pii pair<int,int>

int32_t main(){
    int n, t; cin >> n >> t;
    vector<int> v(n);
    for(auto&n:v) cin >> n;
    map<int,pii> mp;
    for(int i = 0; i < n; i++){
        for(int j = i+1; j < n; j++){
            if(mp.count(t-v[i]-v[j])){
                cout << mp[t-v[i]-v[j]].first+1 << " " << mp[
                    t-v[i]-v[j]].second+1 << " " << i+1 << "
                     " << j+1 << endl;
                return 0;
            }
        }
        for(int j = 0; j < i; j++)mp[v[i]+v[j]] = {i,j};

    }
```

```
    cout << "IMPOSSIBLE" << endl;
}

// 3 val
#include <bits/stdc++.h>
#pragma GCC optimize("O3","unroll-loops")
#define IO cin.tie(0), ios::sync_with_stdio(0)
#define int long long
using namespace std;
typedef pair<int,int> pii;

int ans = 0;
vector<pii> v;

int binarySearch(int l, int r, int target){
    l++ , r--;
    while(l<=r){
        int mid = (l+r)/2;
        if(v[mid].first == target){
            ans = 1;
            return v[mid].second;
        }
        else if(v[mid].first < target) l = mid+1;
        else r = mid-1;
    }
    return -1;
}

int32_t main(){
    int n, t, l, r, m; cin >> n >> t;
    v.resize(n);
    for(int i = 0; i < n ; i++){
        cin >> v[i].first;
        v[i].second = i+1;
    }
    sort(v.begin(),v.end());

    for(l = 0; l < n; l++)
        for(r = l+1; r < n; r++){
            m = binarySearch( r, n, t - v[l].first - v[r].
                first);
            if(ans){
                cout << v[l].second << " " << m << " " << v[r
                    ].second << endl;
                return 0;
            }
        }
    cout << "IMPOSSIBLE\n";
}
```

10 Other

10.1 精度

```
from decimal import*
getcontext().prec = 1000000
n = Decimal(input())
```

10.2 inversion

```
// 1
int inversion(){
    ll Length,now, ans = 0;
    Multiset place;
    cin >> Length ;
    for(int i = 0 ; i < Length ; i++){
        cin >> now ;
        place.insert(now) ;
        ans +=  i - place.order_of_key(now) ;
    }
    return ans ;
}
// 2
void Merge(int a[],int b,int m,int e){
    int ap = b,bp=m+1,cp=0,sum=0;
    static int c[MAX];
    while(ap<=m&&bp<=e){
        if(a[ap]<=a[bp])c[cp++]=a[ap++],ans+=sum;
        else c[cp++]=a[bp++],sum++;
    }
    while(ap<=m)c[cp++]=a[ap++],ans+=sum;
    while(bp<=e)c[cp++]=a[bp++];
    for(int k = 0;k < cp;k++)a[b+k]=c[k];
}
void Merge_Sort(int a[],int b,int e){
    if(e-b<1)return;
    int m =(b+e)/2;
    Merge_Sort(a,b,m);
    Merge_Sort(a,m+1,e);
    Merge(a,b,m,e);
}
```

10.3 離散化

```
vector<int> v(1000),b(1000);
for(auto&n:v)cin>>n;
sort(v.begin(),v.end());
auto len = unique(v.begin(),v.end())-v.begin();
v.resize(len);
for(int i = 0; i < v.size(); i++){
    b[i] = lower_bound(v.begin(),v.end(),v[i])-v.begin();
}
```

10.4 Convert

```
stoi
stoll
to_string
```

10.5 minSwap

```
int minSwaps(vector<int> &arr, int n) {
    pii arrPos[n];
    for (int i = 0; i < n; i++) {
        arrPos[i].first = arr[i];
        arrPos[i].second = i;
    }
    sort(arrPos, arrPos + n);
    vector<bool> vis(n, false);
    int ans = 0;
    for (int i = 0; i < n; i++) {
        if (vis[i] or arrPos[i].second == i) continue;
        int cycle_size = 0 , j = i ;
        while (!vis[j]){
            vis[j] = 1;
            j = arrPos[j].second;
            cycle_size++;
        }
        if(cycle_size > 0) ans += (cycle_size - 1);
    }
    return ans;
}
```

10.6 莫隊

```
#include"bits/stdc++.h"
using namespace std;

#define ll long long
#define pii pair<int, int>
#define ql first.first
#define qr first.second
#define id second

int N, Q, K;
vector<int> A;
vector<pair<pii, int>> qrys;
ll ans[200007];

void init(){
  cin >> N >> Q;
  K = N / max((int)sqrt(Q), 1); // 塊的大小

  A.clear();A.resize(N);
  qrys.clear();qrys.resize(Q);

  for(auto&i : A)cin >> i;
  int cnt = 0;
  for(auto&i : qrys){
    cin >> i.ql >> i.qr;
    --i.ql;--i.qr; // 轉換成 0-base
    i.id = cnt++;
  }
}

void solve(){
  sort(qrys.begin(), qrys.end(),
  [&](const pair<pii, int> &a, const pair<pii, int> &b){
    if(a.ql/K == b.ql/K) // 左界塊相同，按照右界塊排序
      return a.qr < b.qr;
    return a.ql/K < b.ql/K; // 否則，按照左界塊排序
  });

  int l = 0, r = -1; // 初始答案窗口
  ll sum = 0;
```

```cpp

  for(auto&i : qrys){
    while(l > i.ql){ l--; sum += A[l]; } // 延伸左界
    while(r < i.qr){ r++; sum += A[r]; } // 延伸右界
    while(l < i.ql){ sum -= A[l]; l++; } // 內縮左界
    while(r > i.qr){ sum -= A[r]; r--; } // 內縮右界
    ans[i.id] = sum;
  }

  for(int i = 0; i < Q; i++) cout << ans[i] << '\n';
}

int main(){
  init();
  solve();
}
```

MCU-NL Codebook

Contents