NAMUNAVIY MASALALAR

1-masala. Og‘irliksiz grafda eng qisqa yo‘l

Og‘irliksiz graf berilgan. 1-tugundan n-tugungacha eng qisqa masofani toping.

Yechim (BFS)

```
Queue<Integer> q = new ArrayDeque<>();
int[] dist = new int[n + 1];
Arrays.fill(dist, -1);

dist[1] = 0;
q.add(1);

while (!q.isEmpty()) {
    int u = q.poll();
    for (int v : g[u]) {
        if (dist[v] == -1) {
            dist[v] = dist[u] + 1;
            q.add(v);
        }
    }
}
System.out.println(dist[n]);
```

2-masala. Grafdagi komponentlar soni

Yo‘naltirilmagan graf berilgan. Bog‘langan komponentlar sonini toping.

Yechim (DFS)

```
boolean[] vis = new boolean[n + 1];

void dfs(int u) {
    vis[u] = true;
    for (int v : g[u]) {
        if (!vis[v]) dfs(v);
    }
}

int cnt = 0;
for (int i = 1; i <= n; i++) {
    if (!vis[i]) {
        dfs(i);
        cnt++;
```

```
    }
}
System.out.println(cnt);
```

3-masala. Eng qisqa yo‘l (og‘irlikli)

Og‘irliklari manfiy bo‘lmagan graf berilgan. 1 dan n gacha eng qisqa yo‘l uzunligini toping.

Yechim (Dijkstra)

```
long[] dist = new long[n + 1];
Arrays.fill(dist, Long.MAX_VALUE);
dist[1] = 0;

PriorityQueue<long[]> pq = new
PriorityQueue<>(Comparator.comparingLong(a -> a[0]));
pq.add(new long[]{0, 1});

while (!pq.isEmpty()) {
    long[] cur = pq.poll();
    long d = cur[0];
    int u = (int) cur[1];

    if (d > dist[u]) continue;

    for (long[] e : g[u]) {
        int v = (int) e[0];
        long w = e[1];
        if (dist[v] > d + w) {
            dist[v] = d + w;
            pq.add(new long[]{dist[v], v});
        }
    }
}
System.out.println(dist[n]);
```

4-masala. Sikl bormi?

Yo‘naltirilgan graf berilgan. Grafda sikl mavjudmi aniqlang.

Yechim (DFS + ranglar)

```
int[] color = new int[n + 1]; // 0,1,2
boolean cycle = false;
```

```
void dfs(int u) {
    color[u] = 1;
    for (int v : g[u]) {
        if (color[v] == 0) dfs(v);
        else if (color[v] == 1) cycle = true;
    }
    color[u] = 2;
}
```

5-masala. Topologik tartib

DAG berilgan. Tugunlarni topologik tartibda chiqaring.

Yechim (Kahn algoritmi)

```
Queue<Integer> q = new ArrayDeque<>();
for (int i = 1; i <= n; i++)
    if (indeg[i] == 0) q.add(i);

ArrayList<Integer> order = new ArrayList<>();

while (!q.isEmpty()) {
    int u = q.poll();
    order.add(u);
    for (int v : g[u]) {
        if (--indeg[v] == 0) q.add(v);
    }
}
System.out.println(order);
```

6-masala. Minimal Spanning Tree og‘irligi

Og‘irlikli graf berilgan. MST umumiy og‘irligini toping.

Yechim (Kruskal + DSU)

```
Collections.sort(edges, Comparator.comparingInt(e ->
e.w));

int ans = 0;
for (Edge e : edges) {
    if (find(e.u) != find(e.v)) {
        union(e.u, e.v);
        ans += e.w;
    }
}
```

```
System.out.println(ans);
```

7-masala. SCC soni

Yo‘naltirilgan graf berilgan. Strongly Connected Components sonini toping.

Yechim (Kosaraju – sxema)

```
// 1) DFS → order
// 2) Grafni teskari qilish
// 3) Order bo‘yicha DFS
```

Bu bosqichda sxemani bilish yetarli.

8-masala. DAG’da eng uzun yo‘l

DAG berilgan. 1-tugundan boshlab eng uzun yo‘l uzunligini toping.

Yechim (Topo + DP)

```
int[] dp = new int[n + 1];
Arrays.fill(dp, -1);
dp[1] = 0;

for (int u : topo) {
    if (dp[u] == -1) continue;
    for (int v : g[u]) {
        dp[v] = Math.max(dp[v], dp[u] + 1);
    }
}
```

9-masala. Eng katta komponent o‘lchami

Graf berilgan. Eng katta bog‘langan komponent o‘lchamini toping.

Yechim (DFS)

```
int dfsSize(int u) {
    vis[u] = true;
    int cnt = 1;
    for (int v : g[u]) {
        if (!vis[v]) cnt += dfsSize(v);
    }
    return cnt;
}
```

10-masala. Eng yaqin tugun (multi-source BFS)

Bir nechta boshlang‘ich tugunlar berilgan. Har tugun uchun eng yaqin boshlang‘ich tugungacha masofani toping.

Yechim (Multi-source BFS)

```
Queue<Integer> q = new ArrayDeque<>();
int[] dist = new int[n + 1];
Arrays.fill(dist, -1);

for (int s : sources) {
    dist[s] = 0;
    q.add(s);
}

while (!q.isEmpty()) {
    int u = q.poll();
    for (int v : g[u]) {
        if (dist[v] == -1) {
            dist[v] = dist[u] + 1;
            q.add(v);
        }
    }
}
```