

Competitive Programming Library

Too bad to be Accepted

Contents

1	Dynamic Programming	2
2	Bit Manipulation	3
3	Algorithms	4
4	Data Structures	5
5	Graph Theory	6
5.1	Dijkstra Algorithm	6

1 Dynamic Programming

2 Bit Manipulation

3 Algorithms

4 Data Structures

5 Graph Theory

5.1 Dijkstra Algorithm

```
1 #define INF (1e18)
2
3 int n, m;
4 vector<vector<pair<int, int>>> adj;
5 vector<int> cost;
6 vector<int> parent;
7
8 void dijkstra(int startNode = 1) {
9     priority_queue<pair<ll, int>, vector<pair<ll, int>>, greater<>> pq;
10
11     cost[startNode] = 0;
12     pq.emplace(0, startNode);
13
14     while (!pq.empty()) {
15         int u = pq.top().second;
16         ll d = pq.top().first;
17         pq.pop();
18
19         if (d > cost[u]) continue;
20
21         for (auto &p: adj[u]) {
22             int v = p.first;
23             int w = p.second;
24             if (cost[v] > cost[u] + w) {
25                 cost[v] = cost[u] + w;
26                 parent[v] = u;
27                 pq.emplace(cost[v], v);
28             }
29         }
30     }
31 }
32
33 void run_test_case(int testNum) {
34     cin >> n >> m;
35
36     adj.assign(n + 1, {});
37     cost.assign(n + 1, INF);
38     parent.assign(n + 1, -1);
39
40     while (m--) {
41         // Read Edges
42     }
43
44     dijkstra();
45
46     if (cost[n] == INF) {
47         cout << -1 << endl; // not connected {Depends on you use case}
48         return;
49     }
50
51     stack<int> ans;
52     for (int v = n; v != -1; v = parent[v]) ans.push(v);
53
54     while (!ans.empty()) { // printing the path
55         cout << ans.top() << ' ';
56         ans.pop();
57     }
58     cout << endl;
59 }
```

Dijkstra Implementation