03.3 - Breadth-First Search

BFS

1. 随机从一个起始顶点 s 开始，将其置于第 0 层，并标记为“已探索”
2. 对于第 i 层的任意一个节点，将它所有未被探索的邻居节点放入第 $i+1$ 层，并将这些邻居节点标记为已探索
3. 当第 j 层的所有节点都没有未被探索的邻居时，算法终止

算法伪代码

```
Algorithm BFS(G, s)
    Initialise empty list L_0
    Insert s into L_0

    Set i = 0
    While L_i is not empty
        Initialise empty list L_{i+1}
        for each node v in L_i
            for all edges e incident to v
                if edge e is unexplored
                    let w be the other endpoint of e
                    if node w is unexplored
                        label e as discovery edge
                        insert w into L_{i+1}
                    else
                        label e as cross edge
        i = i + 1
```

1. 初始化空数组 L_0
2. 将初始节点 s 插入 L_0
3. 设置 $i = 0$
4. 当 L_i 非空时：
    1. 初始化空数组 L_{i+1}
    2. 对于 L_i 中的每个节点 v，执行：
        1. 遍历所有与 v 相连的 e：
            1. 如果边 e 未被探索：
                1. 设 w 为边 e 的另一个端点
                2. 如果节点 w 未被探索：
                    1. 将边 e 标记为发现边
                    2. 将节点 w 插入 L_{i+1}
                3. 否则：
                    1. 将边 e 标记为交叉边
    3. $i = i + 1$

Properties of BFS

- 为了简化讨论，假设图是连通的。这意味着图中任意两个顶点之间都有路径相连
- BFS遍历过程中会访问图中的所有顶点
- BFS中的发现边会形成一棵生成树，这棵树包含所有顶点且无环，覆盖了整个连通图
- 从起始顶点 s 到第 i 层的某个顶点 v 的路径包含 i 条边，这是从 s 到 v 的最短路径
- 如果边 $e=(u,v)$ 是交叉边，那么 u 和 v 至多相差一个层级
  - 由于BFS是按层级遍历的，意味着在第 i 层访问的节点，其所有直接相连的邻居节点要么在第 i 层、要么在第 $i+1$ 层
- 如果边 $e=(u,v)$ 是发现边，那么 u 和 v 恰好相差一个层级

BFS 时间复杂度

- BFS按层级遍历节点，每次处理一个层级的节点
  - 由于BFS确保在处理某个层级的节点之前，所有前一层级的节点已经被处理完毕，因此每个节点只会被访问一次
- 每条边会被检查两次
  - 一次是从一个端点访问，另一次是从另一个端点访问

因此 BFS 的运行时间是 $O(n+m)$

BFS 与 DFS

（我直接 GPT）

BFS

- 最短路径问题：
  - BFS适用于无权图中从起始节点到目标节点的最短路径查找。因为BFS按层级遍历节点，第一次访问到目标节点时保证路径是最短的
  - 例如：在迷宫求解、社交网络中找到两个人之间的最短关系链等
- 连通性问题：
  - BFS可以用于查找图中的所有连通分量。通过从每个未被访问的节点开始BFS，可以发现所有与该节点连通的节点
  - 例如：判断一个图是否连通、查找图中的孤立区域等
- 层级遍历：
  - BFS按层级遍历节点，非常适用于需要按层次结构处理的场景，如按层打印树的节点。
  - 例如：计算树的深度、在广度优先的顺序中访问树的节点等

DFS

- 路径查找和可达性：
  - DFS适用于查找图中的所有路径，或检测图中节点的可达性。DFS能够深入图的结构，直到找到所需的目标或路径
  - 例如：在迷宫中寻找所有可能的路径、检查两点是否连通等
- 拓扑排序：
  - 在有向无环图（DAG）中，DFS用于拓扑排序。这在任务调度、编译依赖关系等场景中非常有用
  - 例如：任务执行顺序安排、课程先修关系排序等
- 强连通分量：
  - 在有向图中，DFS用于查找强连通分量（Tarjan算法或Kosaraju算法）

    - 例如：社交网络中找出紧密联系的群体等
- 检测环路：
    - DFS可以用于检测图中的环路。如果在DFS过程中遇到已访问的节点且不是当前路径上的节点，则存在环
    - 例如：判断课程安排中是否有循环依赖等