UPLB Eliens - Pegaraw Notebook

Contents

1 Data Structures

1.1 Union Find

```
class UF {
private: vi p;
public:
    UF(int N) {p.assign(N, -1);}
    int fs(int i ) {return (p[i] < 0) ? i : (p[i] =
        fs(p[i]));}
    bool isSame(int i, int j) {return fs(i) == fs(j
        );}
    void join(int i, int j) {
        int x = fs(i), y = fs(j);
        if (!(x == y)){
            if (x < y) { p[x] += p[y]; p[y] = x; }
            else {p[y] += p[x]; p[x] = y; } } }
};
```

2 Dynamic Programming

2.1 Edit Distance

```
ll edit_distance(string x, string y, ll n, ll m) {
  vector<vector<int>> dp(n + 1, vector<int>(m + 1,
      INF));
  dp[0][0] = 0;
  for (int i = 1; i <= n; i++) {
    dp[i][0] = i;
  }
  for (int j = 1; j <= m; j++) {
    dp[0][j] = j;
  }
  for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= m; j++) {
      dp[i][j] = min({dp[i - 1][j] + 1, dp[i][j -
          1] + 1, dp[i - 1][j - 1] + (x[i - 1] !=
          y[j - 1])});
    }
  }
  return dp[n][m];
}
```

2.2 Knapsack

```
ll knapsack(ll W, vector<ll> &wt, vector<ll> &val,
    ll n) {
  vector<ll> dp(W + 1, 0);
  for (ll i = 1; i <= n; i++) {
    for (ll w = W; w >= 0; w--) {
      if (wt[i - 1] <= w) {
        dp[w] = max(dp[w], dp[w - wt[i - 1]] + val[
            i - 1]);
      }
    }
  }
  return dp[W];
}
```

2.3 Longest Common Subsequence

```
ll LCS(string x, string y, ll n, ll m) {
  vector<vector<ll>> dp(n + 1, vector<ll>(m + 1));
  for (ll i = 0; i <= n; i++) {
    for (ll j = 0; j <= m; j++) {
      if (i == 0 || j == 0) {
        dp[i][j] = 0;
      } else if (x[i - 1] == y[j - 1]) {
        dp[i][j] = dp[i - 1][j - 1] + 1;
      } else {
        dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
      }
    }
  }
  ll index = dp[n][m];
  vector<char> lcs(index + 1);
  lcs[index] = '\0';
  ll i = n, j = m;
  while (i > 0 && j > 0) {
    if (x[i - 1] == y[j - 1]) {
      lcs[index - 1] = x[i - 1];
      i--;
      j--;
      index--;
    } else if (dp[i - 1][j] > dp[i][j - 1]) {
      i--;
    } else {
      j--;
    }
  }
  return dp[n][m];
}
```

2.4 Longest Increasing Subsequence

```
ll get_ceil_idx(vector<ll> &a, vector<ll> &T, ll l,
    ll r, ll x) {
  while (r - l > 1) {
    ll m = l + (r - l) / 2;
    if (a[T[m]] >= x) {
      r = m;
    } else {
      l = m;
    }
  }
  return r;
}
ll LIS(ll n, vector<ll> &a) {
  ll len = 1;
  vector<ll> T(n, 0), R(n, - 1);
  T[0] = 0;
  for (ll i = 1; i < n; i++) {
    if (a[i] < a[T[0]]) {
      T[0] = i;
    } else if (a[i] > a[T[len - 1]]) {
      R[i] = T[len - 1];
      T[len++] = i;
    } else {
      ll pos = get_ceil_idx(a, T, -1, len - 1, a[i
          ]);
      R[i] = T[pos - 1];
      T[pos] = i;
    }
  }
  return len;
}
```

2.5 Subset Sum

```
bool subset_sum(ll n, vector<ll> &arr, ll sum) {
  vector<vector<ll>> dp(n + 1, vector<ll>(sum + 1,
      false));
  dp[0][0] = true;
  for (ll i = 1; i <= n; i++) {
    for (ll j = 0; j <= sum; j++) {
      dp[i][j] = dp[i - 1][j];
      if (j >= arr[i]) {
        dp[i][j] |= dp[i - 1][j - arr[i]];
      }
    }
  }
  return dp[n][sum];
}
```

3 Graph Theory

3.1 Articulation Point

```
void APUtil(vector<vector<ll>> &adj, ll u, vector<
    bool> &visited,
vector<ll> &disc, vector<ll> &low, ll &time, ll
    parent, vector<bool> &isAP) {
  ll children = 0;
  visited[u] = true;
  disc[u] = low[u] = ++time;
  for (auto v : adj[u]) {
    if (!visited[v]) {
      children++;
      APUtil(adj, v, visited, disc, low, time, u,
          isAP);
      low[u] = min(low[u], low[v]);
      if (parent != -1 && low[v] >= disc[u]) {
        isAP[u] = true;
      }
    } else if (v != parent) {
      low[u] = min(low[u], disc[v]);
    }
  }
  if (parent == -1 && children > 1) {
    isAP[u] = true;
  }
}
void AP(vector<vector<ll>> &adj, ll n) {
  vector<ll> disc(n), low(n);
  vector<bool> visited(n), isAP(n);
  ll time = 0, par = -1;
  for (ll u = 0; u < n; u++) {
    if (!visited[u]) {
      APUtil(adj, u, visited, disc, low, time, par,
          isAP);
    }
  }
  for (ll u = 0; u < n; u++) {
    if (isAP[u]) {
      cout << u << " ";
    }
  }
}
```

3.2 Bellman Ford

```
void bellman_ford(vector<vector<ll>> &edges, ll n,
    ll m, ll src, vector<ll> &dis) {
  for (ll i = 0; i < n; i++) {
    dis[i] = INF;
  }
  for (ll i = 0; i < n - 1; i++) {
    for (ll j = 0; j < m; j++) {
      ll u = edges[j][0], v = edges[j][1], w =
          edges[j][2];
      if (dis[u] < INF) {
        dis[v] = min(dis[v], dis[u] + w);
      }
    }
  }
  for (ll i = 0; i < m; i++) {
    ll u = edges[i][0], v = edges[i][1], w = edges[
        i][2];
    if (dis[u] < INF && dis[u] + w < dis[v]) {
      cout << "The graph contains a negative cycle.
          " << '\n';
    }
  }
}
```

3.3 Bridge

```
void bridge_util(vector<vector<ll>> &adj, ll u,
    vector<bool> &visited, vector<ll> &disc,
    vector<ll> &low, vector<ll> &parent) {
  static ll time = 0;
  visited[u] = true;
  disc[u] = low[u] = ++time;
  list<ll>::iterator i;
  for (auto v : adj[u]) {
    if (!visited[v]) {
      parent[v] = u;
      bridge_util(adj, v, visited, disc, low,
          parent);
      low[u] = min(low[u], low[v]);
      if (low[v] > disc[u]) {
        cout << u << ' ' << v << '\n';
      }
    } else if (v != parent[u]) {
      low[u] = min(low[u], disc[v]);
    }
  }
}

void bridge(vector<vector<ll>> &adj, ll n) {
  vector<bool> visited(n, false);
  vector<ll> disc(n), low(n), parent(n, -1);
  for (ll i = 0; i < n; i++) {
    if (!visited[i]) {
      bridge_util(adj, i, visited, disc, low,
          parent);
    }
  }
}
```

3.4 Dijkstra

```
void dijkstra(ll n, vector<vector<pair<ll, ll>>> &
    adj, vector<ll> &dis) {
  priority_queue<pair<ll, ll>, vector<pair<ll, ll
      >>, greater<pair<ll, ll>>> pq;
  for (int i = 0; i < n; i++) {
    dis[i] = INF;
  }
  dis[0] = 0;
  pq.push({0, 0});
  while (!pq.empty()) {
    auto p = pq.top();
    pq.pop();
    ll u = p.second;
    if (dis[u] != p.first) {
      continue;
    }
    for (auto x : adj[u]) {
      ll v = x.first, w = x.second;
      if (dis[v] > dis[u] + w) {
        dis[v] = dis[u] + w;
        pq.push({dis[v], v});
      }
    }
  }
}
```

3.5 Floyd Warshall

```
void floyd_warshall(vector<vector<ll>> &dis, ll n)
    {
  for (ll i = 0; i < n; i++) {
    for (ll j = 0; j < n; j++) {
      dis[i][j] = (i == j ? 0 : INF);
    }
  }
  for (ll k = 0; k < n; k++) {
    for (ll i = 0; i < n; i++) {
      for (ll j = 0; j < n; j++) {
        if (dis[i][k] < INF && dis[k][j] < INF) {
          dis[i][j] = min(dis[i][j], dis[i][k] +
              dis[k][j]);
        }
      }
    }
  }
  for (ll i = 0; i < n; i++) {
    for (ll j = 0; j < n; j++) {
      for (ll k = 0; k < n; k++) {
        if (dis[k][k] < 0 && dis[i][k] < INF && dis
            [k][j] < INF) {
          dis[i][j] = -INF;
        }
      }
    }
  }
}
```

3.6 Hierholzer

```
void print_circuit(vector<vector<ll>> &adj) {
  map<ll, ll> edge_count;
  for (ll i = 0; i< adj.size(); i++) {
    edge_count[i] = adj[i].size();
  }
  if (!adj.size()) {
    return;
  }
  stack<ll> curr_path;
  vector<ll> circuit;
  curr_path.push(0);
  ll curr_v = 0;
  while (!curr_path.empty()) {
    if (edge_count[curr_v]) {
      curr_path.push(curr_v);
      ll next_v = adj[curr_v].back();
      edge_count[curr_v]--;
      adj[curr_v].pop_back();
      curr_v = next_v;
    } else {
      circuit.push_back(curr_v);
      curr_v = curr_path.top();
      curr_path.pop();
    }
  }
  for (ll i = circuit.size() - 1; i >= 0; i--) {
    cout << circuit[i] << ' ';
```

```
  }
}
```

3.7 Is Bipartite

```
bool is_bipartite(vector<ll> &col, vector<vector<ll
    >> &adj, ll n) {
  queue<pair<ll, ll>> q;
  for (ll i = 0; i < n; i++) {
    if (col[i] == -1) {
      q.push({i, 0});
      col[i] = 0;
      while (!q.empty()) {
        pair<ll, ll> p = q.front();
        q.pop();
        ll v = p.first, c = p.second;
        for (ll j : adj[v]) {
          if (col[j] == c) {
            return false;
          }
          if (col[j] == -1) {
            col[j] = (c ? 0 : 1);
            q.push({j, col[j]});
          }
        }
      }
    }
  }
  return true;
}
```

3.8 Is Cyclic

```
bool is_cyclic_util(int u, vector<vector<int>> &adj
    , vector<bool> &vis, vector<bool> &rec) {
  vis[u] = true;
  rec[u] = true;
  for(auto v : adj[u]) {
    if (!vis[v] && is_cyclic_util(v, adj, vis, rec)
        ) {
      return true;
    } else if (rec[v]) {
      return true;
    }
  }
  rec[u] = false;
  return false;
}
bool is_cyclic(int n, vector<vector<int>> &adj) {
  vector<bool> vis(n, false), rec(n, false);
  for (int i = 0; i < n; i++) {
    if (!vis[i] && is_cyclic_util(i, adj, vis, rec)
        ) {
      return true;
    }
  }
  return false;
}
```

3.9 Kahn

```
void kahn(vector<vector<ll>> &adj) {
  ll n = adj.size();
  vector<ll> in_degree(n, 0);
  for (ll u = 0; u < n; u++) {
    for (ll v: adj[u]) {
      in_degree[v]++;
    }
  }
  queue<ll> q;
  for (ll i = 0; i < n; i++) {
    if (in_degree[i] == 0) {
      q.push(i);
    }
  }
  ll cnt = 0;
  vector<ll> top_order;
  while (!q.empty()) {
    ll u = q.front();
    q.pop();
    top_order.push_back(u);
    for (ll v : adj[u]) {
      if (--in_degree[v] == 0) {
        q.push(v);
      }
    }
    cnt++;
  }
  if (cnt != n) {
    cout << -1 << '\n';
    return;
  }
  for (ll i = 0; i < (ll) top_order.size(); i++) {
    cout << top_order[i] << ' ';
  }
  cout << '\n';
}
```

3.10 Maximum Bipartite Matching

```
bool bpm(ll n, ll m, vector<vector<bool>> &bpGraph,
    ll u, vector<bool> &seen, vector<ll> &matchR)
    {
  for (ll v = 0; v < m; v++) {
    if (bpGraph[u][v] && !seen[v]) {
      seen[v] = true;
      if (matchR[v] < 0 || bpm(n, m, bpGraph,
          matchR[v], seen, matchR)) {
        matchR[v] = u;
        return true;
      }
    }
  }
  return false;
}
ll maxBPM(ll n, ll m, vector<vector<bool>> &bpGraph
    ) {
  vector<ll> matchR(m, -1);
  ll result = 0;
  for (ll u = 0; u < n; u++) {
    vector<bool> seen(m, false);
    if (bpm(n, m, bpGraph, u, seen, matchR)) {
      result++;
    }
  }
  return result;
}
```

3.11 Max Flow

```
bool bfs(ll n, vector<vector<ll>> &r_graph, ll s,
    ll t, vector<ll> &parent) {
  vector<bool> visited(n, false);
  queue<ll> q;
  q.push(s);
  visited[s] = true;
  parent[s] = -1;
  while (!q.empty()) {
    ll u = q.front();
    q.pop();
    for (ll v = 0; v < n; v++) {
      if (!visited[v] && r_graph[u][v] > 0) {
        if (v == t) {
          parent[v] = u;
          return true;
        }
        q.push(v);
        parent[v] = u;
        visited[v] = true;
      }
    }
  }
  return false;
}
ll fordFulkerson(ll n, vector<vector<ll>> graph, ll
    s, ll t) {
  ll u, v;
  vector<vector<ll>> r_graph;
  for (u = 0; u < n; u++) {
    for (v = 0; v < n; v++) {
      r_graph[u][v] = graph[u][v];
    }
  }
  vector<ll> parent;
  ll max_flow = 0;
  while (bfs(n, r_graph, s, t, parent)) {
    ll path_flow = INF;
    for (v = t; v != s; v = parent[v]) {
      u = parent[v];
      path_flow = min(path_flow, r_graph[u][v]);
    }
    for (v = t; v != s; v = parent[v]) {
      u = parent[v];
      r_graph[u][v] -= path_flow;
      r_graph[v][u] += path_flow;
    }
    max_flow += path_flow;
  }
  return max_flow;
}
```

3.12 Prim Mst

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

```
    ll u = pq.top().second;
    pq.pop();
    if (in_mst[u]){
      continue;
    }
    in_mst[u] = true;
    for (auto p : adj[u]) {
      ll v = p.first, w = p.second;
      if (in_mst[v] == false && w < key[v]) {
        key[v] = w;
        pq.push(make_pair(key[v], v));
        parent[v] = u;
      }
    }
  }
  return parent;
}
```

3.13 Strongly Connected Component

```
void dfs(ll u, vector<vector<ll>> &adj, vector<bool
    > &visited) {
  visited[u] = true;
  cout << u + 1 << ' ';
  for (ll v : adj[u]) {
    if (!visited[v]) {
      dfs(v, adj, visited);
    }
  }
}
vector<vector<ll>> get_transpose(ll n, vector<
    vector<ll>> &adj) {
  vector<vector<ll>> res(n);
  for (ll u = 0; u < n; u++) {
    for (ll v : adj[u]) {
      res[v].push_back(u);
    }
  }
  return res;
}
void fill_order(ll u, vector<vector<ll>> &adj,
    vector<bool> &visited, stack<ll> &stk) {
  visited[u] = true;
  for(auto v : adj[u]) {
    if(!visited[v]) {
      fill_order(v, adj, visited, stk);
    }
  }
  stk.push(u);
}
void get_scc(ll n, vector<vector<ll>> &adj) {
  stack<ll> stk;
  vector<bool> visited(n, false);
  for (ll i = 0; i < n; i++) {
    if (!visited[i]) {
      fill_order(i, adj, visited, stk);
    }
  }
  vector<vector<ll>> transpose = get_transpose(n,
     adj);
  for (ll i = 0; i < n; i++) {
    visited[i] = false;
  }
  while (!stk.empty()) {
    ll u = stk.top();
    stk.pop();
    if (!visited[u]) {
      dfs(u, transpose, visited);
      cout << '\n';
    }
  }
}
```

3.14 Topological Sort

```
void dfs(ll v) {
  visited[v] = true;
  for (ll u : adj[v]) {
    if (!visited[u]) {
      dfs(u);
    }
  }
  ans.push_back(v);
}
void topological_sort() {
  visited.assign(n, false);
  ans.clear();
  for (ll i = 0; i < n; ++i) {
    if (!visited[i]) {
      dfs(i);
    }
  }
  reverse(ans.begin(), ans.end());
}
```

4 Number Theory

4.1 Extended Euclidean

```
ll gcd_extended(ll a, ll b, ll &x, ll &y) {
  if (b == 0) {
    x = 1;
    y = 0;
    return a;
  }
  ll x1, y1, g = gcd_extended(b, a % b, x1, y1);
  x = y1;
  y = x1 - (a / b) * y1;
  return g;
}
```

4.2 Find All Solutions

```
bool find_any_solution(ll a, ll b, ll c, ll &x0, ll
     &y0, ll &g) {
  g = gcd_extended(abs(a), abs(b), x0, y0);
  if (c % g) {
    return false;
  }
  x0 *= c / g;
  y0 *= c / g;
  if (a < 0) {
    x0 = -x0;
  }
  if (b < 0) {
    y0 = -y0;
  }
  return true;
}
void shift_solution(ll & x, ll & y, ll a, ll b, ll
    cnt) {
  x += cnt * b;
  y -= cnt * a;
}
ll find_all_solutions(ll a, ll b, ll c, ll minx, ll
     maxx, ll miny, ll maxy) {
  ll x, y, g;
  if (!find_any_solution(a, b, c, x, y, g)) {
    return 0;
  }
  a /= g;
  b /= g;
  ll sign_a = a > 0 ? +1 : -1;
  ll sign_b = b > 0 ? +1 : -1;
  shift_solution(x, y, a, b, (minx - x) / b);
  if (x < minx) {
    shift_solution(x, y, a, b, sign_b);
  }
  if (x > maxx) {
    return 0;
  }
  ll lx1 = x;
  shift_solution(x, y, a, b, (maxx - x) / b);
  if (x > maxx) {
    shift_solution(x, y, a, b, -sign_b);
  }
  ll rx1 = x;
  shift_solution(x, y, a, b, -(miny - y) / a);
  if (y < miny) {
    shift_solution(x, y, a, b, -sign_a);
  }
  if (y > maxy) {
    return 0;
  }
  ll lx2 = x;
  shift_solution(x, y, a, b, -(maxy - y) / a);
  if (y > maxy) {
    shift_solution(x, y, a, b, sign_a);
  }
  ll rx2 = x;
  if (lx2 > rx2) {
    swap(lx2, rx2);
  }
  ll lx = max(lx1, lx2), rx = min(rx1, rx2);
  if (lx > rx) {
    return 0;
  }
  return (rx - lx) / abs(b) + 1;
}
```

4.3 Linear Sieve

```
void linear_sieve(ll N, vector<ll> &lowest_prime,
    vector<ll> &prime) {
  for (ll i = 2; i <= N; i++) {
    if (lowest_prime[i] == 0) {
      lowest_prime[i] = i;
      prime.push_back(i);
    }
    for (ll j = 0; i * prime[j] <= N; j++) {
      lowest_prime[i * prime[j]] = prime[j];
      if (prime[j] == lowest_prime[i]) {
        break;
      }
    }
  }
```

```
}
```

4.4 Miller Rabin

```
bool check_composite(u64 n, u64 a, u64 d, ll s) {
  u64 x = binpower(a, d, n);
  if (x == 1 || x == n - 1) {
    return false;
  }
  for (ll r = 1; r < s; r++) {
    x = (u128) x * x % n;
    if (x == n - 1) {
      return false;
    }
  }
  return true;
}
bool miller_rabin(u64 n) {
  if (n < 2) {
    return false;
  }
  ll r = 0;
  u64 d = n - 1;
  while ((d & 1) == 0) {
    d >>= 1;
    r++;
  }
  for (ll a : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
      31, 37}) {
    if (n == a) {
      return true;
    }
    if (check_composite(n, a, d, r)) {
      return false;
    }
  }
  return true;
}
```

4.5 Modulo Inverse

```
ll mod_inv(ll a, ll m) {
  if (m == 1) {
    return 0;
  }
  ll m0 = m, x = 1, y = 0;
  while (a > 1) {
    ll q = a / m, t = m;
    m = a % m;
    a = t;
    t = y;
    y = x - q * y;
    x = t;
  }
  if (x < 0) {
    x += m0;
  }
  return x;
}
```

4.6 Pollard Rho Brent

```
ll mult(ll a, ll b, ll mod) {
  return (__int128_t) a * b % mod;
}
ll f(ll x, ll c, ll mod) {
  return (mult(x, x, mod) + c) % mod;
}
ll pollard_rho_brent(ll n, ll x0 = 2, ll c = 1) {
  ll x = x0, g = 1, q = 1, xs, y, m = 128, l = 1;
  while (g == 1) {
    y = x;
    for (ll i = 1; i < l; i++) {
      x = f(x, c, n);
    }
    ll k = 0;
    while (k < l && g == 1) {
      xs = x;
      for (ll i = 0; i < m && i < l - k; i++) {
        x = f(x, c, n);
        q = mult(q, abs(y - x), n);
      }
      g = __gcd(q, n);
      k += m;
    }
    l *= 2;
  }
  if (g == n) {
    do {
      xs = f(xs, c, n);
      g = __gcd(abs(xs - y), n);
    } while (g == 1);
  }
  return g;
}
```

4.7 Range Sieve

```
vector<bool> range_sieve(ll l, ll r) {
  ll n = sqrt(r);
  vector<bool> is_prime(n + 1, true);
  vector<ll> prime;
  is_prime[0] = is_prime[1] = false;
  prime.push_back(2);
  for (ll i = 4; i <= n; i += 2) {
    is_prime[i] = false;
  }
  for (ll i = 3; i <= n; i += 2) {
    if (is_prime[i]) {
      prime.push_back(i);
      for (ll j = i * i; j <= n; j += i) {
        is_prime[j] = false;
      }
    }
  }
  vector<bool> result(r - l + 1, true);
  for (ll i : prime) {
    for (ll j = max(i * i, (l + i - 1) / i * i); j
        <= r; j += i) {
      result[j - l] = false;
    }
  }
  if (l == 1) {
    result[0] = false;
  }
  return result;
}
```

4.8 Segmented Sieve

```
vector<ll> segmented_sieve(ll n) {
  const ll S = 10000;
  ll nsqrt = sqrt(n);
  vector<char> is_prime(nsqrt + 1, true);
  vector<ll> prime;
  is_prime[0] = is_prime[1] = false;
  prime.push_back(2);
  for (ll i = 4; i <= nsqrt; i += 2) {
    is_prime[i] = false;
  }
  for (ll i = 3; i <= nsqrt; i += 2) {
    if (is_prime[i]) {
      prime.push_back(i);
      for (ll j = i * i; j <= nsqrt; j += i) {
        is_prime[j] = false;
      }
    }
  }
  vector<ll> result;
  vector<char> block(S);
  for (ll k = 0; k * S <= n; k++) {
    fill(block.begin(), block.end(), true);
    for (ll p : prime) {
      for (ll j = max((k * S + p - 1) / p, p) * p -
          k * S; j < S; j += p) {
        block[j] = false;
      }
    }
    if (k == 0) {
      block[0] = block[1] = false;
    }
    for (ll i = 0; i < S && k * S + i <= n; i++) {
      if (block[i]) {
        result.push_back(k * S + i);
      }
    }
  }
  return result;
}
```

4.9 Tonelli Shanks

```
ll legendre(ll a, ll p) {
  return bin_pow_mod(a, (p - 1) / 2, p);
}
ll tonelli_shanks(ll n, ll p) {
  if (legendre(n, p) == p - 1) {
    return -1;
  }
  if (p % 4 == 3) {
    return bin_pow_mod(n, (p + 1) / 4, p);
  }
  ll Q = p - 1, S = 0;
  while (Q % 2 == 0) {
    Q /= 2;
    S++;
  }
  ll z = 2;
  for (; z < p; z++) {
    if (legendre(z, p) == p - 1) {
      break;
    }
  }
```

```
  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);
  while (t % p != 1) {
    if (t % p == 0) {
      return 0;
    }
    ll i = 1, t2 = t * t % p;
    for (; i < M; i++) {
      if (t2 % p == 1) {
        break;
      }
      t2 = t2 * t2 % p;
    }
    ll b = bin_pow_mod(c, bin_pow_mod(2, M - i - 1,
         p), p);
    M = i;
    c = b * b % p;
    t = t * c % p;
    R = R * b % p;
  }
  return R;
}
```

$f(n) = O(g(n))$	iff $\exists$ positive c, n_0 such that $0 \le f(n) \le cg(n)\ \forall n \ge n_0$.
$f(n) = \Omega(g(n))$	iff $\exists$ positive c, n_0 such that $f(n) \ge cg(n) \ge 0\ \forall n \ge n_0$.
$f(n) = \Theta(g(n))$	iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
$f(n) = o(g(n))$	iff $\lim_{n\to\infty} f(n)/g(n) = 0$.
$\lim\limits_{n\to\infty} a_n = a$	iff $\forall \epsilon > 0$, $\exists n_0$ such that $\lvert a_n - a\rvert < \epsilon$, $\forall n \ge n_0$.
$\sup S$	least $b \in \mathbb{R}$ such that $b \ge s$, $\forall s \in S$.
$\inf S$	greatest $b \in \mathbb{R}$ such that $b \le s$, $\forall s \in S$.
$\liminf\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \inf\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\limsup\limits_{n\to\infty} a_n$	$\lim\limits_{n\to\infty} \sup\{a_i \mid i \ge n, i \in \mathbb{N}\}$.
$\binom{n}{k}$	Combinations: Size k subsets of a size n set.
$\left[{n \atop k}\right]$	Stirling numbers (1st kind): Arrangements of an n element set into k cycles.
$\left\{{n \atop k}\right\}$	Stirling numbers (2nd kind): Partitions of an n element set into k non-empty sets.
$\left\langle{n \atop k}\right\rangle$	1st order Eulerian numbers: Permutations $\pi_1\pi_2\ldots\pi_n$ on $\{1, 2, \ldots, n\}$ with k ascents.
$\left\langle\!\!\left\langle{n \atop k}\right\rangle\!\!\right\rangle$	2nd order Eulerian numbers.
C_n	Catalan Numbers: Binary trees with $n+1$ vertices.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \qquad \sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}.$$

In general:

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

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

Geometric series:

$$\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1}, \quad c \ne 1, \qquad \sum_{i=0}^{\infty} c^i = \frac{1}{1-c}, \qquad \sum_{i=1}^{\infty} c^i = \frac{c}{1-c}, \quad |c| < 1,$$

$$\sum_{i=0}^{n} ic^i = \frac{nc^{n+2} - (n+1)c^{n+1} + c}{(c-1)^2}, \quad c \ne 1, \qquad \sum_{i=0}^{\infty} ic^i = \frac{c}{(1-c)^2}, \quad |c| < 1.$$

Harmonic series:

$$H_n = \sum_{i=1}^{n}\frac{1}{i}, \qquad \sum_{i=1}^{n} iH_i = \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4}.$$

$$\sum_{i=1}^{n} H_i = (n+1)H_n - n, \qquad \sum_{i=1}^{n}\binom{i}{m}H_i = \binom{n+1}{m+1}\left(H_{n+1} - \frac{1}{m+1}\right).$$

1. $\displaystyle\binom{n}{k} = \frac{n!}{(n-k)!k!}$, **2.** $\displaystyle\sum_{k=0}^{n}\binom{n}{k} = 2^n$, **3.** $\displaystyle\binom{n}{k} = \binom{n}{n-k}$,

4. $\displaystyle\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$, **5.** $\displaystyle\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$,

6. $\displaystyle\binom{n}{m}\binom{m}{k} = \binom{n}{k}\binom{n-k}{m-k}$, **7.** $\displaystyle\sum_{k=0}^{n}\binom{r+k}{k} = \binom{r+n+1}{n}$,

8. $\displaystyle\sum_{k=0}^{n}\binom{k}{m} = \binom{n+1}{m+1}$, **9.** $\displaystyle\sum_{k=0}^{n}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$,

10. $\displaystyle\binom{n}{k} = (-1)^k\binom{k-n-1}{k}$, **11.** $\displaystyle\left\{{n \atop 1}\right\} = \left\{{n \atop n}\right\} = 1$,

12. $\displaystyle\left\{{n \atop 2}\right\} = 2^{n-1} - 1$, **13.** $\displaystyle\left\{{n \atop k}\right\} = k\left\{{n-1 \atop k}\right\} + \left\{{n-1 \atop k-1}\right\}$,

14. $\displaystyle\left[{n \atop 1}\right] = (n-1)!$, **15.** $\displaystyle\left[{n \atop 2}\right] = (n-1)!H_{n-1}$, **16.** $\displaystyle\left[{n \atop n}\right] = 1$, **17.** $\displaystyle\left[{n \atop k}\right] \ge \left\{{n \atop k}\right\}$,

18. $\displaystyle\left[{n \atop k}\right] = (n-1)\left[{n-1 \atop k}\right] + \left[{n-1 \atop k-1}\right]$, **19.** $\displaystyle\left\{{n \atop n-1}\right\} = \left[{n \atop n-1}\right] = \binom{n}{2}$, **20.** $\displaystyle\sum_{k=0}^{n}\left[{n \atop k}\right] = n!$, **21.** $\displaystyle C_n = \frac{1}{n+1}\binom{2n}{n}$,

22. $\displaystyle\left\langle{n \atop 0}\right\rangle = \left\langle{n \atop n-1}\right\rangle = 1$, **23.** $\displaystyle\left\langle{n \atop k}\right\rangle = \left\langle{n \atop n-1-k}\right\rangle$, **24.** $\displaystyle\left\langle{n \atop k}\right\rangle = (k+1)\left\langle{n-1 \atop k}\right\rangle + (n-k)\left\langle{n-1 \atop k-1}\right\rangle$,

25. $\displaystyle\left\langle{0 \atop k}\right\rangle = \begin{cases} 1 & \text{if } k = 0, \\ 0 & \text{otherwise} \end{cases}$ **26.** $\displaystyle\left\langle{n \atop 1}\right\rangle = 2^n - n - 1$, **27.** $\displaystyle\left\langle{n \atop 2}\right\rangle = 3^n - (n+1)2^n + \binom{n+1}{2}$,

28. $\displaystyle x^n = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{x+k}{n}$, **29.** $\displaystyle\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{m}\binom{n+1}{k}(m+1-k)^n(-1)^k$, **30.** $\displaystyle m!\left\{{n \atop m}\right\} = \sum_{k=0}^{n}\left\langle{n \atop k}\right\rangle\binom{k}{n-m}$,

31. $\displaystyle\left\langle{n \atop m}\right\rangle = \sum_{k=0}^{n}\left\{{n \atop k}\right\}\binom{n-k}{m}(-1)^{n-k-m}k!$, **32.** $\displaystyle\left\langle\!\!\left\langle{n \atop 0}\right\rangle\!\!\right\rangle = 1$, **33.** $\displaystyle\left\langle\!\!\left\langle{n \atop n}\right\rangle\!\!\right\rangle = 0 \quad \text{for } n \ne 0$,

34. $\displaystyle\left\langle\!\!\left\langle{n \atop k}\right\rangle\!\!\right\rangle = (k+1)\left\langle\!\!\left\langle{n-1 \atop k}\right\rangle\!\!\right\rangle + (2n-1-k)\left\langle\!\!\left\langle{n-1 \atop k-1}\right\rangle\!\!\right\rangle$, **35.** $\displaystyle\sum_{k=0}^{n}\left\langle\!\!\left\langle{n \atop k}\right\rangle\!\!\right\rangle = \frac{(2n)^{\underline{n}}}{2^n}$,

36. $\displaystyle\left\{{x \atop x-n}\right\} = \sum_{k=0}^{n}\left\langle\!\!\left\langle{n \atop k}\right\rangle\!\!\right\rangle\binom{x+n-1-k}{2n}$, **37.** $\displaystyle\left\{{n+1 \atop m+1}\right\} = \sum_{k}\binom{n}{k}\left\{{k \atop m}\right\} = \sum_{k=0}^{n}\left\{{k \atop m}\right\}(m+1)^{n-k}$, The Chinese remainder theorem: There exists a number C such that:

$$C \equiv r_1 \bmod m_1$$
$$\vdots \quad \vdots \quad \vdots$$
$$C \equiv r_n \bmod m_n$$

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

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

Fermat's theorem:

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

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

$$(n-1)! \equiv -1 \bmod n.$$

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	Each edge has a direction.
Simple	Graph with no loops or multi-edges.
Walk	A sequence $v_0 e_1 v_1 \ldots e_\ell v_\ell$.
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 $k \cdot c(G - S) \leq \lvert S \rvert$.
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 embeded 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
$\mathrm{r}(k, \ell)$	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)	$(x, y, 1)$
$y = mx + b$	$(m, -1, b)$
$x = c$	$(1, 0, -c)$

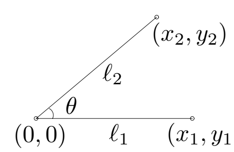
Distance formula, L_p and L_∞ metric:

$$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2},$$
$$\left[|x_1 - x_0|^p + |y_1 - y_0|^p \right]^{1/p},$$
$$\lim_{p \to \infty} \left[|x_1 - x_0|^p + |y_1 - y_0|^p \right]^{1/p}.$$

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

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

Angle formed by three points:



$$\cos\theta = \frac{(x_1, y_1) \cdot (x_2, y_2)}{\ell_1 \ell_2}.$$

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:

$$A = \pi r^2, \qquad V = \tfrac{4}{3} \pi r^3.$$

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

Taylor's series:

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

Expansions:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + x^4 + \cdots = \sum_{i=0}^{\infty} x^i,$$

$$\frac{1}{1-cx} = 1 + cx + c^2x^2 + c^3x^3 + \cdots = \sum_{i=0}^{\infty} c^i x^i,$$

$$\frac{1}{1-x^n} = 1 + x^n + x^{2n} + x^{3n} + \cdots = \sum_{i=0}^{\infty} x^{ni},$$

$$\frac{x}{(1-x)^2} = x + 2x^2 + 3x^3 + 4x^4 + \cdots = \sum_{i=0}^{\infty} i x^i,$$

$$x^k \frac{d^n}{dx^n}\left(\frac{1}{1-x}\right) = x + 2^n x^2 + 3^n x^3 + 4^n x^4 + \cdots = \sum_{i=0}^{\infty} i^n x^i,$$

$$e^x = 1 + x + \tfrac{1}{2}x^2 + \tfrac{1}{6}x^3 + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!},$$

$$\ln(1+x) = x - \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 - \tfrac{1}{4}x^4 - \cdots = \sum_{i=1}^{\infty} (-1)^{i+1} \frac{x^i}{i},$$

$$\ln \frac{1}{1-x} = x + \tfrac{1}{2}x^2 + \tfrac{1}{3}x^3 + \tfrac{1}{4}x^4 + \cdots = \sum_{i=1}^{\infty} \frac{x^i}{i},$$

$$\sin x = x - \tfrac{1}{3!}x^3 + \tfrac{1}{5!}x^5 - \tfrac{1}{7!}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)!},$$

$$\cos x = 1 - \tfrac{1}{2!}x^2 + \tfrac{1}{4!}x^4 - \tfrac{1}{6!}x^6 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!},$$

$$\tan^{-1} x = x - \tfrac{1}{3}x^3 + \tfrac{1}{5}x^5 - \tfrac{1}{7}x^7 + \cdots = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i+1}}{(2i+1)},$$

$$(1+x)^n = 1 + nx + \tfrac{n(n-1)}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{n}{i} x^i,$$

$$\frac{1}{(1-x)^{n+1}} = 1 + (n+1)x + \tbinom{n+2}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{i+n}{i} x^i,$$

$$\frac{x}{e^x - 1} = 1 - \tfrac{1}{2}x + \tfrac{1}{12}x^2 - \tfrac{1}{720}x^4 + \cdots = \sum_{i=0}^{\infty} \frac{B_i x^i}{i!},$$

$$\frac{1}{2x}(1 - \sqrt{1-4x}) = 1 + x + 2x^2 + 5x^3 + \cdots = \sum_{i=0}^{\infty} \frac{1}{i+1}\binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}} = 1 + x + 2x^2 + 6x^3 + \cdots = \sum_{i=0}^{\infty} \binom{2i}{i} x^i,$$

$$\frac{1}{\sqrt{1-4x}}\left(\frac{1-\sqrt{1-4x}}{2x}\right)^n = 1 + (2+n)x + \tbinom{4+n}{2}x^2 + \cdots = \sum_{i=0}^{\infty} \binom{2i+n}{i} x^i,$$

$$\frac{1}{1-x}\ln\frac{1}{1-x} = x + \tfrac{3}{2}x^2 + \tfrac{11}{6}x^3 + \tfrac{25}{12}x^4 + \cdots = \sum_{i=1}^{\infty} H_i x^i,$$

$$\frac{1}{2}\left(\ln\frac{1}{1-x}\right)^2 = \tfrac{1}{2}x^2 + \tfrac{3}{4}x^3 + \tfrac{11}{24}x^4 + \cdots = \sum_{i=2}^{\infty} \frac{H_{i-1}x^i}{i},$$

$$\frac{x}{1-x-x^2} = x + x^2 + 2x^3 + 3x^4 + \cdots = \sum_{i=0}^{\infty} F_i x^i,$$

$$\frac{F_n x}{1-(F_{n-1}+F_{n+1})x - (-1)^n x^2} = F_n x + F_{2n}x^2 + F_{3n}x^3 + \cdots = \sum_{i=0}^{\infty} F_{ni} x^i.$$

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 $b_i = \sum_{j=0}^{i} a_i$ 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