Universidad Nacional Autónoma de México

Facultad de Ciencias

Competitive Programming Template

Ui Chul Shin
shuc@ciencias.unam.mx

2023

1. Initial Template

```
#include<bits/stdc++.h>
using namespace std;
//using namespace __gnu__pbds;
using bint = __int128;
using ll = long long; // %lld
using ld = long double; // %0.5lf
using vi = vector<ll>;
using pi = pair<ll, ll>; // mp(a,b); {a,b};
using ps = pair<string, string>;
using ti = tuple<ll, ll, ll>; // mt(a,b,c); {a,b,c};
using ts = tuple<string, string, string>;
ll gcd(ll a, ll b) {return b ? gcd(b, a %b) : a;}
ll lcm(ll a, ll b) {return (a / gcd(a, b)) * b;}
#define endl '\n'
#define npos string::npos
#define LSOne(a) ((a) & -(a))
#define sq(a) (a)*(a)
#define sz(a) ((int)(a).size())
#define fst first
#define snd second
#define pb push_back
#define rep(i,a,b) for(int i = a; i < b; i++)
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()
#define fo(a) find_by_order(a)
#define ok(a) order_of_key(a)
#define lb(v,a) lower_bound(v.begin(),v.end(),a)-v.begin()
#define ub(v,a) upper_bound(v.begin(),v.end(),a)-v.begin()-1
#define bitcount(a) __builtin_popcountll(a) // 1LL << 62-1
#define tzcount(a) __builtin_ctzll(a) // 1LL << 62
#define median(a,b,c) max(min(a,b), min(max(a,b),c))

const ll MOD = 1e9+7;
const ll INF = 1e9; // 10^9 = 1B is < 2^31-1
const ll LLINF = 4e18; // 4*10^18 is < 2^63-1
const ld EPS = 1e-9; // 10^-9
const ld PI = 3.14159265358979323846264338327950288841971;

/** ==Notes==
 * # of bits of n :: (int)log2(n);
 * char to int :: c-'0';
 * Check if bit is on (i--) :: 1 & (n >> i);
 * When nothing left, cursor next line :: cin.ignore();
 */
```

```
// < ascending, > descending; sort(all(v), cmp);
bool cmpPi(pi a, pi b) {return a.f == b.f ? a.s > b.s : a.f > b.f;}

bool cmpStr(str a, str b) {
        return a.size() == b.size() ? a > b : a.size() > b.size();}

bool cmpTi(ti a, ti b) {
        return get<0>(a) == get<0>(b) ?
         (get<1>(a) == get<1>(b) ? get<2>(a) < get<2>(b)
        : get<1>(a) < get<1>(b)) : get<0>(a) < get<0>(b);}

// priority_queue > ascending, < descending;
// priority_queue<ll, vi, greater<ll>> pq;
struct cmpq {bool operator() (int a, int b) {return a > b;}};

struct cmpqPi {
        bool operator() (pi a, pi b) {
                return a.f == b.f ? a.s > b.s : a.f > b.f;}};

struct cmpqTi {
        bool operator() (ti a, ti b) {
                return get<0>(a) == get<0>(b) ? (get<1>(a) == get<1>(b)
                ? get<2>(a) > get<2>(b) : get<1>(a) > get<1>(b))
                : get<0>(a) > get<0>(b);}};

int digits(ll n) {return int(log10(n)+1);}

bool isnumber(string a) {rep(i,0,a.length()) {
        if(!isdigit(a[i])) return false;}
        return true;}

string tobinary(ll n) {string s = "";
        for(int i = (int)log2(n); i >= 0; i--) {
                (1&(n>>i)) ? s+="1" : s+="0";}
        return s;}

int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    // freopen("input.txt", "r", stdin);
    // freopen("output.txt", "w", stdout);
    int t; cin >> t;
    while(t--) solve();
    return 0;
}
```

2. Syntax

```
// ==Char to string==
string s(1,c);

// ==stringstream delimiter==
rep(i,0,s.length()) {if(s[i]=='/') s[i] = ' ';}
stringstream ss(s); string t;
while(ss >> t) {...} (1)
while(getline(ss,t,'/')) {...} (2)

// ==Count num of this char in string s==
count(all(s),'a');

// ==erase() certain character==
s.erase(remove(all(s), c), s.end());

// ==Check if string contains a char==
s.find(c) != string::npos ? found : !found;

// ==Print for loop==
#define prtfor(a) rep(i,0,a.size()) cout << a[i] << " \n"[i==a.size()-1];

// ==Print for each loop==
#define prtfore(a) for(auto x : a) {cout << x << " \n"[x==*(--a.end())];}

// ==How to switch on and off==
bool io = 0;
io ^= 1;

// ==Sort k characters starting from i-th==
for(int i = 0; i <= n-k; i++) {
        sort(v.begin()+i,v.begin()+i+k);
}

// ==How to not print a newline in the last case==
while(cin >> a) {
        (flag) ? flag=0 : printf("\n");
        solve(a);}

// ==Regex for ip address==
string inputString = "172.16.254.1";
regex a("(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\\.)
{3}([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])");
return regex_match(inputString, a);
```

3. Bitmask

```
// ==Binary to decimal==
stoi(s,0,2);

// ==Decimal to binary==
for(int i = (int)log2(n); i >= 0; i--) {
        (1&(n>>i)) ? s+="1" : s+="0";}

// ==ASCII Char to binary==
bitset<7>(c).to_string();

// ==Left and Right shift operator==
n << 1 - n*2^1
n << 2 - n*2^2
n >> 1 - n/2^1
n >> 2 - n/2^2

// i-th bit is counting from right to left
// ==Turn on i-th bit and add==
n |= (1 << i);

// ==Check if i-th bit is set on==
n & (1 << i);

// ==Clear/turn off i-th bit==
n &= ~ (1 << i);

// ==Flip i-th bit with XOR==
n ^ = (1 << i);

// ==Value of LSB that is on==
((n) & -(n)); // LSOne(n);

// ==Turn on all n bits==
(1 << n) - 1;

// ==Enumerate all proper subsets
for(int i = n; i; i = (n & (i-1))) {cout << i << '\n';}

// ==Count how many bits are on==
__builtin_popcount(n);

// ==Count how many trailing zeroes==
__builtin_ctz(n);
```

4. Order Statistics Tree

```
//==Sorts first k+1 elems in ascending or descending order==
nth_element(v.begin(),v.begin()+k,v.end());
nth_element(v.begin(),v.begin()+k,v.end(), greater<int>());

//==Policy-Based Data Structures (pbds)==
#include <bits/extc++.h>                                  // pbds
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>, rb_tree_tag,
             tree_order_statistics_node_update> ost;

vi v = {2,4,7,10,15,23,50,65,71};
ost tree;
rep(i,0,n) tree.insert(v[i]);
*tree.find_by_order(0)    // smallest = 2
*tree.find_by_order(n-1) // largest = 71
*tree.find_by_order(4)    // 5th smallest = 15
tree.order_of_key(2)      // index 0 (rank 1)
tree.order_of_key(71)     // index 8 (rank 9)
tree.order_of_key(15)     // index 4 (rank 5)

//==Kadane's Algorithm==
tuple<ll,int,int> maxSubarray(vi &v) {
    int n = v.size(), l=0, r=0;
    ll ans = v[0], sum = 0, mp = -1;
    rep(i,0,n) {
        sum += v[i];
        if(sum > ans) {ans = sum; l = mp+1; r = i;}
        if(sum <= 0) {sum = 0; mp = i;}
    }
    return {ans, l, r};
}

tuple<ll,int,int> minSubarray(vi &v) {
    int n = v.size(), l=0, r=0;
    ll ans = v[0], sum = 0, mp = -1;
    rep(i,0,n) {
        sum += v[i];
        if(sum < ans) {ans = sum; l = mp+1; r = i;}
        if(sum >= 0) {sum = 0; mp = i;}
    }
    return {ans, l, r};
}
```

5. Strings

```
// ==Longest Prefix Suffix==
vi LPS(string s) {
    vi v(s.length(),0);
    rep(i,1,s.length()) {
        int j = v[i-1];
        while(j && s[i]!=s[j]) j = v[j-1];
        if(s[i]==s[j]) j++;
        v[i] = j;
    } return v;
}

// ==Knuth-Morris-Pratt==
vi KMP(string s, string t) {
    int n = s.length(), m = t.length();
    vi lps = LPS(t), kmp;
    int i=0, j=0;
    while(n-i >= m-j) {
        if(s[i] == t[j]) {i++; j++;}
        if(j==m) {kmp.pb(i-j); j = lps[j-1];}
        else if(i < n && s[i] != t[j]) {
            j ? j = lps[j-1] : i++;
        }
    }
    return kmp;
}

// ==Z==
vi Z(string s, string t) {
    string a = t+"$"+s; int n = a.length(), m = t.length();
    vi p(n), z;
    for(int i=1, l=0, r=0; i < n; ++i) {
        if(i <= r) p[i] = min(r-i+1, p[i-l]);
        while(i+p[i] < n && a[p[i]] == a[i+p[i]]) ++p[i];
        if(i+p[i]-1 > r) l=i, r=i+p[i]-1;
    }
    rep(i,0,n) {if(p[i] == m) z.pb(i-m-1);}
    return z;
}
```

```
// ==Longest Palindromic Subsequence==
string LPS(string &s) {
    string t = s;
    reverse(all(t));
    return LCS(s,t);
}

// ==Longest Palindromic Subsequence==
// a, b = s.length(); vvi dp(MAXN, vi(MAXN,-1));
int lps(string &s, int a, int b) {
    if(a==0 || b==0) return 0;
    if(dp[a][b] != -1) return dp[a][b];
    if(s[a-1] == s[n-b]) {
        return dp[a][b] = 1+lps(s,a-1,b-1);
    } else {
        return dp[a][b] = max(lps(s,a-1,b),lps(s,a,b-1));}}

// ==Longest Substring without Repeating Chars==
int lengthOfLongestSubstring(string s) {
        set<char> alphabet;
        int ans = 0, l = 0;
        for(int i = 0; i < s.length(); i++) {
            while(alphabet.count(s[i])) {
                alphabet.erase(s[l]); l++;}
            alphabet.insert(s[i]);
            ans = max(ans, i-l+1);}
        return ans;}

// ==Longest Substring without repeating chars==
string LSWRC(string &s) {
    int l=0, r=0, i=0, j=0, m=0;
    unordered_set<char> a;
    while(r < s.size()) {
        if(a.find(s[r]) == a.end()) {
            a.insert(s[r]);
            if(r-l+1 > m) {m = r-l+1; i = l; j = r;}
            r++;
        } else {a.erase(s[l]); l++;}
    }
    return s.substr(i,m);}
```

```
// ==Longest Substring with K unique chars==
string LKS(string &s, int k) {
    int i=0, j=0, ans=-1;
    unordered_map<char,int> m;
    while(j < s.size()) {
        m[s[j]]++;
        while(m.size() > k) {
            m[s[i]]--;
            if(!m[s[i]]) m.erase(s[i]);
            i++;}
        if(m.size()==k) ans=max(ans,j-i+1);
        j++;}
    return s.substr(i,ans);}

// ==Minimum substring of s that contains all chars of t==
string minWindow(string s, string t) {
    vi m(256,0);
    int a=0, b=INT_MAX, c=0, i=0, j=0;
    rep(i,0,t.length()) {if(!m[t[i]]) c++; m[t[i]]++;}
    while(j < s.length()) {
        m[s[j]]--;
        if(!m[s[j]]) c--;
        while(!c) {
            if(b > j-i+1) {a = i; b = min(b,j-i+1);}
            m[s[i]]++;
            if(m[s[i]] > 0) c++;
            i++;}
        j++;}
    return (b != INT_MAX) ? s.substr(a,b) : "-1";}
```

6. Range Queries

```
//==Sparse Table==
vvi sparse_table(vi &v) {
    int n = v.size(), k = (int)log2(n)+1; vvi st(n,vi(k));
    rep(i,0,n) st[i][0] = v[i];
    rep(j,1,k+1) {
        for(int i=0; i+(1<<j)<=n; i++) {
            // RMQ :: Minimum <, Maximum >
            st[i][j-1] < st[i+(1<<(j-1))][j-1] ? st[i][j]=st[i][j-1]
                                               : st[i][j]=st[i+(1<<(j-1))][j-1];
            // RSQ
            // st[i][j] = st[i][j-1] + st[i+(1<<(j-1))][j-1];
        }
    } return st;}

// Minimum <=, Maximum >=
int RMQ(int l, int r, vvi &st) {
    int m, j = (int)log2(r-l+1);
    st[l][j] <= st[r-(1<<j)+1][j] ? m=st[l][j] : m=st[r-(1<<j)+1][j];
    return m;}

ll RSQ(int l, int r, vvi &st) {
    ll ans = 0; int k = (int)log2(st.size())+1;
    for(int j=k; j>=0; j--) {
        if(l+(1<<j)-1 <= r) {ans += st[l][j]; l += 1<<j;}
    } return ans;}
```

```
//==Fenwick Tree==
class Fenwick {
private:
    vll ft;
public:
    Fenwick(int m) {ft.assign(m+1,0);}
    void build(const vll &f) {
        int m = (int)f.size()-1; ft.assign(m+1,0);
        for(int i=1; i <= m; ++i) {
                ft[i] += f[i];
                if(i+LSOne(i) <= m) ft[i+LSOne(i)] += ft[i];}}
    Fenwick(const vll &f) {build(f);}
    Fenwick(int m, const vi &s) {
        vll f(m+1,0);
        for(int i=0; i < (int)s.size(); ++i) ++f[s[i]];
        build(f);}
    ll rsq(int j) {ll sum = 0;
        for(;j;j-=LSOne(j)) sum += ft[j]; return sum;}
    ll rsq(int i, int j) {return rsq(j) - rsq(i-1);}
    void update(int i, ll v) {
        for(; i < (int)ft.size(); i+=LSOne(i)) ft[i] += v;}
    int select(ll k) {
        int l = 1, h = ft.size()-1;
        for(int i=0; i < 30; ++i) {
                int m = (l+h)/2; (rsq(1,m) < k) ? l=m : h=m;} return h;}
};
class RUPQ {
private:
    Fenwick ft;
public:
    RUPQ(int m) : ft(Fenwick(m)) {}
    void range_update(int ui, int uj, int v) {ft.update(ui, v);
                                              ft.update(uj+1, -v);}
    ll point_query(int i) {return ft.rsq(i);}
};
class RURQ  {
private:
    RUPQ rupq; Fenwick purq;
public:
    RURQ(int m) : rupq(RUPQ(m)), purq(Fenwick(m)) {}
    void range_update(int ui, int uj, int v) {
        rupq.range_update(ui, uj, v);
        purq.update(ui, v*(ui-1));
        purq.update(uj+1, -v*uj);}
    ll rsq(int j) {return rupq.point_query(j)*j - purq.rsq(j);}
    ll rsq(int i, int j) {return rsq(j) - rsq(i-1);}
};
```

```
//==Segment Tree RMQ==
class Segtree {
private:
    int n; vll A, st, lazy;
    int l(int p) {return p << 1;}  int r(int p) {return(p << 1)+1;}
    ll conquer(ll a, ll b) {
        if(a==-1) return b; if(b==-1) return a;
        return min(a,b);}
    void build(int p, int L, int R) {
        if(L==R) st[p] = A[L];
        else {
            int m = (L+R)/2;
            build(l(p), L, m); build(r(p),m+1,R);
            st[p] = conquer(st[l(p)],st[r(p)]);}}
    void propagate(int p, int L, int R) {
        if(lazy[p]!=0) {
            // st[p] = lazy[p];
            // if(L!=R) lazy[l(p)] = lazy[r(p)] = lazy[p];
            st[p]+=lazy[p];
            if(L!=R) {lazy[l(p)]+=lazy[p]; lazy[r(p)]+=lazy[p];}
            lazy[p] = 0;}
    }
    ll RMQ(int p, int L, int R, int i, int j) {
        propagate(p, L, R);
        if(i > j) return -1;
        if((L >= i) && (R <= j)) return st[p];
        int m = (L+R)/2;
        return conquer(RMQ(l(p), L, m, i, min(m,j)),
                       RMQ(r(p),m+1,R, max(i, m+1), j));
    }
    void update(int p, int L, int R, int i, int j, ll val) {
        propagate(p, L, R);
        if(i > j) return;
        if((L >= i) && (R <= j)) {
            lazy[p] = val;
            propagate(p, L, R);
        } else {
            int m = (L+R)/2;
            update(l(p), L, m, i, min(m,j), val);
            update(r(p), m+1, R, max(i, m+1), j, val);
            int lsubtree = (lazy[l(p)] != 0) ? lazy[l(p)] : st[l(p)];
            int rsubtree = (lazy[r(p)] != 0) ? lazy[r(p)] : st[r(p)];
            st[p] = (lsubtree <= rsubtree) ? st[l(p)] : st[r(p)];
            // <= min, >= max
        }
    }
```

```
public:
    Segtree(int sz) : n(sz), st(4*n), lazy(4*n, 0) {}
    Segtree(const vll &initialA) : Segtree((int)initialA.size()) {
        A = initialA;
        build(1, 0, n-1);
    }
    void update(int i, int j, ll val) {update(1,0,n-1,i,j,val);}
    ll RMQ(int i, int j) {return RMQ(1,0,n-1,i,j);}
};

//==Segment Tree RSQ==
const int MAXN = 2e5+5;
struct node {ll val, lzAdd, lzSet; node(){};} st[MAXN << 2];

class Segtree {
private:
    int l(int p) {return p << 1;}
    int r(int p) {return(p << 1)+1;}
    void conquer(int p) {st[p].val = st[l(p)].val + st[r(p)].val; return;}

    void propagate(int p, int L, int m, int R) {
        if(st[p].lzSet != 0) {
            st[l(p)].lzSet = st[r(p)].lzSet = st[p].lzSet;
            st[l(p)].val = (m-L+1)*st[p].lzSet;
            st[r(p)].val = (R-m)*st[p].lzSet;
            st[l(p)].lzAdd = st[r(p)].lzAdd = 0;
            st[p].lzSet = 0;
        } else if(st[p].lzAdd != 0) {
            if(st[l(p)].lzSet == 0) st[l(p)].lzAdd += st[p].lzAdd;
            else {
                st[l(p)].lzSet += st[p].lzAdd;
                st[l(p)].lzAdd = 0;
            }
            if(st[r(p)].lzSet == 0) st[r(p)].lzAdd += st[p].lzAdd;
            else {
                st[r(p)].lzSet += st[p].lzAdd;
                st[r(p)].lzAdd = 0;
            }
            st[l(p)].val += (m-L+1)*st[p].lzAdd;
            st[r(p)].val += (R-m)*st[p].lzAdd;
            st[p].lzAdd = 0;
        } return;
    }
```

```
    void build(int p, int L, int R) {
        st[p].lzAdd = st[p].lzSet = 0;
        if(L==R) {st[p].val = A[L]; return;}
        int m = (L+R) >> 1;
        build(l(p), L, m); build(r(p),m+1,R);
        conquer(p); return;}
    void add(int p, int L, int R, int i, int j, ll val) {
        if (i > R || j < L) return;
        if (i <= L && R <= j) {
            st[p].val += (R-L+1)*val;
            if(st[p].lzSet == 0) st[p].lzAdd += val;
            else st[p].lzSet += val; return;}
        int m = (L+R) >> 1;
        propagate(p, L, m, R);
        add(l(p), L, m, i, j, val); add(r(p), m+1, R, i, j, val);
        conquer(p); return;}
    void set(int p, int L, int R, int i, int j, ll val) {
        if (i > R || j < L) return;
        if (i <= L && R <= j) {
            st[p].val = (R-L+1) * val;
            st[p].lzAdd = 0; st[p].lzSet = val; return;}
        int m = (L+R) >> 1;
        propagate(p, L, m, R);
        set(l(p), L, m, i, j, val); set(r(p), m+1, R, i, j, val);
        conquer(p); return;}
    ll query(int p, int L, int R, int i, int j) {
        if(i > R || j < L) return 0;
        if(i <= L && R <= j) return st[p].val;
        int m = (L+R) >> 1;
        propagate(p, L, m, R);
        return query(l(p),L,m,i,j) + query(r(p),m+1,R,i,j);}
public:
    int n;
    ll A[MAXN];
    Segtree(int sz) : n(sz) {}
    void build() {build(1,1,n);}
    void add(int i, int j, ll val) {add(1,1,n,i,j,val);}
    void set(int i, int j, ll val) {set(1,1,n,i,j,val);}
    ll query(int i, int j) {return query(1,1,n,i,j);}
};
```

7. Math

```
const ld pi = 3.14159265358979323846264338327950288419 71;
//==Spiral Square Perimeter==
2 + n*(n+1) + n; // n is length of side of square

//==Dice Moves to get n==
if(n == 1) {cout << -1 << endl; return;}
int ans = (n/11)*2;
ans += ((n%11)/5);
if(((n%11)%5) != 0) ans++;
if(n == 7) ans++;

//==a^b < c^d==
(b*log(a) <= d*log(c))

//==Gauss's Circle Problem==
// # of lattice points within circle of radius r.
ll lattice_points(ll r) { // 1e9 lattice points with r = 18000
    ll points = 0;
    rep(i,1,r+1) points += floor(sqrt(sq(r)-sq(i)));
    return 1ll+4ll*(r+points);
}
//==Circumscribed Circle==
// Polygon of v vertices, each side length s, return area.
ld circumscribed_circle(ld v, ld s) {
    ld area = s/(2*sin(pi/v));
    area*=(area*pi);
    return area;
}
//==XOR from 1 to n==
int computeXOR(int n) {
    if(n%4==0) return n;
    if(n%4==1) return 1;
    if(n%4==2) return n+1;
    return 0;}
//==Two nums that are sum s and xor x==
pi sumxor(ll s, ll x) {
    ll z = (s-x)/2, a = 0, b = 0;
    for(int i = 0; i < (int)log2(s); i++) {
        ll xi = (x & (1 << i));
        ll zi = (z & (1 << i));
        if(xi && zi) {return {-1,-1};}
        if(!xi && zi) {
            a = ((1 << i) | a);
            b = ((1 << i) | b);}
        if(xi && !zi) {a = ((1 << i) | a);}}
    return {a,b};}
```

8. Graph

```
//==Graph (int)==
vi graph[n];
vi visited(n, 0);
vi dist(n, 0);

//==Graph (string)==
map<string, vector<string>> graph;
map<string, bool> visited;
map<string, int> distance;

//==Graph (matrix)==
vector<pi> graph[n][m];
vvi visited(n,vi(m,0));

//==Generating Edges (matrix graph)
 graph[i][j].pb(mp(i,j));
 // left
 if(j-1 >= 0) {graph[i][j].pb({i,j-1}); graph[i][j-1].pb({i,j});}
 // right
 if(j+1 <= m-1) {graph[i][j].pb({i,j+1}); graph[i][j+1].pb({i,j});}
 // up
 if(i-1 >= 0) {graph[i][j].pb({i-1,j}); graph[i-1][j].pb({i,j});}
 // up left
 if(i-1 >= 0 && j-1 >= 0) {graph[i][j].pb({i-1,j-1});
                           graph[i-1][j-1].pb({i,j});}
 // up right
 if(i-1 >= 0 && j+1 <= m-1) {graph[i][j].pb({i-1,j+1});
                             graph[i-1][j+1].pb({i,j});}

void bfs(int s) {
    visited[s] = 1;
    queue<int> q; q.push(s);
    while(!q.empty()) {
        int u = q.front(); q.pop();
        for(auto v : graph[u]) {
            if(!visited[v]) {
                visited[v] = 1;
                dist[v] = dist[u]+1;
                q.push(v);}
        }
    }
}
```

```
void dfs(int u) {
    visited[u] = 1;
    for (int v : graph[u]) {
        if (!visited[v]) {
            dfs(v); dist[v] = dist[u]+1;
        }
    }
}

void dfs(int a) {
    visited[a] = 1;
    stack<int> s; s.push(a);
    while(!s.empty()) {
        int u = s.top(); s.pop();
        visited[u] = 1;
        for(auto v : graph[u]) {
            if(!visited[v]) {
                dist[v] = dist[u]+1;
                s.push(v);
            }
        }
    }
}

//==Graph (dijkstra)==
vector<pi> graph[n];
vi dist(n, INF);
//==Edges (w :: weight)==
graph[u].pb({v,w});

void dijkstra(int s) {
    priority_queue<pi, vector<pi>, greater<pi> > pq;
    pq.push({0,s}); dist[s] = 0;
    while(!pq.empty()) {
        auto p = pq.top(); pq.pop();
        int u = p.second;
        if(p.first != dist[u]) continue;
        for(auto z : graph[u]) {
            int v = z.first;
            int weight = z.second;
            if(dist[v] > dist[u] + weight) {
                dist[v] = dist[u] + weight;
                pq.push({dist[v], v});
            }
        }
    }
}
```

```
//==Disjoint Set Union==
class DSU {
private:
    vi p, rank, sz; int numSets;
public:
    DSU(int n) {
        p.assign(n,0);
        rep(i,0,n) p[i] = i;
        rank.assign(n,0);
        sz.assign(n,1);
        numSets = n;
    }
    int find(int i) {return (p[i]==i) ? i : (p[i] = find(p[i]));}
    bool same(int i, int j) {return find(i) == find(j);}
    int numDisjoint() {return numSets;}
    ll size(int i) {return sz[find(i)];}
    void unite(int i, int j) {
        if(same(i,j)) return;
        int x = find(i), y = find(j);
        if(rank[x] > rank[y]) swap(x,y);
        p[x] = y;
        if(rank[x] == rank[y]) ++rank[y];
        sz[y] += sz[x];
        --numSets;}};
//==Strongly Connected Components==
class SCC {
private:
    vector<vi> AL, AL_T; vi A, dfs_num; ll MAXN, ans = 0;
public:
    SCC(int n) {
        AL.assign(n,{}); AL_T.assign(n,{});
        dfs_num.assign(n,0);
        MAXN = n;
    }
    void Kosaraju(int u, int pass) { //==pass== 1 original, 2 transpose
        dfs_num[u] = 1;
        vi &neighbor = (pass==1) ? AL[u] : AL_T[u];
        for(auto &v : neighbor) {
            if(!dfs_num[v]) Kosaraju(v,pass);
        } A.pb(u);}
    void addEdge(int u, int v) {AL[u].pb(v); AL_T[v].pb(u);}
    ll components() {
        rep(u,0,MAXN) {if(!dfs_num[u]) Kosaraju(u,1);}
        dfs_num.assign(MAXN,0);
        for(int i = MAXN-1; i >= 0; i--) {
            if(!dfs_num[A[i]]) Kosaraju(A[i],2), ans++;}
        return ans;}};
```

```
//==Minimum Spanning Tree==
struct Edge {ll u, v, w;};
// < Minimum Spanning Tree, > Maximum Spanning Tree
bool cmp(Edge a, Edge b) {return a.w > b.w;}

class MST {
private:
    vi p, rank, sz;
    ll sets, cost = 0;
public:
    vector<Edge> edges, mst;
    MST(int n) {
        p.assign(n,0);
        rep(i,0,n) p[i] = i;
        rank.assign(n,0);
        sz.assign(n,1);
        sets = n;
    }
    int find(int i) {return (p[i]==i) ? i : (p[i] = find(p[i]));}
    bool same(int i, int j) {return find(i) == find(j);}
    ll size(int i) {return sz[find(i)];}
    void unite(int i, int j) {
        int a = find(i), b = find(j);
        if(a==b) return;
        if(rank[a] < rank[b]) swap(a,b);
        p[b] = a;
        if(rank[a] == rank[b]) ++rank[a];
        sz[a] = sz[b];
        --sets;
    }
    ll total_cost() {return cost;}
    void addEdge(int u, int v, int w) {Edge e; e.u = u; e.v = v;
                                       e.w = w; edges.pb(e);}
    vector<Edge> Kruskal() {
        sort(all(edges),cmp);
        for(Edge e : edges) {
            --e.u; --e.v;
            if(!same(e.u,e.v)) {
                cost += e.w; cost %= MOD; if(cost < 0) cost += MOD;
                mst.pb(e);
                unite(e.u,e.v);
            }
        }
        return mst;
    }
};
```

```
//==Articulation Points and Bridges==
class Bridges {
private:
    vector<vi> graph; ll dfsNumberCounter, dfsRoot, rootChildren,
                                    MAXN, UNVISITED = -1;
    vi dfs_num, dfs_low, dfs_parent, articulation_vertex,
                                    articulation_points;
    vector<pi> bridges;
public:
    Bridges(int n) {
        dfsNumberCounter = 0, MAXN = n;
        graph.assign(n,{});
        dfs_num.assign(n,UNVISITED);
        dfs_low.assign(n,0);
        dfs_parent.assign(n,-1);
        articulation_vertex.assign(n,0);
    }
    void addEdge(int u, int v) {graph[u].pb(v); graph[v].pb(u);}
    void articulationPointAndBridge(int u) {
        dfs_num[u] = dfsNumberCounter++;
        dfs_low[u] = dfs_num[u];
        for(auto &v : graph[u]) {
            if(dfs_num[v] == UNVISITED) {
                dfs_parent[v] = u;
                if(u == dfsRoot) ++rootChildren;
                articulationPointAndBridge(v);
                if(dfs_low[v] >= dfs_num[u]) articulation_vertex[u] = 1;
                if(dfs_low[v] > dfs_num[u]) {bridges.pb({u,v});}
                dfs_low[u] = min(dfs_low[u], dfs_low[v]);
            } else if(v != dfs_parent[u]) dfs_low[u] = min(dfs_low[u],
                                                    dfs_num[v]);
        }}
    void artibridge() {
        rep(u,0,MAXN) {
            if(dfs_num[u] == UNVISITED) {
                dfsRoot = u; rootChildren = 0;
                articulationPointAndBridge(u);
                articulation_vertex[dfsRoot] = (rootChildren > 1);
            }
        }
        rep(u,0,MAXN) {
            if(articulation_vertex[u]) articulation_points.pb(u);}
    }
    vector<pi> getBridges() {return bridges;}
    vi getArticulation() {return articulation_points;}
};
```

```
//==Topological Sort==
class Toposort {
private:
    vector<vi> graph; int MAXN;
    vi ts, tl, vis, indegree;
    priority_queue<ll, vi, greater<ll> > pq;
public:
    Toposort(int n) {
        MAXN = n;
        tl.assign(MAXN,0);
        graph.assign(n,{});
        vis.assign(n,0);
        indegree.assign(n,0);
    }
    void addEdge(int u, int v) {graph[u].pb(v); indegree[v]++;}
    void addEdgeTL(int v, int u) {graph[u].pb(v); indegree[v]++;}
    void topo(int a) {
        vis[a] = 1;
        for(auto &u : graph[a]) {if(!vis[u]) topo(u);}
        ts.pb(a);}
    vi toposort() {
        for(int i = MAXN-1; i >= 0; i--) {if(!vis[i]) topo(i);}
        //rep(i,0,MAXN) {if(!vis[i]) topo(i);}
        reverse(all(ts)); return ts;
    }
    vi khan() {
        rep(i,0,MAXN) {if(!indegree[i]) pq.push(i);}
        while(!pq.empty()) {
            int u = pq.top(); pq.pop(); ts.pb(u);
            for(auto &v : graph[u]) {
                --indegree[v];
                if(!indegree[v]) pq.push(v);}
        } return ts;
    }
    vi topolabel() { // order of node appearance lexicographically
        set<ll,greater<>> s; int c = MAXN-1;
        rep(i,0,MAXN) {if(!indegree[i]) s.insert(i);}
        while(!s.empty()) {
            ll u = *s.begin(); s.erase(u); tl[u] = c;
            for(auto &x : graph[u]) {
                if(--indegree[x]==0) s.insert(x);
            } c--;
        }
        return tl;
    }
};
```

```
//==Lowest Common Ancestor==
class LCA {
private:
    vector<vector<pi> > graph; vector<vi> up;
    vi depth, dist; ll l, MAXN;
public:
    LCA(ll n) {
        graph.assign(n,{}); up.assign(n,vi(20));
        depth.assign(n,0); dist.assign(n,0);
        FOR(i,0,n) {up[i][0] = i;}
        l = ceil(log2(n)); MAXN = n;
    }
    void addEdge(ll u, ll v, ll w) {graph[u].pb({v,w}); graph[v].pb({u,w})
    void init() {
        dfs(0);
        FOR(i,1,l) {FOR(j,0,MAXN) up[j][i] = up[up[j][i-1]][i-1];}
    }

    void dfs(ll u) {
        for(auto &[v,w] : graph[u]) {
            if(v != up[u][0]) {
                depth[v] = depth[u]+1;
                dist[v] = dist[up[v][0]=u]+w;
                dfs(v);
            }}}

    ll jump(ll x, ll d) {
        FOR(i,0,l) {if((d>>i)&1) x = up[x][i];}
        return x;
    }

    ll lca(ll u, ll v) {
        if(depth[u] < depth[v]) swap(u,v);
        u = jump(u, depth[u]-depth[v]);
        if(u==v) return u;
        RFOR(i,0,l) {
            if(up[u][i] != up[v][i])
                u = up[u][i], v = up[v][i];
        }
        return up[u][0];
    }

    ll distance(int u, int v) {
        return dist[u]+dist[v]-2*dist[lca(u,v)];
    }};
```

9. Combinatorics

```
const ll MOD = 1e9+7;
const ll MAXN = 1e6;
vll fac(MAXN), inv(MAXN);
// ==Binary Exponentiation for Modular Multiplicative Inverse==
// bcpow(a, MOD-2) :: Moduler Inverse of a
ll bcpow(ll a, ll b){
    ll r = 1;
    while(b) {
        if(b&1ll) r = r * a %MOD;
        a = a * a %MOD; b >>= 1ll;
    } return r;}

// ==Binary Exponentiation==
ll binpow(ll a, ll b) {
    ll r = 1;
    while(b) {
        if(b & 1ll) r = r * a;
        a = a * a; b >>= 1ll;
    } return r;}

//==Binomial Coefficient==
ll nCk(ll n, ll k) {
    ld r = 1;
    rep(i,1,k+1) r = r * (n-k+i)/i;
    return (ll)(r+0.01);}

// ==Calculates Factorials and Inverses==
void facinv() {
    fac[0] = 1; fac[1] = 1;
    rep(i,2,MAXN) fac[i] = fac[i-1] * i %MOD;
    inv[MAXN-1] = bcpow(fac[MAXN-1], MOD-2);
    for(ll i = MAXN-2; i >= 0; i--) inv[i] = inv[i+1] * (i+1ll) %MOD;}

// ==Binomial Coefficient with Modular Arithmetic==
ll nCk(ll n, ll k) {
    return n < k ? 0 : fac[n] * inv[k] %MOD * inv[n-k] %MOD;}

vll inverses(ll n, vll &f) {
    vll inv(n+1); inv[n] = bcpow(f[n], MOD-2);
    for(ll i = n-1; i >= 0; i--) inv[i] = inv[i+1] * (i+1ll) %MOD;
    return inv;}
```

10. Geometry

```
//==Convex Hull Graham Scan==
struct pt{ld x, y;};
bool operator<(const pt &p, const pt &q) {
        return mp(p.y, p.x) < mp(q.y, q.x);
}

// Direction :: collinear 0, clockwise -1, counterclockwise 1;
int orientation(pt p, pt q, pt r) {
    ld val = (r.y-p.y)*(q.x-p.x)-(q.y-p.y)*(r.x-p.x);
    return !val ? 0 : (val < 0 ? -1 : 1);
}

vector<pt> grahamScan(vector<pt> &pts) {
    pt p0 = *min_element(all(pts), [](pt p, pt q) {
             return mp(p.y, p.x) < mp(q.y, q.x);});
    sort(all(pts), [&p0](const pt &p, const pt &q) {
        int o = orientation(p0,p,q);
        if(!o) return (p0.x-p.x)*(p0.x-p.x)+(p0.y-p.y)*(p0.y-p.y)
                    < (p0.x-q.x)*(p0.x-q.x)+(p0.y-q.y)*(p0.y-q.y);
        return o > 0; // o < 0 for cw
    });
    vector<pt> ch;
    rep(i,0,pts.size()) {
        while(ch.size() > 1 && orientation(ch[ch.size()-2],
              ch.back(), pts[i]) != 1 /* -1 for cw */) ch.pop_back();
        ch.pb(pts[i]);
    }
    return ch;
}

set<pt> tmp;
vector<pt> pts;
rep(i,0,n) {
    cin >> x >> y;
    pt p; p.x = x; p.y = y;
    tmp.insert(p);
}
for(auto a : tmp) pts.pb(a);

vector<pt> ans = grahamScan(pts);
cout << ans.size() << endl;
for(auto a : ans) {
    cout << a.x << " " << a.y << endl;
}
```

11. Number Theory

```
// ==Factorials==
vll factorials(ll n) {
    vll f(n+1,1);
    rep(i,1,n+1) f[i] = f[i-1] * i %MOD;
    return f;}
// ==Trailing Zeros for n!==
ll trailingZeros(ll n) {
    ll z = 0; for(ll i = 5; i <= n; i*=5) z += n/i; return z;}

//==Check if n is prime==
bool isPrime(ll n) {
    if(n <= 1) return 0;
    if(n <= 3) return 1;
    if(n%2==0 || n%3==0) return 0;
    for(ll i = 5; i*i <= n; i+=6) {
        if(n%i==0 || n%(i+2)==0) return 0;
    } return 1;
}

//==Check if a^2-b^2 is prime==
bool isDiffPrime(ll a, ll b) {
    return (isPrime(a+b) && a-b==1) ? 1 : 0;
}

//==Sieve of Eratosthenes==
vll primes(ll n) {
        vll p(n+1,1); p[0]=0; p[1]=0;
        for(ll i=2; i*i<=n; i++) {
        if(p[i]) {for(ll j=i*i; j<=n; j+=i) p[j]=0;}}
        rep(i,0,n+1) {if(p[i]) q.pb(i);}
        return q;}

//==Divisors of N==
vll divisors(ll n) {
        vll d; d.pb(1); d.pb(n);
        for(ll i=2; i*i<=n; i++) {
                if(n%i==0) i*i==n ? d.pb(i) : (d.pb(i), d.pb(n/i));}
        return d;}

//==Practical Numbers==
bool practicalNum(vll d, ll n) {
        ll r = 0; bool p;
        for(auto a : d) {if(r+1 < a) break; r+=a;}
        (r+1<n*2) ? p=0 : p=1;
        return p;}
```

```
// ==K-divisible Sum==
// Sum of n numbers div by k and max elem of arr is min possible
(n+(k*((n+k-1)/k))-1)/n

//==Divisible pairs in an Array==
int divisible_pairs(vi &A) {
    int n = A.size(), ans = 0;
    unordered_map<ll,ll> frq;
    rep(i,0,n) frq[A[i]]++;
    rep(i,0,n) {
        for(int j = 1; sq(j) <= A[i]; j++) {
            if(A[i]%j==0) {
                (A[i]==sq(j)) ? ans+=frq[j] : ans+=frq[j]+frq[A[i]/j];}
        } ans--;
    }return ans;}

//==Numbers of sums of distinct powers of 3 til 1e5==
vector<int> v;
rep(i,0,9) {
    int a = pow(3,i); v.pb(a);
    rep(j,0,pow(2,i)) {
        int tmp = a;
        rep(k,0,i) {
            if(j & (1<<k)) {tmp += pow(3,k);}
        }
        v.pb(tmp);
    }
} v.pb(pow(3,9));

//==Number of pairs that can matrix multiply==
map<int,ll> snd; vi v(n);
rep(i,0,n) {cin >> v[i] >> b; snd[b]++;}
ll ans = 0;
rep(i,0,n) {if(snd.count(v[i])) ans += snd[v[i]];}

//==Number of numers that have only 1 zero in binary==
ll a, b; cin >> a >> b;
ll i = (ll)log2(a), j = (ll)log2(b);
ll first = (i*(i-1))/2, last = (j*(j-1))/2;
i++; j++;
ll x = (1LL << i)-1, y = (1LL << j)-1;
for(ll k = i-2; k >= 0; k--) {
    if((x & ~(1LL << k)) < a) first++;
}
for(ll k = j-2; k >= 0; k--) {
    if((y & ~(1LL << k)) <= b) last++;
} cout << last-first << endl;
```

```
//==Maximal AND==
// can turn on k bits in total in any elem in array
int n, k; cin >> n >> k;
    vi bits(31,0);
    vector<ll> v(n);
    rep(i,0,n) {
        cin >> v[i];
        for(int j = (int)log2(v[i]); j >= 0; j--) {
            if(v[i] & (1 << j)) bits[j]++;
        }
    }
    int bit = 30;
    while(k > 0) {
        while(bits[bit] == n || n-bits[bit] > k) bit--;
        if(bit < 0) break;
        rep(i,0,n) {
            v[i] |= (1 << bit);
            k -= (n-bits[bit]);
            bits[bit] = n;
        }
    }
    ll ans = pow(2,31)-1;
    rep(i,0,n) ans &= v[i];
    cout << ans << endl;

//==Partition Problem==
bool findPartition(vi &v) {
    int n = v.size(), sum = 0;
    rep(i,0,n) sum += v[i];
    if(sum%2) return false;
    int p = sum/2;
    vector<bool> part(p+1,0);
    rep(i,0,n) {
        for(int j = p; j >= v[i]; j--) {
            if(part[j-v[i]]==1 || j==v[i]) part[j] = 1;
        }
    }
    return part[p];
}

//==2 max elems==
int a = 0, b = 0;
rep(i,0,n) {
        cin >> c;
        if(c > a) {b=a; a=c;}
        else if(c > b) {b=c;}
}
```

```
//==Smallest number with length m and sum of digits s==
string getMin(int m, int s) {
    if(m == 1) {return to_string(s);}
    string a = ""; bool first = 1;
    while((m-1)*9 > s && m > 0) {
        if(first) {first = 0; a += "1"; s--; m--; continue;}
        a += "0"; m--;
    }
    a += to_string(s-(m-1)*9); m--;
    rep(i,0,m) a += "9";
    return a;
}

//==Largest number with length m and sum of digits s==
string getMax(int m, int s) {
    string a = "";
    while(s >= 9) {a += "9"; m--; s-=9;}
    if(s >= 1) {a += to_string(s); m--;}
    rep(i,0,m) a += "0";
    return a;
}

//==2D Prefix Sum==
int c = 0; // til 1e6
rep(i,0,1414) {
    int a = i, b = 0;
    while(a >= 0) {
        ll can = 0;
        if(a > 0) {can += pyramid[a-1][b];} // up
        if(b > 0) {can += pyramid[a][b-1];} // left
        if(a > 0 && b > 0) {can -= pyramid[a-1][b-1];} // up-left
        can += binpow(c,2);
        pyramid[a][b] = can;
        dp[c] = can;
        c++;
        a--; b++;
    }
}
```

12. Implementation

```
// ==Make Palindrome with given string==
string palindrome(string s) {
        unordered_map<char,int> m;
        rep(i,0,s.length()) m[s[i]]++;
        int odd=0; char c;
        for(auto a : m) {
                if(a.second%2!=0) {odd++; c=a.first;}}
        if(odd>1 || odd&&s.length()%2==0) return "Null";
        string x="", y="";
        for(auto a : m) {
                string s(a.second/2,a.first);
                x=x+s; y=s+y;}
        return (odd) ? (x+c+y) : (x+y);}

// ==All triplets v[i]+v[j]+v[k] that equal n==
vvi threeSum(vi &v, int n) {
    sort(all(v)); vvi ans;
    rep(i,0,v.size()-2) {
        if(i > 0 && v[i-1] == v[i]) continue;
        int k = -v[i]+n, a = i+1, b = v.size()-1;
        while(a < b) {
            if(v[a] + v[b] == k) {
                ans.push_back({v[i], v[a], v[b]});
                a++, b--;
                while(a < b && v[a] == v[a-1]) a++;
                while(a < b && v[b] == v[b+1]) b--;
            }
            if(v[a] + v[b] > k) b--;
            if(v[a] + v[b] < k) a++;
        }
    } return ans;}

//==Number of cars(b) that overtook other cars(a)
int overrun(vi &a, vi &b) {
    rep(i,0,n) c[b[i]] = i; // turn of that index that goes out
    rep(i,0,n) res[i] = c[a[i]]; // order of that car that goes out
    int fined = 0, x=-1;
    rep(i,0,n) {
        (res[i] > x) ? x = res[i] : fined++;
    }
    return fined;
}
```

```
//==Maximize difference with other player==
ll asum=0, bsum=0;
int player = 0;
while(!a.empty() || !b.empty()) {
    int ai = a.top(), bi = b.top();
    if(ai > bi && !player) {asum += ai; a.pop();}
    else if(ai < bi && player) {bsum += bi; b.pop();}
    else {player ? a.pop() : b.pop();}
    player ^= 1;
}

// Number of different strings by removing 2 consecutive characters
int diffStrings(string s) {
    int ans = 1;
    rep(i,0,s.length()-2) ans += s[i]!=s[i+2];
    return ans;
}

// Number of pairs of lower and upper case
// Can swap lower to upper or viceversa max k times
int pairsLowerUpper(int k, string s) {
    map<char,int> m; int ans = 0; int available = 0;
    rep(i,0,s.length()) m[s[i]]++;
    for(auto a : m) {
        char l = tolower(a.first), u = toupper(a.first);
        if(!m[l] && !m[u]) continue;
        else if(m[l] && m[u]) {
            ans += min(m[l], m[u]);
            available += abs(m[l]-m[u])/2;
            m[l] = 0; m[u] = 0;
        } else {
            available += max(m[l], m[u])/2;
            m[l] = 0; m[u] = 0;
        }}
    ans += min(available, k);
    return ans;}

// Prefix needed from s to form t
map<char,vi> letters;
rep(i,0,n) letters[s[i]].pb(i+1);
vi alpha(26,0);
rep(i,0,t.length()) alpha[t[i]-'a']++;
int ans = 0;
for(int i = 0; i < 26; i++) {
    if(alpha[i]) {ans = max(ans,letters[i+'a'][alpha[i]-1]);}
}
cout << ans << endl;
```

```
// # of operations to make s anti-palindrome
map<char,int> m;
rep(i,0,n) m[s[i]]++;
bool cant = 0;
for(auto a : m) {
    if(m[a.first]*2 > s.length()) cant=1;
}
if(n%2 || cant) {cout << -1 << endl; return;}

vi same(26);
rep(i,0,n/2) {
    same[s[i]-'a'] += s[i]==s[n-i-1];
    // add to index of the alphabet if same on both sides
}
// doing just #same+1/2 won't work cuz what if aaaabbcdefghbbaaaa
int ans = 0;
priority_queue<int> pq;
rep(i,0,26) {
    if(same[i]) pq.push(same[i]);
    // if there is repeating alphabet, add to pq
}
while(pq.size() > 1) {
    int a = pq.top(); pq.pop();
    int b = pq.top(); pq.pop();
    a--; b--; ans++;
    if(a) pq.push(a);
    if(b) pq.push(b);
}
if(!pq.empty()) ans += pq.top();


// Swap letters of s at distance k or k+1 to make same as t
void canBeSame(string s, string t, int k) {
    bool can = 1; int n = s.length();
    if(k > n) {
        (s==t) ? printf("YES\n") : printf("NO\n"); return;}
    if(k > n/2) {
        if(s.substr(n-k,n-(n-k)*2) != t.substr(n-k,n-(n-k)*2)) can=0;}
    map<char,int> ms, mt; rep(i,0,n) ms[s[i]]++; rep(i,0,n) mt[t[i]]++;
    for(auto a : ms) {if(ms[a.first] != mt[a.first]) {can=0; break;}}
    can ? printf("YES\n") : printf("NO\n");
}
```

13. DP

```
//==Dice Combinations==
vi dp(n+1,0); dp[0] = 1;
    FOR(i,1,n+1) {
        FOR(j,1,7) {
            if(i < j) break;
            dp[i] += dp[i-j]%MOD;
            dp[i] %= MOD;
            if(dp[i] < 0) dp[i] += MOD;
        }
    }
    cout << dp[n] << endl;

//==Coin Combinations I==
vi coin(n); FOR(i,0,n) cin >> coin[i];
    vi dp(x+1); dp[0] = 1;
    FOR(i,1,x+1) {
        FOR(j,0,n) {
            if(i-coin[j] < 0) continue;
            dp[i] += dp[i-coin[j]];
            dp[i] %= MOD;
            if(dp[i] < 0) dp[i] += MOD;
        }
    }
    cout << dp[x] << endl;

//==Coin Combinations II==
vi coin(n); FOR(i,0,n) cin >> coin[i];
    vi dp(x+1); dp[0] = 1;
    FOR(i,0,n) {
        FOR(j,coin[i],x+1) {
            dp[j] += dp[j-coin[i]];
            dp[j] %= MOD;
            if(dp[j] < 0) dp[j] += MOD;
        }
    }
    cout << dp[x] << endl;
```

```
// ==Longest Common Subsequence==
string LCS(string s, string t) {
    int n = s.length(), m = t.length();
    vvi dp(n+1,vi(m+1));
    rep(i,0,n+1) {
        rep(j,0,m+1) {
            !i||!j ? dp[i][j] = 0 : (s[i-1]==t[j-1]
                   ? dp[i][j]=dp[i-1][j-1]+1
                   : dp[i][j]=max(dp[i-1][j],dp[i][j-1]));
        }
    }
    int i = n, j = m, k = dp[n][m]; vector<char> c(k+1); c[k] = '\0';
    while(i && j) {
        s[i-1]==t[j-1] ? (c[k-1] = s[i-1], i--, j--, k--)
                       : (dp[i-1][j]>dp[i][j-1] ? i-- : j--);
    }
    string lcs(all(c)); lcs.erase(lcs.length()-1,1);
    return lcs;
}

//==Longest Increasing Subsequence==
vi LIS(vi &v) {
    int n = v.size();
    vi ans, sub, si, path(n, -1);
    rep(i,0,n) {
        if(sub.empty() || sub[sub.size()-1] < v[i]) {
            path[i] = sub.empty() ? -1 : si[sub.size()-1];
            sub.pb(v[i]);
            si.pb(i);
        } else {
            int idx = lb(sub,v[i]);
            path[i] = idx == 0 ? -1 : si[idx-1];
            sub[idx] = v[i];
            si[idx] = i;
        }
    }
    int t = si[si.size()-1];
    while(t != -1) {
        ans.pb(v[t]);
        t = path[t];
    }
    reverse(all(ans));
    return ans;
}
```

```
//==Longest Subsequence of A having LCM at most K
int LongestSubsequenceLCM(vi A, int k) {
    int n = A.size(); map<ll,ll> m;
    rep(i,0,n) m[A[i]]++;
    vi v(k+1,0);
    for(auto a : m) {
        if(a.fst <= k) {
            for(int i=1;;++i) {
                if(a.fst*i > k) break;
                v[a.fst*i] += a.snd;
            }
        } else break;
    }
    ll lcm = 0, ans = 0; //if lcm==0, no answer
    rep(i,1,n+1) {if(v[i] > ans) {ans = v[i]; lcm = i;}}
    vi u; rep(i,0,n) if(lcm%A[i]==0) u.pb(A[i]); // subsequence
    return ans;
}
```

14. Search

```
//==Binary Search==
int binarySearch(vi &v, int x) {
    int l = 0, r = v.size()-1;
    while(l <= r) {
        int m = l+(r-l)/2;
        v[m] < x ? l = m+1 : r = m-1;
        if(v[m] == x) return m;
    }
    return -1;}

//==Ternary Search==
int ternarySearch(vi &v, int x) {
    int l = 0, r = v.size()-1;
    while(r >= l) {
        int m1 = l+(r-l)/3, m2 = r-(r-l)/3;
        if(v[m1] > x) r = m1-1;
        else if(v[m2] < x) l = m2+1;
        else {l = m1+1; r = m2-1;}
        if(v[m1] == x) return m1;
        if(v[m2] == x) return m2;
    }
    return -1;}
```

15. Prefix

```
vll v = {1,2,3,4,5,6,7,8,9,10};
int n = v.size();
//==Prefix Sum==
vll pfx(n+1,0);
rep(i,1,n+1) {pfx[i] = pfx[i-1] + v[i-1];}

ll max_subarray_sum = pfx[1], min_prefix_sum = pfx[0];
rep(i,1,n+1) {
    max_subarray_sum = max(max_subarray_sum, pfx[i]-min_prefix_sum);
    min_prefix_sum = min(min_prefix_sum, pfx[i]);
}

//==2D Prefix Sum==
vector<vll> v2d = {
    {0,0,0,0,0,0},
    {0,1,2,3,4,5},
    {0,1,2,3,4,5},
    {0,1,2,3,4,5},
    {0,1,2,3,4,5},
    {0,1,2,3,4,5}
}; n = v2d.size()-1;
vector<vll> pfx2d(n+1,vll(n+1,0));
rep(i,1,n+1) {
    rep(j,1,n+1) {
        pfx2d[i][j] = v2d[i][j]+pfx2d[i-1][j]+
                      pfx2d[i][j-1]-pfx2d[i-1][j-1];
    }
}
// x = 3~5, y = 2~3
int x1 = 3, x2 = 5, y1 = 2, y2 = 3;
ll sum = pfx2d[x2][y2] - pfx2d[x1-1][y2] -
         pfx2d[x2][y1-1] + pfx2d[x1-1][y1-1];
cout << sum << endl;

//==Prefix Multiplication Except Self==
vi productExceptSelf(vi &v) {
    int n = v.size();
    vi prefix(n), suffix(n), ans;
    int p = 1, s = 1;
    rep(i,0,n) {prefix[i] = p *= v[i]; suffix[n-1-i] = s *= v[n-1-i];}
    rep(i,0,n) {
        if(!i) {ans.pb(suffix[i+1]); continue;}
        if(i==n-1) {ans.pb(prefix[i-1]); continue;}
        ans.pb(prefix[i-1]*suffix[i+1]);
    } return ans;}
```

16. **Swaps**

```
// Number of bubble sort swaps to sort array
//==Inversion Count with Merge Sort==
ll merge(vi &v, vi &A, int l, int m, int r) {
    int i = l, j = m, k = l; ll inv = 0;
    while(i <= m-1 && j <= r) {
        if(v[i] <= v[j]) A[k++] = v[i++];
        else {A[k++] = v[j++]; inv += (m-i);}
    }
    while(i <= m-1) A[k++] = v[i++];
    while(j <= r) A[k++] = v[j++];
    rep(i,l,r+1) v[i] = A[i];
    return inv;
}

ll mergeSort(vi &v, vi &A, int l, int r) {
    int m; ll inv = 0;
    if(l < r) {
        m = (l+r)/2;
        inv += mergeSort(v,A,l,m);
        inv += mergeSort(v,A,m+1,r);
        inv += merge(v,A,l,m+1,r);
    } return inv;
}

ll inversionCount(vi &v) {
    vi A(v.size());
    return mergeSort(v,A,0,v.size()-1);
}

//==Inversion Index==
struct comparepq {bool operator() (pi a, pi b) {return a.first > b.first;}
int inversionIndex(vi v) {
    if(v.size() <= 1) return 0;
    /*==Ascending==
    struct comparepq {bool operator() (pi a, pi b) {return a.first > b.fir
    priority_queue<pi, vector<pi>, comparepq > pq; */
    priority_queue<pi> pq;
    rep(i,0,v.size()) {pq.push({v[i],i});} v.clear(); int ans = 0;
    while(!pq.empty()) {
        pi p = pq.top(); pq.pop();
        int y = lb(v,p.second); ans += p.second - y;
        v.insert(lower_bound(all(v),p.second),p.second);
    } return ans;
}
```

```
//==Swaps(any i,j) needed to sort array==
void swap(vi &A, int i, int j) {
    ll tmp = A[i];
    A[i] = A[j];
    A[j] = tmp;
}

int min_swaps(vi A, int n) {
    int ans = 0; vi tmp = A; sort(all(tmp));
    map<ll,ll> m; // indexes of A
    rep(i,0,n) m[A[i]] = i; // m[5] = index of 5
    rep(i,0,n) {
        if(A[i] != tmp[i]) {
            ans++;
            int j = A[i];
            // swap elem with idx it should be in
            swap(A,i,m[tmp[i]]);
            // update idx
            m[j] = m[tmp[i]];
            m[tmp[i]] = i;
        }
    }
    return ans;
}

//==Swaps in a binary tree to make sorted perfect binary tree==
//==int ans = 0; swapsBT(v,0,m-1,ans); m = # elems.
bool mergeBT(vi &v, int l, int m, int r) {
    if(v[m] < v[m+1]) return 0;
    rep(i,l,m+1) swap(v[i],v[i+(r-l+1)/2]);
    return 1;
}

void swapsBT(vi &v, int l, int r, int &ans) {
    if(l < r) {
        int m = l+(r-l)/2;
        swapsBT(v,l,m, ans);
        swapsBT(v,m+1,r, ans);
        if(mergeBT(v,l,m,r)) ans++;
    }
}

is_sorted(all(v)) ? printf("%d\n",ans) : printf("-1\n");
```