**NAMUNAVIY MASALALAR**

1-masala. Dijkstra — eng qisqa yo‘l

Og‘irlikli graf berilgan (og‘irliklar manfiy emas). 1-tugundan n-tugungacha eng qisqa masofani toping.

Yechim (Dijkstra + heapq)

```
import heapq

n, m = map(int, input().split())
g = [[] for _ in range(n + 1)]

for _ in range(m):
    u, v, w = map(int, input().split())
    g[u].append((v, w))
    g[v].append((u, w))

INF = 10**18
dist = [INF] * (n + 1)
dist[1] = 0

pq = [(0, 1)]
while pq:
    d, u = heapq.heappop(pq)
    if d > dist[u]:
        continue
    for v, w in g[u]:
        if dist[v] > d + w:
            dist[v] = d + w
            heapq.heappush(pq, (dist[v], v))

print(dist[n] if dist[n] != INF else -1)
```

2-masala. DSU — komponentlar soni

n ta tugun va m ta bog‘lanish berilgan. Grafdagi bog‘langan komponentlar sonini toping.

Yechim (Union–Find)

```
parent = []

def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]

def union(a, b):
    ra, rb = find(a), find(b)
    if ra != rb:
        parent[rb] = ra
```

```
n, m = map(int, input().split())
parent = list(range(n + 1))

for _ in range(m):
    u, v = map(int, input().split())
    union(u, v)

roots = set(find(i) for i in range(1, n + 1))
print(len(roots))
```

3-masala. MST — minimal spanning tree

Og‘irlikli graf berilgan. Grafni bog‘lash uchun kerak bo‘lgan minimal umumiy og‘irlikni toping.

Yechim (Kruskal)

```
edges = []
n, m = map(int, input().split())

for _ in range(m):
    u, v, w = map(int, input().split())
    edges.append((w, u, v))

edges.sort()

parent = list(range(n + 1))

def find(x):
    if parent[x] != x:
        parent[x] = find(parent[x])
    return parent[x]

ans = 0
for w, u, v in edges:
    ru, rv = find(u), find(v)
    if ru != rv:
        parent[rv] = ru
        ans += w

print(ans)
```

4-masala. Topological sort

Yo‘naltirilgan graf berilgan (DAG). Tugunlarni topologik tartibda chiqaring.

Yechim (Kahn algoritmi)

```
from collections import deque

n, m = map(int, input().split())
g = [[] for _ in range(n + 1)]
indeg = [0] * (n + 1)
```

```
for _ in range(m):
    u, v = map(int, input().split())
    g[u].append(v)
    indeg[v] += 1

dq = deque()
for i in range(1, n + 1):
    if indeg[i] == 0:
        dq.append(i)

res = []
while dq:
    u = dq.popleft()
    res.append(u)
    for v in g[u]:
        indeg[v] -= 1
        if indeg[v] == 0:
            dq.append(v)

print(*res)
```

5-masala. BFS — panjarada yo‘l

n×m panjara (`0` – bo‘sh, `1` – devor). (1,1) dan (n,m) gacha eng qisqa yo‘l uzunligini toping.

Yechim (BFS)

```
from collections import deque

n, m = map(int, input().split())
a = [list(map(int, input().split())) for _ in range(n)]

dist = [[-1]*m for _ in range(n)]
dq = deque([(0, 0)])
dist[0][0] = 0

dx = [1, -1, 0, 0]
dy = [0, 0, 1, -1]

while dq:
    x, y = dq.popleft()
    for k in range(4):
        nx, ny = x + dx[k], y + dy[k]
        if 0 <= nx < n and 0 <= ny < m:
            if a[nx][ny] == 0 and dist[nx][ny] == -1:
                dist[nx][ny] = dist[x][y] + 1
                dq.append((nx, ny))

print(dist[n-1][m-1])
```

6-masala. Fenwick Tree — prefix sum

Massiv berilgan. Ikki tur so‘rov bor:

- `1 i x` → `a[i] += x`
- `2 l r` → `[l, r]` yig‘indisi

Yechim (Fenwick / BIT)

```
n, q = map(int, input().split())
a = [0] + list(map(int, input().split()))

bit = [0] * (n + 1)

def add(i, v):
    while i <= n:
        bit[i] += v
        i += i & -i

def sum_(i):
    s = 0
    while i > 0:
        s += bit[i]
        i -= i & -i
    return s

for i in range(1, n + 1):
    add(i, a[i])

for _ in range(q):
    t, x, y = map(int, input().split())
    if t == 1:
        add(x, y)
    else:
        print(sum_(y) - sum_(x - 1))
```

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

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

Yechim

```
from collections import deque

n, m, k = map(int, input().split())
g = [[] for _ in range(n + 1)]
for _ in range(m):
    u, v = map(int, input().split())
    g[u].append(v)
    g[v].append(u)

sources = list(map(int, input().split()))
dist = [-1] * (n + 1)
dq = deque()

for s in sources:
    dist[s] = 0
    dq.append(s)

while dq:
    u = dq.popleft()
    for v in g[u]:
        if dist[v] == -1:
            dist[v] = dist[u] + 1
            dq.append(v)

print(*dist[1:])
```

8-masala. Sikl bormi?

Yo‘naltirilgan graf berilgan. Grafda sikl mavjudmi aniqlang.

Yechim (DFS + ranglar)

```
import sys
sys.setrecursionlimit(10**7)

n, m = map(int, input().split())
g = [[] for _ in range(n + 1)]
for _ in range(m):
    u, v = map(int, input().split())
    g[u].append(v)

color = [0] * (n + 1)  # 0=unvisited, 1=visiting, 2=done
ok = True

def dfs(u):
    global ok
    color[u] = 1
    for v in g[u]:
        if color[v] == 0:
            dfs(v)
        elif color[v] == 1:
            ok = False
    color[u] = 2

for i in range(1, n + 1):
    if color[i] == 0:
        dfs(i)

print("YES" if not ok else "NO")
```

9-masala. Eng arzon yo‘l (2 holatli Dijkstra)

Grafda 1 ta qirrani bepul o‘tish mumkin. 1 dan n gacha minimal narxni toping.

Yechim

```
import heapq
```

```
n, m = map(int, input().split())
g = [[] for _ in range(n + 1)]
for _ in range(m):
    u, v, w = map(int, input().split())
    g[u].append((v, w))
    g[v].append((u, w))

INF = 10**18
dist = [[INF]*2 for _ in range(n + 1)]
dist[1][0] = 0

pq = [(0, 1, 0)]
while pq:
    d, u, used = heapq.heappop(pq)
    if d > dist[u][used]:
        continue
    for v, w in g[u]:
        if dist[v][used] > d + w:
            dist[v][used] = d + w
            heapq.heappush(pq, (dist[v][used], v, used))
        if used == 0 and dist[v][1] > d:
            dist[v][1] = d
            heapq.heappush(pq, (d, v, 1))

print(min(dist[n]))
```

10-masala. Eng katta bog‘langan komponent

Graf berilgan. Eng ko‘p tugunga ega komponent o‘lchamini toping.

Yechim (DFS)

```
n, m = map(int, input().split())
g = [[] for _ in range(n + 1)]
for _ in range(m):
    u, v = map(int, input().split())
    g[u].append(v)
    g[v].append(u)

vis = [False] * (n + 1)

def dfs(u):
    stack = [u]
    cnt = 0
    vis[u] = True
    while stack:
        x = stack.pop()
        cnt += 1
        for v in g[x]:
            if not vis[v]:
                vis[v] = True
                stack.append(v)
    return cnt

ans = 0
for i in range(1, n + 1):
    if not vis[i]:
        ans = max(ans, dfs(i))

print(ans)
```