



Daffodil International University

# DIU\_Pawgrammers

Nezu, SiyamBhuiyan, NoObMin

Team Reference Document

# Contents

## 1 Setup

|     |                         |   |
|-----|-------------------------|---|
| 1.1 | CP_Ubuntu               | 1 |
| 1.2 | CP_Windows              | 1 |
| 1.3 | StressTesting(check.sh) | 1 |
| 1.4 | StressTesting(gen.cpp)  | 2 |

## 2 Data Structures

|      |                               |   |
|------|-------------------------------|---|
| 2.1  | Custom Hash                   | 2 |
| 2.2  | Fast Unordered Map            | 2 |
| 2.3  | GP Hash Table                 | 2 |
| 2.4  | Kahn's Algorithm              | 2 |
| 2.5  | Manacher Algorithm in O(N)    | 2 |
| 2.6  | Mex of All Subarray           | 2 |
| 2.7  | Pbds                          | 3 |
| 2.8  | Segment Tree(BSUA)            | 3 |
| 2.9  | Segment Tree(LzP)             | 3 |
| 2.10 | Segment Tree                  | 4 |
| 2.11 | Sparse Table                  | 4 |
| 2.12 | Tarjan's Algorithm            | 4 |
| 2.13 | Topological sort              | 4 |
| 2.14 | Tree Rerooting(A simple Path) | 4 |
| 2.15 | Tree Rerooting                | 5 |
| 2.16 | main                          | 5 |
| 2.17 | pbds(iterator)                | 5 |
| 2.18 | string hashing                | 7 |

## 3 Dynamic Programming

|     |                                     |   |
|-----|-------------------------------------|---|
| 3.1 | Coin Change(Number of Ways)         | 7 |
| 3.2 | Digit DP                            | 7 |
| 3.3 | LIS                                 | 7 |
| 3.4 | Longest Increasing Subsequence(Set) | 7 |
| 3.5 | Maximum Subarray Sum(Kadanes)       | 7 |

## 4 Geometry

|     |                            |   |
|-----|----------------------------|---|
| 4.1 | Convex Hull                | 8 |
| 4.2 | Integer Points in a Circle | 8 |

## 5 Graph

|      |  |    |
|------|--|----|
| 5.1  | 0-1 BFS (Directed graph)                                     | 8  |
| 5.2  | Articulation Point   | 8  |
| 5.3  | BFS  | 8  |
| 5.4  | Basic Representation of Grap                                 | 8  |
| 5.5  | Bellman Ford   | 8  |
| 5.6  | Bridge Finding DFS   | 9  |
| 5.7  | Count Connected Components                                   | 9  |
| 5.8  | Cycle Detection in DAG                                       | 9  |
| 5.9  | DFS on Tree ( Height Depth )                                 | 9  |
| 5.10 | DFS  | 9  |
| 5.11 | DSU  | 9  |
| 5.12 | Diameter of a Tree   | 9  |
| 5.13 | Dijkstra   | 10 |
| 5.14 | Edge Deletion of Tree  | 10 |
| 5.15 | Euler Tour   | 10 |
| 5.16 | Find Cycle in Graph  | 10 |
| 5.17 | Find shortest path in Grid using BFS (Find Level in 2D Grid) | 10 |
| 5.18 | Flood Fill Algorithm ( DFS in 2D Grid)                       | 11 |
| 5.19 | Floyd Warshall   | 11 |
| 5.20 | LCA (O(logN))  | 11 |
| 5.21 | LCA in a Tree ( Lowest Common Ancestor)                      | 11 |

|      |   |    |
|------|---|----|
| 5.22 | MST   | 11 |
| 5.23 | Max Bipartite Matching[Hopcroft Karp]         | 11 |
| 5.24 | Max Bipartite Matching[Kuhn's]                | 12 |
| 5.25 | Multisource BFS                               | 12 |
| 5.26 | Print all Connected Components                | 12 |
| 5.27 | Subtree Quarries ( Pre-computation using DFS) | 13 |
| 5.28 | Topological Sorting                           | 13 |
| 5.29 | Weighted Union Find                           | 13 |

## 6 Math

|     |                          |    |
|-----|--------------------------|----|
| 6.1 | Formula                  | 13 |
| 6.2 | Matrix Exponentiation    | 13 |
| 6.3 | Matrix Rotation          | 13 |
| 6.4 | Polynomial Interpolation | 14 |
| 6.5 | Sqrt Distinct Floor      | 14 |

## 7 Miscellaneous

|     |                               |    |
|-----|-------------------------------|----|
| 7.1 | Max Subarray Size Sum equal K | 14 |
| 7.2 | Merge Sort                    | 14 |
| 7.3 | Number of Subarray Sum is K   | 14 |

## 8 Number Theory

|      |   |    |
|------|---|----|
| 8.1  | All In One NT                             | 15 |
| 8.2  | Divisor Sieve                             | 15 |
| 8.3  | Euler Totient Sieve in $O(N \log \log N)$ | 15 |
| 8.4  | Factorization                             | 15 |
| 8.5  | Mobius Function                           | 15 |
| 8.6  | Nth recurrence exponentiation             | 16 |
| 8.7  | Number of Pairs with GCD equal g          | 16 |
| 8.8  | Phi(1toN)                                 | 16 |
| 8.9  | Phi                                       | 16 |
| 8.10 | Ranged Coprime in $O(\sqrt{x}+k2^k)$      | 16 |
| 8.11 | SOD NOD                                   | 16 |
| 8.12 | Segmented Sieve                           | 17 |
| 8.13 | Sieve                                     | 17 |
| 8.14 | Smallest prime factor                     | 17 |
| 8.15 | Spf                                       | 17 |
| 8.16 | UniquePF of all elements till MX          | 17 |
| 8.17 | int128                                    | 17 |
| 8.18 | nCr and nPr                               | 17 |
| 8.19 | nCr anup                                  | 17 |

## 9 String

|     |                     |    |
|-----|---------------------|----|
| 9.1 | Aho Corasic         | 17 |
| 9.2 | LCS for 3 Strings   | 17 |
| 9.3 | Manacher Palindrome | 18 |
| 9.4 | String Hashing 2    | 18 |
| 9.5 | String Hashing      | 18 |
| 9.6 | Suffix Array        | 18 |
| 9.7 | Suffix Automata     | 19 |
| 9.8 | Suffix Automation   | 19 |
| 9.9 | Trie                | 19 |

## 10 Tree

|      |                          |    |
|------|--------------------------|----|
| 10.1 | Centroid Decomposition   | 19 |
| 10.2 | DSUOnTrees               | 19 |
| 10.3 | LCA using binary Lifting | 20 |
| 10.4 | LCA                      | 20 |

## 11 Notes

|      |                      |    |
|------|----------------------|----|
| 11.1 | Geometry             | 20 |
| 11.2 | Binomial Coefficient | 20 |
| 11.3 | Fibonacci Number     | 21 |
| 11.4 | Sums                 | 21 |

|       |                         |    |
|-------|-------------------------|----|
| 11.5  | Series                  | 21 |
| 11.6  | Pythagorean Triples     | 21 |
| 11.7  | Number Theory           | 21 |
| 11.8  | Permutations            | 22 |
| 11.9  | Partitions and subsets  | 22 |
| 11.10 | Coloring                | 23 |
| 11.11 | General purpose numbers | 23 |
| 11.12 | Ballot Theorem          | 23 |
| 11.13 | Classical Problem       | 23 |
| 11.14 | Matching Formula        | 23 |
| 11.15 | Inequalities            | 23 |
| 11.16 | Games                   | 23 |
| 11.17 | Tree Hashing            | 23 |
| 11.18 | Permutation             | 23 |
| 11.19 | String                  | 23 |
| 11.20 | Bit                     | 23 |
| 11.21 | Convolution             | 23 |

## 1 Setup

|                |   |    |
|----------------|---|----|
| 1.1            | CP_Ubuntu                                     | 15 |
| {              |   | 15 |
| "cmd":         | ["ulimit -s 268435456; g++ -std=c++20         | 15 |
|                | \$file_name -o \$file_base_name && timeout 4s | 15 |
|                | ./\$file_base_name < inputf.in >              | 15 |
|                | = outputf.in"]                                | 15 |
| "selector":    | "source.cpp",                                 | 16 |
| "shell":       | true,   | 16 |
| "working_dir": | "\$file_path"                                 | 16 |
| }              |   | 16 |

## 1.2 CP\_Windows

|                |  |    |
|----------------|--|----|
| {              |  | 17 |
| "cmd":         | ["g++.exe", "-std=c++20", "\${file}",        | 17 |
|                | "-o", "\${file_base_name}.exe", "&&", "\${f} | 17 |
|                | = ile_base_name}.exe<inputf.in>outputf.in"]  | 17 |
| "selector":    | "source.cpp",                                | 17 |
| "shell":       | true,  | 17 |
| "working_dir": | "\$file_path"                                | 17 |
| }              |  | 17 |

## 1.3 StressTesting(check.sh)

|      |                                 |    |
|------|---------------------------------|----|
| //   | chmod u+x check.sh              | 18 |
| //   | ./check.sh                      | 18 |
| set  | -e                              | 18 |
| g++  | gen.cpp -o gen                  | 18 |
| g++  | code.cpp -o code                | 19 |
| g++  | brute.cpp -o brute              | 19 |
| for  | ((i = 1; ; ++i)); do            | 19 |
|      | echo "Passed on TestCase: " \$i | 19 |
|      | ./gen \$i > in                  | 19 |
|      | ./code < in > out1              | 19 |
|      | ./brute < in > out2             | 19 |
|      | diff -Z out1 out2    break      | 20 |
| done |                                 | 20 |
| echo | -e "WA on the following test:"  | 20 |
| cat  | in                              | 20 |
| echo | -e "\nExpected:"                | 20 |
| cat  | out2                            | 20 |
| echo | -e "\nFound:"                   | 21 |
| cat  | out1                            | 21 |

**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) {
        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**

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

**2.3 GP Hash Table**

```
#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 Kahn's Algorithm**

```
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
using namespace std;
const int N = 1e5;
vector<int> g[N];
int n, m;
int deg[N];
void solve()
{
    cin >> n >> m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
        deg[v]++;
    }
    queue<int> q;
    for(int i = 0; i < n; i++) {
        if(deg[i] == 0) {
            q.push(i);
        }
    }
    while(!q.empty()) {
        auto vertex = q.front(); q.pop();
        cout << vertex << " ";
        for(auto child : g[vertex]) {
            deg[child]--;
            if(deg[child] == 0) q.push(child);
        }
    }
    cout << endl;
}
signed main()
{
    ios_base::sync_with_stdio(0), cin.tie(0);
    int tt = 1;
    while(tt--)
        solve();
}
```

**2.5 Manacher Algorithm in O(N)**

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string s;
    cin >> s;
    int n = s.size();
    vector<int> d1(n); // odd-length palindromes
    int l = 0, r = -1;
    for (int i = 0; i < n; i++) {
        int k = (i > r) ? 1 : min(d1[l + r - i], r
            _ - i + 1);
        while (0 <= i - k && i + k < n && s[i - k]
            _ == s[i + k])
            k++;
        d1[i] = k;
        if (i + k - 1 > r) {
            l = i - k + 1;
            r = i + k - 1;
        }
    }
    vector<int> d2(n); // even-length palindromes
    l = 0, r = -1;
```

```
for (int i = 0; i < n; i++) {
    int k = (i > r) ? 0 : min(d2[l + r - i +
    _ 1], r - i + 1);
    while (0 <= i - k - 1 && i + k < n && s[i
    _ - k - 1] == s[i + k])
        k++;
    d2[i] = k;
    if (i + k - 1 > r) {
        l = i - k;
        r = i + k - 1;
    }
}
// Output longest palindrome length centered
    _ at each index
for (int i = 0; i < n; i++) {
    int longest = max(2 * d1[i] - 1, 2 *
    _ d2[i]);
    cout << longest << " ";
}
}
```

**2.6 Mex of All Subarray**

```
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;
        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; j++) {
            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;
}

```

## 2.7 Pbds

```

#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,
    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)

```

## 2.8 Segment Tree(BSUA)

```

// 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] << " ";
    }
}

```

## 2.9 Segment Tree(LzP)

```

#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 7;
int segTree[4 * N];
int lazy[4 * N];
int n;
int ar[N];
void buildTree(int i, int l, int r) {
    if (l == r) {
        segTree[i] = ar[r]; return;
    }
    int mid = l + (r - l) / 2;
    buildTree(2 * i + 1, l, mid);
    buildTree(2 * i + 2, mid + 1, r);
    segTree[i] = segTree[2 * i + 1] + segTree[2 * i + 2];
}
int Query(int start, int end, int i = 0, int l
    - = 0, int r = n - 1) {
    if (lazy[i] != 0) {
        segTree[i] += (r - l + 1) * lazy[i];
        if (l != r) {

```

```

            lazy[2 * i + 1] += lazy[i];
            lazy[2 * i + 2] += lazy[i];
        }
        lazy[i] = 0;
    }
    if (l > end or r < start) return 0;
    if (l >= start and r <= end) return segTree[i];
    else {
        int mid = l + (r - l) / 2;
        return Query(start, end, 2 * i + 1, l, mid) +
            - Query(start, end, 2 * i + 2, mid + 1, r);
    }
}
// LAZY PROPAGATION
void updateRange(int start, int end, int val,
    - int i = 0, int l = 0, int r = n - 1) {
    if (lazy[i] != 0) {
        segTree[i] += (r - l + 1) * lazy[i];
        if (l != r) {
            lazy[2 * i + 1] += lazy[i];
            lazy[2 * i + 2] += lazy[i];
        }
        lazy[i] = 0;
    }
    // out of range
    if (l > end or r < start or l > r) return;
    // in the range
    if (l >= start and r <= end) {
        segTree[i] += (r - l + 1) * val;
        if (l != r) {
            lazy[2 * i + 1] += val;
            lazy[2 * i + 2] += val;
        }
        return;
    }
    // overlapping
    int mid = l + (r - l) / 2;
    updateRange(start, end, val, 2 * i + 1, l, mid);
    updateRange(start, end, val, 2 * i + 2, mid + 1, r);
    segTree[i] = segTree[2 * i + 1] + segTree[2 * i + 2];
}
void solve() {
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> ar[i];
    }
    buildTree(0, 0, n - 1);
    int k; cin >> k;
    while (k--) {
        char x; cin >> x;
        if (x == 'R') {
            int l, r, val; cin >> l >> r >> val;
            updateRange(l, r, val);
            continue;
        }
        else if (x == 'X') {
            int l, r; cin >> l >> r;
            cout << Query(l, r) << endl;
        }
    }
}
signed main() {
    int tt = 1;

```

```

while(tt--){
    solve();
}

2.10 Segment Tree
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
using namespace std;
const int N = 1e6+7;
int segTree[4*N];
int n;
int ar[N];
void buildTree(int i, int l, int r){
    if(l == r){
        segTree[i] = ar[r]; return;
    }
    int mid = l + (r - l) / 2;
    buildTree(2*i+1, l, mid);
    buildTree(2*i+2, mid+1, r);
    segTree[i] = segTree[2*i+1] + segTree[2*i+2];
}
void updateSegTree(int index, int val, int i = 0, int l = 0, int r = n - 1) {
    if(l == r){
        segTree[i] = val; return;
    }
    int mid = l + (r - l) / 2;
    if(index <= mid){
        updateSegTree(index, val, 2*i+1, l, mid);
    }
    else updateSegTree(index, val, 2*i+2, mid+1, r);
    segTree[i] = segTree[2*i+1] + segTree[2*i+2];
}
int Query(int start, int end, int i = 0, int l = 0, int r = n - 1) {
    if(l > end or r < start) return 0;
    if(l >= start and r <= end) return segTree[i];
    else {
        int mid = l + (r - l)/2;
        return Query(start, end, 2*i+1, l, mid) +
            Query(start, end, 2*i+2, mid+1, r);
    }
}
void solve()
{
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> ar[i];
    }
    buildTree(0, 0, n-1);
    int k; cin >> k;
    while(k--) {
        char u; cin >> u;
        int l, r; cin >> l >> r;
        if(u == 'u') {
            updateSegTree(l, r);
            continue;
        }
        cout << Query(l, r) << endl;
    }
}
signed main()
{
    int tt = 1;

```

```

while(tt--){
    solve();
}

2.11 Sparse Table
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]);
}

2.12 Tarjan's Algorithm
#include <bits/stdc++.h>
#define int long long
#define endl '\n'
using namespace std;
const int N = 55;
vector<int> g[N];
vector<int> timee(N, 0), low(N, 0);
int vis[N];
int n, m;
int cnt = 0;
int timer = 1;
void dfs(int vertex, int par = -1) {
    vis[vertex] = 1;
    timee[vertex] = low[vertex] = timer;
    ++timer;
    for(auto child : g[vertex]) {
        if(child == par) continue;
        if(vis[child] == 1) {
            low[vertex] = min(low[vertex], low[child]);
            cnt = continue;
        }
        dfs(child, vertex);
        low[vertex] = min(low[vertex], low[child]);
        if(low[child] > timee[vertex]) ++cnt;
    }
}
void solve()
{
    cin >> n >> m;
    for(int i = 0; i < m; i++) {
        int x, y; cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    dfs(1);
    cout << cnt << endl;
}
signed main()

```

```

{
    ios_base::sync_with_stdio(0), cin.tie(0);
    int tt = 1;
    // cin >> tt;
    while(tt--){
        solve();
    }
}

```

## 2.13 Topological sort

```

#include <bits/stdc++.h>
#define int long long
#define endl '\n'
using namespace std;
const int N = 1e5;
vector<int> g[N];
int vis[N];
int n, m;
stack<int> tmp;
void dfs(int vertex) {
    vis[vertex] = 1;
    for(auto child : g[vertex]) {
        if(vis[child]) continue;
        dfs(child);
    }
    tmp.push(vertex);
}
void solve()
{
    cin >> n >> m;
    for(int i = 1; i <= m; i++) {
        int u, v; cin >> u >> v;
        g[u].push_back(v);
    }
    for(int i = 0; i < n; i++) {
        if(vis[i]) continue;
        dfs(i);
    }
    vector<int> ans;
    while(tmp.size()) {
        ans.push_back(tmp.top());
        tmp.pop();
    }
    for(auto &it:ans) cout << it << " ";
    cout << endl;
}
signed main()
{
    ios_base::sync_with_stdio(0), cin.tie(0);
    int tt = 1;
    while(tt--){
        solve();
    }
}

```

## 2.14 Tree Rerooting(A simple Path)

```

#include <bits/stdc++.h>
#define int long long
#define endl '\n'
using namespace std;
const int N = 3e5+5;
vector<int> tree[N];
int ar[N];
int dp1[N];
int ans[N];

```

```

int n,k,q;
void clear(int n) {
    for(int i = 0; i <= n; i++) {
        tree[i].clear();
        ar[i] = 0;
        ans[i] = 0;
        dpl[i] = 0;
    }
}
void pre(int vertex, int par) {
    dpl[vertex] = 0;
    for(auto child : tree[vertex]) {
        if(child == par) continue;
        pre(child, vertex);
        dpl[vertex] = max(dpl[vertex], dpl[child]);
    }
    dpl[vertex] += ar[vertex];
}
void reroot(int vertex, int par, int max_val) {
    ans[vertex] =
        max(dpl[vertex], max_val + ar[vertex]);
    int maxi = 0, low = 0;
    for(auto child : tree[vertex]) {
        if(child == par) continue;
        int val = dpl[child];
        if(val > maxi) {
            low = maxi;
            maxi = val;
        }
        else if(val > low) {
            low = val;
        }
    }
    for(auto child : tree[vertex]) {
        if(child == par) continue;
        int x = (dpl[child] == maxi) ? low : maxi;
        int boro = max(max_val, x) + ar[vertex];
        reroot(child, vertex, boro);
    }
}
void solve()
{
    cin >> n >> k >> q;
    clear(n);
    for(int i = 1; i <= k; i++) {
        int x; cin >> x;
        ar[x] = 1;
    }
    for(int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    vector<int> qr;
    while(q--) {
        int x; cin >> x;
        qr.push_back(x);
    }
    pre(1, -1);
    reroot(1, -1, 0);
    int maxi = 0;
    for(int i = 1; i <= n; i++) {
        maxi = max(maxi, ans[i]);
    }
    for(auto &it:qr) {

```

```

        if(ans[it] == maxi) cout << "JA" << endl;
        else cout << "NEIN" << endl;
    }
}
signed main()
{
    ios_base::sync_with_stdio(0), cin.tie(0);
    int tt = 1;
    cin >> tt;
    while(tt--)
        solve();
}

```

### 2.15 Tree Rerooting

```

#include <bits/stdc++.h>
using namespace std;
const int N = 2e5+5;
vector<int> tree[N];
vector<int> subtree_sum(N, 1);
vector<int> dist(N, 0);
int ans[N];
int ar[N];
int n;
void pre(int vertex, int par) {
    subtree_sum[vertex] = ar[vertex];
    for(auto child : tree[vertex]) {
        if(child == par) continue;
        pre(child, vertex);
        dist[vertex] += dist[child] +
            subtree_sum[child];
        subtree_sum[vertex] += subtree_sum[child];
    }
}
void reroot(int vertex, int par) {
    ans[vertex] = dist[vertex];
    for(auto child : tree[vertex]) {
        if(child == par) continue;
        dist[vertex] -= (dist[child] +
            subtree_sum[child]);
        subtree_sum[vertex] -= subtree_sum[child];
        dist[child] += (dist[vertex] +
            subtree_sum[vertex]);
        subtree_sum[child] += subtree_sum[vertex];
        reroot(child, vertex);
        dist[child] -= (dist[vertex] +
            subtree_sum[vertex]);
        subtree_sum[child] -= subtree_sum[vertex];
        dist[vertex] += (dist[child] +
            subtree_sum[child]);
        subtree_sum[vertex] += subtree_sum[child];
    }
}
void solve()
{
    cin >> n;
    for(int i = 1; i <= n; i++) {
        cin >> ar[i];
    }
    for(int i = 1; i < n; i++) {
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
}

```

```

    }
    pre(1, -1);
    reroot(1, -1);
    int maxi = 0;
    for(int i = 1; i <= n; i++) {
        maxi = max(maxi, ans[i]);
    }
    cout << maxi << endl;
}
signed main()
{
    int tt = 1;
    while(tt--)
        solve();
}

```

### 2.16 main

```

// sjdafllksjf
// hello new changes
// more changes
// hello world

```

### 2.17 pbds(iterator)

```

#include <bits/stdc++.h>
using namespace std;
template<typename T, typename Comp =
    std::less<T>>
struct OST {
    struct Node {
        T key;
        int prior, cnt, sz;
        Node *l, *r, *p;
        Node(const T &v){
            key = v;
            prior = rand();
            cnt = 1;
            sz = 1;
            l = r = p = NULL;
        }
    } *root = NULL;
    Comp cmp;
    int getsz(Node* t){ return t ? t->sz : 0; }
    void upd(Node* t){
        if(t){
            t->sz = getsz(t->l) + getsz(t->r) +
                t->cnt;
            if(t->l) t->l->p = t;
            if(t->r) t->r->p = t;
        }
    }
    void rotate_left(Node* &t){
        Node* r = t->r;
        t->r = r->l;
        if(r->l) r->l->p = t;
        r->l = t;
        r->p = t->p;
        t->p = r;
        upd(t);
        upd(r);
        t = r;
    }
}

```

```

void rotate_right(Node* &t){
    Node* l = t->l;
    t->l = l->r;
    if(l->r) l->r->p = t;
    l->r = t;
    l->p = t->p;
    t->p = l;
    upd(t);
    upd(l);
    t = l;
}

void insert(Node* &t, const T &x, Node*
    parent = NULL){
    if(!t){
        t = new Node(x);
        t->p = parent;
        return;
    }
    if(!cmp(x,t->key) && !cmp(t->key,x))
        t->cnt++;
    else if(cmp(x,t->key)){
        insert(t->l, x, t);
        if(t->l->prior > t->prior)
            rotate_right(t);
    }
    else{
        insert(t->r, x, t);
        if(t->r->prior > t->prior)
            rotate_left(t);
    }
    upd(t);
}

void erase(Node* &t, const T &x){
    if(!t) return;
    if(cmp(x,t->key)) erase(t->l,x);
    else if(cmp(t->key,x)) erase(t->r,x);
    else{
        if(t->cnt > 1){
            t->cnt--;
        }
        else{
            if(!t->l) t = t->r;
            else if(!t->r) t = t->l;
            else{
                if(t->l->prior >
                    t->r->prior){
                    rotate_right(t);
                    erase(t->r,x);
                }
                else{
                    rotate_left(t);
                    erase(t->l,x);
                }
            }
        }
        if(t) t->p = NULL;
    }
    if(t) upd(t);
}

void insert(const T &x){ insert(root,x); }
void erase(const T &x){ erase(root,x); }
int size(){ return getsz(root); }
bool empty(){ return size()==0; }

```

```

// ----- order + k-th -----
int order_of_key(const T &x){
    Node* t = root;
    int ans = 0;
    while(t){
        if(cmp(t->key,x)){
            ans += getsz(t->l) + t->cnt;
            t = t->r;
        }
        else t = t->l;
    }
    return ans;
}

T find_by_order(int k){
    Node* t = root;
    if(k < 0 || k >= size()) throw
        out_of_range("index");
    while(t){
        int left = getsz(t->l);
        if(k < left) t = t->l;
        else if(k < left + t->cnt) return
            t->key;
        else{
            k -= left + t->cnt;
            t = t->r;
        }
    }
    throw out_of_range("index");
}

// ----- iterator -----
struct iterator {
    Node* cur;
    iterator(Node* n = NULL): cur(n) {}
    T& operator*(){ return cur->key; }
    T* operator->(){ return &cur->key; }
    bool operator==(const iterator &o) const
        { return cur == o.cur; }
    bool operator!=(const iterator &o) const
        { return cur != o.cur; }
    // next inorder
    iterator& operator++(){
        if(!cur) return *this;
        if(cur->r){
            cur = cur->r;
            while(cur->l) cur = cur->l;
        }
        else{
            Node* p = cur->p;
            while(p && cur == p->r){
                cur = p;
                p = p->p;
            }
            cur = p;
        }
        return *this;
    }
    // prev inorder
    iterator& operator--(){
        if(!cur) return *this;
        if(cur->l){
            cur = cur->l;
            while(cur->r) cur = cur->r;
        }
        else{
            Node* p = cur->p;

```

```

        while(p && cur == p->l){
            cur = p;
            p = p->p;
        }
        cur = p;
        return *this;
    }
};

iterator begin(){
    Node* t = root;
    if(!t) return iterator(NULL);
    while(t->l) t = t->l;
    return iterator(t);
}

iterator end(){
    return iterator(NULL);
}

iterator lower_bound(const T &x){
    Node* t = root;
    Node* ans = NULL;
    while(t){
        if(!cmp(t->key,x)){
            ans = t;
            t = t->l;
        }
        else t = t->r;
    }
    return iterator(ans);
}

iterator upper_bound(const T &x){
    Node* t = root;
    Node* ans = NULL;
    while(t){
        if(cmp(x,t->key)){
            ans = t;
            t = t->l;
        }
        else t = t->r;
    }
    return iterator(ans);
}

};

/*
<-----Using int----->
OST<int> st;
st.insert(5);
st.insert(2);
st.insert(10);
st.insert(5);
cout << st.find_by_order(1) << endl; // 5
cout << st.order_of_key(6) << endl; // 3
cout << st.lower_bound(6) << endl; // 10
cout << st.upper_bound(5) << endl; // 10
*/

/*
<-----Using string----->
OST<string> st;
st.insert("apple");
st.insert("banana");
st.insert("banana");
st.insert("orange");
cout << st.find_by_order(1) << endl; //
    banana
cout << st.order_of_key("banana") << endl; // 1

```

```
cout << st.lower_bound("ball") << endl; //
    banana
*/
/*
<---Using pair<int,int> (lexicographic)--->
OST<pair<int,int>> st;
st.insert({2,3});
st.insert({1,5});
st.insert({2,1});
cout << st.find_by_order(0).first << endl; // 1
cout << st.order_of_key({2,0}) << endl; // 1
*/
```

## 2.18 string hashing

```
#include<bits/stdc++.h>
using namespace std;
#define ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);
#define ll long long
#define endl "\n"
const int p1 = 137, mod1 = 127657753, p2 = 277,
    mod2 = 987654319; // 911382323, 972663749
const int N = 1e6 + 3;
array<int, 2> pref[N], rev[N];
int pw1[N], pw2[N], ipw1[N], ipw2[N];
int power(int a, int n, int mod) {
    int ans = 1 % mod;
    while (n) {
        if (n & 1) ans = 1LL * ans * a % mod;
        a = 1LL * a * a % mod;
        n >>= 1;
    }
    return ans;
}
void prec() {
    pw1[0] = pw2[0] = ipw1[0] = ipw2[0] = 1;
    int ip1 = power(p1, mod1 - 2, mod1);
    int ip2 = power(p2, mod2 - 2, mod2);
    for (int i = 1; i < N; ++i) {
        pw1[i] = 1LL * pw1[i - 1] * p1 % mod1;
        pw2[i] = 1LL * pw2[i - 1] * p2 % mod2;
        ipw1[i] = 1LL * ipw1[i - 1] * ip1 % mod1;
        ipw2[i] = 1LL * ipw2[i - 1] * ip2 % mod2;
    }
}
void build(string& s) {
    int n = s.size();
    for (int i = 0; i < n; ++i) {
        pref[i][0] = 1LL * s[i] * pw1[i] % mod1;
        if (i) pref[i][0] = (pref[i][0] + pref[i - 1][0]) % mod1;
        pref[i][1] = 1LL * s[i] * pw2[i] % mod2;
        if (i) pref[i][1] = (pref[i][1] + pref[i - 1][1]) % mod2;
        rev[i][0] = 1LL * s[i] * ipw1[i] % mod1;
        if (i) rev[i][0] = (rev[i][0] + rev[i - 1][0]) % mod1;
        rev[i][1] = 1LL * s[i] * ipw2[i] % mod2;
        if (i) rev[i][1] = (rev[i][1] + rev[i - 1][1]) % mod2;
    }
}
array<int, 2> get_hash(int i, int j) {
```

```
array<int, 2> ans = {0, 0};
ans[0] = pref[j][0];
if (i) ans[0] = (pref[j][0] - pref[i - 1][0] +
    mod1) % mod1;
ans[1] = pref[j][1];
if (i) ans[1] = (pref[j][1] - pref[i - 1][1] +
    mod2) % mod2;
ans[0] = 1LL * ans[0] * ipw1[i] % mod1;
ans[1] = 1LL * ans[1] * ipw2[i] % mod2;
return ans;
}
array<int, 2> get_rev_hash(int i, int j) {
    array<int, 2> ans = {0, 0};
    ans[0] = rev[j][0];
    if (i) ans[0] = (rev[j][0] - rev[i - 1][0] +
        mod1) % mod1;
    ans[1] = rev[j][1];
    if (i) ans[1] = (rev[j][1] - rev[i - 1][1] +
        mod2) % mod2;
    ans[0] = 1LL * ans[0] * pw1[j] % mod1;
    ans[1] = 1LL * ans[1] * pw2[j] % mod2;
    return ans;
}
void solve(){}
int main(){
    int tc=1;
    prec();
    //cin>>tc;
    while(tc--){
        solve();
    }
}
```

## 3 Dynamic Programming

### 3.1 Coin Change(Number of Ways)

```
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

```
vector<int> nmbrs;
int dp[10][10][2];
int dgt_dp(int idx, int tight, int oneCnt) {
    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 & i == nmbrs[idx]);
    sum += dgt_dp(idx + 1, newTight, oneCnt + (i
        == 1));
}
return dp[idx][oneCnt][tight] = sum;
}
```

### 3.3 LIS

```
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 Longest Increasing Subsequence(Set)

```
int LIS(vector<int> &v){
    multiset<int> st;
    for(int i=0; i<(int)v.size(); i++){
        auto it = st.lower_bound(v[i]);
        if(it != st.end())
            st.erase(it);
        st.insert(v[i]);
    }
    return (int)st.size();
}
```

### 3.5 Maximum Subarray Sum(Kadanes)

```
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

```
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);
    });
    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

```
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 0-1 BFS (Directed graph)

```
#include "bits/stdc++.h"
#define int long long
using namespace std;
const int INF = 1e9 + 10;
const int N = 1e5 + 10;
vector<int> level(N, INF);
vector<pair<int, int>> g[N];
int n, m;
int bfs() {
    deque<int> q;
    q.push_back(1);
    level[1] = 0;
    while(!q.empty()) {
        int cur_v = q.front();
        q.pop_front();
        for(auto child : g[cur_v]) {
            int child_v = child.first;
            int wt = child.second;
            if (level[cur_v] + wt < level[child_v]) {
                level[child_v] = level[cur_v] + wt;
                if (wt == 0) {
                    q.push_front(child_v);
                } else q.push_back(child_v);
            }
        }
    }
    return level[n] == INF ? -1 : level[n];
}
```

### signed main()

```
{
    cin >> n >> m;
    for(int i = 0; i < m; i++) {
        int x, y; cin >> x >> y;
        if(x == y) continue;
        g[x].push_back({y, 0});
        g[y].push_back({x, 1});
    }
    cout << bfs() << endl;
}
```

### 5.2 Articulation Point

```
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);
        }
    }
}
```

### 5.3 BFS

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5;
bool vis[N];
int level[N];
std::vector<int> tree[N];
void bfs(int source) {
    queue<int> q;
    q.push(source);
```

```
vis[source] = true;
while(!q.empty()) {
    int cur_v = q.front(); q.pop();
    for(int child : tree[cur_v]) {
        if(vis[child] == false) {
            q.push(child);
            vis[child] = true;
            level[child] = level[cur_v] + 1;
        }
    }
}
```

### 5.4 Basic Representation of Graph

```
#include "bits/stdc++.h"
#define int long long
using namespace std;
signed main() {
    // For Adjacency Matrix
    // SC -> O(V*V);
    int v, e; cin >> v >> e;
    int graph[n+1][n+1];
    for(int i = 0; i < e; i++) {
        int v1, v2; cin >> v1 >> v2;
        graph[v1][v2] = 1;
        graph[v2][v1] = 1;
    }
}
```

### 5.5 Bellman Ford

```
#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];
            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';
}

```

#### 5.6 Bridge Finding DFS

```

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);
            }
        }
    }
}

```

#### 5.7 Count Connected Components

```

int cnt = 0;
for (int i = 1; i <= v; i++) {
    if (vis[i] == true) continue;
    dfs(i);
    ++cnt;
}
cout << cnt << endl;

```

#### 5.8 Cycle Detection in DAG

```

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]) {
            return true;
        }
    }
    return false;
}

```

```

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

```

#### 5.9 DFS on Tree (Height Depth)

```

const int N = 1e5+5;
vector<int> tree[N];
int depth[N], height[N];
bool dfs(int vertex, int parent = -1) {
    for (int child : tree[vertex]) {
        if (child == parent) continue;
        depth[child] = depth[vertex] + 1;
        dfs(child, vertex);
        height[vertex] = max(height[vertex],
            height[child] + 1);
    }
}

```

#### 5.10 DFS

```

#include "bits/stdc++.h"
#define int long long
using namespace std;
const int N = 1e5+5;
vector<int> g[N];
bool vis[N];
void dfs(int vertex) {
    cout << vertex << " ";
    vis[vertex] = true;
    for (auto child : g[vertex]) {
        if (vis[child] == true) continue;
        dfs(child);
    }
}
signed main() {
    int v, e; cin >> v >> e;
    for (int i = 0; i < e; i++) {
        int x, y; cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    dfs(1);
    cout << endl;
}

```

#### 5.11 DSU

```

#include <bits/stdc++.h>
#define int long long
using namespace std;
const int N = 100005;
int parent[N];
int size[N];
// make, find, union
void make(int v) {

```

```

parent[v] = v;
size[v] = 1;
}
int find (int v) {
    if (parent[v] == v) return v;
    return parent[v] = find(parent[v]);
}
void Union(int a, int b) {
    a = find(a);
    b = find(b);
    if (a != b) {
        if (size[a] < size[b]) {
            swap(a, b);
        }
        parent[b] = a;
        size[a] += size[b];
    }
}
void solve() {
    int n, k; cin >> n >> k;
    for (int i = 1; i <= n; i++) {
        make(i);
    }
    cout << "Before union " << endl;
    for (int i = 1; i <= n; i++) {
        cout << "For " << i << ": " << parent[i] <<
            endl;
    }
    while (k--) {
        int u, v; cin >> u >> v;
        Union(u, v);
    }
    int cnt = 0;
    cout << "After union " << endl;
    for (int i = 1; i <= n; i++) {
        if (parent[i] == i) ++cnt;
        cout << "For " << i << ": " << parent[i] <<
            endl;
    }
    cout << cnt << endl;
}
signed main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int t = 1;
    while (t--) {
        solve();
    }
    return 0;
}

```

#### 5.12 Diameter of a Tree

```

#include "bits/stdc++.h"
#define int long long
using namespace std;
const int N = 1e5+5;
vector<int> tree[N];
int depth[N];
void dfs(int vertex, int parent = -1) {
    for (int child : tree[vertex]) {
        if (child == parent) continue;
        depth[child] = depth[vertex] + 1;
        dfs(child, vertex);
    }
}

```

```

}
signed main() {
    int v; cin >> v;
    for(int i = 0; i < v-1; i++) {
        int x,y; cin >> x >> y;
        tree[x].push_back(y);
        tree[y].push_back(x);
    }
    dfs(1);
    int max_depth = -1;
    int max_vertex = -1;
    for(int i = 1; i <= v; i++) {
        if(max_depth < depth[i]) {
            max_depth = depth[i];
            max_vertex = i;
        }
        depth[i] = 0;
    }
    int diameter = -1;
    dfs(max_vertex);
    for(int i = 1; i <= n; i++) {
        diameter = max(diameter, depth[i]);
    }
    cout << "Maximum diameter : " << diameter <<
    endl;
}

```

#### 5.13 Dijkstra

```

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

```

#### 5.14 Edge Deletion of Tree

```

#include <bits/stdc++.h>
using namespace std;
const int mod = (int)1e9+7;
const int N = (int)1e5+10;
vector<int> tree[N];
int val[N];

```

```

int subtree_sum[N];
void dfs(int vertex, int parent = -1) {
    subtree_sum[vertex] += val[vertex];
    for(int child : tree[vertex]) {
        dfs(child, vertex);
        subtree_sum[vertex] += subtree_sum[child] +
        0LL;
    }
}
int main()
{
    int n; cin >> n;
    for(int i = 1; i <= n; i++) cin >> val[i];
    dfs(1);
    int maxi = 0;
    for(int i = 2; i <= n; i++) {
        int sum1 = subtree_sum[i];
        int sum2 = subtree_sum[1] - sum1;
        maxi = max(maxi, (sum1*sum2*1LL) % mod);
    }
    cout << "Maximum sum possible : " << maxi <<
    endl;
}

```

#### 5.15 Euler Tour

```

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;
}

```

#### 5.16 Find Cycle in Graph

```

bool dfs(int vertex, int parent) {
    vis[vertex] = true;
    bool isLoopExist = false;
    for(auto child : g[vertex]) {
        if(vis[child] == true and child == parent)
            continue;
        if(vis[child] == true) return true;
        isLoopExist = isLoopExist | dfs(child,
            vertex);
    }
    return isLoopExist;
}

```

#### 5.17 Find shortest path in Grid using BFS (Find Level in 2D Grid)

```

#include "bits/stdc++.h"
#define int long long
using namespace std;
const int INF = 1e9+10;
int level[8][8];
int vis[8][8];
void reset() {

```

```

for(int i = 0; i < 8; i++) {
    for(int j = 0; j < 8; j++) {
        level[i][j] = INF;
        vis[i][j] = 0;
    }
}
bool isValid(int x, int y) {
    return (x >= 0 and y >= 0 and x < 8 and y < 8);
}
vector<pair<int,int>> movements = {
    {1,2},{-1,2},
    {2,1},{2,-1},
    {-2,1},{-2,-1},
    {1,-2},{-1,-2}
};
int getX(string &a) {
    return a[0] - 'a';
}
int getY(string &a) {
    return a[1] - '1';
}
int bfs(string &source, string &dest) {
    reset();
    int sourceX = getX(source);
    int sourceY = getY(source);
    int destX = getX(dest);
    int destY = getY(dest);
    queue<pair<int,int>> q;
    q.push({sourceX, sourceY});
    vis[sourceX][sourceY] = 1;
    level[sourceX][sourceY] = 0;
    while(!q.empty()) {
        auto v = q.front();
        int x = v.first;
        int y = v.second;
        q.pop();
        for(auto move : movements) {
            int childX = move.first + x;
            int childY = move.second + y;
            if(!isValid(childX, childY)) continue;
            if(!vis[childX][childY]) {
                q.push({childX, childY});
                vis[childX][childY] = 1;
                level[childX][childY] = level[x][y] + 1;
            }
        }
        if(level[destX][destY] != INF) {
            break;
        }
    }
    return level[destX][destY];
}
signed main () {
    string s1,s2; cin >> s1 >> s2;
    cout << bfs(s1,s2) << endl;
}

```

**5.18 Flood Fill Algorithm (DFS in 2D Grid)**

```
vector < vector<int> > image;
int initialColor, newColor;
void dfs(int i, int j) {
    int row = image.size();
    int col = image[0].size();

    if(i < 0 or j < 0) return;
    if(i >= row or j >= col) return;
    if(image[i][j] != initialColor) return;
    image[i][j] = newColor;
    dfs(i+1,j);
    dfs(i-1,j);
    dfs(i,j+1);
    dfs(i,j-1);
}
```

**5.19 Floyd Warshall**

```
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]);
        }
    }
}
```

**5.20 LCA (O(logN))**

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 200005;
const int LOG = 20; // since 2^20 > 2e5
vector<int> adj[MAXN];
int up[MAXN][LOG];
int depth[MAXN];
int n;
/* DFS to build depth[] and up[][] */
void dfs(int v, int p) {
    up[v][0] = p; // immediate parent
    for (int i = 1; i < LOG; i++) {
        up[v][i] = up[ up[v][i-1] ][i-1];
    }
    for (int to : adj[v]) {
        if (to == p) continue;
        depth[to] = depth[v] + 1;
        dfs(to, v);
    }
}
/* Find LCA of u and v */
int lca(int u, int v) {
    if (depth[u] < depth[v])
        swap(u, v);
    // 1 Bring u and v to same depth
    int diff = depth[u] - depth[v];
    for (int i = 0; i < LOG; i++) {
        if (diff & (1 << i))
            u = up[u][i];
    }
}
```

```
if (u == v) return u;
// 2 Lift both nodes up until parents differ
for (int i = LOG - 1; i >= 0; i--) {
    if (up[u][i] != up[v][i]) {
        u = up[u][i];
        v = up[v][i];
    }
}
// 3 Parent is LCA
return up[u][0];
}
int main() {
    cin >> n;
    for (int i = 0; i < n - 1; i++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
    depth[1] = 0;
    dfs(1, 0); // root at node 1
    int q;
    cin >> q;
    while (q--) {
        int u, v;
        cin >> u >> v;
        cout << lca(u, v) << "\n";
    }
}
```

**5.21 LCA in a Tree (Lowest Common Ancestor)**

```
#include "bits/stdc++.h"
#define int long long
using namespace std;
const int N = 1e5+5;
vector < int > tree[N];
int par[N];
void dfs(int vertex, int parent = -1) {
    par[vertex] = parent;
    for (int child : tree[vertex]) {
        if (child == parent) continue;
        dfs(child, vertex);
    }
}
vector < int > path(int vertex) {
    vector < int > ans;
    while (vertex != -1) {
        ans.push_back(par[vertex]);
        vertex = par[vertex];
    }
    reverse(ans.begin(), ans.end());
    return ans;
}
signed main() {
    int v; cin >> v;
    for (int i = 0; i < v-1; i++) {
        int x,y; cin >> x >> y;
        tree[x].push_back(y);
        tree[y].push_back(x);
    }
    dfs(1);
    int x,y; cin >> x >> y;
    vector < int > x_par = path(x);
    vector < int > y_par = path(y);
}
```

```
int len = min(x_par.size(), y_par.size());
int lca = -1;
for (int i = 0; i < len; i++) {
    if (x_par[i] == y_par[i]) {
        lca = x_par[i];
    }
    else {
        break;
    }
}
cout << "Lowest common ancestor : " << lca <<
    endl;
```

**5.22 MST**

```
// 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;
}
```

**5.23 Max Bipartite Matching[Hopcroft Karp]**

```
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;
        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;
    }
}
}
}

```

#### 5.24 Max Bipartite Matching[Kuhn's]

```

// 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) {
    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;
    }
}
}
}

```

#### 5.25 Multisource BFS

```

#include "bits/stdc++.h"
using namespace std;
const int INF = 1e9 + 10;
const int N = 1e3 + 10;
int level[N][N];
int vis[N][N];
int g[N][N];
int n, m;

void reset() {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            vis[i][j] = 0;
            level[i][j] = 9;
        }
    }
}

vector < pair < int, int > > movement = {
    {0, 1}, {0, -1}, {1, 0}, {-1, 0},
    {1, 1}, {1, -1}, {-1, 1}, {-1, -1}
};

bool isValid(int x, int y) {
    return (x >= 0 and y >= 0 and x < n and y < m);
}

int bfs() {
    reset();
    int maxi = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            maxi = max(maxi, g[i][j]);
        }
    }
    queue < pair < int, int > > q;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (g[i][j] == maxi) {

```

```

q.push({i, j});
level[i][j] = 0;
vis[i][j] = 1;
}
}
}
int ans = 0;
while (!q.empty()) {
    auto v = q.front(); q.pop();
    int x = v.first;
    int y = v.second;
    for (auto move : movement) {
        int child_x = move.first + x;
        int child_y = move.second + y;
        if (!isValid(child_x, child_y)) {
            continue;
        }
        if (vis[child_x][child_y]) continue;
        q.push({child_x, child_y});
        vis[child_x][child_y] = 1;
        level[child_x][child_y] = level[x][y] + 1;
        ans = max(ans, level[child_x][child_y]);
    }
}
return ans;
}
signed main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> g[i][j];
        }
    }
    cout << bfs() << endl;
}

```

#### 5.26 Print all Connected Components

```

#include "bits/stdc++.h"
#define int long long
using namespace std;
const int N = 1e5 + 5;
vector < int > g[N];
bool vis[N];
vector < vector < int > > cc;
vector < int > current_cc;

void dfs(int vertex) {
    vis[vertex] = true;
    current_cc.push_back(vertex);
    for (auto child : g[vertex]) {
        if (vis[child] == true) continue;
        dfs(child);
    }
}

signed main() {
    int v, e; cin >> v >> e;
    for (int i = 0; i < e; i++) {
        int x, y; cin >> x >> y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    for (int i = 1; i <= v; i++) {

```

```

    if(vis[i] == true) continue;
    current_cc.clear();
    dfs(i);
    cc.push_back(current_cc);
}
cout << "Total connected componenets : " <<
    cc.size() << endl;
for(auto u:cc) {
    for(int vertex : u) {
        cout << vertex << " ";
    }
    cout << endl;
}
}

```

### 5.27 Subtree Quarries ( Pre-computation using DFS)

```

const int N = 1e5+5;
vector < int > tree[N];
int subtreeSum[N];
int evenCount[n], oddCount[N];
void dfs(int vertex, int parent = -1) {
    subtreeSum[vertex] += vertex;
    if(vertex % 2 == 0) {
        evenCount[vertex] += 1;
    }
    else {
        oddCount[vertex] += 1;
    }
    for(int child : tree[vertex]) {
        if(child == parent) continue;
        dfs(child, vertex);
        subtreeSum[vertex] += subtreeSum[child];
        evenCount[vertex] += evenCount[child];
        oddCount[vertex] += oddCount[child];
    }
}

```

### 5.28 Topological Sorting

```

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);
            }
        }
    }
    return order;
}

```

### 5.29 Weighted Union Find

```

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 Formula

```

/*
1. Sum of Pair Sums
  (n - 1) * a[k]
2. Sum of Subarray Sums
  ( a[k] * k * (n - k + 1) )
3. Sum of Subset Sums
  2^(n - 1) * a[k]
4. Product of Pair Products
  ( a[k]^(n - 1) )
5. XOR of Subarray XORs
  { a[k] | k(n - k + 1) is odd }
6. Sum of Max - Min over all Subsets
  ( a[k] * (2^(k - 1) - 2^(n - k)) )
7. Sum of Sum*Length over all Subarrays
  ( a[k] * ( k(n - k + 1)(n + 1) / 2 ) )
8. Sum of Pair XORs
  ( cnt1(b) * cnt0(b) * 2^b )
9. Sum of Pair ANDs
  ( C(cnt1(b), 2) * 2^b )
10. Sum of Pair ORs
  ( (C(n, 2) - C(cnt0(b), 2)) * 2^b )
11. Sum of Subset XORs

```

```

( (cnt1(b) > 0) * 2^(n - 1) * 2^b )
12. Sum of Subset ANDs
  ( (2^(cnt1(b)) - 1) * 2^b )
13. Sum of Subset ORs
  ( (2^n - 2^(cnt0(b))) * 2^b )
*/

```

### 6.2 Matrix Exponentiation

```

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();
    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;
}

```

### 6.3 Matrix Rotation

```

//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

```

**6.4 Polynomial Interpolation**

```
// 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];
    }
    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;
}
```

**6.5 Sqrt Distinct Floor**

```
//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;
}
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){
            i++;
            q = i-prev;
        }
        else{
            q = n-prev+1;
        }
        sum+=p*q;
    }
    cout<<sum<<endl;
}
```

**7 Miscellaneous****7.1 Max Subarray Size Sum equal K**

```
//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**

```
// 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++;
    }
    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**

```
//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_subarray = 0;
    gp_hash_table<int, int, customHash> table;
    table[0] = 1;
    for (int i = 1; i <= n; i++) {
        cnt_subarray += table[pre[i] - k];
        table[pre[i]]++;
    }
    cout << cnt_subarray << endl;
}
```

**8 Number Theory****8.1 All In One NT**

```

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);
        }
        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**

```

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

```

**8.3 Euler Totient Sieve in O(N log log N)**

```

#include <bits/stdc++.h>
using namespace std;
int main() {
    int N;
    cin >> N;
    vector<int> phi(N + 1);
    // Step 1: Initialize
    for (int i = 1; i <= N; i++)
        phi[i] = i;
    // Step 2: Sieve
    for (int i = 2; i <= N; i++) {
        if (phi[i] == i) { // i is prime
            for (int j = i; j <= N; j += i) {
                phi[j] -= phi[j] / i;
            }
        }
    }
    // Step 3: Output
    for (int i = 1; i <= N; i++) {
        cout << "phi(" << i << ") = " << phi[i] <<
            "\n";
    }
    return 0;
}

```

**8.4 Factorization**

```

int lp[1000001];
void factorization(int n) {
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            int cnt = 0;
            while (n % i == 0) {
                ++cnt;
                n /= i;
            }
            cout << i << "^" << cnt << endl;
        }
    }
    if (n > 1) cout << n << "^" << 1 << endl;
}

```

**8.5 Mobius Function**

```

//FULL MÖBIUS FUNCTION TEMPLATE
//Möbius Sieve (O(N log log N))
//Computes:
//mu[n] -> Möbius value
//primes[] -> list of primes
//isPrime[]
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 1000000;
int mu[MAXN + 1];
bool isPrime[MAXN + 1];
vector<int> primes;
void mobius_sieve() {
    for (int i = 1; i <= MAXN; i++)
        mu[i] = 1, isPrime[i] = true;
    for (int i = 2; i <= MAXN; i++) {
        if (isPrime[i]) {
            primes.push_back(i);
            for (int j = i; j <= MAXN; j += i) {
                isPrime[j] = false;
            }
        }
    }
}

```

```

        mu[j] *= -1;
    }
    long long sq = 1LL * i * i;
    if (sq <= MAXN) {
        for (int j = sq; j <= MAXN; j +=
            sq)
            mu[j] = 0;
    }
}
mu[1] = 1;
//Count Numbers n Coprime with x
long long count_coprime(long long n, int x) {
    long long ans = 0;
    for (int d = 1; d * d <= x; d++) {
        if (x % d == 0) {
            ans += 1LL * mu[d] * (n / d);
            if (d != x / d)
                ans += 1LL * mu[x / d] * (n / (x
                    / d));
        }
    }
    return ans;
}
//Count Pairs (i, j) with gcd(i, j) = 1 (1 <= i, j
    <= n)
long long count_coprime_pairs(int n) {
    long long ans = 0;
    for (int d = 1; d <= n; d++) {
        long long t = n / d;
        ans += 1LL * mu[d] * t * t;
    }
    return ans;
}
//Count Coprime Pairs in an Array (ICPC CLASSIC)
//Problem:
//Count (i, j) such that gcd(a[i], a[j]) = 1.
long long count_array_coprime_pairs(vector<int> &
    a) {
    int maxA = *max_element(a.begin(), a.end());
    vector<int> cnt(maxA + 1, 0);
    for (int x : a)
        cnt[x]++;
    vector<int> freq(maxA + 1, 0);
    for (int i = 1; i <= maxA; i++) {
        for (int j = i; j <= maxA; j += i)
            freq[i] += cnt[j];
    }
    long long ans = 0;
    for (int d = 1; d <= maxA; d++) {
        if (mu[d] != 0)
            ans += 1LL * mu[d] * freq[d] *
                (freq[d] - 1) / 2;
    }
    return ans;
}
//Mobius Inversion (Template)
vector<long long> mobius_inversion(vector<long
    long> & f, int n) {
    vector<long long> g(n + 1, 0);
}

```

```

    for (int i = 1; i <= n; i++) {
        for (int j = i; j <= n; j += i) {
            g[j] += mu[i] * f[j / i];
        }
    }
    return g;
}

//When to Use This Template
//Use Möbius when problem mentions:
//gcd = 1
//coprime pairs
//divisor sums
//inclusion-exclusion
//multiplicative functions
//Complexity Summary
//Task
//Time
//Sieve
//O(N log log N)
//Coprime count
//O(x)
//Pair counting
//O(N)
//Array coprime
//O(A log A)

//Contest Tips (IMPORTANT)
//Always precompute once
//Combine with prefix sums
//Watch out for overflow
// values are only {-1, 0, +1}

```

#### 8.6 Nth recurrence exponentiation

```

void multiply(vector<vector<int>>&mat1, vector<vector<int>>&mat2) {
    int x = mat1[0][0] * mat2[0][0] + mat1[0][1] * mat2[1][0];
    int y = mat1[0][0] * mat2[0][1] + mat1[0][1] * mat2[1][1];
    int z = mat1[1][0] * mat2[0][0] + mat1[1][1] * mat2[1][0];
    int w = mat1[1][0] * mat2[0][1] + mat1[1][1] * mat2[1][1];
    mat1[0][0] = x;
    mat1[0][1] = y;
    mat1[1][0] = z;
    mat1[1][1] = w;
}

int func(int n) {
    vector<int> ar(2);
    vector<vector<int>> temp(2, vector<int>(2));
    vector<vector<int>> I(2, vector<int>(2, 0));
    ar[0] = 0, ar[1] = 1;
    I[0][0] = I[1][1] = 1;
    temp[0][0] = 0;
    temp[0][1] = 1;
    temp[1][0] = 1;
    temp[1][1] = 1;

    while(n) {
        if(n&1) {
            multiply(I, temp);
            n--;
        }
        else {

```

```

        multiply(temp, temp);
        n /= 2;
    }
    return (ar[0]*I[0][0])+(ar[1]*I[1][0]);
}

```

#### 8.7 Number of Pairs with GCD equal g

```

/*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
}

```

#### 8.8 Phi(1toN)

```

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;
            }
        }
    }
}

```

#### 8.9 Phi

```

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;
}

```

#### 8.10 Ranged Coprime in $O(\sqrt{x} + k^2)$

```

//Prime factorization of x: O(x)
//Inclusion-Exclusion:  $O(2^k)$ 
//k 9 for x 10
#include <bits/stdc++.h>
using namespace std;
/* get distinct prime factors of x */
vector<long long> prime_factors(long long x) {

```

```

vector<long long> primes;
for (long long i = 2; i * i <= x; i++) {
    if (x % i == 0) {
        primes.push_back(i);
        while (x % i == 0) x /= i;
    }
}
if (x > 1) primes.push_back(x);
return primes;
}

/* count numbers in [1..n] coprime with x */
long long coprime_upto(long long n, long long x) {
    if (n <= 0) return 0;
    vector<long long> p = prime_factors(x);
    int k = p.size();
    long long res = n;
    // Inclusion-Exclusion over subsets
    for (int mask = 1; mask < (1 << k); mask++) {
        long long prod = 1;
        int bits = 0;
        for (int i = 0; i < k; i++) {
            if (mask & (1 << i)) {
                prod *= p[i];
                bits++;
            }
        }
        if (bits % 2 == 1)
            res -= n / prod;
        else
            res += n / prod;
    }
    return res;
}

int main() {
    long long l, r, x;
    cin >> l >> r >> x;
    cout << coprime_upto(r, x) - coprime_upto(l
        - 1, x);
    return 0;
}

```

#### 8.11 SOD NOD

```

// 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;
            sod *= (pown - 1) / (i - 1); // p^e+1
            nod *= (pows + 1); // (p^e+1)-1 /
        }
    }
    return {sod, nod};
}

```

```

}
if (n > 1) {
    sod *= (n + 1);
    nod *= 2;
}
return {sod, nod};
}

```

### 8.12 Segmented Sieve

```

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];
        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;
            }
        }
    }
}

```

### 8.13 Sieve

```

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;
            }
        }
    }
}

```

### 8.14 Smallest prime factor

```

int lp[1000001];
void sieve() {
    int maxN = 1000000;
    for(int i = 0; i <= maxN; i++) lp[i] = -1;
    for(int i = 2; i*i <= maxN; i++) {
        if(lp[i] == -1) {
            for(int j = i; j <= maxN; j+=i) {
                if(lp[j] == -1) lp[j] = i;
            }
        }
    }
}

```

### 8.15 Spf

```

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);
            }
        }
    }
}

```

### 8.16 UniquePF of all elements till MX

```

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);
        }
    }
}

```

### 8.17 int128

```

__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');
}

```

### 8.18 nCr and nPr

```

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;
}

```

### 8.19 nCr anup

```

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);
    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

```

//number of occurrence 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;
    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

```

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

```

// pal[l][i] = longest odd (half rounded down)
// palindrome around pos i and
// starts at i - pal[l][i] and ends at i +
// pal[l][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-->0) {
        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

```

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

```

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;
        hsh2 = (hsh2 + 1LL * str[i] * pow_b2[i] %
            mod2) % mod2;
    }
    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

// 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 +
    1] == y[b + 1]) ? 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

```

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++;
    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

```

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

```

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

```

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){
                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

```

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

```

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

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):  $s_a =$

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

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

#### 11.1.2 Quadrilaterals

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

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

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

#### 11.1.3 Spherical coordinates

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

#### 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  $(x_1, y_1)$  to  $(x_2, y_2)$  is:  $\gcd(\text{abs}(x_2 - x_1), \text{abs}(y_2 - y_1)) + 1$

#### 11.1.5 Polygon

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

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

**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,  $\theta$  is the central angle subtended by the chord, in radians.

**11.2 Binomial Coefficient**

- 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:  $\sum_{k=1}^n k \binom{n}{k} = n 2^{n-1}$
- Connection with the fibonacci numbers:  $\sum_{k=0}^n \binom{n-k}{k} = F_{n+1}$
- Vandermonde's Identity:  $\sum_{i=0}^k \binom{m}{i} \binom{n}{k-i} = \binom{m+n}{k}$
- If  $f(n, k) = C(n, 0) + C(n, 1) + \dots + C(n, k)$ , Then  $f(n+1, k) = 2 * f(n, k) - C(n, k)$  [For multiple  $f(n, k)$  queries, use Mo's algo]

**Lucas Theorem**

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

- $\binom{m}{n}$  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  $n$ th 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 divisible by  $p$ .
- $\binom{n}{m} \equiv \lfloor \frac{n}{p} \rfloor \pmod{p}$

**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.  $\sum_{i=0}^n F_i F_{i-1} = F_{n+1}^2 - (-1)^n$
  6.  $\gcd(F_m, F_n) = F_{\gcd(m, n)}$
  7.  $\sum_{0 \leq k \leq n} \binom{n-k}{k} = F_{n+1}$
  8.  $\gcd(F_n, F_{n+1}) = \gcd(F_n, F_{n+2}) = \gcd(F_{n+1}, F_{n+2}) = 1$
- 11.4 Sums**
- $$1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(2n+1)(n+1)}{6}$$
- $$1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^2(n+1)^2}{4}$$
- $$1^4 + 2^4 + 3^4 + \dots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$
- $$\sum_{i=1}^n i^m = \frac{1}{m+1} \left[ (n+1)^{m+1} - 1 - \sum_{i=1}^n ((i+1)^{m+1} - i^{m+1} - (m+1)i^m) \right]$$
- $$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}$$
- $$\sum_{k=0}^n k x^k = (x - (n+1)x^{n+1} + nx^{n+2}) / (x-1)^2$$
- 11.5 Series**

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

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

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

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

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

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

**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

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

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 primitive root exists, there are  $\phi(\phi(n))$  primitive roots of  $n$ .
- The numbers from 1 to  $n$  have in total  $O(n \log \log n)$  unique prime factors.
- $x \equiv r_1 \pmod{m_1}$  and  $x \equiv r_2 \pmod{m_2}$  has a solution iff  $\gcd(m_1, m_2) | (r_1 - r_2)$  Solution of  $x^2 \equiv a \pmod{p}$
- $ca \equiv cb \pmod{m} \iff a \equiv b \pmod{\frac{n}{\gcd(n, c)}}$
- $ax \equiv b \pmod{m}$  has a solution  $\iff \gcd(a, m) | b$
- If  $ax \equiv b \pmod{m}$  has a solution, then it has  $\frac{m}{\gcd(a, m)}$  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 exactly 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  $a(1 \leq a \leq N)$  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  $\text{lcm}, \sum_{i=1}^n \text{lcm}(i, n) = \frac{n}{2}(\sum_{d|n} (d\phi(d)) + 1)$
- $\phi(1) = 1$  and  $\phi(2) = 1$  which two are only odd  $\phi$
- $\phi(3) = 2$  and  $\phi(4) = 2$  and  $\phi(6) = 2$  which three are only prime  $\phi$
- 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  $n \in \mathbb{N}$ ,  $F(n) = \sum_{d|n} f(d)$ , 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  $\sum_{d|n} \mu(d)f(d) = \prod_{p|n} (1 - f(p))$ ,  $\sum_{d|n} \mu(d)^2 f(d) = \prod_{p|n} (1 + f(p))$ .

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

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

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

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

### 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  $n = \lfloor \sqrt{m} \rfloor$ , 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 |
| $\frac{n!}{n}$  |       | 1     | 11    | 12     | 13     | 14     | 15     | 16       | 17     |         |
| $\frac{n!}{n!}$ | 4.0e7 | 4.8e8 | 6.2e9 | 8.7e10 | 1.3e12 | 2.1e13 | 3.6e14 |          |        |         |
| $\frac{n!}{n!}$ | 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  $\frac{n}{k} \binom{n-k-1}{k-1} = \frac{n-k}{n-k} \binom{n-k}{k}$

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

|   |  |  |
|---|--|--|
| <p><b>11.10 Coloring</b></p> <p>The number of labeled undirected graphs with <math>n</math> vertices, <math>G_n = 2^{\binom{n}{2}}</math></p> <p>The number of labeled directed graphs with <math>n</math> vertices, <math>G_n = 2^{n(n-1)}</math></p> <p>The number of connected labeled undirected graphs with <math>n</math> vertices, <math>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} \frac{n-1}{k-1} 2^{\binom{n-k}{2}} C_k</math></p> <p>The number of <math>k</math>-connected labeled undirected graphs with <math>n</math> vertices, <math>D[n][k] = \sum_{s=1}^n \binom{n-1}{s-1} C_s D[n-s][k-1]</math></p> <p>Cayley's formula: the number of trees on <math>n</math> labeled vertices = the number of spanning trees of a complete graph with <math>n</math> labeled vertices = <math>n^{n-2}</math></p> <p>Number of ways to color a graph using <math>k</math> color such that no two adjacent nodes have same color</p> <p>Complete graph = <math>k(k-1)(k-2)\dots(k-n+1)</math></p> <p>Tree = <math>k(k-1)^{n-1}</math></p> <p>Cycle = <math>(k-1)^n + (-1)^n(k-1)</math></p> <p>Number of trees with <math>n</math> labeled nodes: <math>n^{n-2}</math></p> <p><b>11.11 General purpose numbers</b></p> <p><b>11.11.1 Eulerian numbers</b></p> <p>Number of permutations <math>\pi \in S_n</math> in which exactly <math>k</math> elements are greater than the previous element. <math>k</math> <math>j</math>:s s.t. <math>\pi(j) &gt; \pi(j+1)</math>, <math>k+1</math> <math>j</math>:s s.t. <math>\pi(j) \geq j</math>, <math>k</math> <math>j</math>:s s.t. <math>\pi(j) &gt; j</math>.</p> $E(n, k) = (n-k)E(n-1, k-1) + (k+1)E(n-1, k)$ $E(n, 0) = E(n, n-1) = 1$ $E(n, k) = \sum_{j=0}^k (-1)^j \binom{n+1}{j} (k+1-j)^n$ <p><b>11.11.2 Bell numbers</b></p> <p>Total number of partitions of <math>n</math> distinct elements. <math>B(n) = 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, \dots</math>. For <math>p</math> prime,</p> $B(p^m + n) \equiv mB(n) + B(n+1) \pmod{p}$ <p><b>11.11.3 Bernoulli numbers</b></p> $\sum_{j=0}^m \binom{m+1}{j} B_j = 0. \quad B_0 = 1, B_1 = -\frac{1}{2}. B_n = 0, \text{ for all odd } n \neq 1.$ <p><b>11.11.4 Catalan numbers</b></p> $C_n = \frac{1}{n+1} \binom{2n}{n} = \binom{2n}{n} - \binom{2n}{n+1} = \frac{(2n)!}{(n+1)!n!}$ $C_0 = 1, C_{n+1} = \frac{2(2n+1)}{n+2} C_n, C_{n+1} = \sum C_i C_{n-i}$ <ul style="list-style-type: none"> <li><math>C_n = 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, \dots</math></li> <li>sub-diagonal monotone paths in an <math>n \times n</math> grid.</li> <li>strings with <math>n</math> pairs of parenthesis, correctly nested.</li> <li>binary trees with <math>n+1</math> leaves (0 or 2 children).</li> <li>ordered trees with <math>n+1</math> vertices.</li> <li>ways a convex polygon with <math>n+2</math> sides can be cut into triangles by connecting vertices with straight lines.</li> <li>permutations of <math>[n]</math> with no 3-term increasing subseq.</li> </ul> | <ul style="list-style-type: none"> <li>Find the count of balanced parentheses sequences consisting of <math>n+k</math> pairs of parentheses where the first <math>k</math> symbols are open brackets.</li> </ul> $C_n^{(k)} = \frac{k+1}{n+k+1} \binom{2n+k}{n}$ <ul style="list-style-type: none"> <li>Recursive formula of Catalan Numbers:</li> </ul> $C_n^{(k)} = \frac{(2n+k-1) \cdot (2n+k)}{n \cdot (n+k+1)} C_{n-1}^{(k)}$ <p><b>11.11.5 Lucas Number</b></p> <p>Number of edge cover of a cycle graph <math>C_n</math> is <math>L_n</math></p> $L(n) = L(n-1) + L(n-2); L(0) = 2, L(1) = 1$ <p><b>11.12 Ballot Theorem</b></p> <p>Suppose that in an election, candidate A receives <math>a</math> votes and candidate B receives <math>b</math> votes, where <math>a &gt; b</math> for some positive integer <math>k</math>. Compute the number of ways the ballots can be ordered so that A maintains more than <math>k</math> times as many votes as B throughout the counting of the ballots.</p> <p>The solution to the ballot problem is <math>\frac{a-kb}{a+b} \times C(a+b, a)</math></p> <p><b>11.13 Classical Problem</b></p> <p><math>F(n, k)</math> = number of ways to color <math>n</math> objects using exactly <math>k</math> colors</p> <p>Let <math>G(n, k)</math> be the number of ways to color <math>n</math> objects using no more than <math>k</math> colors.</p> <p>Then, <math>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) \dots</math></p> <p><b>Determining <math>G(n, k)</math> :</b></p> <p>Suppose, we are given a <math>1 * n</math> grid. Any two adjacent cells can not have same color. Then, <math>G(n, k) = k * ((k-1)^{n-1})</math></p> <p>If no such condition on adjacent cells. Then, <math>G(n, k) = k^n</math></p> <p><b>11.14 Matching Formula</b></p> <p><b>11.14.1 Normal Graph</b></p> <p><math>MM + MEC = n</math> (excluding vertex), <math>IS + VC = G</math>, <math>MIS + MVC = G</math></p> <p><b>11.14.2 Bipartite Graph</b></p> <p><math>MIS = n - MBM</math>, <math>MVC = MBM</math>, <math>MEC = n - MBM</math></p> <p><b>11.15 Inequalities</b></p> <p><b>11.15.1 Titu's Lemma</b></p> <p>For positive reals <math>a_1, a_2, \dots, a_n</math> and <math>b_1, b_2, \dots, b_n</math>,</p> $\frac{a_1^2}{b_1} + \frac{a_2^2}{b_2} + \dots + \frac{a_n^2}{b_n} \geq \frac{a_1 + a_2 + \dots + a_n^2}{b_1 + b_2 + \dots + b_n}$ <p>Equality holds if and only if <math>a_i = kb_i</math> for a non-zero real constant <math>k</math>.</p> <p><b>11.16 Games</b></p> <p><b>11.16.1 Grundy numbers</b></p> <p>For a two-player, normal-play (last to move wins) game on a graph <math>(V, E)</math>: <math>G(x) = \text{mex}(\{G(y) : (x, y) \in E\})</math>, where <math>\text{mex}(S) = \min\{n \geq 0 : n \notin S\}</math>. <math>x</math> is losing iff <math>G(x) = 0</math>.</p> <p><b>11.16.2 Sums of games</b></p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul> | <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> </ul> <p><b>11.16.3 Misère Nim</b></p> <p>A position with pile sizes <math>a_1, a_2, \dots, a_n \geq 1</math>, not all equal to 1, is losing iff <math>a_1 \oplus a_2 \oplus \dots \oplus a_n = 0</math> (like in normal nim.) A position with <math>n</math> piles of size 1 is losing iff <math>n</math> is odd.</p> <p><b>11.17 Tree Hashing</b></p> $f(u) = sz[u] * \sum_{i=0} f(v) * p^i; f(v) \text{ are sorted } f(child) = 1$ <p><b>11.18 Permutation</b></p> <p>To maximize the sum of adjacent differences of a permutation, 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.</p> <p><b>11.19 String</b></p> <ul style="list-style-type: none"> <li>If the sum of length of some strings is <math>N</math>, there can be at most <math>\sqrt{N}</math> distinct length.</li> <li>A Text can have at most <math>O(N \times \sqrt{N})</math> distinct substrings that match with given patterns where the sum of the length of the given patterns is <math>N</math>.</li> <li>Period = <math>n \% (n - \text{pi.back}() == 0)? n - \text{pi.back}() : n</math></li> <li>The first (<i>period</i>) cyclic rotations of a string are distinct. Further cyclic rotations repeat the previous strings.</li> <li><math>S</math> is a palindrome if and only if its period is a palindrome.</li> <li>If <math>S</math> and <math>T</math> are palindromes, then the periods of <math>S \ T</math> are same if and only if <math>S + T</math> is a palindrome.</li> </ul> <p><b>11.20 Bit</b></p> <ul style="list-style-type: none"> <li><math>(a \text{ xor } b)</math> and <math>(a + b)</math> has the same parity</li> <li><math>(a + b) = (a \text{ xor } b) + 2(a \text{ and } b)</math></li> <li><math>\text{gcd}(a, b) \leq a - b \leq \text{xor}(a, b)</math></li> </ul> <p><b>11.21 Convolution</b></p> <ul style="list-style-type: none"> <li>Hamming Distance: Replace 0 with -1 - SQRT Decomposition: Find block size, <math>B = \sqrt{8 * n}</math></li> </ul> |
|---|--|--|