

Competitive Programming Library

Too bad to be Accepted

Contents

1	Dynamic Programming	1
2	Bit Manipulation	2
3	Algorithms	2
4	Data Structures	3
5	Counting Principles	3
5.1	nCr	3
5.1.1	Fast nCr	3
6	Graph Theory	4
6.1	Dijkstra Algorithm	4

1 Dynamic Programming

2 Bit Manipulation

3 Algorithms

4 Data Structures

5 Counting Principles

5.1 nCr

5.1.1 Fast nCr

$$C(n, k) = \frac{n!}{(n-k)!k!} = \frac{n * (n-1) * (n-2) * \dots * (n-k+1)}{k!}$$

$$C(n, k) = \frac{n * (n-1) * (n-2) * \dots * (n-k+1)}{1 * 2 * 3 * \dots * k} = \prod_{i=0}^{k-1} \frac{n-i}{i+1} = \prod_{i=0}^{k-1} (n-i)(i+1)^{-1}$$

Fast nCr

```
int nCr(const int& n, const int& r) {  
    double res = 1;  
    for (int i = 1; i <= r; ++i)  
        res = res * (n - r + i) / i;  
    return (int)(res + 0.01);  
}
```

6 Graph Theory

6.1 Dijkstra Algorithm

```
#define INF (1e18) // for int defined as ll

int n, m;
vector<vector<pair<int, int>>> adj;
vector<int> cost;
vector<int> parent;

void dijkstra(int startNode = 1) {
    priority_queue<pair<ll, int>, vector<pair<ll, int>>,
greater<>> pq;

    cost[startNode] = 0;
    pq.emplace(0, startNode);

    while (!pq.empty()) {
        int u = pq.top().second;
        ll d = pq.top().first;
        pq.pop();

        if (d > cost[u]) continue;

        for (auto &p: adj[u]) {
            int v = p.first;
            int w = p.second;
            if (cost[v] > cost[u] + w) {
                cost[v] = cost[u] + w;
                parent[v] = u;
                pq.emplace(cost[v], v);
            }
        }
    }
}

void run_test_case(int testNum) {
    cin >> n >> m;

    adj.assign(n + 1, {});
    cost.assign(n + 1, INF);
    parent.assign(n + 1, -1);
}
```

```
while (m--) {
    // Read Edges
}

dijkstra();

if (cost[n] == INF) {
    cout << -1 << endl; // not connected {Depends on you
use case}
    return;
}

stack<int> ans;
for (int v = n; v != -1; v = parent[v]) ans.push(v);

while (!ans.empty()) { // printing the path
    cout << ans.top() << ' ';
    ans.pop();
}
cout << endl;
}
```

Dijkstra Implementation