Daffodil International University

DIU_Pawgrammers

Nezu, SiyamBhuiyan, NoObMin

Team Reference Document

Contents

1 Setup

1.1 *CP_Ubuntu*

```
{
  "cmd": ["ulimit -s 268435456; g++ -std=c++20
      $file_name -o $file_base_name && timeout 4s
      ./$file_base_name < inputf.in >
  ⤶  outputf.in"],
  "selector": "source.cpp",
  "shell": true,
  "working_dir": "$file_path"
}
```

1.2 *CP_Windows*

```
{
  "cmd": ["g++.exe","-std=c++20", "${file}",
      "-o", "${file_base_name}.exe", "&&", "${f⌐
  ⤶  ile_base_name}.exe<inputf.in>outputf.in"],
  "selector": "source.cpp",
  "shell": true,
  "working_dir": "$file_path"
}
```

1.3 *StressTesting(check.sh)*

```
// chmod u+x check.sh
// ./check.sh
set -e
g++ gen.cpp -o gen
g++ code.cpp -o code
g++ brute.cpp -o brute
for ((i = 1; ; ++i)); do
  echo "Passed on TestCase: " $i
  ./gen $i > in
  ./code < in > out1
  ./brute < in > out2
  diff -Z out1 out2 || break
done
echo -e "WA on the following test:"
cat in
echo -e "\nExpected:"
cat out2
echo -e "\nFound:"
cat out1
```

1.4 *StressTesting(gen.cpp)*

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
mt19937_64 rng(chrono::steady_clock::now().time⌐
  ↪  since_epoch().count());
inline ll gen_random(ll l, ll r) {
  return uniform_int_distribution<ll>(l, r)(rng);
}
inline double gen_random_real(double l, double
  ↪  r) {
  return uniform_real_distribution<double>(l,
  ↪  r)(rng);
}
int main(int argc, char* args[]) {
  int _ = atoi(args[1]);
  rng.seed(_);
  int n = gen_random(1, 5);
  vector<int> per;
  for (int i = 0; i < n; ++i) {
    per.push_back(i + 1);
  }
  shuffle(per.begin(), per.end(), rng);
  return 0;
}
```

2 Data Structures

2.1 Custom Hash

```
#include <ext/pb_ds/assoc_container.hpp>
using namespace __gnu_pbds;
struct customHash {
  static uint64_t Meaw(uint64_t x) {
```

```cpp
      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_e↵
      ↪  poch().count();
      return Meaw(x + FIXED_RANDOM);
    }
};  // gp_hash_table<int, int> table;
```

2.2 Fast Unordered Map

```cpp
mp.reserve(1<<20);      // about 1M buckets
mp.max_load_factor(0.7); // safe and fast
```

2.3 GP Hash Table

```cpp
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
const int RANDOM = chrono::high_resolution_clock↵
  ↪  ::now().time_since_epoch().count();
struct custom_hash {
  int operator()(int x) const { return x ^
    ↪  RANDOM; }
};
//gp_hash_table<int, int, custom_hash> mp;
```

2.4 Mex of All Subarray

```cpp
const int N = 1e5 + 9, inf = 1e9;
struct ST {
  int t[4 * N];
  ST() {}
  void build(int n, int b, int e) {
    t[n] = 0;
    if (b == e) {
      return;
    }
    int mid = (b + e) >> 1, l = n << 1, r = l |
      ↪  1;
    build(l, b, mid);
    build(r, mid + 1, e);
    t[n] = min(t[l], t[r]);
  }
  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;
      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] = min(t[l], t[r]);
  }
  int get_min(int n, int b, int e, int i, int j)
    ↪  {
    if (b > j || e < i) return inf;
    if (b >= i && e <= j) return t[n];
    int mid = (b + e) >> 1, l = n << 1, r = l |
      ↪  1;
```

```cpp
    int L = get_min(l, b, mid, i, j);
    int R = get_min(r, mid + 1, e, i, j);
    return min(L, R);
  }
  int get_mex(int n, int b, int e, int i) {  //
    ↪  mex of [i... cur_id] if (b == e) return b;
    int mid = (b + e) >> 1, l = n << 1, r = l |
      ↪  1;
    if (t[l] >= i) return get_mex(r, mid + 1, e,
      ↪  i);
    return get_mex(l, b, mid, i);
  }
} t;
int a[N], f[N];
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];
    --a[i];
  }
  t.build(1, 0, n);
  set<array<int, 3>> seg;  // for cur_id = i,
    ↪  [x[0]... i], [x[0] + 1...i], ...[x[1]... i]
    ↪  has mex, , x[2]
  for (int i = 1; i <= n; i++) {
    int x = a[i];
    int r = min(i - 1, t.get_min(1, 0, n, 0, x -
      ↪  1));
    int l = t.get_min(1, 0, n, 0, x) + 1;
    if (l <= r) {
      auto it = seg.lower_bound({l, -1, -1});
      while (it != seg.end() && (*it)[1] <= r) {
        auto x = *it;
        it = seg.erase(it);
      }
    }
    t.upd(1, 0, n, x, i);
    for (int j = r; j >= l;) {
      int m = t.get_mex(1, 0, n, j);
      int L = max(l, t.get_min(1, 0, n, 0, m) +
        ↪  1);
      f[m] = 1;
      seg.insert({L, j, m});
      j = L - 1;
    }
    int m = !a[i];
    seg.insert({i, i, m});
    f[m] = 1;
  }
  int ans = 0;
  while (f[ans]) ++ans;
  cout << ans + 1 << '\n';
  return 0;
}
```

2.5 Pbds

```cpp
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <functional>
using namespace __gnu_pbds;
typedef tree<int, null_type, less<int>,
  ↪  rb_tree_tag,
```

```cpp
      tree_order_statistics_node_update>
    ordered_set;
// s.order_of_key(x) = number of elements
  ↪  strictly less than x
// *s.find_by_order(i) = ith element in set (0
  ↪  index)
```

2.6 Segment Tree(BSUA)

```cpp
// CSES - 1749
const int MX = 2e5 + 10;
int n;
int arr[MX], st[MX << 2];
void assign(int i, int x, int u = 1, int s = 0,
  ↪  int e = n - 1) {
  if (s == e) {
    st[u] = x;
    return;
  }
  int v = u << 1, w = v | 1, m = (s + e) >> 1;
  if (i <= m) assign(i, x, v, s, m);
  else assign(i, x, w, m + 1, e);
  st[u] = st[v] + st[w];
}
int kth(int k, int u = 1, int s = 0, int e = n -
  ↪  1) {
  if (st[u] < k) return -1;
  if (s == e) {
    return s;
  }
  int v = u << 1, w = v | 1, m = (s + e) >> 1;
  if (st[v] >= k) return kth(k, v, s, m);
  else return kth(k - st[v], w, m + 1, e);
}
void solve() {
  cin >> n;
  for (int i = 0; i < n; ++i) {
    cin >> arr[i];
  }
  for (int i = 0; i < n; ++i) {
    assign(i, 1);
  }
  for (int i = 0; i < n; ++i) {
    int x;
    cin >> x;
    int ind = kth(x);
    assign(ind, 0);
    cout << arr[ind] << " ";
  }
}
```

2.7 Segment Tree(LzP)

```cpp
class stree {
  vector<ll> st, lazy;

 public:
  stree(int n) {
    st.assign((n << 2) + 10, 0);
    lazy.assign((n << 2) + 10, 0);
  }
  void push(int u, int s, int e) {
    if (!lazy[u]) return;
    st[u] += (e - s + 1) * 1LL * lazy[u];
    if (s != e) {
```

```cpp
    int v = u << 1, w = v | 1, m = (s + e) >>
    ↪    1;
    lazy[v] += lazy[u];
    lazy[w] += lazy[u];
    }
    lazy[u] = 0;
}
void build(int u, int s, int e, int arr[]) {
    if (s == e) {
        st[u] = arr[s];
        return;
    }
    int v = u << 1, w = v | 1, m = (s + e) >> 1;
    build(v, s, m, arr);
    build(w, m + 1, e, arr);
    st[u] = st[v] + st[w];
}
void update(int l, int r, int x, int u, int s,
↪    int e) {
    push(u, s, e);
    if (e < l or r < s) return;
    int v = u << 1, w = v | 1, m = (s + e) >> 1;
    if (l <= s and e <= r) {
        st[u] += (e - s + 1) * 1LL * x;
        if (s != e) {
            lazy[v] += x;
            lazy[w] += x;
        }
        return;
    }
    update(l, r, x, v, s, m);
    update(l, r, x, w, m + 1, e);
    st[u] = st[v] + st[w];
}
ll query(int l, int r, int u, int s, int e) {
    push(u, s, e);
    if (e < l or r < s) return 0;
    if (l <= s and e <= r) return st[u];
    int v = u << 1, w = v | 1, m = (s + e) >> 1;
    return query(l, r, v, s, m) + query(l, r, w,
    ↪    m + 1, e);
    }
};
```

2.8 Segment Tree

```cpp
class stree {
    vector<ll> st;
public:
    stree(int n) {
        st.assign((n << 2) + 10, 0);
    }
    void build(int u, int s, int e, int arr[]) {
        if (s == e) {
            st[u] = arr[s];
            return;
        }
        int v = u << 1, w = v | 1, m = (s + e) >> 1;
        build(v, s, m, arr);
        build(w, m + 1, e, arr);
        st[u] = st[v] + st[w];
    }
    ll query(int l, int r, int u, int s, int e) {
        if (e < l or r < s) return 0;
        if (l <= s and e <= r) return st[u];
        int v = u << 1, w = v | 1, m = (s + e) >> 1;
```

```cpp
        return query(l, r, v, s, m) + query(l, r, w,
        ↪    m + 1, e);
    }
    void update(int i, int x, int u, int s, int e)
    ↪    {
        if (s == e) {
            st[u] = x;
            return;
        }
        int v = u << 1, w = v | 1, m = (s + e) >> 1;
        if (i <= m) update(i, x, v, s, m);
        else update(i, x, w, m + 1, e);
        st[u] = st[v] + st[w];
    }
};
```

2.9 Sparse Table

```cpp
const int mxN = 1e5 + 10, M = 21;
int sparse[mxN][M];
void build_sparse(int n, vector<int>& v) {
    for (int i = 0; i < n; i++) sparse[i][0] =
    ↪    v[i];
    for (int k = 1; k < M; k++) {
        for (int i = 0; i + (1 << k) <= n; i++) {
            sparse[i][k] = max(sparse[i][k - 1],
            ↪    sparse[i + (1 << (k - 1))][k - 1]);
        }
    }
}
int query(int l, int r) {  // 0 based index
    if (l > r) swap(l, r);
    int b = __bit_width(r - l + 1) - 1;
    return max(sparse[l][b], sparse[r - (1 << b) +
    ↪    1][b]);
}
```

2.10 main

```cpp
// sjdaflksjf
// hello new changes
// more changes
// hello world
```

3 Dynamic Programming
3.1 Coin Change(Number of Ways)

```cpp
const int mod = 1e9+7;
void solve(){
    int n, k; cin>>n>>k;
    vector<int> coin(n);
    for(int i = 0; i<n; i++){ cin>>coin[i];  }
    vector<int> dp(k+1, 0); dp[0] = 1;
    for(int i = 1; i<=k; i++){
        for(int j = 0; j<n; j++){
            if(i-coin[j]>=0){
                dp[i] = (dp[i]+dp[i-coin[j]])%mod;
            }
        }
    } cout<<dp[k]<<endl;
}
```

3.2 Digit DP

```cpp
vector<int> nmbrs;
int dp[10][10][2];
int dgt_dp(int idx, int tight, int onecnt) {
```

```cpp
    if (idx == nmbrs.size()) {
        return onecnt;
    }
    if (dp[idx][onecnt][tight] != -1) return
    ↪    dp[idx][onecnt][tight];
    int lmt = (tight ? nmbrs[idx] : 9);
    int sum = 0;
    for (int i = 0; i <= lmt; i++) {
        bool newTight = (tight and i == nmbrs[idx]);
        sum += dgt_dp(idx + 1, newTight, onecnt + (i
        ↪    == 1));
    }
    return dp[idx][onecnt][tight] = sum;
}
```

3.3 LIS

```cpp
vector<int> lis(int n, vector<int>& v) {
    vector<int> parent(n, -1), ind(n);
    vector<int> lis;
    for (int i = 0; i < n; i++) {
        int it = lower_bound(lis.begin(), lis.end(),
        ↪    v[i]) - lis.begin();
        if (it == lis.size()) {
            lis.push_back(v[i]);
            ind[lis.size() - 1] = i;
            parent[i] = (lis.size() == 1 ? -1 : ind[it
            ↪    - 1]);
        } else {
            lis[it] = v[i];
            ind[it] = i;
            parent[i] = (it == 0 ? -1 : ind[it - 1]);
        }
    }
    vector<int> LIS;
    int it = ind[lis.size() - 1];
    LIS.push_back(lis.back());
    while (parent[it] != -1) {
        it = parent[it];
        LIS.push_back(v[it]);
    }
    return LIS;
}
```

3.4 Maximum Subarray Sum(Kadanes)

```cpp
int max_sum_of(vector<int> &vct){
    int mx = INT_MIN, till = 0;
    for (int i = 0; i<vct.size(); i++) {
        till = till + vct[i];
        mx = max(mx, till);
        till = max(till, 1LL*0);
    }
    return mx;
}
```

4 Geometry
4.1 Convex Hull

```cpp
vector <PT> convexHull (vector <PT> p) {
    int n = p.size(), m = 0;
    if (n < 3) return p;
    vector <PT> hull(n + n);
    sort(p.begin(), p.end(), [&] (PT a, PT b) {
        return (a.x==b.x? a.y<b.y: a.x<b.x);
```

```cpp
});
for (int i = 0; i < n; ++i) {
    while (m > 1 and cross(hull[m - 2] - p[i],
    ↪    hull[m - 1] - p[i]) <= 0) --m;
    hull[m++] = p[i];
}
for (int i = n - 2, j = m + 1; i >= 0; --i) {
    while (m >= j and cross(hull[m - 2] - p[i],
    ↪    hull[m - 1] - p[i]) <= 0) --m;
    hull[m++] = p[i];
}
hull.resize(m - 1); return hull;
}
```

4.2 Integer Points in a Circle

```cpp
ll latticeInCircle(ll r){
    ll ans = (4*r) +1; // 1 for center
    for(int i = 1; i*i<=r*r; i++){
        for(int j = 1; j*j+i*i<=r*r; j++){  ans+=4;
        ↪    }
    } return ans;
}
```

5 Graph
5.1 Articulation Point

```cpp
int n;                      // number of nodes
vector<vector<int>> lst;  // adjacency list of
↪    graph
vector<bool> vis;
vector<int> tin, low;
int timer;
void dfs(int u, int p = -1) {
    vis[u] = true;
    tin[u] = low[u] = timer++;
    int children = 0;
    for (int v : lst[u]) {
        if (v == p) continue;
        if (vis[v]) {
            low[u] = min(low[u], tin[v]);
        } else {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] >= tin[u] && p != -1){
                IS_CUTPOINT(u);
            }
            ++children;
        }
    }
    // if no vertex below v can reach u or higher
    ↪    removing u disconnects that subtree
    if (p == -1 && children > 1){
        IS_CUTPOINT(u);
    }
}
void find_cutpoints() {
    timer = 0;
    vis.assign(n, false);
    tin.assign(n, -1);
    low.assign(n, -1);
    for (int i = 0; i < n; ++i) {
        if (!vis[i]){
            dfs(i);
        }
    }
}
```

```cpp
    }
}
```

5.2 BFS

```cpp
vector<vector<int>> adj;  // adjacency list
↪    representation
int n;                      // number of nodes
int s;                      // source vertex
void bfs() {
    queue<int> q;
    vector<int> d(n), p(n);
    vector<bool> used(n);

    q.push(s);
    used[s] = true;
    p[s] = -1;
    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int u : adj[v]) {
            if (!used[u]) {
                used[u] = true;
                q.push(u);
                d[u] = d[v] + 1;
                p[u] = v;
            }
        }
    }
}

// retrieving shortest path

if (!used[u]) {
    cout << "No path!";
} else {
    vector<int> path;
    for (int v = u; v != -1; v = p[v])
        path.push_back(v);
    reverse(path.begin(), path.end());
    cout << "Path: ";
    for (int v : path)
        cout << v << " ";
}
```

5.3 Bellman Ford

```cpp
#define ll long long
#define INF 1e18

void solve() {
    int n, m, v;
    cin >> n >> m >> v;  // n = nodes, m = edges, v
    ↪    = source (0-indexed)
    vector<array<ll, 3>> edges(m);  // each edge:
    ↪    {a, b, cost}
    for (int i = 0; i < m; i++) cin >> edges[i][0]
    ↪    >> edges[i][1] >> edges[i][2];
    vector<ll> d(n, INF);
    vector<int> p(n, -1);
    d[v] = 0;

    int x = -1;
    for (int i = 0; i < n; i++) {
        x = -1;
        for (auto& e : edges) {
            int a = e[0], b = e[1];
            ll cost = e[2];
```

```cpp
            if (d[a] < INF && d[b] > d[a] + cost) {
                d[b] = max(-INF, d[a] + cost);
                p[b] = a;
                x = b;
            }
        }
    }
    if (x == -1) {
        cout << "No negative cycle from vertex " <<
        ↪    v << '\n';
        return;
    }
    int y = x;
    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 u : path) cout << u << ' ';
    cout << '\n';
}
```

5.4 Bridge Finding DFS

```cpp
const int MX = 1e5 + 10;
int n, m, timer = 0;
vector<int> adj[MX];
vector<int> tin(MX, -1), low(MX, -1);
vector<bool> vis(MX, false);
void is_bridge(int u, int v) {
    // do something with the edge
}
void dfs(int u, int p = -1) {
    vis[u] = true;
    tin[u] = low[u] = timer++;
    for (int v : adj[u]) {
        if (v == p) continue;
        if (vis[v]) {
            low[u] = min(low[u], tin[v]);
        } else {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
            if (low[v] > tin[u]) {
                is_bridge(u, v);
            }
        }
    }
}
```

5.5 Cycle Detection in DAG

```cpp
const int MX = 1e5 + 10;
bool vis[MX], pathVis[MX];
vector<int> lst[MX];
bool dfs(int u) {
    vis[u] = true;
    pathVis[u] = true;
    for (auto v : lst[u]) {
        if (!vis[v]) {
            if (dfs(v))
                return true;
        } else if (pathVis[v]) {
```

```cpp
            return true;
        }
    }
    pathVis[u] = false;
    return false;
}
void solve() {
    // take graph input
    for (int i = 0; i < n; ++i) {
        if (!vis[i])
            dfs(i);
    }
}
```

5.6 DSU

```cpp
const int MX = 1e5 + 10;
int par[MX], sz[MX];
void init() {
    for (int i = 1; i < MX; i++) {
        par[i] = i;
        sz[i] = 1;
    }
}
int findpar(int x) {
    if (par[x] == x) return x;
    return par[x] = findpar(par[x]);
}
void unite(int u, int v) {
    u = findpar(u);
    v = findpar(v);
    if (u != v) {
        if (sz[u] < sz[v]) {
            swap(u, v);
        }
        sz[u] += sz[v];
        par[v] = u;
    }
}
```

5.7 Dijkastra

```cpp
const int N = 1e5 + 5, INF = 1e18 + 7;
vector<pair<int, int>> g[N];
bool visited[N];
vector<int> dist(N, INF), parent(N);
bool dijkstra(int source) {
    priority_queue<pair<int, int>,
        vector<pair<int, int>>, greater<pair<int,
     ↪  int>>> pq;
    pq.push({0, source});
    dist[source] = 0;
    parent[source] = -1;
    while (pq.size()) {
        int x = pq.top().second;
        pq.pop();
        if (visited[x]) continue;
        visited[x] = 1;
        for (auto [child_x, child_wt] : g[x]) {
            if (dist[x] + child_wt < dist[child_x]) {
                parent[child_x] = x;
                dist[child_x] = child_wt + dist[x];
                pq.push({dist[child_x], child_x});
            }
        }
    }
}
```

```cpp
    return (dist[n] == INF);
}
```

5.8 Euler Tour

```cpp
const int MX = 2e5 + 10;
int timer = -1;
// s = start pos, e = end pos
int val[MX], s[MX], e[MX], flat[MX];
vector<int> lst[MX];
void dfs(int u, int p) {
    s[u] = ++timer;
    flat[timer] = val[u];
    for (auto v : lst[u]) {
        if (v != p)
            dfs(v, u);
    }
    e[u] = timer;
}
```

5.9 Floyd Warshall

```cpp
vector<vector<int>> d(n, vector<int>(n, INF));
// take graph input into d
for (int k = 0; k < n; ++k) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (d[i][k] < INF && d[k][j] < INF)
                d[i][j] = min(d[i][j], d[i][k] +
                 ↪  d[k][j]);
        }
    }
}
```

5.10 MST

```cpp
// DSU first
void solve() {
    int n, m;
    cin >> n >> m;
    vector<tuple<int, int, int>> edges;
    for (int i = 0; i < m; ++i) {
        int u, v, wt;
        cin >> u >> v >> wt;
        edges.push_back({wt, u, v});
    }
    sort(edges.begin(), edges.end());
    init(n);
    int cost = 0;
    for (tuple& [ wt, u, v ] : edges) {
        if (findpar(u) == findpar(v)) continue;
        unite(u, v);
        cost += wt;
    }
    cout << cost << endl;
}
```

5.11 Max Bipartite Matching[Hopcroft Karp]

```cpp
const int INF = 1e9;
void hopcroftCarp() {
    int n, m, e;
    cin >> n >> m >> e;
    vector<int> adj[n];
    for (int i = 0; i < e; ++i) {
        int u, v;
        cin >> u >> v;
        --u;
        --v;
```

```cpp
        adj[u].push_back(v);
    }
    vector<int> ml(m, -1), mr(n, -1), dist(n);
    auto bfs = [&]() -> bool {
        queue<int> q;
        for (int u = 0; u < n; ++u) {
            if (mr[u] == -1) {
                dist[u] = 0;
                q.push(u);
            } else {
                dist[u] = INF;
            }
        }
        bool foundAugmenting = false;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int v : adj[u]) {
                int pairedLeft = ml[v];
                if (pairedLeft == -1) {
                    foundAugmenting = true;
                } else if (dist[pairedLeft] == INF) {
                    dist[pairedLeft] = dist[u] + 1;
                    q.push(pairedLeft);
                }
            }
        }
        return foundAugmenting;
    };
    function<bool(int)> dfs = [&](int u) -> bool {
        for (int v : adj[u]) {
            int pairedLeft = ml[v];
            if (pairedLeft == -1 or (dist[pairedLeft]
             ↪  == dist[u] + 1 and dfs(pairedLeft))) {
                mr[u] = v;
                ml[v] = u;
                return true;
            }
        }
        dist[u] = INF;
        return false;
    };
    int matching = 0;
    while (bfs()) {
        for (int u = 0; u < n; ++u) {
            if (mr[u] == -1) {
                if (dfs(u)) matching++;
            }
        }
    }
    cout << matching << el;
    for (int u = 0; u < n; ++u) {
        if (mr[u] != -1) {
            cout << u << " " << mr[u] << el;
        }
    }
}
```

5.12 Max Bipartite Matching[Kuhn's]

```cpp
// left set size, right set size, edge count
int n, k, m, visToken = 1;
vector<int> lst[MX];
int mr[MX], ml[MX], vis[MX];
bool try_kuhn(int u) {
```

```cpp
    if (vis[u] == visToken)
      return false;
    vis[u] = visToken;
    for (auto v : lst[u]) {
      if (ml[v] == -1 or try_kuhn(ml[v])) {
        ml[v] = u;
        mr[u] = v;
        return true;
      }
    }
    return false;
}
void solve() {
    cin >> n >> k >> m;
    for (int i = 0; i < m; ++i) {
      int u, v;
      cin >> u >> v;
      --u, --v;
      lst[u].push_back(v);
    }
    fill(mr, mr + n, -1);
    fill(ml, ml + k, -1);
    int ans = 0;
    for (int u = 0; u < n; ++u) {
      for (auto v : lst[u]) {
        if (ml[v] == -1) {
          ml[v] = u;
          mr[u] = v;
          ans++;
          break;
        }
      }
    }
    for (int u = 0; u < n; ++u) {
      if (mr[u] != -1) continue;
      visToken++;
      if (try_kuhn(u))
        ans++;
    }
    cout << ans << el;
    for (int v = 0; v < k; ++v) {
      if (ml[v] != -1) {
        cout << ml[v] + 1 << " " << v + 1 << el;
      }
    }
}
```

5.13 Topological Sorting

```cpp
const int N = 1e5 + 10;
vector<int> g[N], indegree(N, 0);
vector<int> topSort(int n) {
    queue<int> q;
    vector<int> order;
    for (int i = 1; i <= n; i++) {
      if (indegree[i] == 0) {
        q.push(i);
      }
    }
    while (!q.empty()) {
      int u = q.front();
      q.pop();
      order.push_back(u);
      for (int v : g[u]) {
        indegree[v]--;
        if (indegree[v] == 0) {
          q.push(v);
        }
      }
```

```cpp
    }
  }
  return order;
}
```

5.14 Weighted Union Find

```cpp
const int MX = 2e5 + 10;
int par[MX], sz[MX];
ll d[MX];
void init() {
    for (int i = 0; i < MX; ++i) {
      par[i] = i;
      sz[i] = 1;
      d[i] = 0;
    }
}
int findpar(int x) {
    if (par[x] == x) return x;
    int p = par[x];
    par[x] = findpar(p);
    d[x] += d[p];
    return par[x];
}
bool unite(int a, int b, ll w) {
    int ra = findpar(a);
    int rb = findpar(b);
    if (ra == rb) {
      return (d[b] - d[a] == w);
    }
    if (sz[ra] < sz[rb]) {
      swap(a, b);
      swap(ra, rb);
      w = -w;
    }
    par[rb] = ra;
    d[rb] = d[a] + w - d[b];
    sz[ra] += sz[rb];
    return true;
}
ll dist(int a, int b) {
    findpar(a), findpar(b);
    return d[b] - d[a];
}
```

6 Math
6.1 Matrix Exponentiation

```cpp
const ll mod = 1e9;
vector<vector<ll>> matMul(vector<vector<ll>>& a,
↪  vector<vector<ll>>& b) {
    ll row1 = a.size(), col1 = a[0].size();
    ll row2 = b.size(), col2 = b[0].size();
    vector<vector<ll>> res(row1, vector<ll>(col2,
↪  0));
    for (ll i = 0; i < row1; i++) {
      for (ll j = 0; j < col1; j++) {
        for (ll k = 0; k < row2; k++) {
          res[i][j] = (res[i][j] + (1LL * a[i][k]
↪   * b[k][j]) % mod) % mod;
        }
      }
    }
    return res;
}
vector<vector<ll>> matExpo(vector<vector<ll>>&
↪  Mat, ll exp) {
    ll row = Mat.size(), col = Mat[0].size();
```

```cpp
    ll p = row;
    vector<vector<ll>> res(p, vector<ll>(p, 0));
    for (ll i = 0; i < p; i++) res[i][i] = 1;
    while (exp) {
      if (exp & 1) res = matMul(res, Mat);
      Mat = matMul(Mat, Mat);
      exp >>= 1;
    }
    return res;
}
// b = (A(i), A(i-1), A(i-2), A(i-3))
// M = Magic matrix, nth = nth term, known =
↪   known value
ll get_nth(ll nth, ll known, vector<ll>& b,
↪   vector<vector<ll>>& M) {
    if (nth <= known) return b[nth - 1] % mod;
    reverse(b.begin(), b.end());
    vector<vector<ll>> me = matExpo(M, nth -
↪   known);   // MAT^(nth-known)
    ll ans = 0;
    for (int i = 0; i < known; i++) {
      ans = (ans + (b[i] * me[i][0]) % mod) % mod;
    }
    return ans;
}
```

6.2 Matrix Rotation

```cpp
//90* clock-wise
now = {{0, 1, 0}, { -1, 0, 0}, {0, 0, 1}};
//90* anti-clock
now = {{0, -1, 0}, {1, 0, 0}, {0, 0, 1}};
//mirror with x axis at point p
now = {{ -1, 0, 2 * p}, {0, 1, 0}, {0, 0, 1}};
//mirror with y axis at point p
now = {{1, 0, 0}, {0, -1, 2 * p}, {0, 0, 1}};
op[i + 1] = matMul(now, op[i]); // this
// op[i + 1] = matMul(op[i], now); //not this
```

6.3 Polynomial Interpolation

```cpp
// P(x) = a0 + a1x + a2x^2 + ... + anx^n
// y[i] = P(i)
const int mod = 1e9 + 7;
ll BigMod(ll a, ll b) {
    ll res = 1;
    while (b) {
      if (b & 1) res = 1ll * res * a % mod;
      a = 1ll * a * a % mod;
      b >>= 1;
    }
    return res;
}
ll inv(ll x) {
    if (x < 0) x += mod;
    return BigMod(x, mod - 2);
}
ll add(ll& a, ll b) {
    a += b;
    if (a >= mod) a -= mod;
    return a;
}
ll eval(vector<ll> y, ll k) {
    int n = y.size() - 1;
    if (k <= n) {
      return y[k];
```

```cpp
  }
  vector<ll> L(n + 1, 1);
  for (int x = 1; x <= n; ++x) {
    L[0] = L[0] * (k - x) % mod;
    L[0] = L[0] * inv(-x) % mod;
  }
  for (int x = 1; x <= n; ++x) {
    L[x] = L[x - 1] * inv(k - x) % mod * (k - (x
    ↪    - 1)) % mod;
    L[x] = L[x] * ((x - 1) - n + mod) % mod *
    ↪    inv(x) % mod;
  }
  ll yk = 0;
  for (int x = 0; x <= n; ++x) {
    yk = add(yk, L[x] * y[x] % mod);
  }
  return yk;
}
```

6.4 Sqrt Distinct Floor

```cpp
//1st problem
const ll mod = 1e9+7;
void solution(){
  ll n; cin>>n;
  ll i = 1;
  ll l = 0, r = 0;
  ll sum = 0;
  while(i<=n){
    ll p = n/i;
    ll l = i-1;
    i = (n/p)+1;
    ll r;
    if(i<=n){
      r = i-1;
    }
    else{
      r = n;
    }
    ll s1 = (__int128(l)*(l+1)/2)%mod;
    ll s2 = (__int128(r)*(r+1)/2)%mod;
    // cout<<l<<" "<<r<<" "<<s1<<' '<<s2<<endl;
    sum = ((sum%mod) +
    ↪    (((s2-s1+mod)%mod)*(p%mod))%mod)%mod;
  }
  cout<<sum<<endl;
}

//2nd problem
void solution(){
  ll n; cin>>n;
  vector<ll> v;
  ll i = 1;
  ll sum = 0;
  while(i<=n){
    ll p = n/i;
    ll prev = i;
    v.push_back(p);
    i = (n/p)+1;
    ll q;
    if(i<=n){
      q = i-prev;
    }
    else{
      q = n-prev+1;
    }
    sum+=p*q;
  }
}
```

```cpp
  cout<<sum<<endl;
}
```

7 Miscellaneous
7.1 Max Subarray Size Sum equal K

```cpp
//write gpHashTable code before this part
void solution(){
  int n, k; cin >> n >> k;
  int total_sum = 0;
  vector < int > pre(n + 7, 0);
  for (int i = 1; i <= n; i++) {
    int temp; cin >> temp;
    total_sum += temp;
    if (i == 1) pre[i] = temp;
    else pre[i] = pre[i - 1] + temp;
  }
  if (total_sum < k) {
    cout << "-1" << endl; return;
  }
  if (total_sum == k) {
    cout << "0" << endl; return;
  }
  int maximum_subSize = 0;
  gp_hash_table < int, int, customHash> table;
  for (int i = 1; i <= n; i++) {
    if (pre[i] >= k) {
      int subSUM = pre[i] - k;
      if (subSUM == 0){
        maximum_subSize = max(maximum_subSize, i);
      }
      else if (table[subSUM]) {
        int left = table[subSUM];
        int right = i; int subSize = right - left;
        maximum_subSize = max(subSize,
        ↪    maximum_subSize);
      }
    } if (!table[pre[i]]) table[pre[i]] = i;
  } cout << maximum_subSize << endl; }
```

7.2 Merge Sort

```cpp
// use array of elements, if multiple testcase
↪    make inv = 0 each time
// int inv = 0;
void merge(int vct[], int l, int m, int r) {
  int left = m - l + 1, right = r - m, lv[left],
  ↪    rv[right];
  for (int i = 0; i < left; i++) {
    lv[i] = vct[l + i];
  }
  for (int i = 0; i < right; i++) {
    rv[i] = vct[m + 1 + i];
  }
  int i = 0, j = 0, to = l;
  while (i < left && j < right) {
    if (lv[i] <= rv[j]) {
      vct[to] = lv[i];
      i++;
    } else {
      vct[to] = rv[j];
      j++;
      // inversion count
      // int pore = left-i; inv+=pore;
    }
    to++;
```

```cpp
  }
  while (i < left) {
    vct[to] = lv[i];
    i++;
    to++;
  }
  while (j < right) {
    vct[to] = rv[j];
    j++;
    to++;
  }
}
void merge_sort(int vct[], int l, int r) {
  if (r <= l) return;
  int m = l + ((r - l) / 2);
  merge_sort(vct, l, m);
  merge_sort(vct, m + 1, r);
  merge(vct, l, m, r);
}
```

7.3 Number of Subarray Sum is K

```cpp
//write gpHashTable code before this part
void solution(){
  int n, k; cin >> n >> k;
  int total_sum = 0;
  vector < int > pre(n + 7, 0);
  for (int i = 1; i <= n; i++) {
    int temp; cin >> temp;
    total_sum += temp;
    if (i == 1) pre[i] = temp;
    else pre[i] = pre[i - 1] + temp; }
  int cnt_subarry = 0;
  gp_hash_table < int, int, customHash> table;
  table[0] = 1;
  for (int i = 1; i <= n; i++) {
    cnt_subarry += table[pre[i] - k];
    table[pre[i]]++;
  } cout << cnt_subarry << endl; }
```

8 Number Theory
8.1 All In One NT

```cpp
const int MAXN = 1e6 + 9;
typedef struct info {
  int lowest_prime = 0, greatest_prime = 0,
  ↪    distinct_prime = 0;
  int total_prime = 0, NOD = 0, SOD = 0;
} info;
info num[MAXN];
void preStore() {
  for (int i = 2; i < MAXN; i++) {
    int n = i;
    map<int, int> factors;  // Key->Factor,
    ↪    Val->count
    int SOD = 1, NOD = 1, total_p_factor = 0;
    if (n % 2 == 0) {
      while (n % 2 == 0) {
        n /= 2;
        factors[2]++;
        total_p_factor++;
      }
      SOD *= (1 << (factors[2] + 1)) - 1;
      NOD *= (factors[2] + 1);
```

```cpp
    }
    for (int i = 3; i * i <= n; i += 2) {
      if (n % i == 0) {
        while (n % i == 0) {
          n /= i;
          factors[i]++;
          total_p_factor++;
        }
        SOD *= (pow(i, factors[i] + 1) - 1) / (i
        ↪    - 1);
        NOD *= (factors[i] + 1);
      }
    }
    if (n > 1) {
      factors[n]++;
      SOD *= (pow(n, 2) - 1) / (n - 1);
      NOD *= 2;
      total_p_factor++;
    }
    num[i].distinct_prime = factors.size();
    num[i].total_prime = total_p_factor;
    num[i].NOD = NOD;
    num[i].SOD = SOD;
    auto lowest_prime = factors.begin();
    auto greatest_prime = factors.rbegin();
    num[i].lowest_prime = lowest_prime->first;
    num[i].greatest_prime =
    ↪    greatest_prime->first;
  }
}
```

8.2 Divisor Sieve

```cpp
const int mxN = 1e5 + 10;
vector<int> divisors[mxN];
void divisorSeive() {
  for (int i = 1; i < mxN; i++) {
    for (int j = i; j < mxN; j += i) {
      divisors[j].push_back(i);
    }
  }
}
```

8.3 Number of Pairs with GCD equal g

```cpp
/*a[i] <= 1e6
for all 1<=g<=n, how many pairs exist such that g
↪    = gcd(a[i], a[j]);
complexity : nlogn
*/
ll n; cin >> n;
ll a[n + 1];
ll cnt[n + 1]; memset(cnt, 0, sizeof cnt);
for (ll i = 1; i <= n; i++) {cin >> a[i];
↪    cnt[a[i]]++;}
ll gcd[n + 1]; memset(gcd, 0, sizeof gcd);
for (ll i = n; i >= 1; i--) {
  ll pair = 0, invalid_pair = 0;
  for (ll j = i; j <= n; j += i) {
    pair += cnt[j];
    invalid_pair += gcd[j];}
  pair = (pair * (pair - 1)) / 2;
  gcd[i] = pair - invalid_pair;
  // how many pairs exist whose gcd is i
}
```

8.4 Phi(1toN)

```cpp
const int mxN = 1e7+10;
vector<int> phi(mxN);
void phi_till() { //O(n.log.log(n))
  for (int i = 0; i < mxN; i++) phi[i] = i;
  for (int i = 2; i < mxN; i++) {
    if (phi[i] == i) {
      for (int j = i; j < mxN; j += i){
        phi[j] -= phi[j] / i;
      }
    }
  }
}
```

8.5 Phi

```cpp
int phi(int n) {  // sqrt(n)
  int result = n;
  for (int i = 2; i * i <= n; i++) {
    if (n % i == 0) {
      while (n % i == 0) n /= i;
      result -= result / i;
    }
  }
  if (n > 1) result -= result / n;
  return result;
}
```

8.6 SOD NOD

```cpp
// SOD = ((P^(x+1)-1)/(P-1)) *
↪    ((Q^(y+1)-1)/(Q-1)) * ((R^(z+1)-1)/(R-1))
// NOD = P^x * Q^y * R^z   => here, P, Q, R are
↪    prime factors & x, y, z are
// powers NOD = (x + 1) (y + 1) (z + 1)
pair<int, int> SOD_NOD(int n) {
  int sod = 1, nod = 1;
  for (int i = 2; i * i <= n; ++i) {
    if (n % i == 0) {
      int pown = 1, pows = 0;
      while (n % i == 0) {
        pown *= i;   // p^e
        pows++;
        n /= i;
      }
      pown *= i;                  // p^e+1
      sod *= (pown - 1) / (i - 1);  //(p^e+1)-1 /
      ↪    p-1
      nod *= (pows + 1);
    }
  }
  if (n > 1) {
    sod *= (n + 1);
    nod *= 2;
  }
  return {sod, nod};
}
```

8.7 Segmented Sieve

```cpp
void segSeive(ll low, ll high) {
  vector < bool > area((high - low) + 1, true);
  for (ll i = 0; primes[i]*primes[i] <= high;
  ↪    i++) {
    ll start = ((low / primes[i]) * primes[i]);
    if (start < low) start += primes[i];
```

```cpp
    for (ll j = start; j <= high; j +=
    ↪    primes[i]) {
      if (j == primes[i]) continue;
      area[j - low] = false;
    }
  }
  for (ll i = 0; i < (high - low) + 1; i++) {
    if (area[i]) {
      if (i + low != 1 and i + low != 0) {
        cout << i + low << endl;
      }
    }
  }
}
```

8.8 Sieve

```cpp
const ll MAXN = 1e7 + 10;
bool prime[MAXN];
vector<ll> prm;
void sieve() {
  prime[0] = prime[1] = true;
  for (ll i = 2; i < MAXN; i++) {
    if (!prime[i]) {
      prm.push_back(i);
      for (ll j = i + i; j < MAXN; j += i) {
        prime[j] = true;
      }
    }
  }
}
```

8.9 Spf

```cpp
const int MAXN = 1e6 + 2;
int spf[MAXN];
vector<int> prms;
void preStore() {
  for (int i = 1; i < MAXN; i++) spf[i] = i;
  for (int i = 2; i < MAXN; i++) {
    if (spf[i] == i) {
      prms.push_back(i);
      for (int j = i + i; j < MAXN; j += i) {
        spf[j] = min(spf[j], i);
      }
    }
  }
}
```

8.10 UniquePF of all elements till MX

```cpp
const int MX = 2e5 + 10;
vector<int> pfac[MX];
void factorize() {
  for (int i = 2; i < MX; i++) {
    if (!pfac[i].empty()) continue;
    for (int j = i; j < MX; j += i)
      pfac[j].push_back(i);
  }
}
```

8.11 int128

```cpp
__int128 read() {
    __int128 x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x * f;
}
void print(__int128 x) {
    if (x < 0) {
        putchar('-');
        x = -x;
    }
    if (x > 9) print(x / 10);
    putchar(x % 10 + '0');
}
```

8.12 nCr and nPr

```cpp
int fact[N], ifact[N];
void prec() {
    fact[0] = 1;
    for (int i = 1; i < N; i++) {
        fact[i] = 1LL * fact[i - 1] * i % mod;
    }
    ifact[N - 1] = power(fact[N - 1], -1);
    for (int i = N - 2; i >= 0; i--) {
        ifact[i] = 1LL * ifact[i + 1] * (i + 1) %
        ↪   mod;
    }
}
int nPr(int n, int r) {
    if (n < r) return 0;
    return 1LL * fact[n] * ifact[n - r] % mod;
}
int nCr(int n, int r) {
    if (n < r) return 0;
    return 1LL * fact[n] * ifact[r] % mod *
    ↪   ifact[n - r] % mod;
}
```

8.13 nCr anup

```cpp
const int MX = 1e6 + 10;
const int M = 1e9 + 7;
int fact[MX], inv_fact[MX];
int modPow(int a, int b) {
    int ans = 1;
    while (b) {
        if (b & 1) ans = (1LL * ans * a) % M;
        a = (1LL * a * a) % M;
        b >>= 1;
    }
    return ans;
}
void precalFact() {
    fact[0] = inv_fact[0] = 1;
    for (int i = 1; i < MX; i++) {
        fact[i] = (1LL * fact[i - 1] * i) % M;
    }
    inv_fact[MX - 1] = modPow(fact[MX - 1], M - 2);
```

```cpp
    for (int i = MX - 2; i >= 1; i--) {
        inv_fact[i] = (1LL * inv_fact[i + 1] * (i +
        ↪   1)) % M;
    }
}
int nCr(int n, int r) {
    if (r < 0 or r > n) return 0;
    return 1LL * fact[n] * inv_fact[r] % M *
    ↪   inv_fact[n - r] % M;
}
```

9 String
9.1 Aho Corasic

```cpp
//number of occourence of word in a text
const ll N = 1e6+10, A = 26;
ll trie[N][A], pos[N], slink[N], dp[N], tot = 1;
vector<int> order;
void initTrie(){
    order.clear();
    while(tot--){
        memset(trie[tot], 0, sizeof(trie[tot]));
    }
    memset(pos, 0, sizeof(pos));
    memset(slink, 0, sizeof(slink));
    memset(dp, 0, sizeof(dp)); tot = 1;
}
void addStr(string &s, int ind){
    ll u = 0;
    for(auto it: s){
        ll n = it-'a';
        if(trie[u][n]==0) trie[u][n] = tot++;
        u = trie[u][n];
    } pos[ind] = u;
}
void build(){
    queue<ll> q; q.push(0);
    while(!q.empty()){
        ll p = q.front(); q.pop();
        order.push_back(p);
        for(ll c = 0; c<A; c++){
            ll u = trie[p][c];
            if(!u) continue;
            q.push(u);
            if(!p) continue;
            ll v = slink[p];
            while(v && !trie[v][c]) v = slink[v];
            slink[u] = trie[v][c];
        }
    }
}
void trav(string &s){
    ll u = 0;
    for(char c: s){
        c-='a';
        while(u && !trie[u][c]) u = slink[u];
        u = trie[u][c]; dp[u]++;
    }
    reverse(order.begin(), order.end());
    for(auto u: order){
        dp[slink[u]]+=dp[u];
    }
}
void solve(){
    ll n; cin>>n;
    string text; cin>>text;
    string s;
```

```cpp
    for(ll i = 0; i<n; i++){
        cin>>s; addStr(s, i);
    }
    build(); trav(text);
    for(ll i = 0; i<n; i++){
        cout<<dp[pos[i]]<<endl;
    }
}
int32_t main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    ll tc = 1;
    cin>>tc;
    for(ll i = 1; i<=tc;i++){
        cout<<"Case "<<i<<": \n";
        initTrie();
        solve();
    }
}
```

9.2 LCS for 3 Strings

```cpp
string a, b, c;
ll dp[55][55][55];
ll lcs(ll i, ll j, ll k) {
    if (i == a.size() or j == b.size() or k ==
    ↪   c.size()) return 0;
    if (dp[i][j][k] != -1) return dp[i][j][k];
    if (a[i] == b[j] and a[i] == c[k]) return 1 +
    ↪   lcs(i + 1, j + 1, k + 1);
    ll ans = 0;
    ans = max(ans, lcs(i, j , k + 1));
    ans = max(ans, lcs(i, j + 1 , k));
    ans = max(ans, lcs(i + 1, j, k));
    return dp[i][j][k] = ans;
}
```

9.3 Manacher Palindrome

```cpp
// pal[1][i] = longest odd (half rounded down)
↪   palindrome around pos i and
// starts at i - pal[1][i] and ends at i +
↪   pal[1][i] pal[0][i] = half length of
// longest even palindrome around pos i, i + 1
↪   and starts at i - par[0][i] + 1
// and ends at i + pal[0][i]
const int N = 5e5 + 10;
int pal[2][N];
void manacher(string& s) {
    int n = s.size(), idx = 2;
    while (idx--) {
        for (int l = -1, r = -1, i = 0; i < n - 1;
        ↪   ++i) {
            if (i > r)
                l = r = i;
            else {
                int k = min(r - i, pal[idx][l + r - i]);
                l = i - k, r = i + k;
            }
            while (l - idx >= 0 and r + 1 < n and s[l
            ↪   - idx] == s[r + 1]) l--, r++;
            pal[idx][i] = r - i;
            // [l - 1 + idx : r] palindrome
        }
    }
}
```

9.4 String Hashing 2

```cpp
const int N = 1000010, MOD = 1e9 + 7;
const ll P[] = {97, 1000003};
ll bigMod(ll a, ll e) {
    if (e == -1) e = MOD - 2;
    ll ret = 1;
    while (e) {
        if (e & 1) ret = ret * a % MOD;
        a = a * a % MOD, e >>= 1;
    }
    return ret;
}
ll pwr[2][N], inv[2][N];
void initHash() {
    for (int it = 0; it < 2; ++it) {
        pwr[it][0] = inv[it][0] = 1;
        ll INV_P = bigMod(P[it], -1);
        for (int i = 1; i < N; ++i) {
            pwr[it][i] = pwr[it][i - 1] * P[it] % MOD;
            inv[it][i] = inv[it][i - 1] * INV_P % MOD;
        }
    }
}
struct RangeHash {
    vector<ll> h[2], rev[2];
    RangeHash(const string S, bool revFlag = 0) {
        for (int it = 0; it < 2; ++it) {
            h[it].resize(S.size() + 1, 0);
            for (int i = 0; i < S.size(); ++i) {
                h[it][i + 1] = (h[it][i] + pwr[it][i +
                 ↪ 1] * (S[i] - 'a' + 1)) % MOD;
            }
            if (revFlag) {
                rev[it].resize(S.size() + 1, 0);
                for (int i = 0; i < S.size(); ++i) {
                    rev[it][i + 1] =
                        (rev[it][i] + inv[it][i + 1] *
                         ↪ (S[i] - 'a' + 1)) % MOD;
                }
            }
        }
    }
    inline ll get(int l, int r) {
        ll one = (h[0][r + 1] - h[0][l]) * inv[0][l
         ↪ + 1] % MOD;
        ll two = (h[1][r + 1] - h[1][l]) * inv[1][l
         ↪ + 1] % MOD;
        if (one < 0) one += MOD;
        if (two < 0) two += MOD;
        return one << 31 | two;
    }
    inline ll getReverse(int l, int r) {
        ll one = (rev[0][r + 1] - rev[0][l]) *
         ↪ pwr[0][r + 1] % MOD;
        ll two = (rev[1][r + 1] - rev[1][l]) *
         ↪ pwr[1][r + 1] % MOD;
        if (one < 0) one += MOD;
        if (two < 0) two += MOD;
        return one << 31 | two;
    }
};
```

9.5 String Hashing

```cpp
const int mod1 = 911382323, mod2 = 972663749, b1
 ↪ = 137, b2 = 139;
const int mxN = 5000010;
int pow_b1[mxN], pow_b2[mxN], inv_b1[mxN],
 ↪ inv_b2[mxN];
int binExp(int base, int power, int mod) {
    int res = 1;
    while (power) {
        if (power & 1) res = (1LL * res * base) %
         ↪ mod;
        base = (1LL * base * base) % mod;
        power >>= 1;
    }
    return res;
}
void pre() {
    pow_b1[0] = pow_b2[0] = 1;
    for (int i = 1; i < mxN; i++) {
        pow_b1[i] = (1LL * pow_b1[i - 1] * b1) %
         ↪ mod1;
        pow_b2[i] = (1LL * pow_b2[i - 1] * b2) %
         ↪ mod2;
    }
    inv_b1[mxN - 1] = binExp(pow_b1[mxN - 1], mod1
     ↪ - 2, mod1);
    inv_b2[mxN - 1] = binExp(pow_b2[mxN - 1], mod2
     ↪ - 2, mod2);
    for (int i = mxN - 2; i >= 0; i--) {
        inv_b1[i] = (1LL * inv_b1[i + 1] * b1) %
         ↪ mod1;
        inv_b2[i] = (1LL * inv_b2[i + 1] * b2) %
         ↪ mod2;
    }
}
vector<pair<int, int>> getPref(string& s) {
    int qq = s.size();
    vector<pair<int, int>> hsh(qq);
    for (int i = 0; i < qq; i++) {
        if (i == 0) {
            hsh[i].first = (1LL * s[i] * pow_b1[i]) %
             ↪ mod1;
            hsh[i].second = (1LL * s[i] * pow_b2[i]) %
             ↪ mod2;
        } else {
            hsh[i].first =
                (hsh[i - 1].first + (1LL * s[i] *
                 ↪ pow_b1[i]) % mod1) % mod1;
            hsh[i].second =
                (hsh[i - 1].second + (1LL * s[i] *
                 ↪ pow_b2[i]) % mod2) % mod2;
        }
    }
    return hsh;
}
pair<int, int> getHash(string& str) {
    int hsh1 = 0, hsh2 = 0, sz = str.size();
    for (int i = 0; i < sz; ++i) {
        hsh1 = (hsh1 + 1LL * str[i] * pow_b1[i] %
         ↪ mod1) % mod1;
    }
    for (int i = 0; i < sz; ++i) {
        hsh2 = (hsh2 + 1LL * str[i] * pow_b2[i] %
         ↪ mod2) % mod2;
```

```cpp
    }
    return {hsh1, hsh2};
}
pair<int, int> getSub(int l, int r,
 ↪ vector<pair<int, int>>& v) {
    pair<int, int> q;
    if (l == 0) {
        q = {v[r].first, v[r].second};
    } else {
        int x = (1LL * ((v[r].first - v[l - 1].first
         ↪ + mod1) % mod1) * inv_b1[l]) %
                mod1;
        int y =
            (1LL * ((v[r].second - v[l - 1].second +
             ↪ mod2) % mod2) * inv_b2[l]) %
            mod2;
        q = {x, y};
    }
    return q;
}
```

9.6 Suffix Array

```cpp
// fahimcp495
array<vector<int>, 2> get_sa(string& s, int lim
 ↪ = 128) {  // for integer, just change string
 ↪ to vector<int> and minimum value of vector
 ↪ must be >= 1
    int n = s.size() + 1, k = 0, a, b;
    vector<int> x(begin(s), end(s) + 1), y(n),
     ↪ sa(n), lcp(n), ws(max(n, lim)), rank(n);
    x.back() = 0;
    iota(begin(sa), end(sa), 0);
    for (int j = 0, p = 0; p < n; j = max(1, j *
     ↪ 2), lim = p) {
        p = j, iota(begin(y), end(y), n - j);
        for (int i = 0; i < n; ++i)
            if (sa[i] >= j) y[p++] = sa[i] - j;
        fill(begin(ws), end(ws), 0);
        for (int i = 0; i < n; ++i) ws[x[i]]++;
        for (int i = 1; i < lim; ++i) ws[i] += ws[i
         ↪ - 1];
        for (int i = n; i--;) sa[--ws[x[y[i]]]] =
         ↪ y[i];
        swap(x, y), p = 1, x[sa[0]] = 0;
        for (int i = 1; i < n; ++i) a = sa[i - 1], b
         ↪ = sa[i], x[b] = (y[a] == y[b] && y[a +
         ↪ j] == y[b + j]) ? p - 1 : p++;
    }
    for (int i = 1; i < n; ++i) rank[sa[i]] = i;
    for (int i = 0, j; i < n - 1; lcp[rank[i++]] =
     ↪ k)
        for (k && k--, j = sa[rank[i] - 1]; s[i + k]
         ↪ == s[j + k]; k++);
    sa.erase(sa.begin()), lcp.erase(lcp.begin());
    return {sa, lcp};
}
```

9.7 Suffix Automata

```cpp
const int N = 2e5 + 10;  // max string size
int len[N], lnk[N]{-1}, last, sz = 1;
unordered_map<char, int> to[N];
void add(char c) {
    int cur = sz++;
```

```cpp
    len[cur] = len[last] + 1;
    int u = last;
    while (u != -1 and !to[u].count(c)) {
        to[u][c] = cur;
        u = lnk[u];
    }
    if (u == -1) {
        lnk[cur] = 0;
    } else {
        int v = to[u][c];
        if (len[v] == len[u] + 1) {
            lnk[cur] = v;
        } else {
            int w = sz++;
            len[w] = len[u] + 1, lnk[w] = lnk[v],
            ↪   to[w] = to[v];
            while (u != -1 and to[u][c] == v) {
                to[u][c] = w;
                u = lnk[u];
            }
            lnk[cur] = lnk[v] = w;
        }
    }
    last = cur;
}
```

9.8 Suffix Automation

```cpp
int len[N], lnk[N]{-1}, last, sz = 1;
unordered_map<char, int> to[N];
void init() {
    while (sz) {
        sz--;
        to[sz].clear();
    }
    last = 0, sz = 1;
}
void add(char c) {
    int cur = sz++;
    int u = last;
    len[cur] = len[last] + 1;
    while (u != -1 and !to[u].count(c)) {
        to[u][c] = cur;
        u = lnk[u];
    }
    if (u == -1) {
        lnk[cur] = 0;
    } else {
        int v = to[u][c];
        if (len[v] == len[u] + 1) {
            lnk[cur] = v;
        } else {
            int w = sz++;
            len[w] = len[u] + 1, lnk[w] = lnk[v],
            ↪   to[w] = to[v];
            while (u != -1 and to[u][c] == v) {
                to[u][c] = w;
                u = lnk[u];
            }
            lnk[cur] = lnk[v] = w;
        }
    }
    last = cur;
}
```

9.9 Trie

```cpp
const ll N = 1e6 + 5, A = 26;
ll trie[N][A], cnt[N], tot = 1, root = 1;
```

```cpp
void initTrie() {
    cnt[tot] = 0;
    root = 1;
}
void addStr(string& s) {
    ll u = 1;
    for (auto it : s) {
        ll n = it - 'a';
        if (trie[u][n] == 0) {
            trie[u][n] = ++tot;
        }
        u = trie[u][n];
        cnt[u]++;
    }
}
ll wordCount(string& s) {
    ll u = 1;
    for (auto it : s) {
        int n = it - 'a';
        if (trie[u][n] == 0) return 0;
        u = trie[u][n];
    }
    return cnt[u];
}
```

10 Tree
10.1 Centroid Decomposition

```cpp
const int N = 2e5+5;
int n, k, sz[N], centered[N], ans = 0;
vector<int> adj[N];
void dfs_sz(int u, int p) {
    sz[u] = 1;
    for (auto v: adj[u]) {
        if (v != p && !centered[v]) {
            dfs_sz(v, u); sz[u] += sz[v];
        }
    }
}
int get_cen(int u, int p, int n) {
    for (auto v: adj[u]) {
        if (v != p && !centered[v] && sz[v] > n/2) {
            return get_cen(v, u, n);
        }
    } return u;
}
int t, tin[N], tout[N], nodes[N], dis[N];
void dfs(int u, int p){
    nodes[t] = u;
    tin[u] = t++;
    for(auto v: adj[u]){
        if(v!=p && !centered[v]){
            dis[v] = dis[u]+1; dfs(v, u);
        }
    } tout[u] = t-1;
}
void go(int u){
    dfs_sz(u, u);
    int c = get_cen(u, u, sz[u]);
    centered[c] = 1; sz[c] = sz[u];
    t = 0; dis[c] = 0; dfs(c, c);
    int cnt[t]{1};
    for(auto v: adj[c]){
        if(centered[v]) continue;
        for(int i = tin[v]; i<=tout[v]; ++i){
            int w = nodes[i];
            if(k-dis[w]>=0 && k-dis[w]<t){
```

```cpp
                ans+=cnt[k-dis[w]];
            }
        }
        for(int i = tin[v]; i<=tout[v]; ++i){
            int w = nodes[i]; cnt[dis[w]]++;
        }
    }
    for(auto v: adj[c]){
        if(!centered[v]){  go(v); }
    }
}
void solve() {
    cin>>n>>k;
    for(ll i = 1; i<n; i++){
        ll u, v; cin>>u>>v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    } go(1);
    cout<<ans<<endl;
}
```

10.2 DSUOnTrees

```cpp
int n, color[MX], ans[MX];
vector<int> g[MX];
set<int> bucket[MX];
int merge(int a, int b) {
    if (bucket[a].size() < bucket[b].size())
    ↪   swap(a, b);
    bucket[a].insert(bucket[b].begin(),
    ↪   bucket[b].end());
    bucket[b].clear();
    return a;
}
int dfs(int u, int p = -1) {
    int cur = u;
    for (int v : g[u])
        if (v != p)
            cur = merge(cur, dfs(v, u));
    ans[u] = (int)bucket[cur].size();
    return cur;
}
void solve() {
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cin >> color[i];
        bucket[i].insert(color[i]);
    }
    // graph input
    dfs(0);
    // print output
}
```

10.3 LCA using binary Lifting

```cpp
int n, l;
vector<vector<int>> adj;
int timer;
vector<int> tin, tout;
vector<vector<int>> up;
void dfs(int v, int p) {
    tin[v] = ++timer;
    up[v][0] = p;
    for (int i = 1; i <= l; ++i)
        up[v][i] = up[up[v][i - 1]][i - 1];
```

```
for (int u : adj[v]) {
    if (u != p)
        dfs(u, v);
    }
    tout[v] = ++timer;
}
bool is_ancestor(int u, int v) {
    return tin[u] <= tin[v] && tout[u] >= tout[v];
}
int lca(int u, int v) {
    if (is_ancestor(u, v))
        return u;
    if (is_ancestor(v, u))
        return v;
    for (int i = l; i >= 0; --i) {
        if (!is_ancestor(up[u][i], v))
            u = up[u][i];
    }
    return up[u][0];
}
void preprocess(int root) {
    tin.resize(n);
    tout.resize(n);
    timer = 0;
    l = ceil(log2(n));
    up.assign(n, vector<int>(l + 1));
    dfs(root, root);
}
```

10.4 LCA

```
const int N = 1e5 + 5;
vector<int> g[N], parent(N), depth(N, 0);
void dfs(int vertex, int par = -1) {
    parent[vertex] = par;
    for (auto child : g[vertex]) {
        if (child != par) {
            depth[child] = depth[vertex] + 1;
            ⌐  dfs(child, vertex);
        }
    }
}
int lca(int x, int y) {
    int diff = min(depth[x], depth[y]);
    while (depth[x] > diff) x = parent[x];
    while (depth[y] > diff) y = parent[y];
    while (x != y) { x = parent[x]; y = parent[y];
    ⌐  }  return x;
}
```

11 Notes
11.1 Geometry
11.1.1 Triangles

Circumradius: $R = \dfrac{abc}{4A}$, Inradius: $r = \dfrac{A}{s}$

The area of a triangle using two sides and the included angle can be given as:

$$A = \frac{1}{2}ab\sin\angle C$$

Length of median (divides triangle into two equal-area triangles): $m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$

Length of bisector (divides angles in two): $s_a =$

$$\sqrt{bc\left[1 - \left(\frac{a}{b+c}\right)^2\right]}$$

Law of tangents: $\dfrac{a+b}{a-b} = \dfrac{\tan\dfrac{\alpha+\beta}{2}}{\tan\dfrac{\alpha-\beta}{2}}$

11.1.2 Quadrilaterals
With side lengths a, b, c, d, diagonals e, f, diagonals angle θ, area A and magic flux $F = b^2 + d^2 - a^2 - c^2$:

$$4A = 2ef \cdot \sin\theta = F\tan\theta = \sqrt{4e^2f^2 - F^2}$$

For cyclic quadrilaterals the sum of opposite angles is $180°$, $ef = ac + bd$, and $A = \sqrt{(p-a)(p-b)(p-c)(p-d)}$.

11.1.3 Spherical coordinates

$$
\begin{aligned}
x &= r\sin\theta\cos\phi & r &= \sqrt{x^2+y^2+z^2} \\
y &= r\sin\theta\sin\phi & \theta &= \text{acos}(z/\sqrt{x^2+y^2+z^2}) \\
z &= r\cos\theta & \phi &= \text{atan2}(y,x)
\end{aligned}
$$

11.1.4 Pick's Theorem:
Given a lattice polygon with non-zero area, we define: S as the area of the polygon, I as the number of integer-coordinate points strictly inside the polygon, B as the number of integer-coordinate points on the boundary of the polygon. Then, Pick's Theorem states:

$$S = I + \frac{B}{2} - 1$$

The number of lattice points on segments (x1, y1) to (x2 , y2) is: gcd(abs(x2 - x1) , abs(y2 - y1)) + 1

11.1.5 Polygon
For a regular polygon with n sides and side length a, the circumradius R is given by:

$$R = \frac{a}{2\sin\left(\frac{\pi}{n}\right)}$$

11.1.6 Area of a Circular Segment
The area of a circular segment, which is the region enclosed by a chord and the corresponding arc, can be calculated using the formula:

$$A = \frac{R^2}{2}(\theta - \sin\theta)$$

where: R is the radius of the circle, θ is the central angle subtended by the chord, in radians.

11.2 Binomial Coefficent
- Factoring in: $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$

- Sum over k: $\sum_{k=0}^{n}\binom{n}{k} = 2^n$

- Alternating sum: $\sum_{k=0}^{n}(-1)^k\binom{n}{k} = 0$

- Even and odd sum: $\sum_{k=0}^{n}\binom{n}{2k} = \sum_{k=0}^{n}\binom{n}{2k+1}2^{n-1}$

- The Hockey Stick Identity
  - (Left to right) Sum over n and k: $\sum_{k=0}^{m}\binom{n+k}{k} = \binom{n+m-1}{m}$
  - (Right to left) Sum over n: $\sum_{m=0}^{n}\binom{m}{k} = \binom{n+1}{k+1}$

- Sum of the squares: $\sum_{k=0}^{n}(\binom{n}{k})^2 = \binom{2n}{n}$

- Weighted sum: $\sum_{k=1}^{n}k\binom{n}{k} = n2^{n-1}$

- Connection with the fibonacci numbers: $\sum_{k=0}\binom{n-k}{k} = F_{n+1}$

- Vandermonde's Identity: $\sum_{i=0}^{k}\binom{m}{i}\binom{n}{k-i} = \binom{m+n}{k}$

- If $f(n,k) = C(n,0) + C(n,1) + ... + C(n,k)$, Then $f(n+1,k) = 2*f(n,k) - C(n,k)$ [For multiple $f(n,k)$ queries, use Mo's algo]

Lucas Theorem

$$\binom{m}{n} \equiv \prod_{i=0}^{k}\binom{m_i}{n_i}(\mod p)$$

- $\binom{m}{n}$ is divisible by p if and only if at least one of the base-p digits of n is greater than the corresponding base-p digit of m.

- The number of entries in the nth row of Pascal's triangle that are not divisible by $p = \prod_{i=0}^{k}(n_i + 1)$

- All entries in the $(p^k - 1)th$ row are not divisble by p.

- $\binom{n}{m} \equiv \lfloor\frac{n}{p}\rfloor(\mod p)$

11.3 Fibonacci Number
1. $k = A - B, F_A F_B = F_{k+1}F_A^2 + F_k F_A F_{A-1}$

2.$\sum_{i=0}^{n}F_i^2 = F_{n+1}F_n$ **3.**$\sum_{i=0}^{n}F_i F_{i+1} = F_{n+1}^2 - (-1)^n$

4.$\sum_{i=0}^{n}F_i F_{i+1} = F_{n+1}^2 - (-1)^n$ **5.**$\sum_{i=0}^{n}F_i F_{i-1} = \sum_{i=0}^{n-1}F_i F_{i+1}$

6.$\gcd(F_m, F_n) = F_{\gcd(m,n)}$ **7.**$\sum_{0\le k\le n}\left(\binom{n-k}{k}\right) = F_{n+1}$

8.$\gcd(F_n, F_{n+1}) = \gcd(F_n, F_{n+2}) = \gcd(F_{n+1}, F_{n+2}) = 1$

11.4 Sums
$1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(2n+1)(n+1)}{6}$

$1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$

$1^4 + 2^4 + 3^4 + \cdots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$

$\sum_{i=1}^{n}i^m = \frac{1}{m+1}\left[(n+1)^{m+1} - 1 - \sum_{i=1}^{n}\left((i+1)^{m+1} - i^{m+1} - (m+1)i^m\right)\right]$

$\sum_{i=1}^{n-1}i^m = \frac{1}{m+1}\sum_{k=0}^{m}\binom{m+1}{k}B_k n^{m+1-k}$

$\sum_{k=0}^{n}kx^k = (x - (n+1)x^{n+1} + nx^{n+2})/(x-1)^2$

11.5 Series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots, (-\infty < x < \infty)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \dots, (-1 < x \le 1)$$

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{2x^3}{32} - \frac{5x^4}{128} + \dots, (-1 \le x \le 1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots, (-\infty < x < \infty)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots, (-\infty < x < \infty)$$

$$(x+a)^{-n} = \sum_{k=0}^{\infty} (-1)^k \binom{n+k-1}{k} x^k a^{-n-k}$$

Generating Function

$$1/(1-x) = 1 + x + x^2 + x^3 + \dots$$

$$1/(1-ax) = 1 + ax + (ax)^2 + (ax)^3 + \dots$$

$$1/(1-x)^2 = 1 + 2x + 3x^2 + 4x^3 + \dots$$

$$1/(1-x)^3 = C(2,2) + C(3,2)x + C(4,2)x^2 + C(5,2)x^3 + \dots$$

$$1/(1-ax)^{(k+1)} = 1 + C(1+k,k)(ax) + C(2+k,k)(ax)^2 + C(3+k,k)(ax)^3 + \dots$$

$$x(x+1)(1-x)^{-3} = 1 + x + 4x^2 + 9x^3 + 16x^4 + 25x^5 + \dots$$

$$e^x = 1 + x + (x^2)/2! + (x^3)/3! + (x^4)/4! + \dots$$

11.6 Pythagorean Triples

The Pythagorean triples are uniquely generated by

$$a = k \cdot (m^2 - n^2), \ b = k \cdot (2mn), \ c = k \cdot (m^2 + n^2),$$

with $m > n > 0$, $k > 0$, $m \perp n$, and either m or n even.

11.7 Number Theory

- HCN: 1e6(240), 1e9(1344), 1e12(6720), 1e14(17280), 1e15(26880), 1e16(41472)

- $gcd(a,b,c,d,\dots) = gcd(a, b-a, c-b, d-c, \dots)$

- $gcd(a+k, b+k, c+k, d+k, \dots) = gcd(a+k, b-a, c-b, d-c, \dots)$

- Primitive root exists iff $n = 1, 2, 4, p^k, 2 \times p^k$, where p is an odd prime.

- If primtive root exists, there are $\phi(\phi(n))$ primtive roots of n.

- The numbers from 1 to n have in total $O(n \log\log n)$ unique prime factors.

- $x \equiv r_1 \mod m1$ and $x \equiv r_2 \mod m2$ has a solution iff $gcd(m_1, m_2)|(r_1 - r_2)$ Solution of $x^2 \equiv a (\mod p)$

- $ca \equiv cb \pmod{m} \iff a \equiv b \pmod{\frac{m}{gcd(n,c)}}$

- $ax \equiv b \pmod{m}$ has a solution $\iff gcd(a,m)|b$

- If $ax \equiv b \pmod{m}$ has a solution, then it has $gcd(a,m)$ solutions and they are separated by $\frac{m}{gcd(a,m)}$

- $ax \equiv 1 \pmod{m}$ has a solution or a is invertible $\pmod{m}$ $\iff gcd(a,m) = 1$

- $x^2 \equiv 1 \pmod{p}$ then $x \equiv \pm 1 \pmod{p}$

- There are $\frac{p-1}{2}$ has no solution.

- There are $\frac{p-1}{2}$ has exaclty two solutions.

- When $p\%4 = 3$, $x \equiv \pm a^{\frac{p+1}{4}}$

- When $p\%8 = 5$, $x \equiv a^{\frac{p+3}{8}}$ or $x \equiv 2^{\frac{p-1}{4}} a^{\frac{p+3}{8}}$

11.7.1 Primes

$p = 962592769$ is such that $2^{21} | p - 1$, which may be useful. For hashing use 970592641 (31-bit number), 31443539979727 (45-bit), 3006703054056749 (52-bit). There are 78498 primes less than 1 000 000.
Primitive roots exist modulo any prime power p^a, except for $p = 2, a > 2$, and there are $\phi(\phi(p^a))$ many. For $p = 2, a > 2$, the group $\mathbb{Z}_{2^a}^{\times}$ is instead isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_{2^{a-2}}$.

11.7.2 Estimates

$\sum_{d|n} d = O(n \log\log n)$.
The number of divisors of n is at most around 100 for $n < 5e4$, 500 for $n < 1e7$, 2000 for $n < 1e10$, 200 000 for $n < 1e19$.

11.7.3 Perfect numbers

$n > 1$ is called perfect if it equals sum of its proper divisors and 1. Even n is perfect iff $n = 2^{p-1}(2^p - 1)$ and $2^p - 1$ is prime (Mersenne's). No odd perfect numbers are yet found.

11.7.4 Carmichael numbers

A positive composite n is a Carmichael number ($a^{n-1} \equiv 1 \pmod{n}$ for all a: $gcd(a,n) = 1$), iff n is square-free, and for all prime divisors p of n, $p - 1$ divides $n - 1$.

11.7.5 Totient

- If p is a prime $(p^k) = p^k - p^{k-1}$
- If a b are relatively prime, $\phi(ab) = \phi(a)\phi(b)$
- $\phi(n) = n(1 - \frac{1}{p_1})(1 - \frac{1}{p_2})(1 - \frac{1}{p_3})\dots(1 - \frac{1}{p_k})$
- Sum of coprime to $n = n * \frac{\phi(n)}{2}$
- If $n = 2^k$, $\phi(n) = 2^{k-1} = \frac{n}{2}$
- For a b, $\phi(ab) = \phi(a)\phi(b)\frac{d}{\phi(d)}$
- $\phi(ip) = p\phi(i)$ whenever p is a prime and it divides i
- The number of $a(1 <= a <= N)$ such that $gcd(a,N) = d$ is $\phi(\frac{n}{d})$
- If $n > 2$, $\phi(n)$ is always even
- Sum of gcd, $\sum_{i=1}^{n} gcd(i,n) = \sum_{d|n} d\phi(\frac{n}{d})$
- Sum of lcm, $\sum_{i=1}^{n} nlcm(i,n) = \frac{n}{2}(\sum_{d|n}(d\phi(d)) + 1)$
- $\phi(1) = 1$ and $\phi(2) = 1$ which two are only odd ϕ
- $\phi(3) = 2$ and $\phi(4) = 2$ and $\phi(6) = 2$ which three are only prime ϕ
- Find minimum n such that $\frac{\phi(n)}{n}$ is maximum- Multiple of small primes- $2 * 3 * 5 * 7 * 11 * 13 * \dots$

11.7.6 Mobius function

$\mu(1) = 1$. $\mu(n) = 0$, if n is not squarefree. $\mu(n) = (-1)^s$, if n is the product of s distinct primes. Let f, F be functions on positive integers. If for all $n \in N$, $F(n) = \sum_{d|n} f(d)$, then $f(n) = \sum_{d|n} \mu(d)F(\frac{n}{d})$, and vice versa. $\phi(n) = \sum_{d|n} \mu(d)\frac{n}{d}$. $\sum_{d|n} \mu(d) = 1$.
If f is multiplicative, then $\sum_{d|n} \mu(d)f(d) = \prod_{p|n}(1 - f(p))$, $\sum_{d|n} \mu(d)^2 f(d) = \prod_{p|n}(1 + f(p))$.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} [gcd(i,j) = 1] = \sum_{k=1}^{n} \mu(k)\lfloor \frac{n}{k} \rfloor^2$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} gcd(i,j) = \sum_{k=1}^{n} k \sum_{l=1}^{\lfloor \frac{n}{k} \rfloor} \mu(l) \lfloor \frac{n}{kl} \rfloor^2$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} gcd(i,j) = \sum_{k=1}^{n} (\frac{\lfloor \frac{n}{k} \rfloor(1 + \lfloor \frac{n}{k} \rfloor)}{2})^2 \sum_{d|k} \mu(d)kd$$

11.7.7 Legendre symbol

If p is an odd prime, $a \in \mathbb{Z}$, then $\left(\frac{a}{p}\right)$ equals 0, if $p|a$; 1 if a is a quadratic residue modulo p; and -1 otherwise. Euler's criterion: $\left(\frac{a}{p}\right) = a^{\left(\frac{p-1}{2}\right)} \pmod{p}$.

11.7.8 Jacobi symbol

If $n = p_1^{a_1} \cdots p_k^{a_k}$ is odd, then $\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{k_i}$.

11.7.9 Primitive roots

If the order of g modulo m (min $n > 0$: $g^n \equiv 1 \pmod{m}$) is $\phi(m)$, then g is called a primitive root. If Z_m has a primitive root, then it has $\phi(\phi(m))$ distinct primitive roots. Z_m has a primitive root iff m is one of 2, 4, p^k, $2p^k$, where p is an odd prime. If Z_m has a primitive root g, then for all a coprime to m, there exists unique integer $i = \text{ind}_g(a)$ modulo $\phi(m)$, such that $g^i \equiv a \pmod{m}$. $\text{ind}_g(a)$ has logarithm-like properties: $\text{ind}(1) = 0$, $\text{ind}(ab) = \text{ind}(a) + \text{ind}(b)$.
If p is prime and a is not divisible by p, then congruence $x^n \equiv a \pmod{p}$ has $gcd(n, p-1)$ solutions if $a^{(p-1)/gcd(n,p-1)} \equiv 1 \pmod{p}$, and no solutions otherwise. (Proof sketch: let g be a primitive root, and $g^i \equiv a \pmod{p}$, $g^u \equiv x \pmod{p}$. $x^n \equiv a \pmod{p}$ iff $g^{nu} \equiv g^i \pmod{p}$ iff $nu \equiv i \pmod{p}$.)

11.7.10 Discrete logarithm problem

Find x from $a^x \equiv b \pmod{m}$. Can be solved in $O(\sqrt{m})$ time and space with a meet-in-the-middle trick. Let $n = \lceil\sqrt{m}\rceil$, and $x = ny - z$. Equation becomes $a^{ny} \equiv ba^z \pmod{m}$. Precompute all values that the RHS can take for $z = 0, 1, \dots, n-1$, and brute force y on the LHS, each time checking whether there's a corresponding value for RHS.

11.7.11 Pythagorean triples

Integer solutions of $x^2 + y^2 = z^2$ All relatively prime triples are given by: $x = 2mn, y = m^2 - n^2, z = m^2 + n^2$ where $m > n, gcd(m,n) = 1$ and $m \not\equiv n \pmod 2$. All other triples are multiples of these. Equation $x^2 + y^2 = 2z^2$ is equivalent to $(\frac{x+y}{2})^2 + (\frac{x-y}{2})^2 = z^2$.

11.7.12 Postage stamps/McNuggets problem

Let a, b be relatively-prime integers. There are exactly $\frac{1}{2}(a-1)(b-1)$ numbers *not* of form $ax+by$ $(x,y \geq 0)$, and the largest is $(a-1)(b-1)-1 = ab-a-b$.

11.7.13 Fermat's two-squares theorem

Odd prime p can be represented as a sum of two squares iff $p \equiv 1 \pmod 4$. A product of two sums of two squares is a sum of two squares. Thus, n is a sum of two squares iff every prime of form $p = 4k+3$ occurs an even number of times in n's factorization.

11.8 Permutations
11.8.1 Factorial

n	1	2	3	4	5	6	7	8	9	10
$n!$	1	2	6	24	120	720	5040	40320	362880	3628800

n	11	12	13	14	15	16	17
$n!$	4.0e7	4.8e8	6.2e9	8.7e10	1.3e12	2.1e13	3.6e14

n	20	25	30	40	50	100	150	171
$n!$	2e18	2e25	3e32	8e47	3e64	9e157	6e262	>DBL_MAX

11.8.2 Cycles

Let $g_S(n)$ be the number of n-permutations whose cycle lengths all belong to the set S. Then

$$\sum_{n=0}^{\infty} g_S(n) \frac{x^n}{n!} = \exp\left(\sum_{n \in S} \frac{x^n}{n}\right)$$

11.8.3 Derangements

Permutations of a set such that none of the elements appear in their original position.

$$D(n) = (n-1)(D(n-1)+D(n-2)) = nD(n-1)+(-1)^n = \left\lfloor \frac{n!}{e} \right\rfloor$$

11.8.4 Burnside's lemma

Given a group G of symmetries and a set X, the number of elements of X *up to symmetry* equals

$$\frac{1}{|G|} \sum_{g \in G} |X^g|,$$

where X^g are the elements fixed by g ($g.x = x$).
If $f(n)$ counts "configurations" (of some sort) of length n, we can ignore rotational symmetry using $G = \mathbb{Z}_n$ to get

$$g(n) = \frac{1}{n} \sum_{k=0}^{n-1} f(\gcd(n,k)) = \frac{1}{n} \sum_{k|n} f(k)\phi(n/k)$$

11.9 Partitions and subsets
11.9.1 Partition function

Number of ways of writing n as a sum of positive integers, disregarding the order of the summands.

$$p(0) = 1, \; p(n) = \sum_{k \in \mathbb{Z}\setminus\{0\}} (-1)^{k+1} p(n-k(3k-1)/2)$$

$$p(n) \sim 0.145/n \cdot \exp(2.56\sqrt{n})$$

n	0	1	2	3	4	5	6	7	8	9	20	50	100
$p(n)$	1	1	2	3	5	7	11	15	22	30	627	~2e5	~2e8

11.9.2 Partition Number

- Time Complexity: $O(n\sqrt{n})$

```
for (int i = 1; i <= n; ++i) {
  pent[2 * i - 1] = i * (3 * i - 1) / 2;
  pent[2 * i] = i * (3 * i + 1) / 2;
}
p[0] = 1;
for (int i = 1; i <= n; ++i) {
  p[i] = 0;
  for (int j = 1, k = 0; pent[j] <= i; ++j) {
    if (k < 2) p[i] = add(p[i], p[i - pent[j]]);
    else p[i] = sub(p[i], p[i - pent[j]]); ++k, k
  }
}
```

- The number of partitions of a positive integer n into exactly k parts equals the number of partitions of n whose largest part equals k

$$p_k(n) = p_k(n-k) + p_{k-1}(n-1)$$

11.9.3 2nd Kaplansky's Lemma

The number of ways of selecting k objects, no two consecutive, from n labelled objects arrayed in a circle is $\frac{n}{k}\binom{n-k-1}{k-1} = \frac{n}{n-k}\binom{n-k}{k}$

11.9.4 Distinct Objects into Distinct Bins

- n distinct objects into r distinct bins $= r^n$
- Among n distinct objects, exactly k of them into r distincts bins $= \binom{n}{k}r^k$
- n distinct objects into r distinct bins such that each bin contains at least one object $= \sum_{i=0}^r (-1)^i \binom{r}{i}(r-i)^n$

11.10 Coloring

The number of labeled undirected graphs with n vertices, $G_n = 2^{\binom{n}{2}}$

The number of labeled directed graphs with n vertices, $G_n = 2^{n(n1)}$

The number of connected labeled undirected graphs with n vertices, $C_n = 2^{\binom{n}{2}} - \frac{1}{n}\sum_{k=1}^{n-1} k\binom{n}{k}2^{\binom{n-k}{2}}C_k = 2^{\binom{n}{2}} - \sum_{k=1}^{n-1}\binom{n-1}{k-1}2^{\binom{n-k}{2}}C_k$

The number of k-connected labeled undirected graphs with n vertices, $D[n][k] = \sum_{s=1}^n \binom{n-1}{s-1}C_s D[n-s][k-1]$

Cayley's formula: the number of trees on n labeled vertices = the number of spanning trees of a complete graph with n labeled vertices $= n^{n-2}$

Number of ways to color a graph using k color such that no two adjacent nodes have same color
Complete graph $= k(k-1)(k-2)...(k-n+1)$
Tree $= k(k-1)^{n-1}$
Cycle $= (k-1)^n + (-1)^n(k-1)$

Number of trees with n labeled nodes: n^{n-2}

11.11 General purpose numbers
11.11.1 Eulerian numbers

Number of permutations $\pi \in S_n$ in which exactly k elements are greater than the previous element. k j:s s.t. $\pi(j) > \pi(j+1)$, $k+1$ j:s s.t. $\pi(j) \geq j$, k j:s s.t. $\pi(j) > j$.

$$E(n,k) = (n-k)E(n-1,k-1)+(k+1)E(n-1,k)$$

$$E(n,0) = E(n,n-1) = 1$$

$$E(n,k) = \sum_{j=0}^k (-1)^j \binom{n+1}{j}(k+1-j)^n$$

11.11.2 Bell numbers

Total number of partitions of n distinct elements. $B(n) = 1,1,2,5,15,52,203,877,4140,21147,\ldots$. For p prime,

$$B(p^m+n) \equiv mB(n)+B(n+1) \pmod p$$

11.11.3 Bernoulli numbers

$\sum_{j=0}^m \binom{m+1}{j}B_j = 0.$ $B_0 = 1, B_1 = -\frac{1}{2}$. $B_n = 0$, for all odd $n \neq 1$.

11.11.4 Catalan numbers

$$C_n = \frac{1}{n+1}\binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!n!}$$

$$C_0 = 1, \; C_{n+1} = \frac{2(2n+1)}{n+2}C_n, \; C_{n+1} = \sum C_i C_{n-i}$$

- $C_n = 1,1,2,5,14,42,132,429,1430,4862,16796,58786,\ldots$
- sub-diagonal monotone paths in an $n \times n$ grid.
- strings with n pairs of parenthesis, correctly nested.
- binary trees with with $n+1$ leaves (0 or 2 children).
- ordered trees with $n+1$ vertices.
- ways a convex polygon with $n+2$ sides can be cut into triangles by connecting vertices with straight lines.
- permutations of $[n]$ with no 3-term increasing subseq.
- Find the count of balanced parentheses sequences consisting of $n+k$ pairs of parentheses where the first k symbols are open brackets.

$$C_n^{(k)} = \frac{k+1}{n+k+1}\binom{2n+k}{n}$$

- Recursive formula of Catalan Numbers:

$$C_n^{(k)} = \frac{(2n+k-1)\cdot(2n+k)}{n\cdot(n+k+1)}C_{n-1}^{(k)}$$

11.11.5 Lucas Number

Number of edge cover of a cycle graph C_n is L_n
$$L(n) = L(n-1)+L(n-2); L(0) = 2, L(1) = 1$$

11.12 Ballot Theorem

Suppose that in an election, candidate A receives a votes and candidate B receives b votes, where a kb for some positive integer k. Compute the number of ways the ballots can be ordered so that A maintains more than k times as many votes as B throughout the counting of the ballots.
The solution to the ballot problem is $\frac{a-kb}{a+b} \times C(a+b,a)$

11.13   Classical Problem

$F(n,k)$ = number of ways to color n objects using exactly k colors

Let $G(n,k)$ be the number of ways to color n objects using no more than k colors.

Then, $F(n,k) = G(n,k) - C(k,1)*G(n,k-1) + C(k,2)*G(n,k-2) - C(k,3)*G(n,k-3)...$

Determining G(n, k) :

Suppose, we are given a 1 * n grid. Any two adjacent cells can not have same color. Then, $G(n,k) = k*((k-1)^{(n-1)})$

If no such condition on adjacent cells. Then, $G(n,k) = k^n$

11.14   Matching Formula

11.14.1   Normal Graph

MM + MEC = n (excluding vertex), IS + VC = G, MIS + MVC = G

11.14.2   Bipartite Graph

MIS = n - MBM, MVC = MBM, MEC = n - MBM

11.15   Inequalities

11.15.1   Titu's Lemma

For positive reals $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$,

$$\frac{a_1^2}{b_1} + \frac{a_2^2}{b_2} + \ldots + \frac{a_n^2}{b_n} \geq \frac{a_1 + a_2 + \ldots + a_n^2}{b_1 + b_2 + \ldots + b_n}$$

Equality holds if and only if $a_i = kb_i$ for a non-zero real constant k.

11.16   Games

11.16.1   Grundy numbers

For a two-player, normal-play (last to move wins) game on a graph (V, E): $G(x) = \text{mex}(\{G(y) : (x,y) \in E\})$, where $\text{mex}(S) = \min\{n \geq 0 : n \notin S\}$. x is losing iff $G(x) = 0$.

11.16.2   Sums of games

- *Player chooses a game and makes a move in it* Grundy number of a position is xor of grundy numbers of positions in summed games.

- *Player chooses a non-empty subset of games (possibly, all) and makes moves in all of them* A position is losing iff each game is in a losing position.

- *Player chooses a proper subset of games (not empty and not all), and makes moves in all chosen ones.* A position is losing iff grundy numbers of all games are equal.

- *Player must move in all games, and loses if can't move in some game* A position is losing if any of the games is in a losing position.

11.16.3   Misère Nim

A position with pile sizes $a_1, a_2, \ldots, a_n \geq 1$, not all equal to 1, is losing iff $a_1 \oplus a_2 \oplus \cdots \oplus a_n = 0$ (like in normal nim.) A position with n piles of size 1 is losing iff n is *odd*.

11.17   Tree Hashing

$f(u) = sz[u] * \sum_{i=0} f(v) * p^i$; $f(v)$ are sorted $f(child) = 1$

11.18   Permutation

To maximize the sum of adjacent differences of a permuation, it is necessary and sufficient to place the smallest half numbers in odd position and the greatest half numbers in even position. Or, vice versa.

11.19   String

- If the sum of length of some strings is N, there can be at most $\sqrt{N}$ distinct length.

- A Text can have at most $O(N \times \sqrt{N})$ distinct substrings that match with given patterns where the sum of the length of the given patterns is N.

- Period = n % (n - pi.back() == 0)? n - pi.back(): n

- The first ($period$) cyclic rotations of a string are distinct. Further cyclic rotations repeat the previous strings.

- S is a palindrome if and only if it's period is a palindrome.

- If S and T are palindromes, then the periods of S T are same if and only if $S + T$ is a palindrome.

11.20   Bit

- (a xor b) and (a + b) has the same parity
- (a + b) = (a xor b) + 2 (a b)
- gcd(a, b) <= a - b <= xor(a, b)

11.21   Convolution

- Hamming Distance: Replace 0 with -1 - SQRT Decomposition: Find block size, B = $\sqrt{(8*n)}$