```
// memset (name, 0, sizeof(name));
info p;
p.sum=0, p.prop=-1;
fill_n (tree, sizeof(tree)/sizeof(*tree), p);

// start (number to string)
template <typename T> string toString (T number){
        stringstream ss;
        ss << number;
        return ss.str();

}

// start (string to number)
int stringconvert (string s){
        int p;
        istringstream ss (s);
        ss >> p;
        return p;

}
```

// Biginteger Multiplication

```
string multiply (string a , int b){      // Here a is reversed
    int carry=0;
    for (int i=0; i<a.size(); i++){
        carry+=(a[i]-'0')*b;
        a[i] = (carry%10+48);
        carry/=10
    }
    while (carry){
        a+=(carry%10+'0');
        carry/=10;
    }
    return a;
}
```

```
// operator overloading

struct data{
        int x;
        int y;
        bool operator < (const data& b) const{
                if(x==b.x) return y>b.y;
                return x<b.x;
                return 0;
        }

};
```

Linked list

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

struct node {
    int data;
    node *next;
};
node *root = NULL;
void append(int data){
    if(root == NULL){
        root = new node();
        root->data = data;
        root->next = NULL;
    }
    else {
        node *current_node = root;
        while(current_node->next != NULL){
            current_node = current_node->next;
        }
        node *newnode = new node();
        newnode->data = data;
        newnode->next = NULL;
        current_node->next = newnode;
    }
}
```

```
void print(){
    node *current_node = root;
    while(current_node != NULL){
        cout << current_node->data << endl;
        current_node = current_node->next;
    }
}

void delete_node(int data){
    node *current_node = root;
    node *previous_node = NULL;
    while(current_node->data != data){
        previous_node = current_node;
        current_node = current_node->next;
    }
    if(current_node == root){
        root = root->next;
        delete(current_node); // Library function
    }
    else{
        previous_node->next = current_node->next;
        delete(current_node);
    }
}
```

```
void add_node(int data1, int data2){
    node *current_node = root;
    while (current_node->data != data1){
        current_node = current_node->next;
    }
    node *newnode = new node();
    newnode->data = data2;
    newnode->next = current_node->next;
    current_node->next = newnode;
}

int main(){
    append(1);
    append(2);
    append(5);
    append(10);
    append(12);
    print();

    delete_node(10);
    print();
    add_node(2,3);
    add_node(3,4);
    add_node(12,19);
    print();
    return 0;
}
```

Double Linked list / Bidirectional Linked list

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

struct node{
    int data;
    node *next, *prev;
};
node *root = NULL;
node *tail = NULL;

void append(int data){
    if(root == NULL){
    root = new node();
    root->data = data;
    root->next = NULL;
    root->prev = NULL;
    tail = root;
    }
    else {
        node *newnode = new node();
        newnode->data = data;
        newnode->next = NULL;
        newnode->prev = tail;
        tail->next = newnode;
        tail = tail->next;
}
{
```

```
void print_froot ( ){
        node *current_node = root;
        while (current_node != NULL){
            cout<<current_node->data<<endl;
            current_node = current_node->next;
        }
}

void print_ftail ( ){
        node *current_node = tail;
        while (current_node != NULL){
            cout<<current->data<<endl;
            current_node = current_node->prev;
        }
}
```

```
void delete_node (int data){
    node *current_node = root;
    while (current_node -> data != data){
        current_node = current_node -> next;
    }
    if (current_node == root){
        root = root -> next;
        root -> prev = NULL;
        delete (current_node);
    }
    else if (current_node == tail){
        tail = tail -> prev;
        tail -> next = NULL;
        delete (current_node);
    }
    else {
        current_node -> prev -> next = current_node
                                    -> next;
        current_node -> next -> prev = current_node
                                    -> prev;
        delete (current_node);
    }
}
```

```
void add_node(int data1, int data2){
    node *current_node = root;
    while(current_node -> data != data1){
        current_node = current_node -> next;
    }
    if(current_node == tail){
        node *newnode = new node();
        newnode -> data = data2;
        newnode -> next = NULL;
        newnode -> prev = tail;
        tail -> next = newnode;
        tail = tail -> next;
    }
    else {
        node *newnode = new node();
        newnode -> data = data2;
        newnode -> next = current_node -> next;
        current_node -> next -> prev = newnode;
        newnode -> prev = current_node;
        current_node -> next = newnode;
    }
}
```

```
int main (){
    append(1);
    append (2);
    append (5);
    append (10);
    append (12);

    add_node (12,13);
    delete_node (2);
    print_ftail ();
    return  0;
}
```

Binary search Tree

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

struct node{
    int data;
    node *left, *right;
    node(){
        left = NULL;
        right = NULL;
    }
};
node *root;
void append(int data){
    if(root = NULL){
        root = new node();
        root->data = data;
    }
    else{
        node *current = root, *parent;
        while(current != NULL){
            if(data < current->data){
                parent = current;
                current = current->left;
            }
```

```
        else {
            parent = current;
            current = current->right;
        }
    }
    node * newnode = new node();
    newnode->data = data;
    if (newnode->data < parent->data) parent->left
                                        = newnode;
    else parent->right = newnode;
  }
}
```

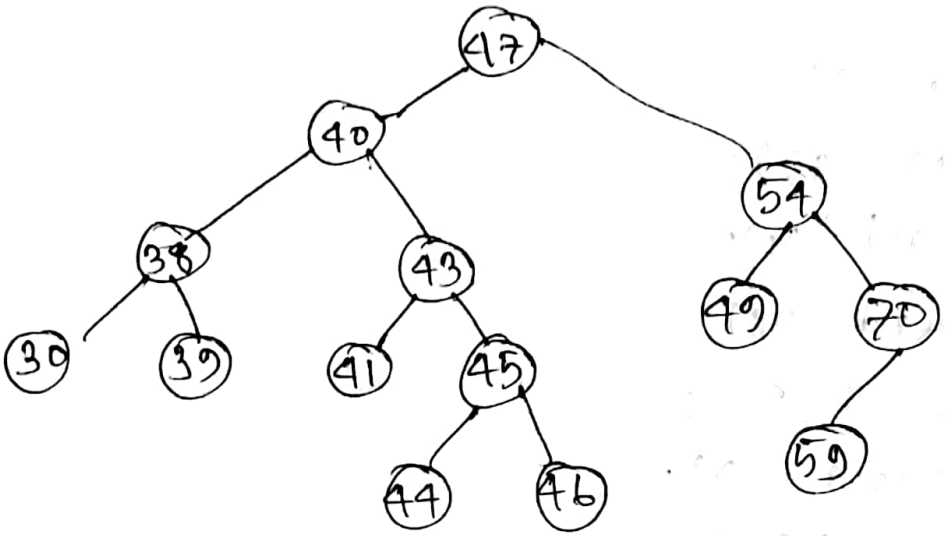
```
void search_node(int data){
    node *current = root;
    while(current != NULL && current->data != data){
        if(data < current->data) current = current->left;
        else current = current->right;
    }
    if(current == NULL) cout << "Not Found\n";
    else if(current->data == data) cout << "Found\n";
}

void printPre(node *root){
    if(root == NULL) return;
    cout << root->data << endl;
    printPre(root->left);
    printPre(root->right);
}
```

```
void printIn(node *root){
    if(root == NULL) return;
    printIn(root->left);
    cout<<root->data<<endl;
    printIn(root->right);
}

void printPost(node *root){
    if(root == NULL) return;
    printPost(root->left);
    printPost(root->right);
    cout<<root->data<<endl;
}
```

```
void delete_node(int data){
    node *current = root, *parent;
    while(current->data != data){
        if(data < current->data){
            parent = current;
            current = current->left;
        }
        else {
            parent = current;
            current = current->right;
        }
    }
if(current->left == NULL && current->right == NULL){
    if (data < parent->data) parent->left = NULL;
    else parent->right = NULL;
}
```

```
else if (current->left != NULL && current->right != NULL){

    node *temp_parent = current;
    node *temp = current->right;
    while (temp->left != NULL){
      temp_parent = temp;
      temp = temp->left;
    }
    current->data = temp->data;
    if (temp_parent == current) temp_parent->right = NULL;
    else temp_parent->left = NULL;

}
else {
   if (current->left == NULL) current = current->right;
   else if (current->right == NULL) current = current->left;
   if (parent->data > current->data) parent->left = current;
   else parent->right = current;
}
}
```

```
int main (){
    append(47);
    append (40);
    append (38);
    append (30);
    append (39);
    append (43);
    append (41);
    append (45);
    append (44);
    append (46);
    append (54);
    append (49);
    append (70);
    append (59);

    // delete_node (40);
    // delete_node (70);
    // delete_node (leaf_node);
    // search_node (46);

    return 0;
}
```



* Pre : Root → left → Right;

17, 40, 38, 30, 39, 43, 41, 45, 44, 46, 54, 49, 70, 59

* In : left → Root → Right; (ছোট থেকে বড়)

30, 38, 39, 40, 41, 43, 44, 45, 46, 47, 49, 54, 59, 70

* Post : left → Right → Root;

30, 39, 38, 41, 44, 46, 45, 43, 40, 49, 59, 70, 54, 47

Trie (Prefix tree / Radix tree)

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

struct node{
    bool endmark;
    node * next[26+1];
    node(){
        endmark = false;
        for(int i=0; i<26; i++), next[i] = NULL;
    }
};

node *root = new node();

void Insert(string s){
    node *current = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i] - 'a';
        if(current->next[id] == NULL){
            node * newnode = new node();
            current->next[id] = newnode;
        }
        current = current->next[id];
    }
    current->endmark = true;
}
```

```
bool search(string s){
    node *current = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'a';
        if(current->next[id]==NULL) return false;
        current=current->next[id];
    }
    return current->endmark;
}

void all_delete(node *current){
    for(int i=0; i<26; i++){
        if(current->next[i]!=NULL){
            all_delete(current->next[i]);
        }
    }
    delete(current);
}
```

```
int main(){
    string s;
    while(cin>>s && s!="bb"){
        insert(s);
    }
    while(cin>>s && s!="bb"){
        if(search(s)==true) cout<<"Found\n";
        else cout<<"Not found\n";
    }
    return 0;
}
```

Segment tree

```
#include <bits/stdc++.h>
# define mx 1000005
using namespace std;
int arra[mx];
int tree[4*mx];

void init(int node, int b, int e){
    if (b==e){
        tree[node] =arra[b];
        return;
    }
    int left = 2 * node;
    int right = 2 * node + 1;
    int mid = (b+e)/2;
    init(left, b, mid);
    init(right, mid+1, e);
    tree[node] = tree[left] + tree[right];
}
```

```
int query(int node, int b, int e, int i, int j){
    if (i>e || j<b) return 0;
    if (b>=i && e<=j) return tree[node];

    int left = 2*node;
    int right = 2*node+1;
    int mid = (b+e)/2;
    int p = query(left, b, mid, i, j);
    int q = query(right, mid+1, e, i, j);
    return p+q;
}

void update(int node, int b, int e, int i, int
                                        newval){
    if (i>e || i<b) return;
    if (b==e){
        tree[node] = newval;
        return;
    }
```

```
    int left = 2*node;
    int right = 2*node+1;
    int mid = (b+e)/2;
    update(left, b, mid, i, newval);
    update(right, mid+1, e, i, newval);
    tree[node] = tree[left] + tree[right];
}

int main(){
    int arr[] = {4,-9,3,7,1,0,2};
    for(int i=0; i<7; i++) arr[i+1]=arr[i];
    init(1,1,7);
    update(1,1,7,4,10);
    cout << tree[1];
    return 0;
}
```

Segment tree (Lazy Propagation)

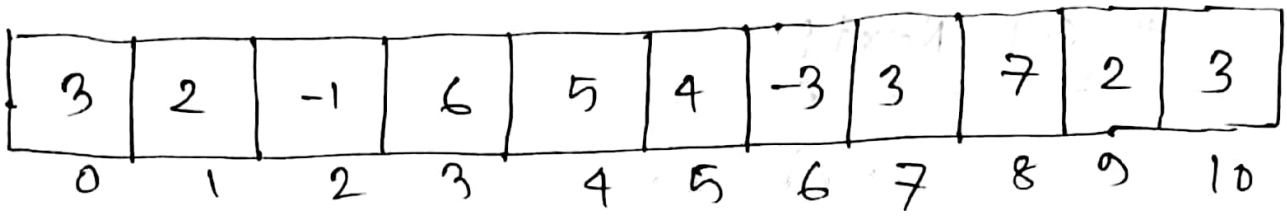
```
#include <bits/stdc++.h>
#define ll long long
#define mx 1000005
using namespace std;

int arra[mx];

struct info{

    ll prop, sum;
} tree[3*mx];


void init(int node, int b, int e){
    if(b==e){
        tree[node].sum = arra[b];
        return;
    }
    int left = 2*node;
    int right = 2*node+1;
    int mid = (b+e)/2;
    init(left, b, mid);
    init(right, mid+1, e);
    tree[node].sum = tree[left].sum + tree[right].sum
}
```

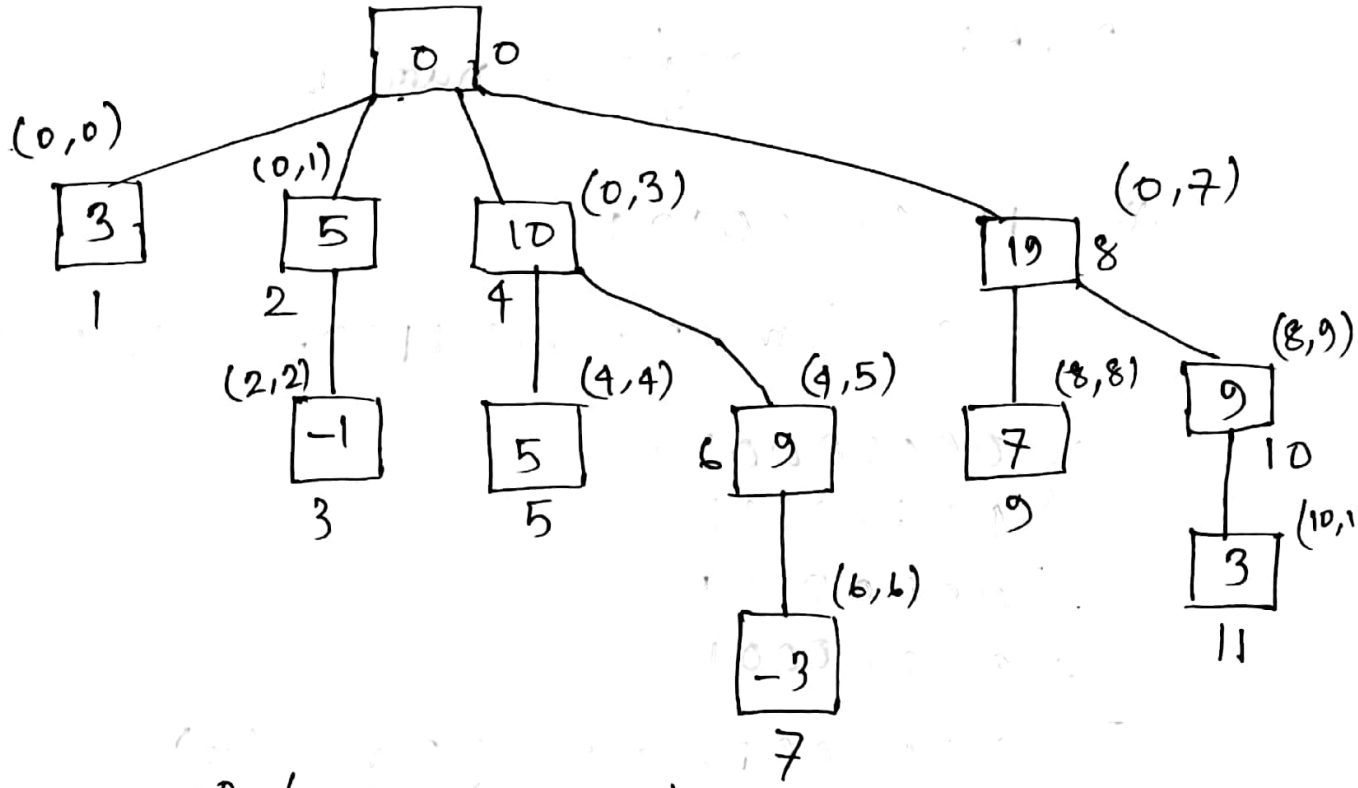
```
void update(int node, int b, int e, int i, int j, ll x){
    if(i>e || j<b) return;
    if(b>=i && e<=j){
        tree[node].sum += ((e-b+1)*x);
        tree[node].prop += x;
        return;
    }
    int left = 2*node;
    int right = 2*node+1;
    int mid = (b+e)/2;
    update(left, b, mid, i, j, x);
    update(right, mid+1, e, i, j, x);
    tree[node].sum = tree[left].sum + tree[right].
                     sum + (e-b+1)*tree[node].prop;
}
```

```
int query(int node, int b, int e, int i, int j, int
                                    carry=0){
    if(i>e || j<b) return 0;
    if(b>=i && e<=j){
        return tree[node].sum+carry*(e-b+1);
    }
    int left = 2*node;
    int right = 2*node+1;
    int mid = (b+e)/2;
    int p = query(left, b, mid, i, j, carry+tree[
                                    node].prop);
    int q = query(right, mid+1, e, i, j,
                carry+tree[node].prop);
    return p+q;
}
```

```
int main (){
    int a[7]={1,2,3,4,5,6,7};
    for(int i=0; i<7; i++) arra[i+1]=a[i];
    init (1,1,7);
    update (1,1,7, 3,5,2);
    cout << query (1,1,7,1,5); // 21
    return 0;
}
```

Binary Indexed Tree (Fenwick Tree)





$1 = 0 + 2^0$ (0 idx (2nd 1 for element)

$2 = 0 + 2^1$ (0 " " 2 " ")

$3 = 2^1 + 2^0$ (2 " " 1 " ")

$4 = 0 + 2^2$

$5 = 2^2 + 2^0$

$6 = 2^2 + 2^1$

$7 = 2^2 + 2^1 + 2^0$ (5 " " 1 " ")

$8 = 0 + 2^3$

Get Next :

1. 2's complement
2. AND with original number
3. Add to original number

1 = 00000001

∴ 2's complement = 11111110 + 1 = 11111111

∴ 00000001

& 11111111

00000001

+ 00000001

00000010 = 2 (2 no. idx)

Get next idx code :

```
idx = idx + (idx & (-idx));
```

Get Parent :

1. 2's complement
2. AND it with original number
3. subtract from original number

* 7 = 111

2's complement = 000 + 1 = 001

```
  111
& 001
-----
  001
```

∴ 111 − 001 = 110 = 6

∴ 7 idx এর parent 6 idx

* 10 = 1010

∴ 2's complement = 0101 + 1 = 0110

```
  1010
& 0110
------
  0010
```

∴ 1010 − 0010 = 1000 = 8 ∴ 10 এর parent 8

Get parent এর code :

$$idx = idx - (idx \;\&\; (-idx))$$

- AND = A & B

  OR = A | B

  XOR = A ^ B

  NOT = ~ A

  left shift => A << 1 ; [A * 2]

  A << 2 ; [A * 2^2]

  A << 3 ; [A * 2^3]

  Right shift =>

  A >> 1 ; [A / 2]

  A >> 2 ; [A / 2^2]

  A >> 3 ; [A / 2^3]

- int a[15];

  memset(a, 0, sizeof a); // sab 0 ho jayega

BIT / Fenwick tree

```
#include <bits/stdc++.h>
using namespace std;
#define mx 1000005

int tree[mx];

void update(int idx, int n, int n){
    while (idx <= n){
        tree[idx] += n;
        idx += idx & (-idx);
    }
}
int query (int idx){
    int sum = 0;
    while (idx > 0){
        sum += tree[idx];
        idx -= idx & (-idx);
    }
    return sum;
}
```

```
int main(){
    bon(int i=0; i<20; i++){ tree[i]=0;
    int n;
    cin>>n;
    bon(int i=1; i<=n; i++){
        int a;
        cin>>a;
        update(i,a,n);
    }
    int x;
    while (scanf("%d", &x)){
        int c;
        cin>>c;
        update(x+1, c, n);
        cout<<query(x+1)<<endl;
    }
    return 0;
}
```

// square root decomposition _

// starting -

```
#include <bits/stdc++.h>
using namespace std;
int n, len, a[mx], block[sqrt(mx)];
void init(){
    for(int i=1; i<=n; i++){
        block[i/len] += a[i];
    }
}

int query(int b, int e){
    int sum=0;
    for(int i=b; i<=e; ){
        if(i%len==0 && i+len-1<=e){
            sum += block[i/len];
            i += len;
        }
        else {
            sum += a[i];
            i++;
        }
    }
```

Scanned by CamScanner

```
    return sum;
}

void update (int idx, int newval){
    block[idx/len] += newval - a[idx];
}

int main (){
    // taking input array
    init ();
    // taking query (p, q) or updating ;
    return 0;
}
```

STL

vector <int>:: iterator it;

vector <int> vt;

1. vt.push_back(value);
2. vt.pop_back();
3. vt.clear();
4. vt.empty();
5. vt.size();
6. vt.begin();
7. vt.end();
8. vt.erase(index position);
9. vt.erase(first, second);
10. vt.insert(vt.begin+3, value)
11. vt.resize(size_value);
12. reverse(vt.begin(), vt.end());
13. vt.sort(vt.begin(), vt.end());
14. it = find(vt.begin(), vt.end(), value);
15. it = lower_bound(vt.begin(), vt.end(), value);
16. it = upper_bound(vt.begin(), vt.end(), value);

deque <int> dq;

1. dq.push_front(value);
2. dq.pop_front(value);
3. dq.push_back(value);
4. dq.pop_back();
5. বাকী সব vector এর মতো।

Set

set <int> st;

অথবা,

int arr[] = {5, 7, 5, 7, 3};
set <int> st(arr, arr+5);

1. st.insert(value);
2. st.erase(element); // Not index

set <int>::iterator it;

3. it = st.find(element); // if not found, it = st.end
4. st.erase(it);

5. it = st.lower_bound (element_value);

6. it = st.upper_bound (element_value);

Map

map <string, int> mp;

1. mp.insert (make_pair ("hello", 9));
2. mp.erase ("hello");
3. it = mp.find ("hello");
   - mp.erase(it);
4. mp ["hello"] // to access
5. mp.size();
6. mp.count (value);

Stack (সবার ও সবার-উপরে)

```
# stack <int> st;
1. st.push(value);
2. st.empty();
3. cout << st.top();
4. st.pop();
5. st.size();
```

Queue (যেমন লাইনে দাঁড়িয়ে থাকা মানুষ)

```
# queue <int> q;
1. cout << q.front();
2. q.pop();
```

Priority_queue <int> q;

```
1. q.top();
```

Scanned by CamScanner

String

1. cin.ignore ();
   getline (cin, string_name);

2. s.find ("some"); // first (প্রথম খুঁজবে)

3 - s.rfind ("word") // last (শেষ থেকে খুঁজবে)

4. s.substr (start_value, value (কতটা));

stringstream ss(line)

// vector <int> v;

1. sort(v.begin(), v.end());
2. reverse(v.begin, v.end());
3. int mx = *max_element(v.begin(), v.end());
4. int mini = *min_element(v.begin(), v.end());
5. int sum = accumulate(v.begin(), v.end(), 0);
6. ~~count(binary~~
   count(v.begin(), v.end(), x); // occurences of x
7. if (bind(v.begin(), v.end(), x) != v.end()){
   cout << "Found";
   }
   else cout << "Not found";

sorting vector

8. bool flag = binary_search(v.begin(), v.end(), x);
9. auto q = lower_bound(v.begin(), v.end(), x);
   cout << "position" << q - v.begin() << endl;
10. auto p = upper_bound(v.begin(), v.end(), x);
    cout << "position" << p - v.begin() << endl;

```
11. v.erase(v.begin()+1); // delete 2nd element
12. v.erase(unique(v.begin(),v.end()),v.end());
13. cout<< distance(v.begin(), max_element(all(v));
14. next_permutation(all(v));
15: prev_permutation(all(v));
```

sort vector

```
#include <bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define db double
#define MIN INT_MIN
#define MAX INT_MAX
#define all(x) (x).begin(), (x).end()
#define ii pair<int, int>
#define iii pair<int, ii>
#define pll pair<ll, ll>
#define plll pair<ll, pll>
#define bt bitset
#define ss second
#define minQueue priority_queue
                 <int, vector<int>, greater<int>>
#define maxQueue priority_queue
                 <int, vector<int>, less<int>>
#define pb push_back
```

```
#define max3 max
#define maxx3(a,b,c) max(a, max(b,c))
#define minn3(a,b,c) min(a, min(b,c))
#dkbine GCD(x,y) __gcd((x),(y))
#define LCM(x,y) ((x/GCD((x),(y)))*y)
#define loop(a,n) bon(int i=a; i<n; i++)
#define mem(x,y) memset((x),(y),sizeob(x))
#define Case int T; scanf("%d",&T);
             bon(int cas=1; cas<=T; cas++)
#define pt printf
#define sf scanf
#define pro1(x) cout<<x<<endl;
#define pro2(x,y) cout<<x<<" "<<y<<endl;
#define Iterator(type) type<int>::iterator
#define mapIterator map<int,int>::iterator
                    // it->binst it->second
using namespace std;
```

```
// Graph moves - - - - - - - - -
// int dx[]={1,0,-1,0}; int dy[]={0,1,0,-1};
                                    // 4 direction

// int dx[]={1,1,0,-1,-1,-1,0,1};
// int dy[]={0,1,1,1,0,-1,-1,-1};      // 8 direction

// int dx[]={2,1,-1,-2,-2,-1,1,2};
// int dy[]={1,2,2,1,-1,-2,-2,-1};     // Knight direction

// int dx[]={2,1,-1,-2,-1,1};
// int dy[]={0,1,1,0,-1,-1};           // Hexagonal direction

/* - - - - - - - - Bitmask - - - - - - */
//int Set(int N, int pos){return N=N | (1<<pos);}
//int reset(int N, int pos){return N=N & ~(1<<pos);}
// bool check(int N, int pos){return (bool)(N & (1<<pos));}
/* - - - - - - - - - - - - - */

const int MAXN = 1.5e7 + 10; // MAXN = 1.5*10^7+10
const double eps = (double) 1e-10;
const double PI = acos(-1.0);
```

```
/*
int bigmod(int a, int b, int m){
    if (b==0) return 1;
    int x = bigmod(a, b/2, m);
    x = (x * x)% m;
    if (b % 2 == 1) x = (x*a) % m;
    return x;
}
*/

inline double getdouble(){
    double x;
    scanf("%lf", &x);
    return x;
}
# define DB getdouble()

inline int getint(){
    int x;
    scanf("%d", &x);
    return x;
}
# define II getint()
```

```
inline long long getlonglong (){
        long long x;
        scanf("%lld", &x);
        return x;
}
#define LL getlonglong()

/* - - - - - - - - - - - - - - */

int main (){


      return 0;
}
```