Contents

1 Basic

1.1 .vimrc

```
filetype indent on
syntax enable
set nu
set cursorline
set ts=2 sts=2 sw=2 et ai
set mouse=a
set wrap
set showcmd
set backspace=indent,eol,start

inoremap ( ()<ESC>i
inoremap [ []<ESC>i
inoremap {<CR> {<CR>}<ESC>ko
```

2 Dynamic Programming

2.1 0/1 Knapsack_problems

```cpp
#include<bits/stdc++.h>
using namespace std;
int f[1000]={0};
int n=0, m=0;
int main(){
    cin >> n >> m;
    for (int i = 1;i <= n;i++){
        int price = 0, value = 0;
        cin >> price >> value;
        for (int j = m;j >= price;j--){
            if (f[j-price]+value>f[j]){
                f[j]=f[j-price]+value;
            }
        }
    }
    cout << f[m] << endl;
    return 0;
}
```

2.2 Complete_Knapsack_problems

```cpp
#include<bits/stdc++.h>
using namespace std;
int f[1000]={0};
int n=0, m=0;
int main(){
    cin >> n >> m;
    for (int i=1;i<=n;i++){
        int price=0, value=0;
        cin >> price >> value;
        for (int j=price; j<=m; j++){
            if (f[j-price]+value>f[j]){
                f[j]=f[j-price]+value;
            }
        }
    }
    cout << f[m] << endl;
    return 0;
}
```

2.3 Longest Common Subsequence(LCS)

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

int dp[1001][1001];
int lcs(const string &s, const string &t){
    int m = s.size(), n = t.size();
    if (m == 0 || n == 0){
        return 0;
    }
    for(int i = 0; i <= m; ++i){
        dp[i][0] = 0;
    }
    for(int j = 1; j <= n; ++j){
        dp[0][j] = 0;
    }
    for(int i = 0; i < m; ++i){
        for (int j = 0; j < n; ++j){
            if(s[i] == t[j]){
                dp[i+1][j+1] = dp[i][j]+1;
            }else{
                dp[i+1][j+1] = max(dp[i+1][j],
                    dp[i][j+1]);
            }
        }
    }
    return dp[m][n];
}
```

2.4 Longest increasing common sequence(LICS)

```cpp
#include<bits/stdc++.h>
using namespace std;
int a[100] = {0};
int b[100] = {0};
```

```
int f[100] = {0};
int n = 0,m = 0;
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> a[i];
    }
    cin >> m;
    for(int i = 1;i <= m;i++){
        cin >> b[i];
    }
    for(int i = 1;i <= n;i++){
        int k = 0;
        for (int j = 1;j <= m;j++){
            if(a[i] > b[j] && f[j] > k){
                k = f[j];
            }else if(a[i] == b[j] && k + 1 > f[j]){
                f[j] = k + 1;
            }
        }
    }
    int ans=0;
    for(int i = 1;i <= m;i++){
        if(f[i] > ans){
            ans = f[i];
        }
    }
    cout << ans << endl;
    return 0;
}
```

2.5 Longest Increasing Subsequence(LIS)

```
#include<bits/stdc++.h>
using namespace std;
int n=0;
int a[100]={0}, f[100]={0}, x[100]={0};
int main(){
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> a[i];
        x[i] = INT_MAX;
    }
    f[0]=0;
    int ans=0;
    for(int i = 1;i <= n;i++){
        int l = 0,r = i;
        while (l+1<r){
            int m=(l+r)/2;
            if (x[m]<a[i]){
                l=m;
            }else{
                r=m;
            }
            // change to x[m]<=a[i] for
                non-decreasing case
        }
        f[i]=l+1;
        x[l+1]=a[i];
        if(f[i]>ans){
            ans=f[i];
        }
    }
    cout << ans << endl;
    return 0;
}
```

2.6 Longest Palindromic Subsequence(LPS)

```
int LPS(vector<vector<int>>& dp, string& s, int l,
    int r){
    if(~dp[l][r]) return dp[l][r];
    if(l == r) return dp[l][r] = 1;
    if(l > r) return dp[l][r] = 0;
    if(s[l] == s[r]) return dp[l][r] = LPS(dp, s,
        l+1, r-1) + 2;
    return dp[l][r] = max(LPS(dp, s, l + 1, r),
        LPS(dp, s, l, r - 1));
}
```

3 Graph Theory

3.1 Lowest Common Ancestor(LCA)

```
#include<bits/stdc++.h>
using namespace std;
const int LOG = 20;
int par[N][LOG];
int tin[N], tout[N];
int timer = 0;
void dfs(int v, int p){
    tin[v] = ++timer;
    par[v][0] = p;
    for (int it : G[v]){
        if (it != p){
            dfs(it, v);
        }
    }
    tout[v] = ++timer;
}
void Doubling(){
    for (int i = 1; i < N; ++i){
        for (int j = 1; j < LOG; ++j){
            par[i][j] = par[par[i][j - 1]][j - 1];
        }
    }
}
bool anc(int v, int u){
    return tin[v] <= tin[u] && tout[u] <= tout[v];
}
int LCA(int v, int u){
    if (anc(v, u)){
        return v;
    }
    for (int j = LOG - 1; j >= 0; --j){
        if (!anc(par[v][j], u)){
            v = par[v][j];
        }
    }
    return par[v][0];
}
```

3.2 bellman-ford

```
void bellman(vector<edge>& edges, vector<int>& dist,
    int n){
    // n - 1 relax
    for(int i = 1; i < n; i++){
        for(edge e : edges){
            int u = e.start, v = e.end, d = e.dist;
            if(dist[u] == INF) continue;
            dist[v] = min(dist[v], d + dist[u]);
        }
    }

    // 偵測負權迴路
    for(edge e : edges){
        int u = e.start, v = e.end, d = e.dist;
        if(dist[u] == INF) continue;
        if(dist[v] > dist[v] + d){
            printf("It is contained negative
                cycle.\n");
            break;
        }
    }
}
```

3.3 dijkstra

```cpp
void dijkstra(ll st){
    vector<ll> dis(n, INF);
    //pll (vetrex, distance)
    priority_queue<pll, vector<pll>, greater<pll>> pq;
    pq.push({st, 0});
    dis[st] = 0;
    while(!pq.empty()){
        pll now = pq.top();
        pq.pop();
        if(now.ss != dis[now.ff]) continue;
        for(pll i : graph[t.ff]){
            if(dis[now.ff] + i.ss < dis[i.ff]){
                dis[i.ff] =  now.ss + i.ss;
                pq.push({i.ff, dis[i.ff]});
            }
        }
    }
}
```

3.4 Topological

```cpp
//此處為建立 Adjacency List 和每個節點的入度點數量
vector<vector<int>> make(vector<Edge>& nodes, int n){
  vector<vector<int>> graph(n + 1);
  vector<int> indegree(n, 0);
  for(auto node : nodes){
    graph[node.src].pb(node.des);
    indegree[node.des] ++; //計算入度點
  }
  graph[n] = indegree;
  return graph;
}

// 拓樸排序
vector<int> TolpologicalOrder(vector<vector<int>>
    graph){
  int n = graph.size();
  queue<int> q;
  vector<int> result;
  for(int i=0; i<graph[n - 1].size(); i++){
    if(!graph[n - 1][i]) q.push(i);
  }
  while(!q.empty()){
    int cnt = q.front();
    result.pb(cnt);
    q.pop();
    for(int i=0; i<graph[cnt].size(); i++){
        graph[n - 1][graph[cnt][i]]--;
        if(!graph[n - 1][graph[cnt][i]])
            q.push(graph[cnt][i]);
    }
  }

    //偵測循環
  for(auto i : graph[n-1]) if(i) return {};

  return result;
}
```

3.5 kruskal

```cpp
struct Edge{
    int a, b, c;
}edge[M];

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

// 要先按照邊權重升序
void kruskal(){
    int ans = 0;
    for (int i = 0; i < m; i ++ ){
        int a = edge[i].a, b = edge[i].b, c =
            edge[i].c;

        int pa = find(a), pb = find(b); // 找節點 A
            與節點 B 所屬的集合

        if (pa != pb){  // 若為不同集合，就合併
            p[pb] = pa;
            cnt ++;
            ans += c;
        }
    }

    if (cnt == n - 1) cout << ans << endl; //
        邊的數量不為 n - 1，則沒有 MST
    else puts("impossible");
}
```

3.6 prime

```cpp
// O(N^2) 類似 dijkstra
void prim(){
    memset(dis, 0x3f, sizeof dis);
    dis[1] = 0; st[1] = 1;  // 抓個初始點
    int ans = 0; // 記錄邊上權重

    for (int i = 0; i < n; i ++ ){
        int t = -1;
        for (int j = 1; j <= n; j ++ ){
            if (!st[j] && (t == -1 || dis[j] <
                dis[t])) t = j; // 尋找距離最近的節點
        }

        if (dis[t] == INF) { // 最短距離為
            INF，則不存在 MST
            puts("impossible");
            return;
        }

        ans += dis[t];
        st[t] = 1; // 將節點加入 MST

        for (int j = 1; j <= n; j ++ ){
            if (!st[j]){
                dis[j] = min(dis[j], dis[t] +
                    g[t][j]); // 更新其他節點到 MST
                    的距離
            }
        }
    }

    return ans;
}
```

4 Algorithm

4.1 Ternary Search

```cpp
int l = -10000;
int r = 10000;
int iterations = 100;
for (int i = 0; i < iterations; i++){
    double mr = (l + r) / 2.0;
    double ml = (l + mr) / 2.0;
    // f( ): 目標函數
    if (f(ml) < f(mr)) r = mr;
    else l = ml;
}
```

5 Number Theory

5.1 質數篩法 Sieve of Eratosthenes

```cpp
bool a[46342];
vector <int> v;
for (int j = 2; j < 46342; j++){
    if (!a[j]){
        v.push_back(j);
        for (int i = j * j; i < 46342; i += j){
            a[i] = true;
        }
    }
}
```

5.2 擴展歐幾里得

```cpp
pair<int,int> extgcd(int a, int b){
    if (b==0) return {1,0};
    int k = a/b;
    pair<int,int> p = extgcd(b,a-k*b);
    return { p.second, p.first - k*p.second };
}
```

5.3 快速冪

```cpp
constexpr int mod = 1e9 + 7;
ll fpow(ll i, int j) {
    ll ret = 1, tmp = i;
    for (; j; j >>= 1, tmp = tmp * tmp % mod)
        if (j & 1) ret = ret * tmp % mod;
    return ret;
}
```

5.4 大質數

```
97, 101, 131, 487, 593, 877, 1087, 1187, 1487, 1787,
    3187, 12721,
13331, 14341, 75577, 123457, 222557, 556679, 999983,
1097774749, 1076767633, 100102021, 999997771,
1001010013, 1000512343, 987654361,999991231,
999888733, 98789101, 987777733,999991921,1000000007,
1000000087, 1000000123,1010101333, 1010102101,
1000000000039,1000000000000037, 2305843009213693951,
4611686018427387847,9223372036854775783,
18446744073709551557
```

6 Data Structure

6.1 Disjoint Set Union-Find

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

vector<int> dsu, rk;

void initDSU(int n){
    dsu.resize(n);
    rk.resize(n);
    for(int i = 0; i < n; i++) dsu[i] = i, rk[i] = 1;
}

int findDSU(int x){
    if(dsu[x] == x) return x;
    dsu[x] = findDSU(dsu[x]);
    return dsu[x];
}

void unionDSU(int a, int b){
    int pa = findDSU(a), pb = findDSU(b);
    if(rk[pa] > rk[pb]) swap(pa, pb);
    if(rk[pa] == rk[pb]) rk[pb]++;
    dsu[pa] = pb;
}
```

6.2 Segment Tree

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

struct segtree {

  vector<ll> sums;
  ll size;

  // 線段樹初始化
  void init(ll n){
    size = 1;
    while(size < n) size<<1;
    sums.assign(size<<1, 0LL);
  }

  // 更新數值
  void update(ll i, ll v, ll x, ll Lptr, ll Rptr){
    if(Rptr - Lptr == 1){
      sums[x] = v;
      return;
    }
    ll m = ( Lptr + Rptr )/2;
    if(i<m) update(i, v, 2*x+1, Lptr, m);
    else update(i, v, 2*x+2, m, Rptr);
    sums[x] = sums[2*x+1] + sums[2*x+2];
  }

  void update(ll a, ll b){
    update(a, b, 0, 0, size);
  }

  // 查詢資訊
  ll query(ll l, ll r, ll x, ll Lptr, ll Rptr){
    if( Lptr >= r || Rptr <= l ) return 0;
    if( Lptr >= l && Rptr <= r ) return sums[x];
    ll m = (Lptr + Rptr) /2;
    ll s1 = query(l, r, 2*x+1, Lptr, m);
    ll s2 = query(l, r, 2*x+2, m, Rptr);
    return s1 + s2;
  }

  ll query(ll a, ll b){
    return query(a, b, 0, 0, size);
  }
};
```

6.3 Segment Tree Lazy

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define MAXN (int)(5e5 + 10)

vector<int> lazy(4 * MAXN, 0);
void build_tree(vector<int>& arr, vector<int>& tree,
    int node, int start, int end) {
  if ( start == end ) {
    tree[node] = arr[start];
    return;
  }
  int mid = ( start + end ) >> 1;
```

```
  int left_node  = 2 * node + 1;
  int right_node = 2 * node + 2;
  build_tree(arr, tree, left_node, start, mid);
  build_tree(arr, tree, right_node, mid + 1, end);
  tree[node] = tree[left_node] + tree[right_node];
  return;
}

void lazy_node(vector<int>& tree, int node, int
    start, int end) {
  if ( lazy[node] == 0 ) return;

  tree[node] += lazy[node] * ( end - start + 1 );
  int left_node  = 2 * node + 1;
  int right_node = 2 * node + 2;
  if ( start != end ) {
    lazy[left_node]  += lazy[node];
    lazy[right_node] += lazy[node];
  }
  lazy[node] = 0;
  return;
}

void update(vector<int>& arr, vector<int>& tree, int
    node, int start, int end, int l, int r, int k) {
  lazy_node(tree, node, start, end);
  if ( start > r || end < l ) return;
  int mid = ( start + end ) >> 1;
  int left_node  = 2 * node + 1;
  int right_node = 2 * node + 2;
  if ( start >= l && end <= r ) {
    tree[node] += k * (end - start + 1);
    if ( start != end ) {
      lazy[left_node]  += k;
      lazy[right_node] += k;
    }
    return;
  }
  update(arr, tree, left_node, start, mid, l, r, k);
  update(arr, tree, right_node, mid + 1, end, l, r,
      k);
  tree[node] = tree[left_node] + tree[right_node];
  return;
}
int query_tree(vector<int>& arr, vector<int>& tree,
    int node, int start, int end, int l, int r) {
  if ( start > r || end < l ) return 0;

  lazy_node(tree, node, start, end);

  int mid = ( start + end ) >> 1;
  int left_node  = 2 * node + 1;
  int right_node = 2 * node + 2;

  if ( start >= l && end <= r  ) {
    return tree[node];
  }

  return query_tree(arr, tree, left_node, start, mid,
      l, r) + query_tree(arr, tree, right_node, mid +
      1, end, l, r);
}

void solve(){
  int n;
  cin >> n;
  vector<int> arr(MAXN), tree(4 * MAXN);
  for(int i = 0; i < n; i++) cin >> arr[i];
  build_tree(arr, tree, 0, 0, n - 1);
  int q, v, l, r, k;
  cin >> q;
  while(q--) {
    cin >> v >> l >> r;
    if ( v == 1 ) {
      cin >> k;
      update(arr, tree, 0, 0, n - 1, l - 1, r - 1, k);
    } else if ( v == 2 ) {
      cout << query_tree(arr, tree, 0, 0, n - 1, l -
          1, r - 1) << "\n";
    }
  }
  return;
}
signed main(){
  solve();
  return 0;
}
```

6.4 PBDS

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

#define int long long

signed main () {
  ordered_set<int> os;

  int k = 1, x = 5;

  os.insert(x); // O(log n)
  os.erase(x);  // O(log n)
  // os.count(x); order_set 沒有 count 方法
  os.find(x); // O(log n)

  // ordered_set 獨有的操作
  // 1. 查找第k小
  int val = *os.find_by_order(k); // O(log n)
  // 2. 求排名
  int order = os.order_of_key(x); // O(log n)

  // set 需要遍歷才能實現上述功能
  set<int> s;
  auto it = next(s.begin(), k);  // O(n)
  distance(s.begin(), s.lower_bound(x));  // O(n)
}
```

7 String

7.1 Suffix Array

```
#include<bits/stdc++.h>
#define int long long

using namespace std;

void count_sort(auto &p, auto &c){
  int n = p.size();
  vector<int> cnt(n);
  for(auto el : c) cnt[el] ++;
  vector<int> p_new(n), pos(n);
  pos[0] = 0;
  for(int i=1;i<n;i++) pos[i] = pos[i-1] + cnt[i-1];
  for(auto el : p){
    int i = c[el];
    p_new[pos[i]] = el;
    pos[i] ++;
  }
  p = p_new;
}

signed main(){
  string s;
```

```
  cin>>s;
  s += "$";
  int n = s.size();
  vector<pair<char, int>> v(n);
  vector<int> p(n), c(n);
  for(int i=0;i<n;i++) v[i] = {s[i], i};
  sort(v.begin(), v.end());

  for(int i=0;i<v.size();i++) p[i] = v[i].second;
  c[p[0]] = 0;
  for(int i=1;i<v.size();i++){
    if(v[i].first == v[i-1].first) c[p[i]] =
        c[p[i-1]];
    else c[p[i]] = c[p[i-1]] + 1;
  }

  int k = 0;
  while((1 << k) < n){
    for(int i=0;i<n;i++) p[i] = (p[i] - (1 << k) + n)
        % n;
    count_sort(p, c);

    vector<int> c_new(n);
    c_new[p[0]] = 0;
    for(int i=1;i<v.size();i++){
      pair<int, int> prev = {c[p[i-1]], c[(p[i-1] +
          (1  << k)) % n]};
      pair<int, int> now = {c[p[i]], c[(p[i] + (1  <<
          k)) % n]};
      if(prev == now) c_new[p[i]] = c_new[p[i-1]];
      else c_new[p[i]] = c_new[p[i-1]] + 1;
    }
    c = c_new;
    k++;
  }
  for(int i=0;i<n;i++) cout<<p[i]<<"\n";
}
```

7.2 Suffix Array LCP

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

vector<int> lcp(n);
int k = 0;
for(int i=0;i<n-1;i++){
    int pi = c[i];
    int j = p[pi - 1];
    while(s[i+k] == s[j+k]) k++;
    lcp[pi] = k;
    k = k-1 > 0 ? k-1 : 0;
}
```

7.3 KMP algorithm

```
vector<int> NEXT;
void getNext(string p){
    int i = 1, j = i - 1;
    while(i < p.size()){
        if(p[i] == p[j]){
            NEXT[i++] = ++j;
        }
        else if(j <= 0){
            NEXT[i++] = 0;
        }
        else{
            j = NEXT[j - 1];
        }
    }
}

int KMP(string s, string p){
  int i = 0, j = 0;
  while(i < s.size() && j < p.size()){
    if(s[i] == p[j]){
      ++i;
      ++j;
    }else if(j <= 0){
      ++i;
    }else{
      j = NEXT[j - 1];
    }
  }

  if(j >= p.size()) return i - p.size();
  else return -1;
}
```

7.4 Manachar's algorithm

```
int P[SIZE * 2];

string convertToNewString(const string &s) {
    string newString = "@";

    for (int i = 0; i < s.size(); i++) {
        newString += "#" + s.substr(i, 1);
    }

    newString += "#$";
    return newString;
}

string longestPalindromeSubstring(const string &s) {
    string Q = convertToNewString(s);
    int c = 0, r = 0;                // current
        center, right limit

    for (int i = 1; i < Q.size() - 1; i++) {
        // find the corresponding letter in the
            palidrome subString
        int iMirror = c - (i - c);

        if(r > i) {
            P[i] = min(r - i, P[iMirror]);
        }

        // expanding around center i
        while (Q[i + 1 + P[i]] == Q[i - 1 - P[i]]){
            P[i]++;
        }

        // Update c,r in case if the palindrome
            centered at i expands past r,
        if (i + P[i] > r) {
            c = i;               // next center = i
            r = i + P[i];
        }
    }

    // Find the longest palindrome length in p.

    int maxPalindrome = 0;
    int centerIndex = 0;

    for (int i = 1; i < Q.size() - 1; i++) {

        if (P[i] > maxPalindrome) {
            maxPalindrome = P[i];
            centerIndex = i;
        }
    }

    cout << maxPalindrome << "\n";
    return s.substr( (centerIndex - 1 -
        maxPalindrome) / 2, maxPalindrome);
}
```

7.5 Z-algorithm

```cpp
void z_build(const char *S, int *Z) {
    Z[0] = 0;
    int bst = 0;
    for(int i = 1; S[i]; i++) {
        if(Z[bst] + bst < i) Z[i] = 0;
        else Z[i] = min(Z[bst]+bst-i, Z[i-bst]);
        while(S[Z[i]] == S[i+Z[i]]) Z[i]++;
        if(Z[i] + i > Z[bst] + bst) bst = i;
    }
}
```

8 Flow

8.1 dinic

```cpp
struct edge{
    int u, v, c, f;
    edge(int u, int v, int c, int f):u(u), v(v),
        c(c), f(f){}
};
vector<edge>e;
vector<int> G[maxn];
int level[maxn]; // 紀錄每個點的層數
int iter[maxn]; // 目前弧優化
int m;
void init(int n){
    for(int i = 0; i <= n; i++)G[i].clear();
    e.clear();
}
void addedge(int u, int v, int c){
    e.push_back(edge(u, v, c, 0));
    e.push_back(edge(v, u, 0, 0));
    m = e.size();
    G[u].push_back(m - 2);
    G[v].push_back(m - 1);
}

void BFS(int s){
    memset(level, -1, sizeof(level));
    queue<int> q;
    level[s] = 0;
    q.push(s);
    while(!q.empty()){
        int u = q.front();
        q.pop();
        for(int v = 0; v < G[u].size(); v++){
            edge& now = e[G[u][v]];
            if(now.c > now.f && level[now.v] < 0){
                level[now.v] = level[u] + 1;
                q.push(now.v);
            }
        }
    }
}

// 尋找增廣路徑
int dfs(int u, int t, int f){
    if(u == t) return f;

    // 用 iter 表示每個節點目前的弧，防止多次遍歷
    for(int &v = iter[u]; v < G[u].size(); v++){
        edge &now = e[G[u][v]];

        // now.c - now.f > 0 表示此路未清空
        // level[u] < level[now.v] 表示這條路是最短路
        if(now.c - now.f > 0 && level[u] <
            level[now.v]) {
            int d = dfs(now.v, t, min(f, now.c -
                now.f));
            if(d > 0) {
                now.f += d; // 正向邊流量 +d
                e[G[u][v] ^ 1].f -= d; // 反向邊 -d
                return d;
            }
        }
    }
    return 0;
}

int Maxflow(int s, int t){
    int flow = 0;
    for(;;) {
        BFS(s);
        // 殘餘的路線達不到 t，增廣路徑不存在
        if(level[t] < 0) return flow;
        memset(iter, 0, sizeof(iter));
        int f;
        while((f = dfs(s, t, INF)) > 0) flow += f;
    }
    return flow;
}
```

9 離散化 Discretization

9.1 Vector ($O(NlogN)$)

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

int main()
{
    vector<int> a = {1561, 777, 89898, 5}; // --> {3,
        2, 4, 1}
    vector<int> b = a;

    sort(b.begin(), b.end());
    b.resize(unique(b.begin(), b.end()) - b.begin());

    for(int i:a)
    {
        cout << lower_bound(b.begin(), b.end(), i) -
            b.begin() + 1 << "\n";
    }
    return 0;
}
```

9.2 Map + Set ($O(NlogN)$)

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

int main(){
    vector<int> a = {1561, 777, 89898, 5}; // -> {3,
        2, 4, 1}

    int now = 1;
    map<int, int>mp;
    set<int> ms;

    for(int i:a){
        ms.insert(i);
    }

    for(int i:ms){
        mp[i] = now++;
    }

    for(int i:a){
        cout << mp[i] << "\n";
    }

    return 0;
}
```