ICPC Team Notebook

Sorbonne Lime

Sorbonne Université

SWERC 2025

A curated reference of algorithms and data structures

October 18, 2025

Contents

1 Data Structures

1.1 union find

Complexity: nearly O(1)

```
typedef struct union_find{
    vector<int> rank, parent;
    union_find(int n){
        rank.resize(n, 0); parent.resize(n);
        for (int i = 0; i < n; i++) parent[i] = i;
    }
    int find(int i){
        int root = parent[i];
        if (parent[root] != root) return parent[i] = find(root);
        return root;
    }
    void unite(int x, int y) {
        int xRoot = find(x);
        int yRoot = find(y);
        if (xRoot == yRoot) return;
        if (rank[xRoot] < rank[yRoot]) parent[xRoot] = yRoot;
        else if (rank[yRoot] < rank[xRoot]) parent[yRoot] = xRoot;
        else{
            parent[yRoot] = xRoot;
            rank[xRoot]++;
        }
    }
} union_find;
```

2 Graph Algorithms

2.1 Shortest Paths

Complexity: O(E+VlogV)
Use: Finds the shortest path in an graph with no negative edges.

```
vi dijkstra(const vector<vector<pii>>& adj, int src) {
    vi dist(adj.size(), INT_MAX);
    priority_queue<pii, vector<pii>, greater<pii>> q;
    dist[src] = 0; q.push({0, src});
    while (!q.empty()) {
        auto [d, u] = q.top(); q.pop();
        if (d != dist[u]) continue;
        for (auto [v, w] : adj[u]) {
            if (d+w < dist[v]) {
                dist[v] = d+w;
                q.push({d+w, v});
            }
        }
    }
    return dist;
}
```

Complexity: O(V*E)
Use: Finds the shortest path in a graph. Detects negative cycles.

```
vector<int> bellmanFord(int n, vector<vector<int>>& edges, int src) {
  vector<int> dist(n, INT_MAX);
  dist[src] = 0;
  for (int i = 0; i < n; i++) {
    for (vector<int> edge : edges) {
      int u = edge[0];int v = edge[1];int wt = edge[2];
      if (dist[u] != INT_MAX && dist[u] + wt < dist[v]) {
                if(i == n - 1) return {-1};
                dist[v] = dist[u] + wt;
            }
    }
  }
    return dist;
}
```