Team notebook

University of Asia Pacific

October 31, 2024

Contents

1 Data Structure

1.1 BIT

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

const int N = 3e5 + 9;

struct BIT { //1-indexed
  int n; vector<int> t;
  BIT() {}
  BIT(int _n) {
```

```
    n = _n; t.assign(n + 5, 0);
  }
  int qry(int i) {
    int ans = 0;
    for (; i >= 1; i -= (i & -i)) ans += t[i];
    return ans;
  }
  void upd(int i, int val) {
    if (i <= 0) return;
    for (; i <= n; i += (i & -i)) t[i] += val;
  }
  void upd(int l, int r, int val) {
    upd(l, val);
    upd(r + 1, -val);
  }
  int qry(int l, int r) {
    return qry(r) - qry(l - 1);
  }
};
int32_t main() {
  int n, q;
  cin >> n >> q;
  BIT bit(n);
  bit.upd(x, 1); // demo
  bit.query(10); // dsemo, change this

  return 0;
}
```

1.2 BST

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

const int N = 3e5 + 9;

//the code returns a BST which will create if we add the values
      one by one
//here nodes are indicated by values and every node must be
      distinct
set<int>se;
map<int, int>l, r; //l contains the left child of the node, r
      contains right child of the node
int main() {
  int n;
  cin >> n;
  int k;
  cin >> k; //root of the tree
  se.insert(k);
  for(int i = 1; i < n; i++) {
    int k;
    cin >> k;
    auto it = se.upper_bound(k);
    if(it != se.end() && l.find(*it) == l.end()) l[*it] = k;
    else --it, r[*it] = k;
    se.insert(k);
  }
  for(int i = 1; i <= n; i++) cout << l[i] << ' ' << r[i] <<
       '\n';
  return 0;
}
```

1.3 Bit Binary Search

```
// --- Bit Binary Search in o(log(n)) ---
const int M = 20
const int N = 1 << M

int lower_bound(int val){
       int ans = 0, sum = 0;
       for(int i = M - 1; i >= 0; i--){
              int x = ans + (1 << i);
              if(sum + bit[x] < val)
                     ans = x, sum += bit[x];
       }

       return ans + 1;
}
```

1.4 DSU

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

const int N = 3e5 + 9;

struct DSU {
  vector<int> par, rnk, sz;
  int c;
  DSU(int n) : par(n + 1), rnk(n + 1, 0), sz(n + 1, 1), c(n) {
    for (int i = 1; i <= n; ++i) par[i] = i;
  }
  int find(int i) {
    return (par[i] == i ? i : (par[i] = find(par[i])));
  }
  bool same(int i, int j) {
    return find(i) == find(j);
  }
  int get_size(int i) {
    return sz[find(i)];
  }
  int count() {
    return c;   //connected components
  }
  int merge(int i, int j) {
    if ((i = find(i)) == (j = find(j))) return -1;
    else --c;
    if (rnk[i] > rnk[j]) swap(i, j);
    par[i] = j;
    sz[j] += sz[i];
    if (rnk[i] == rnk[j]) rnk[j]++;
    return j;
  }
};

int32_t main() {

  return 0;
}
```

1.5 GP Hash Table

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

struct custom_hash {
  static uint64_t splitmix64(uint64_t x) {
    x += 0x9e3779b97f4a7c15;
    x = (x ^ (x >> 30)) * 0xbf58476d1ce4e5b9;
    x = (x ^ (x >> 27)) * 0x94d049bb133111eb;
    return x ^ (x >> 31);
  }
  size_t operator()(uint64_t x) const {
    static const uint64_t FIXED_RANDOM =
         chrono::steady_clock::now().time_since_epoch().count();
    return splitmix64(x + FIXED_RANDOM);
  }
};

gp_hash_table<int, int, custom_hash> mp;

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, x; cin >> n >> x;
  int a[n + 1];
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  for (int i = 1; i <= n; i++) {
    if (mp[x - a[i]]) {
      cout << mp[x - a[i]] << ' ' << i << '\n';
      return 0;
    }
    mp[a[i]] = i;
  }
  cout << "IMPOSSIBLE\n";
  return 0;
}
// https://cses.fi/problemset/task/1640
```

1.6 MO$_s$ *Algorithm*

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

const int N = 1e6 + 9, B = 440;

struct query {
  int l, r, id;
  bool operator < (const query &x) const {
    if(l / B == x.l / B) return ((l / B) & 1) ? r > x.r : r <
        x.r;
    return l / B < x.l / B;
  }
} Q[N];
int cnt[N], a[N];
long long sum;
inline void add_left(int i) {
  int x = a[i];
  sum += 1LL * (cnt[x] + cnt[x] + 1) * x;
  ++cnt[x];
}
inline void add_right(int i) {
  int x = a[i];
  sum += 1LL * (cnt[x] + cnt[x] + 1) * x;
  ++cnt[x];
}
inline void rem_left(int i) {
  int x = a[i];
  sum -= 1LL * (cnt[x] + cnt[x] - 1) * x;
  --cnt[x];
}
inline void rem_right(int i) {
  int x = a[i];
  sum -= 1LL * (cnt[x] + cnt[x] - 1) * x;
  --cnt[x];
}
long long ans[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int n, q;
  cin >> n >> q;
  for(int i = 1; i <= n; i++) cin >> a[i];
  for(int i = 1; i <= q; i++) {
    cin >> Q[i].l >> Q[i].r;
    Q[i].id = i;
  }
  sort(Q + 1, Q + q + 1);
  int l = 1, r = 0;
  for(int i = 1; i <= q; i++) {
    int L = Q[i].l, R = Q[i].r;
    if(R < l) {
      while (l > L) add_left(--l);
      while (l < L) rem_left(l++);
      while (r < R) add_right(++r);
      while (r > R) rem_right(r--);
    } else {
      while (r < R) add_right(++r);
      while (r > R) rem_right(r--);
      while (l > L) add_left(--l);
      while (l < L) rem_left(l++);
    }
    ans[Q[i].id] = sum;
  }
  for(int i = 1; i <= q; i++) cout << ans[i] << '\n';
  return 0;
}
```

1.7 Merge Sort Tree

```cpp
// Mergesort Tree - Time <O(nlogn), O(log^2n)> - Memory O(nlogn)
// Mergesort Tree is a segment tree that stores the sorted
    subarray
// on each node.
vi st[4*N];

void build(int p, int l, int r) {
       if (l == r) { st[p].pb(s[l]); return; }
       build(2*p, l, (l+r)/2);
       build(2*p+1, (l+r)/2+1, r);
       st[p].resize(r-l+1);
       merge(st[2*p].begin(), st[2*p].end(),
                             st[2*p+1].begin(), st[2*p+1].end(),
                             st[p].begin());
}

int query(int p, int l, int r, int i, int j, int a, int b) {
       if (j < l or i > r) return 0;
       if (i <= l and j >= r)
              return upper_bound(st[p].begin(), st[p].end(),
                    b) -
                                  lower_bound(st[p].begin(),
                                       st[p].end(), a);
       return query(2*p, l, (l+r)/2, i, j, a, b) +
                              query(2*p+1, (l+r)/2+1, r, i, j,
                                   a, b);
}
```

1.8 Monotonous Queue

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

const int N = 3e5 + 9;

struct monotonous_queue { //max, stores strictly decreasing
    sequence of the current queue
  int a[N + 10], b[N + 10], l = 0, r = -1;
  void push(int val) {
    int cnt = 0;
    while(l <= r && a[r] <= val) {
      cnt += b[r] + 1;
      r--;
    }
    a[++r] = val, b[r] = cnt;
  };
  int top() {
    return a[l];
  }
  void pop() {
    if(l > r) return;
    if (b[l] > 0) {
      b[l] --;
      return;
    }
    l++;
  }
};

int32_t main() {

  return 0;
}
```

1.9 Ordered Set

```cpp
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
#include<ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
using namespace std;

template <typename T> using o_set = tree<T, null_type, less<T>,
    rb_tree_tag, tree_order_statistics_node_update>;
template <typename T, typename R> using o_map = tree<T, R,
    less<T>, rb_tree_tag, tree_order_statistics_node_update>;
int main() {
  int i, j, k, n, m;
  o_set<int>se;
  se.insert(1);
  se.insert(2);
  cout << *se.find_by_order(0) << endl; ///k th element
  cout << se.order_of_key(2) << endl; ///number of elements
      less than k
  o_map<int, int>mp;
  mp.insert({1, 10});
  mp.insert({2, 20});
  cout << mp.find_by_order(0)->second << endl; ///k th element
  cout << mp.order_of_key(2) << endl; ///number of first
      elements less than k
  return 0;
}
```

1.10 Segment Tree Lazy

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

```
const int N = 5e5 + 9;
int a[N];
struct ST {
  #define lc (n << 1)
  #define rc ((n << 1) + 1)
  long long t[4 * N], lazy[4 * N];
  ST() {
    memset(t, 0, sizeof t);
    memset(lazy, 0, sizeof lazy);
  }
  inline void push(int n, int b, int e) { // change this
    if (lazy[n] == 0) return;
    t[n] = t[n] + lazy[n] * (e - b + 1);
    if (b != e) {
      lazy[lc] = lazy[lc] + lazy[n];
      lazy[rc] = lazy[rc] + lazy[n];
    }
    lazy[n] = 0;
  }
  inline long long combine(long long a,long long b) { // change
      this
    return a + b;
  }
  inline void pull(int n) { // change this
    t[n] = t[lc] + t[rc];
  }
  void build(int n, int b, int e) {
    lazy[n] = 0; // change this
    if (b == e) {
      t[n] = a[b];
      return;
    }
    int mid = (b + e) >> 1;
    build(lc, b, mid);
    build(rc, mid + 1, e);
    pull(n);
  }
  void upd(int n, int b, int e, int i, int j, long long v) {
    push(n, b, e);
    if (j < b || e < i) return;
    if (i <= b && e <= j) {
      lazy[n] = v; //set lazy
      push(n, b, e);
      return;
    }
    int mid = (b + e) >> 1;
    upd(lc, b, mid, i, j, v);
    upd(rc, mid + 1, e, i, j, v);
    pull(n);
  }
  long long query(int n, int b, int e, int i, int j) {
    push(n, b, e);
    if (i > e || b > j) return 0; //return null
    if (i <= b && e <= j) return t[n];
    int mid = (b + e) >> 1;
    return combine(query(lc, b, mid, i, j), query(rc, mid + 1,
        e, i, j));
  }
}t;
```

```
int32_t main() {
  int n = 5;
  for (int i = 1; i <= n; i++) {
    a[i] = i;
  }
  t.build(1, 1, n); // building the segment tree
  t.upd(1, 1, n, 2, 3, 10); // adding 10 to the segment [2, 3]
  cout << t.query(1, 1, n, 1, 5) << '\n'; // range sum query on
      the segment [1, 5]
  return 0;
}
```

1.11 Segment Tree Marge Function

```
// SUM
int t[4 * N];
int marge(int x, int y) {
  return x + y;
}
// MIN
int t[4 * N];
int marge(int x, int y) {
  return min(x , y);
}
// NUMBER OF MINIMUMS
struct node {
  int mn, count;
};
node t[4 * N];
node merge(node l, node r) {
  node ans = {inf , 0};
  ans.mn = min(l.mn, r.mn);
  if(l.mn == ans.mn) {
    ans.count += l.count;
  }
  if(r.mn == ans.mn) {
    ans.count += r.count;
  }
  return ans;
}
// NUMBER OF MAXIMUM
struct node {
  int mx, count;
};
node t[4 * N];
node merge(node l, node r) {
  node ans = {inf , 0};
  ans.mx = max(l.mx, r.mx);
  if(l.mx == ans.mx) {
    ans.count += l.count;
  }
  if(r.mx == ans.mx) {
    ans.count += r.count;
  }
  return ans;
}
// Segemet with maximum sum
```

```
struct node {
    int mx, sum, pref, suf;
};
node t[4 * N];
node marge(node a, node b) {
    int curr_max = max({a.mx, b.mx, a.suf + b.pref});
    int curr_pref = max(a.pref, a.sum + b.pref);
    int curr_suf = max(b.suf, b.sum + a.suf);
    int curr_sum = a.sum + b.sum;
    return {curr_max, curr_sum, curr_pref, curr_suf};
}
```

1.12 Segment Tree

```
const int N = 3e5 + 9;
int a[N];
struct ST {
  int t[4 * N];
  static const int inf = 1e9;
  ST() {
    memset(t, 0, sizeof t);
  }
  int marge(int x, int y) {
    return x + y;
  }
  void build(int n, int b, int e) {
    if (b == e) {
      t[n] = a[b]; // Update
      return;
    }
    int mid = (b + e) >> 1, l = n << 1, r = l | 1;
    build(l, b, mid);
    build(r, mid + 1, e);
    t[n] = marge(t[l], t[r]); // change this
  }
  void upd(int n, int b, int e, int i, int x) {
    if (b > i || e < i) return;
    if (b == e && b == i) {
      t[n] = x; // update
      return;
    }
    int mid = (b + e) >> 1, l = n << 1, r = l | 1;
    upd(l, b, mid, i, x);
    upd(r, mid + 1, e, i, x);
    t[n] = marge(t[l], t[r]); // change this
  }
  int query(int n, int b, int e, int i, int j) {
    if (b > j || e < i) return -inf; // return appropriate value
    if (b >= i && e <= j) return t[n];
    int mid = (b + e) >> 1, l = n << 1, r = l | 1;
    return marge(query(l, b, mid, i, j), query(r, mid + 1, e,
        i, j)); // change this
  }
}t;
int32_t main() {
  int n = 5;
  for (int i = 1; i <= n; i++) {
```

```
    a[i] = i;
  }
  t.build(1, 1, n); // building the segment tree
  t.upd(1, 1, n, 2, 10); // assiging 10 to the index 2 (a[2] :=
       10)
  cout << t.query(1, 1, n, 1, 5) << '\n'; // range max query on
       the segment [1, 5]
  return 0;
}
```

1.13 Sparse Table

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

const int N = 1e5 + 9;

int t[N][18], a[N];
void build(int n) {
  for(int i = 1; i <= n; ++i) t[i][0] = a[i];
  for(int k = 1; k < 18; ++k) {
    for(int i = 1; i + (1 << k) - 1 <= n; ++i) {
      t[i][k] = min(t[i][k - 1], t[i + (1 << (k - 1))][k - 1]);
    }
  }
}

int query(int l, int r) {
  int k = 31 - __builtin_clz(r - l + 1);
  return min(t[l][k], t[r - (1 << k) + 1][k]);
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int n;
  cin >> n;
  for(int i = 1; i <= n; i++) cin >> a[i];
  build(n);
  int q;
  cin >> q;
  while(q--) {
    int l, r;
    cin >> l >> r;
    ++l;
    ++r;
    cout << query(l, r) << '\n';
  }
  return 0;
}
```

1.14 Trie

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

const int N = 3e5 + 9;

struct Trie {
  static const int B = 31;
  struct node {
    node* nxt[2];
    int sz;
    node() {
      nxt[0] = nxt[1] = NULL;
      sz = 0;
    }
  }*root;
  Trie() {
    root = new node();
  }
  void insert(int val) {
    node* cur = root;
    cur -> sz++;
    for (int i = B - 1; i >= 0; i--) {
      int b = val >> i & 1;
      if (cur -> nxt[b] == NULL) cur -> nxt[b] = new node();
      cur = cur -> nxt[b];
      cur -> sz++;
    }
  }
  int query(int x, int k) { // number of values s.t. val ^ x < k
    node* cur = root;
    int ans = 0;
    for (int i = B - 1; i >= 0; i--) {
      if (cur == NULL) break;
      int b1 = x >> i & 1, b2 = k >> i & 1;
      if (b2 == 1) {
        if (cur -> nxt[b1]) ans += cur -> nxt[b1] -> sz;
        cur = cur -> nxt[!b1];
      } else cur = cur -> nxt[b1];
    }
    return ans;
  }
  int get_max(int x) { // returns maximum of val ^ x
    node* cur = root;
    int ans = 0;
    for (int i = B - 1; i >= 0; i--) {
      int k = x >> i & 1;
      if (cur -> nxt[!k]) cur = cur -> nxt[!k], ans <<= 1,
          ans++;
      else cur = cur -> nxt[k], ans <<= 1;
    }
    return ans;
  }
  int get_min(int x) { // returns minimum of val ^ x
    node* cur = root;
    int ans = 0;
    for (int i = B - 1; i >= 0; i--) {
      int k = x >> i & 1;
      if (cur -> nxt[k]) cur = cur -> nxt[k], ans <<= 1;
      else cur = cur -> nxt[!k], ans <<= 1, ans++;
    }
    return ans;
  }
  void del(node* cur) {
    for (int i = 0; i < 2; i++) if (cur -> nxt[i]) del(cur ->
        nxt[i]);
    delete(cur);
  }
} t;
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, k;
  cin >> n >> k;
  int cur = 0;
  long long ans = 1LL * n * (n + 1) / 2;
  t.insert(cur);
  for (int i = 0; i < n; i++) {
    int x;
    cin >> x;
    cur ^= x;
    ans -= t.query(cur, k);
    t.insert(cur);
  }
  cout << ans << '\n';
  return 0;
}
```

2 Dynamic Programming

2.1 Array Description

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

const int N = 1e5 + 9, mod = 1e9 + 7;
int n, m, a[N], dp[N][105];
int f(int i, int last) {
  if(i == n + 1) return 1;
  if(dp[i][last] != -1) return dp[i][last];
  int ans = 0, l, r;
  if(a[i] > 0) {
    l = r = a[i];
  }
  else {
    if(i == 1) {
      l = 1, r = m;
    }
    else {
      l = max(1, last - 1);
      r = min(last + 1, m);
    }
  }
  for(int cur = l; cur <= r; cur++) {
    if(i == 1 or abs(last - cur) <= 1) {
      (ans += f(i + 1, cur)) %= mod;
```

```
    }
  }
  return dp[i][last] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> m;
  for(int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  memset(dp, -1, sizeof dp);
  cout << f(1, 0) << "\n";
  return 0;
}
/*You know that an array has n integers between 1 and m,
and the absolute difference between two adjacent values is at
      most 1.
Given a description of the array where some values may be
      unknown,
your task is to count the number of arrays that match the
      description.*/
```

2.2 Book Shop

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

const int N = 1010, E = 100005;
int dp[N][E];
int v[N], p[N];

int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  int n, m;
  cin >> n >> m;
  for(int i = 1; i <= n; i++) {
    cin >> v[i];
  }
  for(int i = 1; i <= n; i++) {
    cin >> p[i];
  }
  dp[0][0] = 0;
  for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= m; j++) {
      dp[i][j] = dp[i - 1][j];
      if(j - v[i] >= 0) {
        dp[i][j] = max(dp[i][j] , dp[i - 1][j - v[i]] + p[i]);
      }
    }
  }
  cout << dp[n][m] << "\n";
  return 0;
}
/*You are in a book shop which sells n different books.
You know the price and number of pages of each book.
You have decided that the total price of your purchases
will be at most x. What is the maximum number of pages you can
      buy?
You can buy each book at most once.*/
```

2.3 Coin Combinations

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

const int N = 105, X = 1e6 + 9, mod = 1e9 + 7;

int n, dp[X], v[N];
int coin_combination(int c) {
  if(c == 0) return 1;
  int &ans = dp[c];
  if(~ans) return ans;
  ans = 0;
  for(int i = 1; i <= n; i++) {
    if(c >= v[i]) {
      ans += coin_combination(c - v[i]);
      ans %= mod;
    }
  }
  return ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n; int m; cin >> m;
  for(int i = 1; i <= n; i++) {
    cin >> v[i];
  }
  memset(dp, -1, sizeof dp);
  cout << coin_combination(m) << "\n";
  return 0;
}
/*number of distinct ways you can produce a money sum x using
      the available coins.
if the coins are {2,3,5} and the desired sum is 9, there are 8
      ways:

2+2+5
2+5+2
5+2+2
3+3+3
2+2+2+3
2+2+3+2
2+3+2+2
3+2+2+2*/
```

2.4 Coins

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

const int N = 3001;
int n;
bool vis[N][N];
double dp[N][N], p[N];

double f(int i, int head, int tail) {
  if(i == n + 1) {
    if(head > tail) return 1;
    else return 0;
  }
  if(vis[i][head]) return dp[i][head];
  vis[i][head] = true;
  double ans = p[i] * f(i + 1, head + 1, tail);
  ans += (1 - p[i]) * f(i + 1, head, tail + 1);
  return dp[i][head] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  for(int i = 1; i <= n; i++) {
    cin >> p[i];
  }
  cout << fixed << setprecision(10) << f(1, 0, 0) << "\n";
  return 0;
}
/*There are N coins, numbered 1,2,,N. For each i (1  iN ), when
      Coin
i is tossed, it comes up heads with probability pi and tails
      with probability 1  pi.*/
```

2.5 Dice Combinations

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

#define int long long
const int N = 1e6 + 9, mod = 1e9 + 7;
int dp[N];
int ways(int n) {
  if(n < 0) return 0;
  if(n == 1) return 1;
  if(n == 2) return 2;
  if(n == 3) return 4;
  if(n == 4) return 8;
  if(n == 5) return 16;
  if(n == 6) return 32;
  if(dp[n] != -1) return dp[n];
  return dp[n] = (ways(n - 1) % mod + ways(n - 2) % mod +
       ways(n - 3) % mod + ways(n - 4) % mod + ways(n - 5) %
       mod + ways(n - 6) % mod) % mod;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
```

```
  int n;
  cin >> n;
  memset(dp, -1, sizeof dp);
  cout << ways(n) % mod << endl;
  return 0;
}
/*Your task is to count the number of ways to construct sum n
by throwing a dice one or more times. Each throw
produces an outcome between 1 and 6.*/
```

2.6 Frog1

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

#define int long long
const int N = 1e5 + 9, inf = 1e9;
int n, v[N], dp[N];
int frog (int i) {
  if(i > n) return inf;
  if(i == n) return 0;
  if(dp[i] != -1) return dp[i];
  int ans = inf;
  if(i + 1 <= n) ans = min(ans, frog(i + 1) + abs(v[i] - v[i +
       1]));
  if(i + 2 <= n) ans = min(ans, frog(i + 2) + abs(v[i] - v[i +
       2]));
  return dp[i] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  for(int i = 1; i <= n; i++) {
    cin >> v[i];
  }
  memset(dp, -1, sizeof dp);
  cout << frog(1) << endl;
  return 0;
}
/*If the frog is currently on Stone i,
jump to Stone i+1 or Stone i+2. Here,
a cost of  hi    hj    is incurred, where j is the stone to
     land on.*/
```

2.7 Frog2

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

#define int long long
const int N = 1e5 + 9, inf = 1e9;
#define dbg_(x) cout << #x << " = " << x << endl
int n, k, a[N], dp[N];
int frog2(int i) {
  if(i > n) return inf;
  if(i == n) return 0;
  if(dp[i] != -1) return dp[i];
  int ans = inf;
  for(int j = i + 1; j <= i + k; j++) {
    ans = min(ans, frog2(j) + abs(a[i] - a[j]));
  }
  return dp[i] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> k;
  for(int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  memset(dp, -1, sizeof dp);
  cout << frog2(1) << endl;
  return 0;
}
/*If the frog is currently on Stone i,
jump to Stone i+1, i+2...i+k (k <= 100). Here,
a cost of  hi    hj    is incurred, where j is the stone to
     land on.*/
```

2.8 Grid

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

const int N = 1005, mod = 1e9 + 7;
int n, m, dp[N][N];
char g[N][N];

int f(int i, int j) {
  if(i == n and j == m) return 1;
  if(dp[i][j] != -1) return dp[i][j];
  int ans = 0;
  if(i <= n and g[i + 1][j] == '.') (ans = f(i + 1, j)) %= mod;
  if(j <= m and g[i][j + 1] == '.') (ans += f(i, j + 1)) %= mod;
  return dp[i][j] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> m;
  for(int i = 1; i <= n; i++) {
    for(int j = 1; j <= m; j++) {
      cin >> g[i][j];
    }
  }
  memset(dp, -1, sizeof dp);
  cout << f(1, 1) % mod << endl;
  return 0;
}
//Print the number of Taro's paths from Square (1,1) to (H,W),
     modulo 10 ^ 9 +7.
```

2.9 Knapsack

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

#define int long long
const int N = 105, Wight = 1e5 + 9;
int n, W;
int dp[N][Wight], v[N], w[N];

int backpack(int i, int W) {
  if(i > n) return 0;
  if(dp[i][W] != -1) return dp[i][W];
  int ans = backpack(i + 1, W);
  if(W - w[i] >= 0) ans = max(ans, backpack(i + 1, W - w[i]) +
       v[i]);
  return dp[i][W] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> W;
  for(int i = 1; i <= n; i++) {
    cin >> w[i] >> v[i];
  }
  memset(dp, -1, sizeof dp);
  int ans = backpack(1, W);
  cout << ans << endl;
  return 0;
}
```

2.10 LCS

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

const int N = 3030;
string a, b;
int dp[N][N];
int f(int i, int j) {
  if(i >= a.size() || j >= b.size()) return 0;
  if(dp[i][j] != -1) return dp[i][j];
  int ans = 0;
  if(a[i] == b[j]) {
    ans = max(ans, f(i + 1, j + 1) + 1);
  }
  else {
    ans = max(ans, f(i + 1, j));
    ans = max(ans, f(i, j + 1));
  }
  return dp[i][j] = ans;
```

```
}
void print(int i, int j) {
  if(i >= a.size() || j >= b.size()) return;
  if(a[i] == b[j]) {
    cout << a[i];
    print(i + 1, j + 1);
    return;
  }
  int x = f(i + 1, j);
  int y = f(i, j + 1);
  if(x >= y) {
    print(i + 1, j);
  }
  else {
    print(i, j + 1);
  }
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> a >> b;
  memset(dp, -1, sizeof dp);
  print(0, 0);
  return 0;
}
```

2.11 Longest Path

```
#include <bits/stdc++.h>
using namespace std;
vector<int> dp(100001);
vector<vector<int>> adj(100001);
int dfs(int x) {
        if (dp[x]) return dp[x];
        for (auto e : adj[x]){
                        dp[e] = dfs(e);
                        dp[x] = max(dp[e] + 1, dp[x]);
        }
        return dp[x];
}
int main(){
        int n,m;
        cin >> n >> m;
        for(int i = 0; i < m; ++i) {
                int a, b;
                cin >> a >> b;
                a--; b--;
                adj[a].push_back(b);
        }
        for (int i = 0; i < n; ++i) {
                dfs(i);
        }
        int ans = 0;
        for (int i = 0;i < n; ++i) {
                ans = max(dp[i], ans);
        }
        cout << ans;
}
// length of the longest directed path in a Directed Acyclic
     Graph (DAG),
/*There is a directed graph G with N vertices and
M edges. The vertices are numbered 1,2,,N, and for each i
     (1 iM ), the
i-th directed edge goes from Vertex xi to yi. G does not
     contain directed cycles.
Find the length of the longest directed path in
G. Here, the length of a directed path is the number of edges
     in it.*/
```

2.12 Minimizing Coins

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

const int N = 105, X = 1e6 + 9, inf = 1e9;
int n, x, a[N];
int dp[N][X];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cin >> n >> x;
  for (int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  dp[0][0] = 0;
  for (int sum = 1; sum <= x; sum++) {
    dp[0][sum] = inf;
  }
  for (int i = 1; i <= n; i++) {
    for (int sum = 0; sum <= x; sum++) {
      dp[i][sum] = dp[i - 1][sum];
      if (sum >= a[i]) dp[i][sum] = min(dp[i][sum], dp[i][sum -
          a[i]] + 1);
    }
  }
  cout << (dp[n][x] >= inf ? -1 : dp[n][x]) << '\n';
  return 0;
}
/*Your task is to produce a sum of money x using the available
coins in such a way that the number of coins is minimal.
if the coins are {1,5,7} and the desired sum is 11, an
optimal solution is 5+5+1 which requires 3 coins.*/
```

2.13 Money Sum

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

const int N = 105, MX_SUM = 1e5 + 9;
int a[N], n;
bool dp[N][MX_SUM], vis[N][MX_SUM];
int f (int i, int sum) {
  if(i == n + 1) return sum == 0;
  if(vis[i][sum]) return dp[i][sum];
  bool is_possible = f(i + 1, sum);
  if(sum >= a[i]) is_possible |= f(i + 1, sum - a[i]);
  vis[i][sum] = true;
  return dp[i][sum] = is_possible;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  for(int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  std::vector<int> ans;
  for(int sum = 1; sum < MX_SUM; sum++) {
    if(f(1, sum)) {
      ans.push_back(sum);
    }
  }
  cout << ans.size() << "\n";
  for(auto it : ans) {
    cout << it << " ";
  }
  cout << endl;
  return 0;
}
/*You have n coins with certain values.
Your task is to find all money sums you can create using these
     coins.*/
```

2.14 Removal Game

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

#define int long long
const int N = 5005;
int a[N], n;
int dp[N][N], dp1[N][N];

int s(int i, int j);
int f(int i, int j) {
  if(i > j) return 0;
  if(dp[i][j] != -1) return dp[i][j];
  int u = a[i] + s(i + 1, j);
  int v = a[j] + s(i, j - 1);
  return dp[i][j] = max(u, v);
}
int s(int i, int j) {
  if(i > j) return 0;
  if(dp1[i][j] != -1) return dp1[i][j];
  int u = f(i + 1, j);
  int v = f(i, j - 1) ;
  return dp1[i][j] = min(u, v);
}
```

```
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  for(int i = 1; i <= n; i++) {
    cin >> a[i];
  }
  memset(dp, -1, sizeof dp);
  memset(dp1, -1, sizeof dp1);
  cout << f(1, n) << "\n";
  return 0;
}
/*There is a list of n numbers and two players who move
     alternately.
On each move, a player removes either the first or last number
     from the list,
and their score increases by that number. Both players try to
     maximize their scores.
What is the maximum possible score for the first player when
     both players play optimally?*/
```

2.15 Removing Digits

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

const int N = 1e6 + 9, inf = 1e9;
int dp[N];
int RDX (int n) {
  if(n < 0) return inf;
  if(n == 0) return 0;
  if(dp[n] != -1) return dp[n];
  int ans = inf;
  int f = n;
  while (n != 0) {
    int r = n % 10;
    if(r != 0) ans = min(ans, RDX(f - r) + 1);
    n = n / 10;
  }
  return dp[f] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  int n;
  cin >> n;
  memset(dp, -1, sizeof dp);
  cout << RDX(n) << endl;
  return 0;
}
/*You are given an integer n. On each step, you may subtract
one of the digits from the number.
How many steps are required to make the number equal to 0?*/
```

2.16 Two Sets Equal Sum

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

const int N = 505, M = 125005, mod = 1e9 + 7;
long long n, dp[N][M], total;

int f (int i, long long s) {
  if(i == n) {
    if(2 * s == total) return 1;
    else return 0;
  }
  long long &ans = dp[i][s];
  if(ans != -1) return ans;
  ans = (f(i + 1, s + i) % mod);
  ans += (f(i + 1, s) % mod);
  return ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  total = n * (n + 1) / 2;
  if(total % 2) {
    cout << 0 << "\n";
    return 0;
  }
  memset(dp, -1, sizeof dp);
  cout << (f(1, 0) % mod) << "\n";
  return 0;
}
/*Your task is to count the number of ways numbers 1,2,....,n
     can be divided into
two sets of equal sum. For example, if n=7, there are four
     solutions:

{1,3,4,6} and {2,5,7}
{1,2,5,6} and {3,4,7}
{1,2,4,7} and {3,5,6}
{1,6,7} and {2,3,4,5}*/
```

2.17 Vaction

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

const int N = 1e5 + 5;
int dp[N][5];
int n, a[N], b[N], c[N];

int f(int i, int last) {
  if(i == n + 1) return 0;
  if(dp[i][last] != -1) return dp[i][last];
  int ans = 0;
  for(int j = 1; j <= 3; j++) {
    if(j != last and j == 1) ans = max(ans, f(i + 1, j) + a[i]);
    if(j != last and j == 2) ans = max(ans, f(i + 1, j) + b[i]);
    if(j != last and j == 3) ans = max(ans, f(i + 1, j) + c[i]);
  }
  return dp[i][last] = ans;
}
int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cin >> n;
  for(int i = 1; i <= n; i++) {
    cin >> a[i] >> b[i] >> c[i];
  }
  memset(dp, -1, sizeof dp);
  cout << f(1, 0) << '\n';
  return 0;
}

/*The vacation consists of N days. For each i (1iN),
Taro will choose one of the following activities
Find the maximum possible total points of happiness that Taro
     gains.*/
```

3 Graphs

3.1 BFS

```
/*************************************************************************
* BFS (BREADTH-FIRST SEARCH)                                             *
* Time complexity: O(V+E)                                                *
* Usage: bfs(node)                                                       *
* Notation: s: starting node                                             *
*          adj[i]: adjacency list for node i                             *
*          vis[i]: visited state for node i (0 or 1)                     *
*************************************************************************/

const int N = 1e5+10; // Maximum number of nodes
int dist[N], par[N];
vector <int> adj[N];
queue <int> q;

void bfs (int s) {
       memset(dist, 63, sizeof(dist));
       dist[s] = 0;
       q.push(s);

       while (!q.empty()) {
              int u = q.front(); q.pop();
              for (auto v : adj[u]) if (dist[v] > dist[u] + 1)
                   {
```

```
                    par[v] = u;
                    dist[v] = dist[u] + 1;
                    q.push(v);
                }
        }
}
```

3.2 Bellman Ford

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

const int N = 3e5 + 9;

struct st {
  int a, b, cost;
} e[N];
const int INF = 2e9;
int32_t main() {
  int n, m;
  cin >> n >> m;
  for(int i = 0; i < m; i++) cin >> e[i].a >> e[i].b >>
       e[i].cost;
  int s;
  cin >> s;//is there any negative cycle which is reachable
       from s?
  vector<int> d (n, INF);//for finding any cycle(not
       necessarily from s) set d[i] = 0 for all i
  d[s] = 0;
  vector<int> p (n, -1);
  int x;
  for (int i=0; i<n; ++i) {
    x = -1;
    for (int j=0; j<m; ++j) {
      if (d[e[j].a] < INF) {
        if (d[e[j].b] > d[e[j].a] + e[j].cost) {
          d[e[j].b] = max (-INF, d[e[j].a] + e[j].cost);//for
               overflow
          p[e[j].b] = e[j].a;
          x = e[j].b;
        }
      }
    }
  }
  if (x == -1) cout << "No negative cycle from "<<s;
  else {
    int y = x; //x can be on any cycle or reachable from some
         cycle
    for (int i=0; i<n; ++i) y = p[y];

    vector<int> path;
    for (int cur=y; ; cur=p[cur]) {
      path.push_back (cur);
      if (cur == y && path.size() > 1) break;
    }
    reverse (path.begin(), path.end());
    cout << "Negative cycle: ";
    for (int i=0; i<path.size(); ++i) cout << path[i] << ' ';
  }
  return 0;
}
```

3.3 Bipartite Garphs

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

const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N]; int col[N];
bool ok;
void dfs(int u) {
        vis[u] = true;
        for (auto v: g[u]) {
                if (!vis[v]) {
                        col[v] = col[u] ^ 1;
                        dfs(v);
                }
                else {
                        if (col[u] == col[v]) {
                                ok = false;
                        }
                }
        }
}
int32_t main() {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        int n, m; cin >> n >> m;
        while (m--) {
                int u, v; cin >> u >> v;
                g[u].push_back(v);
                g[v].push_back(u);
        }
        ok = true;
        for (int i = 1; i <= n; i++) {
                if (!vis[i]) dfs(i);
        }
        if (ok) {
                cout << "YES\n";
        }
        else {
                cout << "NO\n";
        }
}
```

3.4 Cycle Detection

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

const int N = 5e5 + 9;

vector<pair<int, int>> g[N];
int vis[N], par[N], e_id[N];
vector<int> cycle; // simple cycle, contains edge ids

bool dfs(int u) {
  if (!cycle.empty()) return 1;
  vis[u] = 1;
  for (auto [v, id] : g[u]) {
    if (v != par[u]) {
      if (vis[v] == 0) {
        par[v] = u;
        e_id[v] = id;
        if (dfs(v)) return 1;
      }
      else if (vis[v] == 1) {
        // cycle here
        cycle.push_back(id);
        for (int x = u; x != v; x = par[x]) {
          cycle.push_back(e_id[x]);
        }
        return 1;
      }
    }
  }
  vis[u] = 2;
  return 0;
}

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  for (int i = 1; i <= m; i++) {
    int u, v; cin >> u >> v;
    ++u; ++v;
    g[u].push_back({v, i});
  }
  for (int u = 1; u <= n; u++) {
    if (vis[u] == 0 and dfs(u)) {
      cout << cycle.size() << '\n';
      for (auto x: cycle) cout << x - 1 << '\n';
      return 0;
    }
  }
  cout << -1 << '\n';
  return 0;
}
// How to check if an undirected graph has a cycle or not?
// https://judge.yosupo.jp/problem/cycle_detection
```

3.5 DFS

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

```cpp
const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N];
void dfs(int u) {
  vis[u] = true;
  for (auto v: g[u]) {
    if (!vis[v]) {
      dfs(v);
    }
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs(u);
}
```

3.6 Dijkstra

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

const int N = 3e5 + 9, mod = 998244353;

int n, m;
vector<pair<int, int>> g[N], r[N];
vector<long long> dijkstra(int s, int t, vector<int> &cnt) {
  const long long inf = 1e18;
  priority_queue<pair<long long, int>, vector<pair<long long,
       int>>, greater<pair<long long, int>>> q;
  vector<long long> d(n + 1, inf);
  vector<bool> vis(n + 1, 0);
  q.push({0, s});
  d[s] = 0;
  cnt.resize(n + 1, 0); // number of shortest paths
  cnt[s] = 1;
  while(!q.empty()) {
    auto x = q.top();
    q.pop();
    int u = x.second;
    if(vis[u]) continue;
    vis[u] = 1;
    for(auto y: g[u]) {
      int v = y.first;
      long long w = y.second;
      if(d[u] + w < d[v]) {
        d[v] = d[u] + w;
        q.push({d[v], v});
        cnt[v] = cnt[u];
      } else if(d[u] + w == d[v]) cnt[v] = (cnt[v] + cnt[u]) %
           mod;
    }
  }
  return d;
}

int u[N], v[N], w[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int s, t;
  cin >> n >> m >> s >> t;
  for(int i = 1; i <= m; i++) {
    cin >> u[i] >> v[i] >> w[i];
    g[u[i]].push_back({v[i], w[i]});
    r[v[i]].push_back({u[i], w[i]});
  }
  vector<int> cnt1, cnt2;
  auto d1 = dijkstra(s, t, cnt1);
  auto d2 = dijkstra(t, s, cnt2);

  long long ans = d1[t];
  for(int i = 1; i <= m; i++) {
    int x = u[i], y = v[i];
    long long nw = d1[x] + w[i] + d2[y];
    if(nw == ans && 1LL * cnt1[x] * cnt2[y] % mod == cnt1[t])
         cout << "YES\n";
    else if(nw - ans + 1 < w[i]) cout << "CAN " << nw - ans + 1
         << '\n';
    else cout << "NO\n";
  }
  return 0;
}
```

3.7 Floyd Warshall

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

const int N = 105;
int d[N][N];
int main() {
  int n = 10;
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n; j++) {
      if (i != j) {
        d[i][j] = 1e9;
      }
    }
  }
  for (int k = 1; k <= n; ++k) {
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) {
        d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
      }
    }
  }
  return 0;
}
```

3.8 Krushkals MST

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

const int N = 3e5 + 9, mod = 1e9;

struct dsu {
  vector<int> par, rnk, size; int c;
  dsu(int n) : par(n+1), rnk(n+1,0), size(n+1,1), c(n) {
    for (int i = 1; i <= n; ++i) par[i] = i;
  }
  int find(int i) { return (par[i] == i ? i : (par[i] =
       find(par[i]))); }
  bool same(int i, int j) { return find(i) == find(j); }
  int get_size(int i) { return size[find(i)]; }
  int count() { return c; } //connected components
  int merge(int i, int j) {
    if ((i = find(i)) == (j = find(j))) return -1; else --c;
    if (rnk[i] > rnk[j]) swap(i, j);
    par[i] = j; size[j] += size[i];
    if (rnk[i] == rnk[j]) rnk[j]++;
    return j;
  }
};

int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  vector<array<int, 3>> ed;
  for(int i = 1; i <= m; i++){
    int u, v, w; cin >> u >> v >> w;
    ed.push_back({w, u , v});
  }
  sort(ed.begin(), ed.end());
  long long ans = 0;
  dsu d(n);
  for (auto e: ed){
    int u = e[1], v = e[2], w = e[0];
    if (d.same(u, v)) continue;
    ans += w;
    d.merge(u, v);
  }
  cout << ans << '\n';
  return 0;
}
```

3.9 LCA

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

const int N = 3e5 + 9, LG = 18;

vector<int> g[N];
int par[N][LG + 1], dep[N], sz[N];
void dfs(int u, int p = 0) {
  par[u][0] = p;
  dep[u] = dep[p] + 1;
  sz[u] = 1;
  for (int i = 1; i <= LG; i++) par[u][i] = par[par[u][i -
       1]][i - 1];
  for (auto v: g[u]) if (v != p) {
    dfs(v, u);
    sz[u] += sz[v];
  }
}
int lca(int u, int v) {
  if (dep[u] < dep[v]) swap(u, v);
  for (int k = LG; k >= 0; k--) if (dep[par[u][k]] >= dep[v]) u
       = par[u][k];
  if (u == v) return u;
  for (int k = LG; k >= 0; k--) if (par[u][k] != par[v][k]) u =
       par[u][k], v = par[v][k];
  return par[u][0];
}
int kth(int u, int k) {
  assert(k >= 0);
  for (int i = 0; i <= LG; i++) if (k & (1 << i)) u = par[u][i];
  return u;
}
int dist(int u, int v) {
  int l = lca(u, v);
  return dep[u] + dep[v] - (dep[l] << 1);
}
//kth node from u to v, 0th node is u
int go(int u, int v, int k) {
  int l = lca(u, v);
  int d = dep[u] + dep[v] - (dep[l] << 1);
  assert(k <= d);
  if (dep[l] + k <= dep[u]) return kth(u, k);
  k -= dep[u] - dep[l];
  return kth(v, dep[v] - dep[l] - k);
}
int32_t main() {
  int n; cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs(1);
  int q; cin >> q;
  while (q--) {
    int u, v; cin >> u >> v;
    cout << dist(u, v) << '\n';
  }
  return 0;
}
```

3.10 Prims MST

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

const int N = 2020;
int g[N][N], w[N], to[N], selected[N];
long long Prims(int n, vector< pair<int, int> > &edges) {
  long long ans = 0;
  for(int i = 1; i <= n; i++) w[i] = 1e9, selected[i] = 0,
       to[i] = -1;
  w[1] = 0;
  for(int i = 1; i <= n; i++) {
    int u = -1;
    for(int j = 1; j <= n; j++) if(!selected[j] && (u == -1 ||
         w[j] < w[u])) u = j;
    if (w[u] == 1e9) return - 1; //NO MST
    selected[u] = 1;
    ans += w[u];
    if(to[u] != -1) edges.emplace_back(u, to[u]); //order of
         the edges may be changed
    for(int v = 1; v <= n; v++) if(g[u][v] < w[v]) w[v] =
         g[u][v], to[v] = u;
  }
  return ans;
}
string s[N];
int main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int n, m; cin >> n >> m;
  for(int i = 1; i <= n; i++) for(int j = 1; j <= n; j++)
       g[i][j] = 1e9;
  for(int i = 1; i <= n; i++) cin >> s[i];
  for(int i = 1; i <= n; i++){
    for(int j = i + 1; j <= n; j++){
      int w = 0;
      for(int k = 0; k < m; k++) w = max(w, (int)abs(s[i][k] -
           s[j][k]));
      g[i][j] = min(g[i][j], w);
      g[j][i] = min(g[j][i], w);
    }
  }
  vector< pair<int, int> > ed;
  long long ans = Prims(n, ed);
  int res = 0; for(auto e: ed) res = max(res,
       g[e.first][e.second]);
  cout << res << '\n';
  return 0;
}
/*
https://www.codechef.com/ICL2016/problems/ICL16A
*/
```

3.11 Strongly Connected Components

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 3e5 + 9;

// given a directed graph return the minimum number of edges to
     be added so that the whole graph become an SCC
bool vis[N];
vector<int> g[N], r[N], G[N], vec; //G is the condensed graph
void dfs1(int u) {
  vis[u] = 1;
  for(auto v: g[u]) if(!vis[v]) dfs1(v);
  vec.push_back(u);
}

vector<int> comp;
void dfs2(int u) {
  comp.push_back(u);
  vis[u] = 1;
  for(auto v: r[u]) if(!vis[v]) dfs2(v);
}

int idx[N], in[N], out[N];
int main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);

  int n, m;
  cin >> n >> m;
  for(int i = 1; i <= m; i++) {
    int u, v;
    cin >> u >> v;
    g[u].push_back(v);
    r[v].push_back(u);
  }
  for(int i = 1; i <= n; i++) if(!vis[i]) dfs1(i);
  reverse(vec.begin(), vec.end());
  memset(vis, 0, sizeof vis);
  int scc = 0;
  for(auto u: vec) {
    if(!vis[u]) {
      comp.clear();
      dfs2(u);
      scc++;
      for(auto x: comp) idx[x]=scc;
    }
  }
  for(int u = 1; u <= n; u++) {
    for(auto v: g[u]) {
      if(idx[u] != idx[v]) {
        in[idx[v]]++, out[idx[u]]++;
        G[idx[u]].push_back(idx[v]);
      }
    }
  }
  int needed_in=0, needed_out=0;
  for(int i = 1; i <= scc; i++) {
    if(!in[i]) needed_in++;
```

```
    if(!out[i]) needed_out++;
  }
  int ans = max(needed_in, needed_out);
  if(scc == 1) ans = 0;
  cout << ans << '\n';
  return 0;
}
```

3.12 Topological Sorting

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

const int N = 1e5 + 9;
vector<int> g[N];
bool vis[N];
vector<int> ord;
void dfs(int u) {
  vis[u] = true;
  for (auto v: g[u]) {
    if (!vis[v]) {
      dfs(v);
    }
  }
  ord.push_back(u);
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n, m; cin >> n >> m;
  while (m--) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
  }
  for (int i = 1; i <= n; i++) {
    if (!vis[i]) {
      dfs(i);
    }
  }
  reverse(ord.begin(), ord.end());

  // check is feasible
  vector<int> pos(n + 1);
  for (int i = 0; i < (int) ord.size(); i++) {
    pos[ord[i]] = i;
  }
  for (int u = 1; u <= n; u++) {
    for (auto v: g[u]) {
      if (pos[u] > pos[v]) {
        cout << "IMPOSSIBLE\n";
        return 0;
      }
    }
  }

  // print the order
  for (auto u: ord) cout << u << ' ';
  cout << '\n';
  return 0;
}
// https://cses.fi/problemset/task/1679
```

3.13 Tree Diameter

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

const int N = 2e5 + 9;

vector<int> g[N];
int farthest(int s, int n, vector<int> &d) {
  static const int inf = N;
  d.assign(n + 1, inf); d[s] = 0;
  vector<bool> vis(n + 1);
  queue<int> q; q.push(s);
  vis[s] = 1; int last = s;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (int v: g[u]) {
      if (vis[v]) continue;
      d[v] = d[u] + 1;
      q.push(v); vis[v] = 1;
    }
    last = u;
  }
  return last;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  int n; cin >> n;
  for (int i = 1; i < n; i++) {
    int u, v; cin >> u >> v;
    g[u].push_back(v);
    g[v].push_back(u);
  }
  vector<int> dx, dy;
  int x = farthest(1, n, dx);
  int y = farthest(x, n, dx);
  farthest(y, n, dy);
  for (int i = 1; i <= n; i++) {
    cout << max(dx[i], dy[i]) << ' ';
  }
  cout << '\n';
  return 0;
}
// https://cses.fi/problemset/task/1132
```

3.14 Zero One BFS

```
// 0-1 BFS - O(V+E)
const int N = 1e5 + 5;

int dist[N];
vector<pii> adj[N];
deque<pii> dq;

void zero_one_bfs (int x){
        cl(dist, 63);
        dist[x] = 0;
        dq.push_back({x, 0});
        while(!dq.empty()){
                int u = dq.front().st;
                int ud = dq.front().nd;
                dq.pop_front();
                if(dist[u] < ud) continue;
                for(auto x : adj[u]){
                        int v = x.st;
                        int w = x.nd;
                        if(dist[u] + w < dist[v]){
                                dist[v] = dist[u] + w;
                                if(w) dq.push_back({v, dist[v]});
                                else dq.push_front({v, dist[v]});
                        }
                }
        }
}
```

4 Mathematics

4.1 BIGMOD

```
// Big Mod
long long binpow(long long a, long long b, long long m) {
        a %= m;
        long long res = 1;
        while (b > 0) {
                if (b & 1)
                        res = res * a % m;
                a = a * a % m;
                b >>= 1;
        }
        return res;
}
// Binary Multiplication with Mod
long long binmul(long long a, long long b, long long m) {
        long long res = 0LL;
        a = a % m;
        while (b > 0) {
                if (b & 1) res = (res + a) % m;
                a = (a + a) % m;
                b >>= 1;
        }
        return res;
}
```

4.2 Basics

```
// Greatest Common Divisor & Lowest Common Multiple
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; }

// Multiply caring overflow
ll mulmod(ll a, ll b, ll m = MOD) {
	ll r=0;
	for (a %= m; b; b>>=1, a=(a*2)%m) if (b&1) r=(r+a)%m;
	return r;
}

// Another option for mulmod is using long double
ull mulmod(ull a, ull b, ull m = MOD) {
	ull q = (ld) a * (ld) b / (ld) m;
	ull r = a * b - q * m;
	return (r + m) % m;
}

// Fast exponential
ll fexp(ll a, ll b, ll m = MOD) {
	ll r=1;
	for (a %= m; b; b>>=1, a=(a*a)%m) if (b&1) r=(r*a)%m;
	return r;
}
```

4.3 Combinatorics Basics

```
const int N = 2e5 + 9, mod = 998244353; // change this
     constrains
long long fact[N], finv[N], inv[N];

void precal_factorial (int n) {
  inv[0] = inv[1] = finv[0] = finv[1] = fact[0] = fact[1] = 1;
  for(int i = 2; i <= n; i++) {
    inv[i] = inv[mod % i] * (mod - mod / i) % mod;
    finv[i] = (finv[i-1] * inv[i]) % mod;
    fact[i] = (fact[i-1] * i) % mod;
  }
}
long long ncr (long long n, long long r) {
  if(r > n) return 0;
  long long a = (finv[r] * finv[n - r]) % mod;
  return (a * fact[n]) % mod;
}
```

4.4 Counting Digits

```
#include <bits/stdc++.h>
using namespace std;
int findDigits(int n){
	if (n < 0)
		return 0;
	if (n <= 1)
		return 1;
	double digits = 0;
	for (int i = 2; i <= n; i++)
		digits += log10(i);
	return floor(digits) + 1;
}
int main() {
	cout << findDigits(120) << endl;
	return 0;
}
```

4.5 Derangement

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

const int N = 1e6 + 9, mod = 1e9 + 7;

int d[N];
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  d[0] = 1; d[1] = 0;
  for (int i = 1; i < N; i++) {
    d[i] = 1LL * (i - 1) * (d[i - 1] + d[i - 2]) % mod;
  }
  int n; cin >> n;
  cout << d[n] << '\n';
  return 0;
}
/*There are n children at a Christmas party,
and each of them has brought a gift.
The idea is that everybody will get a gift brought by someone
     else.
In how many ways can the gifts be distributed?*/
```

4.6 Euler Phi

```
// Euler phi (totient)
int ind = 0, pf = primes[0], ans = n;
while (1ll*pf*pf <= n) {
	if (n%pf==0) ans -= ans/pf;
	while (n%pf==0) n /= pf;
	pf = primes[++ind];
}
if (n != 1) ans -= ans/n;

// Euler Totient Function 1 to n in O(nloglog(n))
const int N = 1e5 + 9;
int phi[N];
void totient() {
  for (int i = 1; i < N; i++) phi[i] = i;
  for (int i = 2; i < N; i++) {
    if (phi[i] == i) {
      for (int j = i; j < N; j += i) phi[j] -= phi[j] / i;
    }
  }
}
```

4.7 Fibonacci Number Faster

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

int fib(long long n, int mod) {
  assert (n >= 0);
  if (n <= 1) return n;
  int a = 0, b = 1;
  long long i = 1ll << (63 - __builtin_clzll(n) - 1);
  for (; i; i >>= 1) {
    int na = (a *(long long) a + b *(long long) b) % mod;
    int nb = (2ll * a + b) * b % mod;
    a = na; b = nb;
    if (n & i) {
      int c = a + b; if (c >= mod) c -= mod;
      a = b; b = c;
    }
  }
  return b;
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  cout << fib(10, 100) << '\n';
  return 0;
}
```

4.8 GCD LCM

```
int gcd(int a, int b){
	if(a == 0) return b;
	return gcd(b%a, a);
}
long long lcm(long long a, long long b) {
	return (a / __gcd(a, b)) * b;
}
```

4.9 Josephus

```
// UFMG
/* Josephus Problem - It returns the position to be, in order
     to not die. O(n)*/
```

```
/* With k=2, for instance, the game begins with 2 being killed
     and then n+2, n+4, ... */
ll josephus(ll n, ll k) {
       if(n==1) return 1;
       else return (josephus(n-1, k)+k-1)%n+1;
}

/* Another Way to compute the last position to be killed - O(d
     * log n) */
ll josephus(ll n, ll d) {
       ll K = 1;
       while (K <= (d - 1)*n) K = (d * K + d - 2) / (d - 1);
       return d * n + 1 - K;
}
```

4.10 Large Number GCD

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;
ll gcd(ll a, ll b) {
       if (!a)
              return b;
       return gcd(b % a, a);
}
ll reduceB(ll a, char b[]) {
       ll mod = 0;
       for (int i = 0; i < strlen(b); i++)
              mod = (mod * 10 + b[i] - '0') % a;
return mod; // return modulo
}
ll gcdLarge(ll a, char b[]) {
       ll num = reduceB(a, b);
       return gcd(a, num);
}
int main(){
       ll a = 1221;
       char b[] =
            "12345678910111213141516171819202122232425262728 29";
       if (a == 0)
              cout << b << endl;
       else
              cout << gcdLarge(a, b) << endl;
       return 0;
}
```

4.11 Lengenders Formula

```
// n and a prime number p, find the largest x such that px
     divides n! (factorial) in O(logn).
#include<bits/stdc++.h>
using namespace std;
int legendre(long long n, long long p) {
       int ans = 0;
       while (n) {
              ans += n / p;
              n /= p;
       }
       return ans;
}
int32_t main() {
       ios_base::sync_with_stdio(0);
       cin.tie(0);
       return 0;
}
```

4.12 Miller Rabin

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

using ll = long long;
namespace MillerRabin {
  mt19937
       rnd(chrono::steady_clock::now().time_since_epoch().count());
  const int P = 1e6 + 9;
  int primes[P], spf[P];
  inline ll mul_mod(ll x, ll y, ll m) {
    ll res = __int128(x) * y % m;
    return res;
    // ll res = x * y - (ll)((long double)x * y / m + 0.5) * m;
    // return res < 0 ? res + m : res;
  }
  inline ll pow_mod(ll x, ll n, ll m) {
    ll res = 1 % m;
    for (; n; n >>= 1) {
      if (n & 1) res = mul_mod(res, x, m);
      x = mul_mod(x, x, m);
    }
    return res;
  }
  // O(it * (logn)^3), it = number of rounds performed (but
       faster in practice)
  inline bool miller_rabin(ll n) {
    if (n <= 2 || (n & 1 ^ 1)) return (n == 2);
    if (n < P) return spf[n] == n;
    ll c, d, s = 0, r = n - 1;
    for (; !(r & 1); r >>= 1, s++) {}
    // each iteration is a round
    for (int i = 0; primes[i] < n && primes[i] < 32; i++) {
      c = pow_mod(primes[i], r, n);
      for (int j = 0; j < s; j++) {
        d = mul_mod(c, c, n);
        if (d == 1 && c != 1 && c != (n - 1)) return false;
        c = d;
      }
      if (c != 1) return false;
    }
    return true;
  }
  void init() {
    int cnt = 0;
    for (int i = 2; i < P; i++) {
      if (!spf[i]) primes[cnt++] = spf[i] = i;
      for (int j = 0, k; (k = i * primes[j]) < P; j++) {
        spf[k] = primes[j];
        if (spf[i] == spf[k]) break;
      }
    }
  }
}
int32_t main() {
  ios_base::sync_with_stdio(0);
  cin.tie(0);
  MillerRabin::init();
  int t; cin >> t;
  while (t--) {
    ll n; cin >> n;
    if (MillerRabin::miller_rabin(n)) {
      cout << "Yes\n";
    } else {
      cout << "No\n";
    }
  }
  return 0;
}
// https://judge.yosupo.jp/problem/primality_test
```

4.13 Modular Arithmetic

```
int vagsesh = (a % m - b % m + m) % m; // For Substraction
long long res = 1; // For Multiplicaion
for(int i = 1; i <= n; i++) {
       res = (res * a) % m;
}
cout << res << endl;
```

4.14 NOD

```
// Number of divisors using the prime factorization of n.
long long numberOfDivisors(long long num) {
       long long total = 1;
       for (int i = 2; (long long)i * i <= num; i++) {
              if (num % i == 0) {
                     int e = 0;
                     do {
                            e++;
                            num /= i;
                     } while (num % i == 0);
                     total *= e + 1;
              }
       }
       if (num > 1) {
              total *= 2;
       }
```

```
	return total;
}
```

4.15 Optimized Sieve

```
// (Fast Sieve, Using bit set, Works till 10^8 in less than 1s,
     Memory Complexity: O (n/64))
#include<bits/stdc++.h>
using namespace std;
const int N = 1e8 + 9;
bitset<N> f;
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	int n = N - 9;
	vector<int> primes;
	f[1] = true;
	for (int i = 2; i * i <= n; i++) {
		if (!f[i]) {
			for (int j = i * i; j <= n; j += i) {
				f[j] = true;
			}
		}
	}
	for (int i = 2; i <= n; i++) {
		if (!f[i]) {
			primes.push_back(i);
		}
	}
	cout << primes.size() << '\n';
	return 0;
}
```

4.16 Prime Factorization Spf

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6 + 9;
int spf[N];
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	for (int i = 2; i < N; i++) {
		spf[i] = i;
	}
	for (int i = 2; i < N; i++) {
		for (int j = i; j < N; j += i) {
			spf[j] = min(spf[j], i);
		}
	}
	int q; cin >> q; // queries q <= 1e6
	while (q--) {
		int n; cin >> n; // find prime factorization of
		    n <= 1e6
		vector<int> ans;
		while (n > 1) {
			ans.push_back(spf[n]);
			n /= spf[n];
		}
		for (auto x: ans) cout << x << ' '; cout << '\n';
	}
return 0;
}
```

4.17 Prime Factors

```
// Prime factors (up to 9*10^13. For greater see Pollard Rho)
vi factors;
int ind=0, pf = primes[0];
while (pf*pf <= n) {
	while (n%pf == 0) n /= pf, factors.pb(pf);
	pf = primes[++ind];
}
if (n != 1) factors.pb(n);
```

4.18 SOD

```
// Sum of divisors using the prime factorization of n.
long long SumOfDivisors(long long num) {
	long long total = 1;
	for (int i = 2; (long long)i * i <= num; i++) {
		if (num % i == 0) {
			int e = 0;
			do {
				e++;
				num /= i;
			} while (num % i == 0);
			long long sum = 0, pow = 1;
			do {
				sum += pow;
				pow *= i;
			} while (e-- > 0);
			total *= sum;
		}
	}
	if (num > 1) {
		total *= (1 + num);
	}
	return total;
}
```

4.19 Sieve

```
// primes which are less than n in O (nloglog(n))
#include<bits/stdc++.h>
using namespace std;
const int N = 1e7 + 9;
bool f[N];
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	int n = N - 9;
	vector<int> primes;
	f[1] = true;
	for (int i = 2; i <= n; i++) {
		if (!f[i]) {
			primes.push_back(i);
			for (int j = i + i; j <= n; j += i) {
				f[j] = true;
			}
		}
	}
	cout << primes.size() << '\n';
	return 0;
}
```

4.20 TrailingZero

```
#include <iostream>
using namespace std;
int findTrailingZeros(int n){
if (n < 0) // Negative Number Edge Case
	return -1;
int count = 0;
for (int i = 5; n / i >= 1; i *= 5)
	count += n / i;
return count;
}
int main(){
	int n = 100;
	cout << findTrailingZeros(n);
	return 0;
}
```

4.21 Upto n NOD

```
#include<bits/stdc++.h>
using namespace std;
int d[104];
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	int n = 100;
	for (int i = 1; i <= n; i++) {
		for (int j = i; j <= n; j += i) {
			d[j]++;
// d[j] += i // for sum of divisors
		}
	}
```

```
        for (int i = 1; i <= n; i++) {
                cout << d[i] << ' ';
        }
        return 0;
}
```

5 Miscellaneous

5.1 Base Conberstion

```
string base_convert(int n, int b) {
  string s = "";
  while(n > 0) {
    s = to_string(n % b) + s;
    n /= b;
  }
  return s;
}
int convert_to_decimal(string s , int base) {
  int n = 0 , power = 1;
  for(int i = (int) s.size() - 1; i >= 0 ; i--) {
    n += power * (s[i] - '0');
    power *= base;
  }
  return n;
}
```

5.2 Bitset

```
//Goes through the subsets of a set x :
int b = 0;
do {
// process subset b
} while (b=(b-x)&x);
```

5.3 Bitwise Property

```
const int inf = numeric_limits <int> :: max() - 5;
#define int long long

struct ST {
  int AND(int a, int b) {
    return (a & b);
  }
  int OR(int a, int b) {
    return (a | b);
  }
  int XOR(int a, int b) {
    return (a ^ b);
  }
  int right_shift (int a , int b) {
    return (a >> b);
  }
  int left_shift (int a , int b) {
    return (a << b);
  }
  int bitwise_not (int a) {
    return (~a);
  }
  int ON_BIT(int a) {
    return __builtin_popcount(a);
  }
  int leading_zero (int a) {
    return __builtin_clz(a);
  }
  int tailing_zero (int a) {
    return __builtin_ctz(a);
  }
  int LSB (int a) {
    return (a & 1);
  }
  bool kth_bit(int a, int k) {
    return (a & (1 << k));
  }
  int msb(int a) {
    return a ? 32 - __builtin_clz(a) : 0;
  }
  string base_convert(int n, int b) {
    string s = ""; while(n > 0) { s = to_string(n % b) + s; n
        /= b;} return s;
  }
  int convert_to_decimal(string s , int base) {
    int n = 0 , power = 1; for(int i = (int) s.size() - 1; i >=
        0 ; i--) { n += power * (s[i] - '0'); power *= base;}
        return n;
  }
}bit;
```

5.4 Builtin

```
__builtin_ctz(x) // trailing zeroes
__builtin_clz(x) // leading zeroes
__builtin_popcount(x) // # bits set
__builtin_ffs(x) // index(LSB) + 1 [0 if x==0]

// Add ll to the end for long long [__builtin_clzll(x)]
```

5.5 Merge Sort

```
// Merge-sort with inversion count - O(nlog n)

int n, inv;
vector<int> v, ans;

void mergesort(int l, int r, vector<int> &v){
       if(l == r) return;
       int mid = (l+r)/2;
       mergesort(l, mid, v), mergesort(mid+1, r, v);
       int i = l, j = mid + 1, k = l;
       while(i <= mid or j <= r){
              if(i <= mid and (j > r or v[i] <= v[j]))
                   ans[k++] = v[i++];
              else ans[k++] = v[j++], inv += j-k;
       }
       for(int i = l; i <= r; i++) v[i] = ans[i];
}

//in main
ans.resize(v.size());
```

5.6 STL

```
auto [a, b] = p;
pair<int, int> p;
pair<int, pair<int, int>> p3;
deque<int> dq; dq.push_front(1); dq.push_back(3); pop_front();
     pop_back();
stack<int> st; st.push(2); st.pop(); st.top();st.size();
queue<int> q; q.push(5); q.pop();
     q.front();q.back();size();empty();
set < int > s; insert(); erase(); begin(); end(); size();
     count(); empty();
priority_queue<int> pq; pq.push(1); top(); pop(); size();
multiset < int > m; insert(); erase(); begin(); end(); size();
     count(); empty();
map<int, int> mp1;
map<int, pair<int, int>> mp2; mp2[0].first; mp2[0].second;
int index = lower_bound(v.begin(), v.end(), val) - v.begin()
int index = upper_bound(v.begin(), v.end(), val) - v.begin();
auto it = s.lower_bound(6); *it; it--;
auto it = s.upper_bound(6); *it; it--;
```

5.7 Sqrt Decomposition

```
// Square Root Decomposition (Mo's Algorithm) - O(n^(3/2))
const int N = 1e5+1, SQ = 500;
int n, m, v[N];

void add(int p) { /* add value to aggregated data structure */ }
void rem(int p) { /* remove value from aggregated data
     structure */ }

struct query { int i, l, r, ans; } qs[N];

bool c1(query a, query b) {
       if(a.l/SQ != b.l/SQ) return a.l < b.l;
       return a.l/SQ&1 ? a.r > b.r : a.r < b.r;
}
```

```
bool c2(query a, query b) { return a.i < b.i; }

/* inside main */
int l = 0, r = -1;
sort(qs, qs+m, c1);
for (int i = 0; i < m; ++i) {
	query &q = qs[i];
	while (r < q.r) add(v[++r]);
	while (r > q.r) rem(v[r--]);
	while (l < q.l) rem(v[l++]);
	while (l > q.l) add(v[--l]);

	q.ans = /* calculate answer */;
}

sort(qs, qs+m, c2); // sort to original order
```

5.8 python

```
# reopen
import sys
sys.stdout = open('out','w')
sys.stdin = open('in' ,'r')

//Dummy example
R = lambda: map(int, input().split())
n, k = R(),
v, t = [], [0]*n
for p, c, i in sorted(zip(R(), R(), range(n))):
	t[i] = sum(v)+c
	v += [c]
	v = sorted(v)[::-1]
	if len(v) > k:
		v.pop()
print(' '.join(map(str, t)))
```

6 Strings

6.1 Double Hashing

```
#include<bits/stdc++.h>
using namespace std;
const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 =
    987654319;
const int N = 1e5 + 9;
int pw1[N], pw2[N];
void prec() {
	pw1[0] = 1;
	for (int i = 1; i < N; i++) {
		pw1[i] = 1LL * pw1[i - 1] * p1 % mod1;
	}
	pw2[0] = 1;
	for (int i = 1; i < N; i++) {
		pw2[i] = 1LL * pw2[i - 1] * p2 % mod2;
	}
}
pair<int, int> get_hash(string s) {
	int n = s.size();
	int hs1 = 0;
	for (int i = 0; i < n; i++) {
		hs1 += 1LL * s[i] * pw1[i] % mod1;
		hs1 %= mod1;
	}
	int hs2 = 0;
	for (int i = 0; i < n; i++) {
		hs2 += 1LL * s[i] * pw2[i] % mod2;
		hs2 %= mod2;
	}
	return {hs1, hs2};
}
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	prec();
	string a, b; cin >> a >> b;
	cout << (get_hash(a) == get_hash(b)) << '\n';
	return 0;
}
```

6.2 Largest Substring More than K

```
#include <bits/stdc++.h>
using namespace std;
int n;
int max_oc(int len) {
	map<pair<int, int>, int> mp;
	for (int i = 0; i + len - 1 < n; i++) {
		mp[get_hash(i, i + len - 1)]++;
	}
	int ans = 0;
	for (auto [x, y]: mp) {
		ans = max(ans, y);
	}
	return ans;
}
int32_t main() {
	ios_base::sync_with_stdio(0);
	cin.tie(0);
	prec();
	string s; cin >> s;
	build(s);
	int k; cin >> k;
	n = s.size();
	int l = 1, r = s.size(), ans = -1;
	while (l <= r) {
		int mid = (l + r) >> 1;
		if (max_oc(mid) >= k) {
			ans = mid;
			l = mid + 1;
		}
		else {
			r = mid - 1;
		}
	}
	cout << ans << '\n';
	return 0;
}
```

6.3 Longest Common Prefix Of Two Substrings

```
int lcp(int i, int j, int x, int y) { // O(log n)
	int l = 1, r = min(j - i + 1, y - x + 1), ans = 0;
	while (l <= r) {
		int mid = l + r >> 1;
		if (get_hash(i, i + mid - 1) == get_hash(x, x +
		    mid - 1)) {
			ans = mid;
			l = mid + 1;
		}
		else {
			r = mid - 1;
		}
	}
	return ans;
}
/*given a string s of size n and q queries of type i, j, x, y.
find the LCP of Substring s[i...j] and s[x...y]. 1 <= n, q <=
    1e5*/
```

6.4 Longest Common Substring

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 9;
const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 =
    987654319;
int power(long long n, long long k, int mod) {
	int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
	while (k) {
		if (k & 1) ans = (long long) ans * n % mod;
		n = (long long) n * n % mod;
		k >>= 1;
	}
	return ans;
}
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() { // O(n)
	pw[0] = {1, 1};
	for (int i = 1; i < N; i++) {
		pw[i].first = 1LL * pw[i - 1].first * p1 % mod1;
```

```
                pw[i].second = 1LL * pw[i - 1].second * p2 %
                    mod2;
        }
        ip1 = power(p1, mod1 - 2, mod1);
        ip2 = power(p2, mod2 - 2, mod2);
        ipw[0] = {1, 1};
        for (int i = 1; i < N; i++) {
                ipw[i].first = 1LL * ipw[i - 1].first * ip1 %
                    mod1;
                ipw[i].second = 1LL * ipw[i - 1].second * ip2 %
                    mod2;
        }
}
pair<int, int> string_hash(string s) { // O(n)
        int n = s.size();
        pair<int, int> hs({0, 0});
        for (int i = 0; i < n; i++) {
                hs.first += 1LL * s[i] * pw[i].first % mod1;
                hs.first %= mod1;
                hs.second += 1LL * s[i] * pw[i].second % mod2;
                hs.second %= mod2;
        }
        return hs;
}
struct Hashing {
        pair<int, int> pref[N];
void build(string s) { // O(n)
        int n = s.size();
        for (int i = 0; i < n; i++) {
                pref[i].first = 1LL * s[i] * pw[i].first % mod1;
                if (i) pref[i].first = (pref[i].first + pref[i -
                    1].first) % mod1;
                pref[i].second = 1LL * s[i] * pw[i].second %
                    mod2;
                if (i) pref[i].second = (pref[i].second + pref[i
                    - 1].second) % mod2;
        }
}
pair<int, int> get_hash(int i, int j) { // O(1)
// assert(i <= j);
        pair<int, int> hs({0, 0});
        hs.first = pref[j].first;
        if (i) hs.first = (hs.first - pref[i - 1].first + mod1)
            % mod1;
        hs.first = 1LL * hs.first * ipw[i].first % mod1;
        hs.second = pref[j].second;
        if (i) hs.second = (hs.second - pref[i - 1].second +
            mod2) % mod2;
        hs.second = 1LL * hs.second * ipw[i].second % mod2;
        return hs;
}
}A, B;
int n;
string a, b;
string res;
bool ok(int k) { // is there a k length substring that occurs
    in both a and b
        set<pair<int, int>> substring_hashes_in_a;
        for (int i = 0; i + k - 1 < n; i++) {
                substring_hashes_in_a.insert(A.get_hash(i, i + k
                    - 1));
        }
        for (int i = 0; i + k - 1 < n; i++) {
                auto substring_hash_in_b = B.get_hash(i, i + k -
                    1);
                if
                    (substring_hashes_in_a.find(substring_hash_in_b)
                    != substring_hashes_in_a.end()) {
                        res = b.substr(i, k);
                        return true;
                }
        }
        return false;
}
int32_t main() {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        prec();
        cin >> n;
        cin >> a >> b;
        A.build(a);
        B.build(b);
        int l = 1, r = n, ans = 0;
        while (l <= r) {
                int mid = (l + r) / 2;
                if (ok(mid)) {
                        ans = mid;
                        l = mid + 1;
                }
                else {
                        r = mid - 1;
                }
        }
// cout << ans << '\n';
        ok(ans);
        cout << res << '\n';
// O(n log^2 n)
        return 0;
}
// Find the Longest Common Substring of Two Strings
```

6.5 Number Of Divisors Of String

6.6 Pattern Matching

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6 + 9;
const int p1 = 137, mod1 = 127657753, p2 = 277, mod2 =
    987654319;
int power(long long n, long long k, int mod) {
        int ans = 1 % mod; n %= mod; if (n < 0) n += mod;
        while (k) {
                if (k & 1) ans = (long long) ans * n % mod;
                n = (long long) n * n % mod;
                k >>= 1;
        }
        return ans;
}
int ip1, ip2;
pair<int, int> pw[N], ipw[N];
void prec() {
        pw[0] = {1, 1};
        for (int i = 1; i < N; i++) {
                pw[i].first = 1LL * pw[i - 1].first * p1 % mod1;
                pw[i].second = 1LL * pw[i - 1].second * p2 %
                    mod2;
        }
        ip1 = power(p1, mod1 - 2, mod1);
        ip2 = power(p2, mod2 - 2, mod2);
        ipw[0] = {1, 1};
        for (int i = 1; i < N; i++) {
                ipw[i].first = 1LL * ipw[i - 1].first * ip1 %
                    mod1;
                ipw[i].second = 1LL * ipw[i - 1].second * ip2 %
                    mod2;
        }
}
pair<int, int> string_hash(string s) {
        int n = s.size();
        pair<int, int> hs({0, 0});
        for (int i = 0; i < n; i++) {
                hs.first += 1LL * s[i] * pw[i].first % mod1;
                hs.first %= mod1;
                hs.second += 1LL * s[i] * pw[i].second % mod2;
                hs.second %= mod2;
        }
        return hs;
}
pair<int, int> pref[N];
void build(string s) {
        int n = s.size();
        for (int i = 0; i < n; i++) {
                pref[i].first = 1LL * s[i] * pw[i].first % mod1;
                if (i) pref[i].first = (pref[i].first + pref[i -
                    1].first) % mod1;
                pref[i].second = 1LL * s[i] * pw[i].second %
                    mod2;
                if (i) pref[i].second = (pref[i].second + pref[i
                    - 1].second) % mod2;
        }
}
pair<int, int> get_hash(int i, int j) {
        assert(i <= j);
        pair<int, int> hs({0, 0});
        hs.first = pref[j].first;
        if (i) hs.first = (hs.first - pref[i - 1].first + mod1)
            % mod1;
        hs.first = 1LL * hs.first * ipw[i].first % mod1;
        hs.second = pref[j].second;
        if (i) hs.second = (hs.second - pref[i - 1].second +
            mod2) % mod2;
```

```
        hs.second = 1LL * hs.second * ipw[i].second % mod2;
        return hs;
}
int32_t main() {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        prec();
        string a, b; cin >> a >> b;
        build(a);
        int ans = 0, n = a.size(), m = b.size();
        auto hash_b = string_hash(b);
        for (int i = 0; i + m - 1 < n; i++) {
                ans += get_hash(i, i + m - 1) == hash_b;
        }
        cout << ans << '\n';
        return 0;
}
/*Given a string a pattern, your task is to count
the number if positions where the pattern occurs in the string
Input :
saippuakauppias
pp
output :
2
*/
```

6.7 Two Strings Are Equal Or Not

```
#include<bits/stdc++.h>
using namespace std;
const int p = 137, mod = 1e9 + 7;
const int N = 1e5 + 9;
int pw[N];
void prec() {
        pw[0] = 1;
        for (int i = 1; i < N; i++) {
                pw[i] = 1LL * pw[i - 1] * p % mod;
        }
}
int get_hash(string s) {
        int n = s.size();
        int hs = 0;
        for (int i = 0; i < n; i++) {
                hs += 1LL * s[i] * pw[i] % mod;
                hs %= mod;
        }
        return hs;
}
int32_t main() {
        ios_base::sync_with_stdio(0);
        cin.tie(0);
        prec();
        string a, b; cin >> a >> b;
        cout << (get_hash(a) == get_hash(b)) << '\n';
        return 0;
}
```

7 geometry

7.1 Basics

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

using namespace std;

#define st first
#define nd second
#define pb push_back
#define cl(x,v) memset((x), (v), sizeof(x))
#define db(x) cerr << #x << " == " << x << endl
#define dbs(x) cerr << x << endl
#define _ << ", " <<

typedef long long ll;
typedef long double ld;
typedef pair<int,int> pii;
typedef pair<int, pii> piii;
typedef pair<ll,ll> pll;
typedef pair<ll, pll> plll;
typedef vector<int> vi;
typedef vector <vi> vii;

const ld EPS = 1e-9, PI = acos(-1.);
const ll LINF = 0x3f3f3f3f3f3f3f3f;
const int INF = 0x3f3f3f3f, MOD = 1e9+7;
const int N = 1e5+5;

typedef long double type;
//for big coordinates change to long long

bool ge(type x, type y) { return x + EPS > y; }
bool le(type x, type y) { return x - EPS < y; }
bool eq(type x, type y) { return ge(x, y) and le(x, y); }
int sign(type x) { return ge(x, 0) - le(x, 0); }

struct point {
        type x, y;

        point() : x(0), y(0) {}
        point(type _x, type _y) : x(_x), y(_y) {}

        point operator -() { return point(-x, -y); }
        point operator +(point p) { return point(x + p.x, y +
             p.y); }
        point operator -(point p) { return point(x - p.x, y -
             p.y); }

        point operator *(type k) { return point(x*k, y*k); }
        point operator /(type k) { return point(x/k, y/k); }

        //inner product
        type operator *(point p) { return x*p.x + y*p.y; }
        //cross product
        type operator %(point p) { return x*p.y - y*p.x; }
        bool operator ==(const point &p) const{ return x == p.x
             and y == p.y; }
        bool operator !=(const point &p) const{ return x != p.x
             or y != p.y; }
        bool operator <(const point &p) const { return (x <
             p.x) or (x == p.x and y < p.y); }

        // 0 => same direction
        // 1 => p is on the left
        //-1 => p is on the right
        int dir(point o, point p) {
                type x = (*this - o) % (p - o);
                return ge(x,0) - le(x,0);
        }

        bool on_seg(point p, point q) {
                if (this->dir(p, q)) return 0;
                return ge(x, min(p.x, q.x)) and le(x, max(p.x,
                     q.x)) and ge(y, min(p.y, q.y)) and le(y,
                     max(p.y, q.y));
        }

        ld abs() { return sqrt(x*x + y*y); }
        type abs2() { return x*x + y*y; }
        ld dist(point q) { return (*this - q).abs(); }
        type dist2(point q) { return (*this - q).abs2(); }

        ld arg() { return atan2l(y, x); }

        // Project point on vector y
        point project(point y) { return y * ((*this * y) / (y *
             y)); }

        // Project point on line generated by points x and y
        point project(point x, point y) { return x + (*this -
             x).project(y-x); }

        ld dist_line(point x, point y) { return dist(project(x,
             y)); }

        ld dist_seg(point x, point y) {
                return project(x, y).on_seg(x, y) ? dist_line(x,
                     y) : min(dist(x), dist(y));
        }

        point rotate(ld sin, ld cos) { return point(cos*x -
             sin*y, sin*x + cos*y); }
        point rotate(ld a) { return rotate(sin(a), cos(a)); }

        // rotate around the argument of vector p
        point rotate(point p) { return rotate(p.y / p.abs(),
             p.x / p.abs()); }

};

int direction(point o, point p, point q) { return p.dir(o, q); }

point rotate_ccw90(point p) { return point(-p.y,p.x); }
point rotate_cw90(point p) { return point(p.y,-p.x); }
```

```
//for reading purposes avoid using * and % operators, use the
     functions below:
type dot(point p, point q)  { return p.x*q.x + p.y*q.y; }
type cross(point p, point q) { return p.x*q.y - p.y*q.x; }

//double area
type area_2(point a, point b, point c) { return cross(a,b) +
     cross(b,c) + cross(c,a); }

//angle between (a1 and b1) vs angle between (a2 and b2)
//1  : bigger
//-1 : smaller
//0  : equal
int angle_less(const point& a1, const point& b1, const point&
     a2, const point& b2) {
       point p1(dot( a1, b1), abs(cross( a1, b1)));
       point p2(dot( a2, b2), abs(cross( a2, b2)));
       if(cross(p1, p2) < 0) return 1;
       if(cross(p1, p2) > 0) return -1;
       return 0;
}

ostream &operator<<(ostream &os, const point &p) {
       os << "(" << p.x << "," << p.y << ")";
       return os;
}
```

7.2 Circle

```
#include "basics.cpp"
#include "lines.cpp"

struct circle {
       point c;
       ld r;
       circle() { c = point(); r = 0; }
       circle(point _c, ld _r) : c(_c), r(_r) {}
       ld area() { return acos(-1.0)*r*r; }
       ld chord(ld rad) { return 2*r*sin(rad/2.0); }
       ld sector(ld rad) { return 0.5*rad*area()/acos(-1.0); }
       bool intersects(circle other) {
              return le(c.dist(other.c), r + other.r);
       }
       bool contains(point p) { return le(c.dist(p), r); }
       pair<point, point> getTangentPoint(point p) {
              ld d1 = c.dist(p), theta = asin(r/d1);
              point p1 = (c - p).rotate(-theta);
              point p2 = (c - p).rotate(theta);
              p1 = p1*(sqrt(d1*d1 - r*r)/d1) + p;
              p2 = p2*(sqrt(d1*d1 - r*r)/d1) + p;
              return make_pair(p1,p2);
       }
};

circle circumcircle(point a, point b, point c) {
       circle ans;
       point u = point((b - a).y, -(b - a).x);
       point v = point((c - a).y, -(c - a).x);
       point n = (c - b)*0.5;
       ld t = cross(u,n)/cross(v,u);
       ans.c = ((a + c)*0.5) + (v*t);
       ans.r = ans.c.dist(a);
       return ans;
}

point compute_circle_center(point a, point b, point c) {
       //circumcenter
       b = (a + b)/2;
       c = (a + c)/2;
       return compute_line_intersection(b, b + rotate_cw90(a -
            b), c, c + rotate_cw90(a - c));
}

int inside_circle(point p, circle c) {
       if (fabs(p.dist(c.c) - c.r)<EPS) return 1;
       else if (p.dist(c.c) < c.r) return 0;
       else return 2;
} //0 = inside/1 = border/2 = outside

circle incircle( point p1, point p2, point p3 ) {
       ld m1 = p2.dist(p3);
       ld m2 = p1.dist(p3);
       ld m3 = p1.dist(p2);
       point c = (p1*m1 + p2*m2 + p3*m3)*(1/(m1 + m2 + m3));
       ld s = 0.5*(m1 + m2 + m3);
       ld r = sqrt(s*(s - m1)*(s - m2)*(s - m3))/s;
       return circle(c, r);
}

circle minimum_circle(vector<point> p) {
       random_shuffle(p.begin(), p.end());
       circle C = circle(p[0], 0.0);
       for(int i = 0; i < (int)p.size(); i++) {
              if (C.contains(p[i])) continue;
              C = circle(p[i], 0.0);
              for(int j = 0; j < i; j++) {
                     if (C.contains(p[j])) continue;
                     C = circle((p[j] + p[i])*0.5,
                          0.5*p[j].dist(p[i]));
                     for(int k = 0; k < j; k++) {
                            if (C.contains(p[k])) continue;
                            C = circumcircle(p[j], p[i], p[k]);
                     }
              }
       }
       return C;
}

// compute intersection of line through points a and b with
// circle centered at c with radius r > 0
vector<point> circle_line_intersection(point a, point b, point
     c, ld r) {
       vector<point> ret;
       b = b - a;
       a = a - c;
       ld A = dot(b, b);
       ld B = dot(a, b);
       ld C = dot(a, a) - r*r;
       ld D = B*B - A*C;
       if (D < -EPS) return ret;
       ret.push_back(c + a + b*(sqrt(D + EPS) - B)/A);
       if (D > EPS)
              ret.push_back(c + a + b*(-B - sqrt(D))/A);
       return ret;
}

vector<point> circle_circle_intersection(point a, point b, ld
     r, ld R) {
       vector<point> ret;
       ld d = sqrt(a.dist2(b));
       if (d > r + R || d + min(r, R) < max(r, R)) return ret;
       ld x = (d*d - R*R + r*r)/(2*d);
       ld y = sqrt(r*r - x*x);
       point v = (b - a)/d;
       ret.push_back(a + v*x + rotate_ccw90(v)*y);
       if (y > 0)
              ret.push_back(a + v*x - rotate_ccw90(v)*y);
       return ret;
}

//GREAT CIRCLE

double gcTheta(double pLat, double pLong, double qLat, double
     qLong) {
       pLat *= acos(-1.0) / 180.0; pLong *= acos(-1.0) /
            180.0; // convert degree to radian
       qLat *= acos(-1.0) / 180.0; qLong *= acos(-1.0) / 180.0;
       return acos(cos(pLat)*cos(pLong)*cos(qLat)*cos(qLong) +
              cos(pLat)*sin(pLong)*cos(qLat)*sin(qLong) +
              sin(pLat)*sin(qLat));
}

double gcDistance(double pLat, double pLong, double qLat,
     double qLong, double radius) {
       return radius*gcTheta(pLat, pLong, qLat, qLong);
}


/*
 * Codeforces 101707B
 */
/*
point A, B;
circle C;

double getd2(point a, point b) {
       double h = dist(a, b);
       double r = C.r;
       double alpha = asin(h/(2*r));
       while (alpha < 0) alpha += 2*acos(-1.0);
       return dist(a, A) + dist(b, B) + r*2*min(alpha,
            2*acos(-1.0) - alpha);
}

int main() {
       scanf("%lf %lf", &A.x, &A.y);
       scanf("%lf %lf", &B.x, &B.y);
```

```
        scanf("%lf %lf %lf", &C.c.x, &C.c.y, &C.r);
        double ans;
        if (distToLineSegment(C.c, A, B) >= C.r) {
                ans = dist(A, B);
        }
        else {
                pair<point, point> tan1 = C.getTangentPoint(A);
                pair<point, point> tan2 = C.getTangentPoint(B);
                ans = 1e+30;
                ans = min(ans, getd2(tan1.first, tan2.first));
                ans = min(ans, getd2(tan1.first, tan2.second));
                ans = min(ans, getd2(tan1.second, tan2.first));
                ans = min(ans, getd2(tan1.second, tan2.second));
        }
        printf("%.18f\n", ans);
        return 0;
}*/
```

7.3 Lines

```
#include "basics.cpp"
//functions tested at:
     https://codeforces.com/group/3qadGzUdR4/contest/101706/problem/B

//WARNING: all distance functions are not realizing sqrt
     operation
//Suggestion: for line intersections check
     line_line_intersection and then use
     compute_line_intersection

point project_point_line(point c, point a, point b) {
        ld r = dot(b - a,b - a);
        if (fabs(r) < EPS) return a;
        return a + (b - a)*dot(c - a, b - a)/dot(b - a, b - a);
}

point project_point_ray(point c, point a, point b) {
        ld r = dot(b - a, b - a);
        if (fabs(r) < EPS) return a;
        r = dot(c - a, b - a) / r;
        if (le(r, 0)) return a;
        return a + (b - a)*r;
}

point project_point_segment(point c, point a, point b) {
        ld r = dot(b - a, b - a);
        if (fabs(r) < EPS) return a;
        r = dot(c - a, b - a)/r;
        if (le(r, 0)) return a;
        if (ge(r, 1)) return b;
        return a + (b - a)*r;
}

ld distance_point_line(point c, point a, point b) {
        return c.dist2(project_point_line(c, a, b));
}

ld distance_point_ray(point c, point a, point b) {
        return c.dist2(project_point_ray(c, a, b));
}

ld distance_point_segment(point c, point a, point b) {
        return c.dist2(project_point_segment(c, a, b));
}

//not tested
ld distance_point_plane(ld x, ld y, ld z,
                                            ld a, ld b, ld c,
                                                 ld d)
{
        return fabs(a*x + b*y + c*z - d)/sqrt(a*a + b*b + c*c);
}

bool lines_parallel(point a, point b, point c, point d) {
        return fabs(cross(b - a, d - c)) < EPS;
}

bool lines_collinear(point a, point b, point c, point d) {
  return lines_parallel(a, b, c, d)
         && fabs(cross(a-b, a-c)) < EPS
         && fabs(cross(c-d, c-a)) < EPS;
}

point lines_intersect(point p, point q, point a, point b) {
        point r = q - p, s = b - a, c(p%q, a%b);
        if (eq(r%s,0)) return point(LINF, LINF);
        return point(point(r.x, s.x) % c, point(r.y, s.y) % c)
             / (r%s);
}

//be careful: test line_line_intersection before using this
     function
point compute_line_intersection(point a, point b, point c,
     point d) {
        b = b - a; d = c - d; c = c - a;
        assert(dot(b, b) > EPS && dot(d, d) > EPS);
        return a + b*cross(c, d)/cross(b, d);
}

bool line_line_intersect(point a, point b, point c, point d) {
        if(!lines_parallel(a, b, c, d)) return true;
        if(lines_collinear(a, b, c, d)) return true;
        return false;
}

//rays in direction a -> b, c -> d
bool ray_ray_intersect(point a, point b, point c, point d){
        if (a.dist2(c) < EPS || a.dist2(d) < EPS ||
                b.dist2(c) < EPS || b.dist2(d) < EPS) return
                     true;
        if (lines_collinear(a, b, c, d)) {
                if(ge(dot(b - a, d - c), 0)) return true;
                if(ge(dot(a - c, d - c), 0)) return true;
                return false;
        }
        if(!line_line_intersect(a, b, c, d)) return false;
        point inters = lines_intersect(a, b, c, d);
        if(ge(dot(inters - c, d - c), 0) && ge(dot(inters - a,
             b - a), 0)) return true;
        return false;
}

bool segment_segment_intersect(point a, point b, point c, point
     d) {
        if (a.dist2(c) < EPS || a.dist2(d) < EPS ||
                b.dist2(c) < EPS || b.dist2(d) < EPS) return
                     true;
        int d1, d2, d3, d4;
        d1 = direction(a, b, c);
        d2 = direction(a, b, d);
        d3 = direction(c, d, a);
        d4 = direction(c, d, b);
        if (d1*d2 < 0 and d3*d4 < 0) return 1;
        return a.on_seg(c, d) or b.on_seg(c, d) or
                     c.on_seg(a, b) or d.on_seg(a, b);
}

bool segment_line_intersect(point a, point b, point c, point d){
        if(!line_line_intersect(a, b, c, d)) return false;
        point inters = lines_intersect(a, b, c, d);
        if(inters.on_seg(a, b)) return true;
        return false;
}

//ray in direction c -> d
bool segment_ray_intersect(point a, point b, point c, point d){
        if (a.dist2(c) < EPS || a.dist2(d) < EPS ||
                b.dist2(c) < EPS || b.dist2(d) < EPS) return
                     true;
        if (lines_collinear(a, b, c, d)) {
                if(c.on_seg(a, b)) return true;
                if(ge(dot(d - c, a - c), 0)) return true;
                return false;
        }
        if(!line_line_intersect(a, b, c, d)) return false;
        point inters = lines_intersect(a, b, c, d);
        if(!inters.on_seg(a, b)) return false;
        if(ge(dot(inters - c, d - c), 0)) return true;
        return false;
}

//ray in direction a -> b
bool ray_line_intersect(point a, point b, point c, point d){
        if (a.dist2(c) < EPS || a.dist2(d) < EPS ||
                b.dist2(c) < EPS || b.dist2(d) < EPS) return
                     true;
        if (!line_line_intersect(a, b, c, d)) return false;
        point inters = lines_intersect(a, b, c, d);
        if(!line_line_intersect(a, b, c, d)) return false;
        if(ge(dot(inters - a, b - a), 0)) return true;
        return false;
}

ld distance_segment_line(point a, point b, point c, point d){
        if(segment_line_intersect(a, b, c, d)) return 0;
        return min(distance_point_line(a, c, d),
             distance_point_line(b, c, d));
```

```
}

ld distance_segment_ray(point a, point b, point c, point d){
	if(segment_ray_intersect(a, b, c, d)) return 0;
	ld min1 = distance_point_segment(c, a, b);
	ld min2 = min(distance_point_ray(a, c, d),
		distance_point_ray(b, c, d));
	return min(min1, min2);
}

ld distance_segment_segment(point a, point b, point c, point d){
	if(segment_segment_intersect(a, b, c, d)) return 0;
	ld min1 = min(distance_point_segment(c, a, b),
		distance_point_segment(d, a, b));
	ld min2 = min(distance_point_segment(a, c, d),
		distance_point_segment(b, c, d));
	return min(min1, min2);
}

ld distance_ray_line(point a, point b, point c, point d){
	if(ray_line_intersect(a, b, c, d)) return 0;
	ld min1 = distance_point_line(a, c, d);
	return min1;
}

ld distance_ray_ray(point a, point b, point c, point d){
	if(ray_ray_intersect(a, b, c, d)) return 0;
	ld min1 = min(distance_point_ray(c, a, b),
		distance_point_ray(a, c, d));
	return min1;
}

ld distance_line_line(point a, point b, point c, point d){
	if(line_line_intersect(a, b, c, d)) return 0;
	return distance_point_line(a, c, d);
}
```

7.4 Polygons

```
#include "basics.cpp"
#include "lines.cpp"

//Monotone chain O(nlog(n))
#define REMOVE_REDUNDANT
#ifdef REMOVE_REDUNDANT
bool between(const point &a, const point &b, const point &c) {
	return (fabs(area_2(a,b,c)) < EPS &&
		(a.x-b.x)*(c.x-b.x) <= 0 && (a.y-b.y)*(c.y-b.y) <=
		0);
}
#endif

//new change: <= 0 / >= 0 became < 0 / > 0 (yet to be tested)

void convex_hull(vector<point> &pts) {
	sort(pts.begin(), pts.end());
	pts.erase(unique(pts.begin(), pts.end()), pts.end());
	vector<point> up, dn;
	for (int i = 0; i < pts.size(); i++) {
		while (up.size() > 1 && area_2(up[up.size()-2],
			up.back(), pts[i]) > 0) up.pop_back();
		while (dn.size() > 1 && area_2(dn[dn.size()-2],
			dn.back(), pts[i]) < 0) dn.pop_back();
		up.push_back(pts[i]);
		dn.push_back(pts[i]);
	}
	pts = dn;
	for (int i = (int) up.size() - 2; i >= 1; i--)
		pts.push_back(up[i]);

	#ifdef REMOVE_REDUNDANT
	if (pts.size() <= 2) return;
	dn.clear();
	dn.push_back(pts[0]);
	dn.push_back(pts[1]);
	for (int i = 2; i < pts.size(); i++) {
		if (between(dn[dn.size()-2], dn[dn.size()-1],
			pts[i])) dn.pop_back();
		dn.push_back(pts[i]);
	}
	if (dn.size() >= 3 && between(dn.back(), dn[0], dn[1]))
		{
		dn[0] = dn.back();
		dn.pop_back();
	}
	pts = dn;
	#endif
}

//avoid using long double for comparisons, change type and
	remove division by 2
type compute_signed_area(const vector<point> &p) {
	type area = 0;
	for(int i = 0; i < p.size(); i++) {
		int j = (i+1) % p.size();
		area += p[i].x*p[j].y - p[j].x*p[i].y;
	}
	return area;
}

ld compute_area(const vector<point> &p) {
	return fabs(compute_signed_area(p) / 2.0);
}

ld compute_perimeter(vector<point> &p) {
	ld per = 0;
	for(int i = 0; i < p.size(); i++) {
		int j = (i+1) % p.size();
		per += p[i].dist(p[j]);
	}
	return per;
}

//not tested
// TODO: test this code. This code has not been tested, please
	do it before proper use.
// http://codeforces.com/problemset/problem/975/E is a good
	problem for testing.
point compute_centroid(vector<point> &p) {
	point c(0,0);
	ld scale = 6.0 * compute_signed_area(p);
	for (int i = 0; i < p.size(); i++){
		int j = (i+1) % p.size();
		c = c + (p[i]+p[j])*(p[i].x*p[j].y -
			p[j].x*p[i].y);
	}
	return c / scale;

}

// TODO: test this code. This code has not been tested, please
	do it before proper use.
// http://codeforces.com/problemset/problem/975/E is a good
	problem for testing.
point centroid(vector<point> &v) {
  int n = v.size();
  type da = 0;
  point m, c;

  for(point p : v) m = m + p;
  m = m / n;

  for(int i=0; i<n; ++i) {
	point p = v[i] - m, q = v[(i+1)%n] - m;
	type x = p % q;
	c = c + (p + q) * x;
	da += x;
  }

  return c / (3 * da);
}

//O(n^2)
bool is_simple(const vector<point> &p) {
	for (int i = 0; i < p.size(); i++) {
		for (int k = i+1; k < p.size(); k++) {
			int j = (i+1) % p.size();
			int l = (k+1) % p.size();
			if (i == l || j == k) continue;
			if (segment_segment_intersect(p[i], p[j],
				p[k], p[l]))
				return false;
		}
	}
	return true;
}

bool point_in_triangle(point a, point b, point c, point cur){
	ll s1 = abs(cross(b - a, c - a));
	ll s2 = abs(cross(a - cur, b - cur)) + abs(cross(b -
		cur, c - cur)) + abs(cross(c - cur, a - cur));
	return s1 == s2;
}

void sort_lex_hull(vector<point> &hull){
```

```
        if(compute_signed_area(hull) < 0) reverse(hull.begin(),
            hull.end());
        int n = hull.size();

        //Sort hull by x
        int pos = 0;
        for(int i = 1; i < n; i++) if(hull[i] < hull[pos]) pos
            = i;
        rotate(hull.begin(), hull.begin() + pos, hull.end());
}

//determine if point is inside or on the boundary of a polygon
    (O(logn))
bool point_in_convex_polygon(vector<point> &hull, point cur){
        int n = hull.size();
        //Corner cases: point outside most left and most right
            wedges
        if(cur.dir(hull[0], hull[1]) != 0 && cur.dir(hull[0],
            hull[1]) != hull[n - 1].dir(hull[0], hull[1]))
              return false;
        if(cur.dir(hull[0], hull[n - 1]) != 0 &&
            cur.dir(hull[0], hull[n - 1]) !=
            hull[1].dir(hull[0], hull[n - 1]))
              return false;

        //Binary search to find which wedges it is between
        int l = 1, r = n - 1;
        while(r - l > 1){
              int mid = (l + r)/2;
              if(cur.dir(hull[0], hull[mid]) <= 0)l = mid;
              else r = mid;
        }
        return point_in_triangle(hull[1], hull[l + 1], hull[0],
            cur);
}

// determine if point is on the boundary of a polygon (O(N))
bool point_on_polygon(vector<point> &p, point q) {
for (int i = 0; i < p.size(); i++)
        if (q.dist2(project_point_segment(p[i],
            p[(i+1)%p.size()], q)) < EPS) return true;
        return false;
}

//Shamos - Hoey for test polygon simple in O(nlog(n))
inline bool adj(int a, int b, int n) {return (b == (a + 1)%n or
    a == (b + 1)%n);}

struct edge{
        point ini, fim;
        edge(point ini = point(0,0), point fim = point(0,0)) :
            ini(ini), fim(fim) {}
};

//< here means the edge on the top will be at the begin
bool operator < (const edge& a, const edge& b) {
        if (a.ini == b.ini) return direction(a.ini, a.fim,
            b.fim) < 0;
        if (a.ini.x < b.ini.x) return direction(a.ini, a.fim,
            b.ini) < 0;
        return direction(a.ini, b.fim, b.ini) < 0;
}

bool is_simple_polygon(const vector<point> &pts){
        vector <pair<point, pii>> eve;
        vector <pair<edge, int>> edgs;
        set <pair<edge, int>> sweep;
        int n = (int)pts.size();
        for(int i = 0; i < n; i++){
                point l = min(pts[i], pts[(i + 1)%n]);
                point r = max(pts[i], pts[(i + 1)%n]);
                eve.pb({l, {0, i}});
                eve.pb({r, {1, i}});
                edgs.pb(make_pair(edge(l, r), i));
        }
        sort(eve.begin(), eve.end());
        for(auto e : eve){
                if(!e.nd.st){
                        auto cur =
                            sweep.lower_bound(edgs[e.nd.nd]);
                        pair<edge, int> above, below;
                        if(cur != sweep.end()){
                                below = *cur;
                                if(!adj(below.nd, e.nd.nd, n) and
                                    segment_segment_intersect(pts[below.nd],
                                    pts[(below.nd + 1)%n],
                                    pts[e.nd.nd], pts[(e.nd.nd +
                                    1)%n]))
                                      return false;
                        }
                        if(cur != sweep.begin()){
                                above = *(--cur);
                                if(!adj(above.nd, e.nd.nd, n) and
                                    segment_segment_intersect(pts[above.nd],
                                    pts[(above.nd + 1)%n],
                                    pts[e.nd.nd], pts[(e.nd.nd +
                                    1)%n]))
                                      return false;
                        }
                        sweep.insert(edgs[e.nd.nd]);
                }
                else{
                        auto below =
                            sweep.upper_bound(edgs[e.nd.nd]);
                        auto cur = below, above = --cur;
                        if(below != sweep.end() and above !=
                            sweep.begin()){
                              --above;
                              if(!adj(below->nd, above->nd, n)
                                    and
                                    segment_segment_intersect(pts[below->nd],
                                    pts[(below->nd + 1)%n],
                                    pts[above->nd],
                                    pts[(above->nd + 1)%n]))
                                      return false;
                        }
                        sweep.erase(cur);
                }
        }
        return true;
}

//code copied from
    https://github.com/tfg50/Competitive-Programming/blob/master/Biblio
int maximize_scalar_product(vector<point> &hull, point vec) {
        // this code assumes that there are no 3 colinear points
        int ans = 0;
        int n = hull.size();
        if(n < 20) {
                for(int i = 0; i < n; i++) {
                        if(hull[i] * vec > hull[ans] * vec) {
                                ans = i;
                        }
                }
        } else {
                if(hull[1] * vec > hull[ans] * vec) {
                        ans = 1;
                }
                for(int rep = 0; rep < 2; rep++) {
                        int l = 2, r = n - 1;
                        while(l != r) {
                                int mid = (l + r + 1) / 2;
                                bool flag = hull[mid] * vec >=
                                    hull[mid-1] * vec;
                                if(rep == 0) { flag = flag &&
                                    hull[mid] * vec >= hull[0] *
                                    vec; }
                                else { flag = flag || hull[mid-1]
                                    * vec < hull[0] * vec; }
                                if(flag) {
                                      l = mid;
                                } else {
                                      r = mid - 1;
                                }
                        }
                        if(hull[ans] * vec < hull[l] * vec) {
                                ans = l;
                        }
                }
        }
        return ans;
}

//find tangents related to a point outside the polygon,
    essentially the same for maximizing scalar product
int tangent(vector<point> &hull, point vec, int dir_flag) {
        // this code assumes that there are no 3 colinear points
        // dir_flag = -1 for right tangent
        // dir_flag = 1 for left taangent
        int ans = 0;
        int n = hull.size();
        if(n < 20) {
                for(int i = 0; i < n; i++) {
                        if(hull[ans].dir(vec, hull[i]) ==
                            dir_flag) {
                              ans = i;
                        }
                }
        } else {
                if(hull[ans].dir(vec, hull[1]) == dir_flag) {
```

```
                ans = 1;
            }
            for(int rep = 0; rep < 2; rep++) {
                int l = 2, r = n - 1;
                while(l != r) {
                    int mid = (l + r + 1) / 2;
                    bool flag = hull[mid - 1].dir(vec,
                        hull[mid]) == dir_flag;
                    if(rep == 0) { flag = flag &&
                        (hull[0].dir(vec, hull[mid])
                        == dir_flag); }
                    else { flag = flag ||
                        (hull[0].dir(vec, hull[mid -
                        1]) != dir_flag); }
                    if(flag) {
                        l = mid;
                    } else {
                        r = mid - 1;
                    }
                }
                if(hull[ans].dir(vec, hull[l]) ==
                    dir_flag) {
                    ans = l;
                }
            }
        }
        return ans;
}
```

7.5 Radial Sort

```
#include "basics.cpp"
point origin;

/*
        below < above
        order: [pi, 2 * pi)
*/

int above(point p){
        if(p.y == origin.y) return p.x > origin.x;
        return p.y > origin.y;
}

bool cmp(point p, point q){
        int tmp = above(q) - above(p);
        if(tmp) return tmp > 0;
        return p.dir(origin,q) > 0;
        //Be Careful: p.dir(origin,q) == 0
}
```

7.6 Ternary Search

```
//Ternary Search - O(log(n))
//Max version, for minimum version just change signals

ll ternary_search(ll l, ll r){
        while(r - l > 3) {
                ll m1 = (l+r)/2;
                ll m2 = (l+r)/2 + 1;
                ll f1 = f(m1), f2 = f(m2);
                //if(f1 > f2) l = m1;
                if (f1 < f2) l = m1;
                else r = m2;
        }
        ll ans = 0;
        for(int i = l; i <= r; i++){
                ll tmp = f(i);
                //ans = min(ans, tmp);
                ans = max(ans, tmp);
        }
        return ans;
}

//Faster version - 300 iteratons up to 1e-6 precision
double ternary_search(double l, double r, int No = 300){
        // for(int i = 0; i < No; i++){
        while(r - l > EPS){
                double m1 = l + (r - l) / 3;
                double m2 = r - (r - l) / 3;
                // if (f(m1) > f(m2))
                if (f(m1) < f(m2))
                        l = m1;
                else
                        r = m2;
        }
        return f(l);
}
```