算法分析和复杂性理论

第 12 次作业

张瀚文 2201212865

1 访问所有节点的最短路径（847）

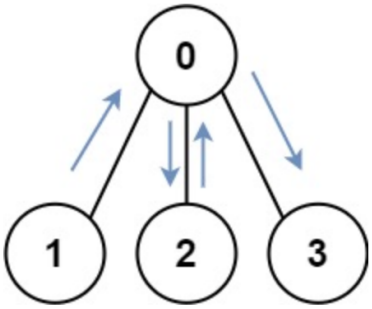
题目描述：

存在一个由 `n` 个节点组成的无向连通图，图中的节点按从 `0` 到 `n - 1` 编号。

给你一个数组 `graph` 表示这个图。其中，`graph[i]` 是一个列表，由所有与节点 `i` 直接相连的节点组成。

返回能够访问所有节点的最短路径的长度。你可以在任一节点开始和停止，也可以多次重访节点，并且可以重用边。

示例 1：




```
输入：graph = [[1,2,3],[0],[0],[0]]
输出：4
解释：一种可能的路径为 [1,0,2,0,3]
```

解题思路：由于题目需要我们求出「访问所有节点的最短路径的长度」，并且图中每一条边的长度均为 111，因此我们可以考虑使用广度优先搜索的方法求出最短路径。

运行截图：

测试代码：

```
class Solution:
    def shortestPathLength(self, graph: List[List[int]]) -> int:
        n = len(graph)
        q = deque((i, 1 << i, 0) for i in range(n))
        seen = {(i, 1 << i) for i in range(n)}
        ans = 0

        while q:
            u, mask, dist = q.popleft()
            if mask == (1 << n) - 1:
                ans = dist
                break
            # 搜索相邻的节点
            for v in graph[u]:
                # 将 mask 的第 v 位置为 1
                mask_v = mask | (1 << v)
                if (v, mask_v) not in seen:
                    q.append((v, mask_v, dist + 1))
                    seen.add((v, mask_v))
```

return ans

2 网络延迟时间（743）

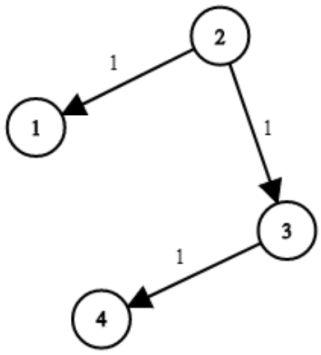
题目描述：

有 `n` 个网络节点，标记为 `1` 到 `n`。

给你一个列表 `times`，表示信号经过 **有向** 边的传递时间。 `times[i] = (u_i, v_i, w_i)`，其中 `u_i` 是源节点，`v_i` 是目标节点， `w_i` 是一个信号从源节点传递到目标节点的时间。

现在，从某个节点 `K` 发出一个信号。需要多久才能使所有节点都收到信号？如果不能使所有节点收到信号，返回 `-1` 。

示例 1：




```
输入: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2
输出: 2
```

解题思路：本题需要用到单源最短路径算法 Dijkstra。每次从「未确定节点」中取一个与起点距离最短的点，将它归类为「已确定节点」，并用它「更新」从起点到其他所有「未确定节点」的距离。直到所有点都被归类为「已确定节点」。

运行截图：

测试代码：

```
class Solution:
    def networkDelayTime(self, times: List[List[int]], n: int, k: int) -> int:
        g = [[float('inf')] * n for _ in range(n)]
        for x, y, time in times:
            g[x - 1][y - 1] = time

        dist = [float('inf')] * n
        dist[k - 1] = 0
        used = [False] * n
        for _ in range(n):
            x = -1
            for y, u in enumerate(used):
                if not u and (x == -1 or dist[y] < dist[x]):
                    x = y
            used[x] = True
            for y, time in enumerate(g[x]):
                dist[y] = min(dist[y], dist[x] + time)

        ans = max(dist)
        return ans if ans < float('inf') else -1
```