

UPLB Eliens - Pegaraw Notebook

Contents

1 Data Structures

1.1 Union Find

2 Dynamic Programming

2.1 Edit Distance

2.2 Knapsack

2.3 Longest Common Subsequence

2.4 Longest Increasing Subsequence

2.5 Subset Sum

3 Graph Theory

3.1 Articulation Point

3.2 Bellman Ford

3.3 Bridge

3.4 Dijkstra

3.5 Floyd Warshall

3.6 Hierholzer

3.7 Is Bipartite

3.8 Is Cyclic

3.9 Kahn

3.10 Maximum Bipartite Matching

3.11 Max Flow

3.12 Prim Mst

3.13 Strongly Connected Component

3.14 Topological Sort

4 Number Theory

4.1 Extended Euclidean

4.2 Find All Solutions

4.3 Linear Sieve

4.4 Miller Rabin

4.5 Modulo Inverse

4.6 Pollard Rho Brent

4.7 Range Sieve

4.8 Segmented Sieve

4.9 Tonelli Shanks

1 Data Structures

1.1 Union Find

```

1 class UF {
2 private: vi p;
3 public:
4     UF(int N) {p.assign(N, -1);}
5     int fs(int i) {return (p[i] < 0) ? i : (p[i] =
6         fs(p[i]));}
7     bool isSame(int i, int j) {return fs(i) == fs(j)
8         );}
9     void join(int i, int j) {
10         int x = fs(i), y = fs(j);
11         if (x == y) {
12             if (x < y) { p[x] += p[y]; p[y] = x; }
13             else { p[y] += p[x]; p[x] = y; } }
14     };

```

2 Dynamic Programming

2.1 Edit Distance

```

1 ll edit_distance(string x, string y, ll n, ll m) {
2     vector<vector<int>> dp(n + 1, vector<int>(m + 1,
3         INF));
4     dp[0][0] = 0;
5     for (int i = 1; i <= n; i++) {
6         dp[i][0] = i;
7     }
8     for (int j = 1; j <= m; j++) {
9         dp[0][j] = j;
10    }
11    for (int i = 1; i <= n; i++) {
12        for (int j = 1; j <= m; j++) {
13            dp[i][j] = min({dp[i - 1][j] + 1, dp[i][j -
14                1] + 1, dp[i - 1][j - 1] + (x[i - 1] !=
15                    y[j - 1])});
16        }
17    }
18    return dp[n][m];
19 }

```

2.2 Knapsack

```

1 ll knapsack(ll W, vector<ll> &wt, vector<ll> &val,
2     ll n) {
3     vector<ll> dp(W + 1, 0);
4     for (ll i = 1; i <= n; i++) {
5         for (ll w = W; w >= 0; w--) {
6             if (wt[i - 1] <= w) {
7                 dp[w] = max(dp[w], dp[w - wt[i - 1]] + val[
8                     i - 1]);
9             }
10        }
11    }
12    return dp[W];
13 }

```

2.3 Longest Common Subsequence

```

1 ll LCS(string x, string y, ll n, ll m) {
2     vector<vector<ll>> dp(n + 1, vector<ll>(m + 1));
3     for (ll i = 0; i <= n; i++) {
4         for (ll j = 0; j <= m; j++) {
5             if (i == 0 || j == 0) {
6                 dp[i][j] = 0;
7             } else if (x[i - 1] == y[j - 1]) {
8                 dp[i][j] = dp[i - 1][j - 1] + 1;
9             } else {
10                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
11            }
12        }
13    }
14    ll index = dp[n][m];
15    vector<char> lcs(index + 1);
16    lcs[index] = '\0';
17    ll i = n, j = m;
18    while (i > 0 && j > 0) {
19        if (x[i - 1] == y[j - 1]) {
20            lcs[index - 1] = x[i - 1];

```

```

21        i--;
22        j--;
23        index--;
24    } else if (dp[i - 1][j] > dp[i][j - 1]) {
25        i--;
26    } else {
27        j--;
28    }
29 }
30 return dp[n][m];
31 }

```

2.4 Longest Increasing Subsequence

```

1 ll get_ceil_idx(vector<ll> &a, vector<ll> &T, ll l,
2     ll r, ll x) {
3     while (r - l > 1) {
4         ll m = l + (r - l) / 2;
5         if (a[T[m]] >= x) {
6             r = m;
7         } else {
8             l = m;
9         }
10    }
11    return r;
12 }
13 ll LIS(ll n, vector<ll> &a) {
14     ll len = 1;
15     vector<ll> T(n, 0), R(n, -1);
16     T[0] = 0;
17     for (ll i = 1; i < n; i++) {
18         if (a[i] < a[T[0]]) {
19             T[0] = i;
20         } else if (a[i] > a[T[len - 1]]) {
21             R[i] = T[len - 1];
22             T[len++] = i;
23         } else {
24             ll pos = get_ceil_idx(a, T, -1, len - 1, a[i]
25                 );
26             R[i] = T[pos - 1];
27             T[pos] = i;
28         }
29     }
30     return len;
31 }

```

2.5 Subset Sum

```

1 bool subset_sum(ll n, vector<ll> &arr, ll sum) {
2     vector<vector<ll>> dp(n + 1, vector<ll>(sum + 1,
3         false));
4     dp[0][0] = true;
5     for (ll i = 1; i <= n; i++) {
6         for (ll j = 0; j <= sum; j++) {
7             dp[i][j] = dp[i - 1][j];
8             if (j >= arr[i]) {
9                 dp[i][j] |= dp[i - 1][j - arr[i]];
10            }
11        }
12    }
13    return dp[n][sum];
14 }

```

3 Graph Theory

3.1 Articulation Point

```

1 void APUtil(vector<vector<ll>> &adj, ll u, vector<
  bool> &visited,
2 vector<ll> &disc, vector<ll> &low, ll &time, ll
  parent, vector<bool> &isAP) {
3   ll children = 0;
4   visited[u] = true;
5   disc[u] = low[u] = ++time;
6   for (auto v : adj[u]) {
7     if (!visited[v]) {
8       children++;
9       APUtil(adj, v, visited, disc, low, time, u,
10              isAP);
11       low[u] = min(low[u], low[v]);
12       if (parent != -1 && low[v] >= disc[u]) {
13         isAP[u] = true;
14       }
15     } else if (v != parent) {
16       low[u] = min(low[u], disc[v]);
17     }
18   }
19   if (parent == -1 && children > 1) {
20     isAP[u] = true;
21   }
22 }
23 void AP(vector<vector<ll>> &adj, ll n) {
24   vector<ll> disc(n), low(n);
25   vector<bool> visited(n), isAP(n);
26   ll time = 0, par = -1;
27   for (ll u = 0; u < n; u++) {
28     if (!visited[u]) {
29       APUtil(adj, u, visited, disc, low, time, par,
30              isAP);
31     }
32     for (ll u = 0; u < n; u++) {
33       if (isAP[u]) {
34         cout << u << " ";
35       }
36     }

```

3.2 Bellman Ford

```

1 void bellman_ford(vector<vector<ll>> &edges, ll n,
  ll m, ll src, vector<ll> &dis) {
2   for (ll i = 0; i < n; i++) {
3     dis[i] = INF;
4   }
5   for (ll i = 0; i < n - 1; i++) {
6     for (ll j = 0; j < m; j++) {
7       ll u = edges[j][0], v = edges[j][1], w =
8         edges[j][2];
9       if (dis[u] < INF) {
10        dis[v] = min(dis[v], dis[u] + w);
11      }
12    }
13    for (ll i = 0; i < m; i++) {
14      ll u = edges[i][0], v = edges[i][1], w = edges[
15        i][2];

```

```

15     if (dis[u] < INF && dis[u] + w < dis[v]) {
16       cout << "The graph contains a negative cycle.
17         " << '\n';
18     }
19   }

```

3.3 Bridge

```

1 void bridge_util(vector<vector<ll>> &adj, ll u,
  vector<bool> &visited, vector<ll> &disc,
  vector<ll> &low, vector<ll> &parent) {
2   static ll time = 0;
3   visited[u] = true;
4   disc[u] = low[u] = ++time;
5   list<ll>::iterator i;
6   for (auto v : adj[u]) {
7     if (!visited[v]) {
8       parent[v] = u;
9       bridge_util(adj, v, visited, disc, low,
10                  parent);
11       low[u] = min(low[u], low[v]);
12       if (low[v] > disc[u]) {
13         cout << u << ' ' << v << '\n';
14       }
15     } else if (v != parent[u]) {
16       low[u] = min(low[u], disc[v]);
17     }
18   }
19 }
20 void bridge(vector<vector<ll>> &adj, ll n) {
21   vector<bool> visited(n, false);
22   vector<ll> disc(n), low(n), parent(n, -1);
23   for (ll i = 0; i < n; i++) {
24     if (!visited[i]) {
25       bridge_util(adj, i, visited, disc, low,
26                  parent);
27     }

```

3.4 Dijkstra

```

1 void dijkstra(ll n, vector<vector<pair<ll, ll>>> &
  adj, vector<ll> &dis) {
2   priority_queue<pair<ll, ll>, vector<pair<ll, ll
3     >>, greater<pair<ll, ll>>> pq;
4   for (int i = 0; i < n; i++) {
5     dis[i] = INF;
6   }
7   dis[0] = 0;
8   pq.push({0, 0});
9   while (!pq.empty()) {
10    auto p = pq.top();
11    pq.pop();
12    ll u = p.second;
13    if (dis[u] != p.first) {
14      continue;
15    }
16    for (auto x : adj[u]) {
17      ll v = x.first, w = x.second;
18      if (dis[v] > dis[u] + w) {
19        dis[v] = dis[u] + w;
20        pq.push({dis[v], v});

```

```

21   }
22   }
23 }

```

3.5 Floyd Warshall

```

1 void floyd_warshall(vector<vector<ll>> &dis, ll n)
2 {
3   for (ll i = 0; i < n; i++) {
4     for (ll j = 0; j < n; j++) {
5       dis[i][j] = (i == j ? 0 : INF);
6     }
7   }
8   for (ll k = 0; k < n; k++) {
9     for (ll i = 0; i < n; i++) {
10      for (ll j = 0; j < n; j++) {
11        if (dis[i][k] < INF && dis[k][j] < INF) {
12          dis[i][j] = min(dis[i][j], dis[i][k] +
13                          dis[k][j]);
14        }
15      }
16    }
17    for (ll i = 0; i < n; i++) {
18      for (ll j = 0; j < n; j++) {
19        for (ll k = 0; k < n; k++) {
20          if (dis[k][i] < 0 && dis[i][k] < INF && dis
21              [k][j] < INF) {
22            dis[i][j] = -INF;
23          }
24        }
25      }

```

3.6 Hierholzer

```

1 void print_circuit(vector<vector<ll>> &adj) {
2   map<ll, ll> edge_count;
3   for (ll i = 0; i < adj.size(); i++) {
4     edge_count[i] = adj[i].size();
5   }
6   if (!adj.size()) {
7     return;
8   }
9   stack<ll> curr_path;
10  vector<ll> circuit;
11  curr_path.push(0);
12  ll curr_v = 0;
13  while (!curr_path.empty()) {
14    if (edge_count[curr_v]) {
15      curr_path.push(curr_v);
16      ll next_v = adj[curr_v].back();
17      edge_count[curr_v]--;
18      adj[curr_v].pop_back();
19      curr_v = next_v;
20    } else {
21      circuit.push_back(curr_v);
22      curr_v = curr_path.top();
23      curr_path.pop();
24    }
25  }
26  for (ll i = circuit.size() - 1; i >= 0; i--) {
27    cout << circuit[i] << ' ';
28  }

```

29 }

3.7 Is Bipartite

```

1 bool is_bipartite(vector<ll> &adj, vector<vector<ll>
  >> &adj, ll n) {
2     queue<pair<ll, ll>> q;
3     for (ll i = 0; i < n; i++) {
4         if (col[i] == -1) {
5             q.push({i, 0});
6             col[i] = 0;
7             while (!q.empty()) {
8                 pair<ll, ll> p = q.front();
9                 q.pop();
10                ll v = p.first, c = p.second;
11                for (ll j : adj[v]) {
12                    if (col[j] == c) {
13                        return false;
14                    }
15                    if (col[j] == -1) {
16                        col[j] = (c ? 0 : 1);
17                        q.push({j, col[j]});
18                    }
19                }
20            }
21        }
22    }
23    return true;
24 }
```

3.8 Is Cyclic

```

1 bool is_cyclic_util(int u, vector<vector<int>> &adj
  , vector<bool> &vis, vector<bool> &rec) {
2     vis[u] = true;
3     rec[u] = true;
4     for(auto v : adj[u]) {
5         if (!vis[v] && is_cyclic_util(v, adj, vis, rec)
6             ) {
7             return true;
8         } else if (rec[v]) {
9             return true;
10        }
11    }
12    rec[u] = false;
13    return false;
14 }
15 bool is_cyclic(int n, vector<vector<int>> &adj) {
16     vector<bool> vis(n, false), rec(n, false);
17     for (int i = 0; i < n; i++) {
18         if (!vis[i] && is_cyclic_util(i, adj, vis, rec)
19             ) {
20             return true;
21         }
22     }
23     return false;
24 }
```

3.9 Kahn

```

1 void kahn(vector<vector<ll>> &adj) {
2     ll n = adj.size();
```

```

3     vector<ll> in_degree(n, 0);
4     for (ll u = 0; u < n; u++) {
5         for (ll v : adj[u]) {
6             in_degree[v]++;
7         }
8     }
9     queue<ll> q;
10    for (ll i = 0; i < n; i++) {
11        if (in_degree[i] == 0) {
12            q.push(i);
13        }
14    }
15    ll cnt = 0;
16    vector<ll> top_order;
17    while (!q.empty()) {
18        ll u = q.front();
19        q.pop();
20        top_order.push_back(u);
21        for (ll v : adj[u]) {
22            if (--in_degree[v] == 0) {
23                q.push(v);
24            }
25        }
26        cnt++;
27    }
28    if (cnt != n) {
29        cout << -1 << '\n';
30        return;
31    }
32    for (ll i = 0; i < (ll) top_order.size(); i++) {
33        cout << top_order[i] << ' ';
34    }
35    cout << '\n';
36 }
```

3.10 Maximum Bipartite Matching

```

1 bool bpm(ll n, ll m, vector<vector<bool>> &bpGraph,
  ll u, vector<bool> &seen, vector<ll> &matchR)
2 {
3     for (ll v = 0; v < m; v++) {
4         if (bpGraph[u][v] && !seen[v]) {
5             seen[v] = true;
6             if (matchR[v] < 0 || bpm(n, m, bpGraph,
7                 matchR[v], seen, matchR)) {
8                 matchR[v] = u;
9                 return true;
10            }
11        }
12    }
13    return false;
14 }
15 ll maxBPM(ll n, ll m, vector<vector<bool>> &bpGraph)
16 {
17     vector<ll> matchR(m, -1);
18     ll result = 0;
19     for (ll u = 0; u < n; u++) {
20         vector<bool> seen(m, false);
21         if (bpm(n, m, bpGraph, u, seen, matchR)) {
22             result++;
23         }
24     }
25     return result;
26 }
```

3.11 Max Flow

```

1 bool bfs(ll n, vector<vector<ll>> &r_graph, ll s,
  ll t, vector<ll> &parent) {
2     vector<bool> visited(n, false);
3     queue<ll> q;
4     q.push(s);
5     visited[s] = true;
6     parent[s] = -1;
7     while (!q.empty()) {
8         ll u = q.front();
9         q.pop();
10        for (ll v = 0; v < n; v++) {
11            if (!visited[v] && r_graph[u][v] > 0) {
12                if (v == t) {
13                    parent[v] = u;
14                    return true;
15                }
16                q.push(v);
17                parent[v] = u;
18                visited[v] = true;
19            }
20        }
21    }
22    return false;
23 }
24 ll fordFulkerson(ll n, vector<vector<ll>> graph, ll
  s, ll t) {
25     ll u, v;
26     vector<vector<ll>> r_graph;
27     for (u = 0; u < n; u++) {
28         for (v = 0; v < n; v++) {
29             r_graph[u][v] = graph[u][v];
30         }
31     }
32     vector<ll> parent;
33     ll max_flow = 0;
34     while (bfs(n, r_graph, s, t, parent)) {
35         ll path_flow = INF;
36         for (v = t; v != s; v = parent[v]) {
37             u = parent[v];
38             path_flow = min(path_flow, r_graph[u][v]);
39         }
40         for (v = t; v != s; v = parent[v]) {
41             u = parent[v];
42             r_graph[u][v] -= path_flow;
43             r_graph[v][u] += path_flow;
44         }
45         max_flow += path_flow;
46     }
47     return max_flow;
48 }
```

3.12 Prim Mst

```

1 vector<ll> prim_mst(ll n, vector<vector<pair<ll, ll>
  >>> &adj) {
2     priority_queue<pair<ll, ll>, vector<pair<ll, ll>
3     >>, greater<pair<ll, ll>>> pq;
4     ll src = 0;
5     vector<ll> key(n, INF), parent(n, -1);
6     vector<bool> in_mst(n, false);
7     pq.push(make_pair(0, src));
8     key[src] = 0;
9     while (!pq.empty()) {
```

```

9     ll u = pq.top().second;
10    pq.pop();
11    if (in_mst[u]) {
12        continue;
13    }
14    in_mst[u] = true;
15    for (auto p : adj[u]) {
16        ll v = p.first, w = p.second;
17        if (in_mst[v] == false && w < key[v]) {
18            key[v] = w;
19            pq.push(make_pair(key[v], v));
20            parent[v] = u;
21        }
22    }
23 }
24 return parent;
25 }

```

3.13 Strongly Connected Component

```

1 void dfs(ll u, vector<vector<ll>> &adj, vector<bool>
  > &visited) {
2     visited[u] = true;
3     cout << u + 1 << ' ';
4     for (ll v : adj[u]) {
5         if (!visited[v]) {
6             dfs(v, adj, visited);
7         }
8     }
9 }
10 vector<vector<ll>> get_transpose(ll n, vector<
  vector<ll>> &adj) {
11     vector<vector<ll>> res(n);
12     for (ll u = 0; u < n; u++) {
13         for (ll v : adj[u]) {
14             res[v].push_back(u);
15         }
16     }
17     return res;
18 }
19 void fill_order(ll u, vector<vector<ll>> &adj,
  vector<bool> &visited, stack<ll> &stk) {
20     visited[u] = true;
21     for (auto v : adj[u]) {
22         if (!visited[v]) {
23             fill_order(v, adj, visited, stk);
24         }
25     }
26     stk.push(u);
27 }
28 void get_scc(ll n, vector<vector<ll>> &adj) {
29     stack<ll> stk;
30     vector<bool> visited(n, false);
31     for (ll i = 0; i < n; i++) {
32         if (!visited[i]) {
33             fill_order(i, adj, visited, stk);
34         }
35     }
36     vector<vector<ll>> transpose = get_transpose(n,
  adj);
37     for (ll i = 0; i < n; i++) {
38         visited[i] = false;
39     }
40     while (!stk.empty()) {
41         ll u = stk.top();
42         stk.pop();
43         if (!visited[u]) {

```

```

44         dfs(u, transpose, visited);
45         cout << '\n';
46     }
47 }
48 }

```

3.14 Topological Sort

```

1 void dfs(ll v) {
2     visited[v] = true;
3     for (ll u : adj[v]) {
4         if (!visited[u]) {
5             dfs(u);
6         }
7     }
8     ans.push_back(v);
9 }
10 void topological_sort() {
11     visited.assign(n, false);
12     ans.clear();
13     for (ll i = 0; i < n; ++i) {
14         if (!visited[i]) {
15             dfs(i);
16         }
17     }
18     reverse(ans.begin(), ans.end());
19 }

```

4 Number Theory

4.1 Extended Euclidean

```

1 ll gcd_extended(ll a, ll b, ll &x, ll &y) {
2     if (b == 0) {
3         x = 1;
4         y = 0;
5         return a;
6     }
7     ll x1, y1, g = gcd_extended(b, a % b, x1, y1);
8     x = y1;
9     y = x1 - (a / b) * y1;
10    return g;
11 }

```

4.2 Find All Solutions

```

1 bool find_any_solution(ll a, ll b, ll c, ll &x0, ll
  &y0, ll &g) {
2     g = gcd_extended(abs(a), abs(b), x0, y0);
3     if (c % g) {
4         return false;
5     }
6     x0 *= c / g;
7     y0 *= c / g;
8     if (a < 0) {
9         x0 = -x0;
10    }
11    if (b < 0) {
12        y0 = -y0;
13    }
14    return true;
15 }

```

```

16 void shift_solution(ll &x, ll &y, ll a, ll b, ll
  cnt) {
17     x += cnt * b;
18     y -= cnt * a;
19 }
20 ll find_all_solutions(ll a, ll b, ll c, ll minx, ll
  maxx, ll miny, ll maxy) {
21     ll x, y, g;
22     if (!find_any_solution(a, b, c, x, y, g)) {
23         return 0;
24     }
25     a /= g;
26     b /= g;
27     ll sign_a = a > 0 ? +1 : -1;
28     ll sign_b = b > 0 ? +1 : -1;
29     shift_solution(x, y, a, b, (minx - x) / b);
30     if (x < minx) {
31         shift_solution(x, y, a, b, sign_b);
32     }
33     if (x > maxx) {
34         return 0;
35     }
36     ll lx1 = x;
37     shift_solution(x, y, a, b, (maxx - x) / b);
38     if (x > maxx) {
39         shift_solution(x, y, a, b, -sign_b);
40     }
41     ll rx1 = x;
42     shift_solution(x, y, a, b, -(miny - y) / a);
43     if (y < miny) {
44         shift_solution(x, y, a, b, -sign_a);
45     }
46     if (y > maxy) {
47         return 0;
48     }
49     ll lx2 = x;
50     shift_solution(x, y, a, b, -(maxy - y) / a);
51     if (y > maxy) {
52         shift_solution(x, y, a, b, sign_a);
53     }
54     ll rx2 = x;
55     if (lx2 > rx2) {
56         swap(lx2, rx2);
57     }
58     ll lx = max(lx1, lx2), rx = min(rx1, rx2);
59     if (lx > rx) {
60         return 0;
61     }
62     return (rx - lx) / abs(b) + 1;
63 }

```

4.3 Linear Sieve

```

1 void linear_sieve(ll N, vector<ll> &lowest_prime,
  vector<ll> &prime) {
2     for (ll i = 2; i <= N; i++) {
3         if (lowest_prime[i] == 0) {
4             lowest_prime[i] = i;
5             prime.push_back(i);
6         }
7         for (ll j = 0; i * prime[j] <= N; j++) {
8             lowest_prime[i * prime[j]] = prime[j];
9             if (prime[j] == lowest_prime[i]) {
10                 break;
11             }
12         }
13     }

```

```
14 }
```

4.4 Miller Rabin

```
1 bool check_composite(u64 n, u64 a, u64 d, ll s) {
2     u64 x = binpower(a, d, n);
3     if (x == 1 || x == n - 1) {
4         return false;
5     }
6     for (ll r = 1; r < s; r++) {
7         x = (u128) x * x % n;
8         if (x == n - 1) {
9             return false;
10        }
11    }
12    return true;
13 }
14 bool miller_rabin(u64 n) {
15     if (n < 2) {
16         return false;
17     }
18     ll r = 0;
19     u64 d = n - 1;
20     while ((d & 1) == 0) {
21         d >>= 1;
22         r++;
23     }
24     for (ll a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
25                31, 37}) {
26         if (n == a) {
27             return true;
28         }
29         if (check_composite(n, a, d, r)) {
30             return false;
31         }
32     }
33     return true;
34 }
```

4.5 Modulo Inverse

```
1 ll mod_inv(ll a, ll m) {
2     if (m == 1) {
3         return 0;
4     }
5     ll m0 = m, x = 1, y = 0;
6     while (a > 1) {
7         ll q = a / m, t = m;
8         m = a % m;
9         a = t;
10        t = y;
11        y = x - q * y;
12        x = t;
13    }
14    if (x < 0) {
15        x += m0;
16    }
17    return x;
18 }
```

4.6 Pollard Rho Brent

```
1 ll mult(ll a, ll b, ll mod) {
2     return ((__int128_t) a * b % mod;
3 }
4 ll f(ll x, ll c, ll mod) {
5     return (mult(x, x, mod) + c) % mod;
6 }
7 ll pollard_rho_brent(ll n, ll x0 = 2, ll c = 1) {
8     ll x = x0, g = 1, q = 1, xs, y, m = 128, l = 1;
9     while (g == 1) {
10        y = x;
11        for (ll i = 1; i < l; i++) {
12            x = f(x, c, n);
13        }
14        ll k = 0;
15        while (k < l && g == 1) {
16            xs = x;
17            for (ll i = 0; i < m && i < l - k; i++) {
18                x = f(x, c, n);
19                q = mult(q, abs(y - x), n);
20            }
21            g = __gcd(q, n);
22            k += m;
23        }
24        l *= 2;
25    }
26    if (g == n) {
27        do {
28            xs = f(xs, c, n);
29            g = __gcd(abs(xs - y), n);
30        } while (g == 1);
31    }
32    return g;
33 }
```

4.7 Range Sieve

```
1 vector<bool> range_sieve(ll l, ll r) {
2     ll n = sqrt(r);
3     vector<bool> is_prime(n + 1, true);
4     vector<ll> prime;
5     is_prime[0] = is_prime[1] = false;
6     prime.push_back(2);
7     for (ll i = 4; i <= n; i += 2) {
8         is_prime[i] = false;
9     }
10    for (ll i = 3; i <= n; i += 2) {
11        if (is_prime[i]) {
12            prime.push_back(i);
13            for (ll j = i * i; j <= n; j += i) {
14                is_prime[j] = false;
15            }
16        }
17    }
18    vector<bool> result(r - l + 1, true);
19    for (ll i : prime) {
20        for (ll j = max(i * i, (l + i - 1) / i * i); j
21              <= r; j += i) {
22            result[j - l] = false;
23        }
24    }
25    if (l == 1) {
26        result[0] = false;
27    }
28    return result;
29 }
```

4.8 Segmented Sieve

```
1 vector<ll> segmented_sieve(ll n) {
2     const ll S = 10000;
3     ll nsqrt = sqrt(n);
4     vector<char> is_prime(nsqrt + 1, true);
5     vector<ll> prime;
6     is_prime[0] = is_prime[1] = false;
7     prime.push_back(2);
8     for (ll i = 4; i <= nsqrt; i += 2) {
9         is_prime[i] = false;
10    }
11    for (ll i = 3; i <= nsqrt; i += 2) {
12        if (is_prime[i]) {
13            prime.push_back(i);
14            for (ll j = i * i; j <= nsqrt; j += i) {
15                is_prime[j] = false;
16            }
17        }
18    }
19    vector<ll> result;
20    vector<char> block(S);
21    for (ll k = 0; k * S <= n; k++) {
22        fill(block.begin(), block.end(), true);
23        for (ll p : prime) {
24            for (ll j = max((k * S + p - 1) / p, p) * p -
25                      k * S; j < S; j += p) {
26                block[j] = false;
27            }
28        }
29        if (k == 0) {
30            block[0] = block[1] = false;
31        }
32        for (ll i = 0; i < S && k * S + i <= n; i++) {
33            if (block[i]) {
34                result.push_back(k * S + i);
35            }
36        }
37    }
38    return result;
39 }
```

4.9 Tonelli Shanks

```
1 ll legendre(ll a, ll p) {
2     return bin_pow_mod(a, (p - 1) / 2, p);
3 }
4 ll tonelli_shanks(ll n, ll p) {
5     if (legendre(n, p) == p - 1) {
6         return -1;
7     }
8     if (p % 4 == 3) {
9         return bin_pow_mod(n, (p + 1) / 4, p);
10    }
11    ll Q = p - 1, S = 0;
12    while (Q % 2 == 0) {
13        Q /= 2;
14        S++;
15    }
16    ll z = 2;
17    for (; z < p; z++) {
18        if (legendre(z, p) == p - 1) {
19            break;
20        }
21    }
```

```

22  ll M = S, c = bin_pow_mod(z, Q, p), t =
    bin_pow_mod(n, Q, p), R = bin_pow_mod(n, (Q
    + 1) / 2, p);
23  while (t % p != 1) {
24      if (t % p == 0) {
25          return 0;
26      }
27      ll i = 1, t2 = t * t % p;
28      for (; i < M; i++) {

```

```

29      if (t2 % p == 1) {
30          break;
31      }
32      t2 = t2 * t2 % p;
33  }
34  ll b = bin_pow_mod(c, bin_pow_mod(2, M - i - 1,
    p), p);
35  M = i;
36  c = b * b % p;

```

```

37      t = t * c % p;
38      R = R * b % p;
39  }
40  return R;
41  }

```

$f(n) = O(g(n))$	iff \exists positive c, n_0 such that $0 \leq f(n) \leq cg(n) \forall n \geq n_0$.	$\sum_{i=1}^n i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^n i^3 = \frac{n^2(n+1)^2}{4}.$
$f(n) = \Omega(g(n))$	iff \exists positive c, n_0 such that $f(n) \geq cg(n) \geq 0 \forall n \geq n_0$.	In general:
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.	$\sum_{i=1}^n i^m = \frac{1}{m+1} \left[(n+1)^{m+1} - 1 - \sum_{i=1}^n ((i+1)^{m+1} - i^{m+1} - (m+1)i^m) \right]$
$f(n) = o(g(n))$	iff $\lim_{n \rightarrow \infty} f(n)/g(n) = 0$.	$\sum_{i=1}^{n-1} i^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}.$
$\lim_{n \rightarrow \infty} a_n = a$	iff $\forall \epsilon > 0, \exists n_0$ such that $ a_n - a < \epsilon, \forall n \geq n_0$.	Geometric series:
$\sup S$	least $b \in \mathbb{R}$ such that $b \geq s, \forall s \in S$.	$\sum_{i=0}^n c^i = \frac{c^{n+1} - 1}{c - 1}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} c^i = \frac{1}{1 - c}, \quad \sum_{i=1}^{\infty} c^i = \frac{c}{1 - c}, \quad c < 1,$
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \leq s, \forall s \in S$.	$\sum_{i=0}^n i c^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \neq 1, \quad \sum_{i=0}^{\infty} i c^i = \frac{c}{(1-c)^2}, \quad c < 1.$
$\liminf_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \inf \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	Harmonic series:
$\limsup_{n \rightarrow \infty} a_n$	$\lim_{n \rightarrow \infty} \sup \{a_i \mid i \geq n, i \in \mathbb{N}\}.$	$H_n = \sum_{i=1}^n \frac{1}{i}, \quad \sum_{i=1}^n i H_i = \frac{n(n+1)}{2} H_n - \frac{n(n-1)}{4}.$
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.	$\sum_{i=1}^n H_i = (n+1)H_n - n, \quad \sum_{i=1}^n \binom{i}{m} H_i = \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right).$
$[n]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.	1. $\binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad 2. \sum_{k=0}^n \binom{n}{k} = 2^n, \quad 3. \binom{n}{k} = \binom{n}{n-k},$
$\left\{ \begin{matrix} n \\ k \end{matrix} \right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.	4. $\binom{n}{k} = \frac{n}{k} \binom{n-1}{k-1}, \quad 5. \binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1},$
$\langle \begin{matrix} n \\ k \end{matrix} \rangle$	1st order Eulerian numbers: Permutations $\pi_1 \pi_2 \dots \pi_n$ on $\{1, 2, \dots, n\}$ with k ascents.	6. $\binom{n}{m} \binom{m}{k} = \binom{n}{k} \binom{n-k}{m-k}, \quad 7. \sum_{k=0}^n \binom{r+k}{k} = \binom{r+n+1}{n},$
$\langle \langle \begin{matrix} n \\ k \end{matrix} \rangle \rangle$	2nd order Eulerian numbers.	8. $\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}, \quad 9. \sum_{k=0}^n \binom{r}{k} \binom{s}{n-k} = \binom{r+s}{n},$
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.	10. $\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad 11. \left\{ \begin{matrix} n \\ 1 \end{matrix} \right\} = \left\{ \begin{matrix} n \\ n \end{matrix} \right\} = 1,$
14. $\left[\begin{matrix} n \\ 1 \end{matrix} \right] = (n-1)!,$	15. $\left[\begin{matrix} n \\ 2 \end{matrix} \right] = (n-1)!H_{n-1},$	16. $\left[\begin{matrix} n \\ n \end{matrix} \right] = 1, \quad 17. \left[\begin{matrix} n \\ k \end{matrix} \right] \geq \left\{ \begin{matrix} n \\ k \end{matrix} \right\},$
18. $\left[\begin{matrix} n \\ k \end{matrix} \right] = (n-1) \left[\begin{matrix} n-1 \\ k \end{matrix} \right] + \left[\begin{matrix} n-1 \\ k-1 \end{matrix} \right],$	19. $\left\{ \begin{matrix} n \\ n-1 \end{matrix} \right\} = \left[\begin{matrix} n \\ n-1 \end{matrix} \right] = \binom{n}{2},$	20. $\sum_{k=0}^n \left[\begin{matrix} n \\ k \end{matrix} \right] = n!, \quad 21. C_n = \frac{1}{n+1} \binom{2n}{n},$
22. $\langle \begin{matrix} n \\ 0 \end{matrix} \rangle = \langle \begin{matrix} n \\ n-1 \end{matrix} \rangle = 1,$	23. $\langle \begin{matrix} n \\ k \end{matrix} \rangle = \langle \begin{matrix} n \\ n-1-k \end{matrix} \rangle,$	24. $\langle \begin{matrix} n \\ k \end{matrix} \rangle = (k+1) \langle \begin{matrix} n-1 \\ k \end{matrix} \rangle + (n-k) \langle \begin{matrix} n-1 \\ k-1 \end{matrix} \rangle,$
25. $\langle \begin{matrix} 0 \\ k \end{matrix} \rangle = \begin{cases} 1 & \text{if } k=0, \\ 0 & \text{otherwise} \end{cases}$	26. $\langle \begin{matrix} n \\ 1 \end{matrix} \rangle = 2^n - n - 1,$	27. $\langle \begin{matrix} n \\ 2 \end{matrix} \rangle = 3^n - (n+1)2^n + \binom{n+1}{2},$
28. $x^n = \sum_{k=0}^n \langle \begin{matrix} n \\ k \end{matrix} \rangle \binom{x+k}{n},$	29. $\langle \begin{matrix} n \\ m \end{matrix} \rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k,$	30. $m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_{k=0}^n \langle \begin{matrix} n \\ k \end{matrix} \rangle \binom{k}{n-m},$
31. $\langle \begin{matrix} n \\ m \end{matrix} \rangle = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} \binom{n-k}{m} (-1)^{n-k-m} k!,$	32. $\langle \langle \begin{matrix} n \\ 0 \end{matrix} \rangle \rangle = 1,$	33. $\langle \langle \begin{matrix} n \\ n \end{matrix} \rangle \rangle = 0 \quad \text{for } n \neq 0,$
34. $\langle \langle \begin{matrix} n \\ k \end{matrix} \rangle \rangle = (k+1) \langle \langle \begin{matrix} n-1 \\ k \end{matrix} \rangle \rangle + (2n-1-k) \langle \langle \begin{matrix} n-1 \\ k-1 \end{matrix} \rangle \rangle,$	35. $\sum_{k=0}^n \langle \langle \begin{matrix} n \\ k \end{matrix} \rangle \rangle = \frac{(2n)n}{2^n},$	
36. $\left\{ \begin{matrix} x \\ x-n \end{matrix} \right\} = \sum_{k=0}^n \langle \langle \begin{matrix} n \\ k \end{matrix} \rangle \rangle \binom{x+n-1-k}{2n},$	37. $\left\{ \begin{matrix} n+1 \\ m+1 \end{matrix} \right\} = \sum_k \binom{n}{k} \left\{ \begin{matrix} k \\ m \end{matrix} \right\} = \sum_{k=0}^n \left\{ \begin{matrix} k \\ m \end{matrix} \right\} (m+1)^{n-k},$	

The Chinese remainder theorem: There exists a number C such that:

$$C \equiv r_1 \pmod{m_1}$$

$$\vdots \quad \vdots \quad \vdots$$

$$C \equiv r_n \pmod{m_n}$$

if m_i and m_j are relatively prime for $i \neq j$.

Euler's function: $\phi(x)$ is the number of positive integers less than x relatively prime to x . If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$\phi(x) = \prod_{i=1}^n p_i^{e_i-1} (p_i - 1).$$

Euler's theorem: If a and b are relatively prime then

$$1 \equiv a^{\phi(b)} \pmod{b}.$$

Fermat's theorem:

$$1 \equiv a^{p-1} \pmod{p}.$$

The Euclidean algorithm: if $a > b$ are integers then

$$\gcd(a, b) = \gcd(a \bmod b, b).$$

If $\prod_{i=1}^n p_i^{e_i}$ is the prime factorization of x then

$$S(x) = \sum_{d|x} d = \prod_{i=1}^n \frac{p_i^{e_i+1} - 1}{p_i - 1}.$$

Perfect Numbers: x is an even perfect number iff $x = 2^{n-1}(2^n - 1)$ and $2^n - 1$ is prime.

Wilson's theorem: n is a prime iff

$$(n-1)! \equiv -1 \pmod{n}.$$

Möbius inversion:

$$\mu(i) = \begin{cases} 1 & \text{if } i = 1. \\ 0 & \text{if } i \text{ is not square-free.} \\ (-1)^r & \text{if } i \text{ is the product of } r \text{ distinct primes.} \end{cases}$$

If

$$G(a) = \sum_{d|a} F(d),$$

then

$$F(a) = \sum_{d|a} \mu(d) G\left(\frac{a}{d}\right).$$

Prime numbers:

$$p_n = n \ln n + n \ln \ln n - n + n \frac{\ln \ln n}{\ln n}$$

$$+ O\left(\frac{n}{\ln n}\right),$$

$$\pi(n) = \frac{n}{\ln n} + \frac{n}{(\ln n)^2} + \frac{2!n}{(\ln n)^3}$$

$$+ O\left(\frac{n}{(\ln n)^4}\right).$$

Definitions:

Loop An edge connecting a vertex to itself.

Directed Simple Each edge has a direction. Graph with no loops or multi-edges.

Walk A sequence $v_0 e_1 v_1 \dots e_\ell v_\ell$.

Trail A walk with distinct edges.

Path A trail with distinct vertices.

Connected A graph where there exists a path between any two vertices.

Component A maximal connected subgraph.

Tree A connected acyclic graph.

Free tree A tree with no root.

DAG Directed acyclic graph.

Eulerian Graph with a trail visiting each edge exactly once.

Hamiltonian Graph with a cycle visiting each vertex exactly once.

Cut A set of edges whose removal increases the number of components.

Cut-set A minimal cut.

Cut edge A size 1 cut.

k-Connected A graph connected with the removal of any $k-1$ vertices.

k-Tough $\forall S \subseteq V, S \neq \emptyset$ we have $k \cdot c(G-S) \leq |S|$.

k-Regular A graph where all vertices have degree k .

k-Factor A k -regular spanning subgraph.

Matching A set of edges, no two of which are adjacent.

Clique A set of vertices, all of which are adjacent.

Ind. set A set of vertices, none of which are adjacent.

Vertex cover A set of vertices which cover all edges.

Planar graph A graph which can be embedded in the plane.

Plane graph An embedding of a planar graph.

$$\sum_{v \in V} \deg(v) = 2m.$$

If G is planar then $n - m + f = 2$, so

$$f \leq 2n - 4, \quad m \leq 3n - 6.$$

Any planar graph has a vertex with degree ≤ 5 .

Notation:

$E(G)$ Edge set

$V(G)$ Vertex set

$c(G)$ Number of components

$G[S]$ Induced subgraph

$\deg(v)$ Degree of v

$\Delta(G)$ Maximum degree

$\delta(G)$ Minimum degree

$\chi(G)$ Chromatic number

$\chi_E(G)$ Edge chromatic number

G^c Complement graph

K_n Complete graph

K_{n_1, n_2} Complete bipartite graph

$r(k, \ell)$ Ramsey number

Geometry

Projective coordinates: triples (x, y, z) , not all x, y and z zero.

$$(x, y, z) = (cx, cy, cz) \quad \forall c \neq 0.$$

Cartesian Projective

$$(x, y) \quad (x, y, 1)$$

$$y = mx + b \quad (m, -1, b)$$

$$x = c \quad (1, 0, -c)$$

Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$

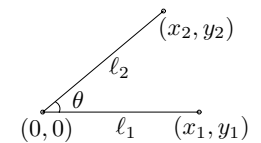
$$[|x_1 - x_0|^p + |y_1 - y_0|^p]^{1/p},$$

$$\lim_{p \rightarrow \infty} [|x_1 - x_0|^p + |y_1 - y_0|^p]^{1/p}.$$

Area of triangle $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2) :

$$\frac{1}{2} \text{abs} \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix}.$$

Angle formed by three points:



$$\cos \theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{l_1 l_2}.$$

Line through two points (x_0, y_0) and (x_1, y_1) :

$$\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0.$$

Area of circle, volume of sphere:

$$A = \pi r^2, \quad V = \frac{4}{3} \pi r^3.$$

If I have seen farther than others, it is because I have stood on the shoulders of giants.

– Issac Newton

Taylor's series:

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \dots = \sum_{i=0}^{\infty} \frac{(x-a)^i}{i!} f^{(i)}(a).$$

Expansions:

$$\begin{aligned} \frac{1}{1-x} &= 1 + x + x^2 + x^3 + x^4 + \dots = \sum_{i=0}^{\infty} x^i, \\ \frac{1}{1-cx} &= 1 + cx + c^2x^2 + c^3x^3 + \dots = \sum_{i=0}^{\infty} c^i x^i, \\ \frac{1}{1-x^n} &= 1 + x^n + x^{2n} + x^{3n} + \dots = \sum_{i=0}^{\infty} x^{ni}, \\ \frac{x}{(1-x)^2} &= x + 2x^2 + 3x^3 + 4x^4 + \dots = \sum_{i=0}^{\infty} ix^i, \\ x^k \frac{d^n}{dx^n} \left(\frac{1}{1-x} \right) &= x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \dots = \sum_{i=0}^{\infty} i^n x^i, \\ e^x &= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}, \\ \ln(1+x) &= x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \dots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i}, \\ \ln \frac{1}{1-x} &= x + \frac{1}{2}x^2 + \frac{1}{3}x^3 + \frac{1}{4}x^4 + \dots = \sum_{i=1}^{\infty} \frac{x^i}{i}, \\ \sin x &= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!}, \\ \cos x &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 - \frac{1}{6!}x^6 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}, \\ \tan^{-1} x &= x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \dots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)}, \\ (1+x)^n &= 1 + nx + \frac{n(n-1)}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{n}{i} x^i, \\ \frac{1}{(1-x)^{n+1}} &= 1 + (n+1)x + \binom{n+2}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i, \\ \frac{x}{e^x - 1} &= 1 - \frac{1}{2}x + \frac{1}{12}x^2 - \frac{1}{720}x^4 + \dots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!}, \\ \frac{1}{2x}(1 - \sqrt{1-4x}) &= 1 + x + 2x^2 + 5x^3 + \dots = \sum_{i=0}^{\infty} \frac{1}{i+1} \binom{2i}{i} x^i, \\ \frac{1}{\sqrt{1-4x}} &= 1 + x + 2x^2 + 6x^3 + \dots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i, \\ \frac{1}{\sqrt{1-4x}} \left(\frac{1 - \sqrt{1-4x}}{2x} \right)^n &= 1 + (2+n)x + \binom{4+n}{2}x^2 + \dots = \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i, \\ \frac{1}{1-x} \ln \frac{1}{1-x} &= x + \frac{3}{2}x^2 + \frac{11}{6}x^3 + \frac{25}{12}x^4 + \dots = \sum_{i=1}^{\infty} H_i x^i, \\ \frac{1}{2} \left(\ln \frac{1}{1-x} \right)^2 &= \frac{1}{2}x^2 + \frac{3}{4}x^3 + \frac{11}{24}x^4 + \dots = \sum_{i=2}^{\infty} \frac{H_{i-1} x^i}{i}, \\ \frac{x}{1-x-x^2} &= x + x^2 + 2x^3 + 3x^4 + \dots = \sum_{i=0}^{\infty} F_i x^i, \\ \frac{F_n x}{1 - (F_{n-1} + F_{n+1})x - (-1)^n x^2} &= F_n x + F_{2n} x^2 + F_{3n} x^3 + \dots = \sum_{i=0}^{\infty} F_{ni} x^i. \end{aligned}$$

Ordinary power series:

$$A(x) = \sum_{i=0}^{\infty} a_i x^i.$$

Exponential power series:

$$A(x) = \sum_{i=0}^{\infty} a_i \frac{x^i}{i!}.$$

Dirichlet power series:

$$A(x) = \sum_{i=1}^{\infty} \frac{a_i}{i^x}.$$

Binomial theorem:

$$(x+y)^n = \sum_{k=0}^n \binom{n}{k} x^{n-k} y^k.$$

Difference of like powers:

$$x^n - y^n = (x-y) \sum_{k=0}^{n-1} x^{n-1-k} y^k.$$

For ordinary power series:

$$\alpha A(x) + \beta B(x) = \sum_{i=0}^{\infty} (\alpha a_i + \beta b_i) x^i,$$

$$x^k A(x) = \sum_{i=k}^{\infty} a_{i-k} x^i,$$

$$\frac{A(x) - \sum_{i=0}^{k-1} a_i x^i}{x^k} = \sum_{i=0}^{\infty} a_{i+k} x^i,$$

$$A(cx) = \sum_{i=0}^{\infty} c^i a_i x^i,$$

$$A'(x) = \sum_{i=0}^{\infty} (i+1) a_{i+1} x^i,$$

$$xA'(x) = \sum_{i=1}^{\infty} i a_i x^i,$$

$$\int A(x) dx = \sum_{i=1}^{\infty} \frac{a_{i-1}}{i} x^i,$$

$$\frac{A(x) + A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i} x^{2i},$$

$$\frac{A(x) - A(-x)}{2} = \sum_{i=0}^{\infty} a_{2i+1} x^{2i+1}.$$

Summation: If $b_i = \sum_{j=0}^i a_j$ then

$$B(x) = \frac{1}{1-x} A(x).$$

Convolution:

$$A(x)B(x) = \sum_{i=0}^{\infty} \left(\sum_{j=0}^i a_j b_{i-j} \right) x^i.$$

God made the natural numbers;
all the rest is the work of man.
– Leopold Kronecker