10. Graph图BFS广度优先搜索套路

LeetCode

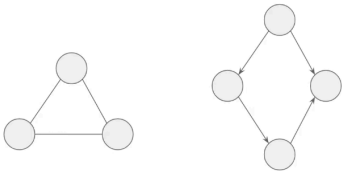
Graph解题模版1

BFS

Graph

类似LinkedList的概念，内存中不一定连续的数据，由各个节点的Reference串起来组成

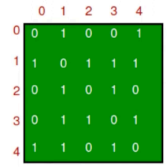
- 可能有环
- 分为无向图和有向图
- 没有固定入口
- 可能有多个入口

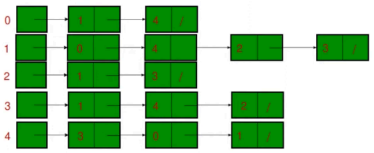


Graph Representation

图该以什么形式存储？最常用的两大类

- Adjacency Matrix
- Adjacency List





Adjacency List 可以为 map, list 或者 set

Adjacency List

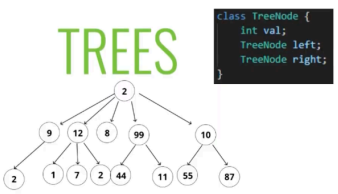
最常用的2种实现方式 (List可用Set代替)

- List<T>[n]
  - adjList[i]: All neighbors of node i
  - Need to know number of nodes (n) beforehand
- Map<T, List<T>>
  - adjList.get(i): All neighbors of node i

Tree

类似LinkedList的概念，内存中不一定连续的数据，由各个节点的Reference串起来组成

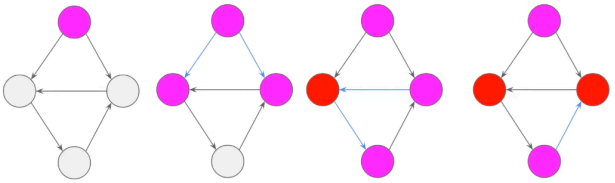
- 节点可以分为Parent和Child两类
- 可以看做一个特殊的无环无向图
- 只有一个入口 (root)



BFS (Breadth-First Search)

以层为概念的搜索方式。因为是水平展开所有nodes，所以适合寻找最短路径

图可能有环，需要查重



找最短路径只适用于Uniform Cost （每条edge的weight一样）

BFS模板

1. Initialize a Queue with all starting points, a HashSet to record visited nodes
2. While queue is not empty
   a. Retrieve current queue size as number of nodes in the current level
   b. for each node in current level
      i. Poll out one node
      ii. If this is the node we want, return it
      iii. Offer all its neighbor to the queue if not visited and valid
   c. Increase level

小技巧: 对于2D Matrix的图，matrix[i][j]的neighbors一般都是上下左右4个，所以预先存一个4 direction array可以帮助访问neighbors → directions = {{0, 1}, {1, 0}, {0, -1}, {-1, 0}}

Time Complexity O(V + E)

542. 01 矩阵

中等 相关标签 相关企业

给定一个由 `0` 和 `1` 组成的矩阵 `mat` ，请输出一个大小相同的矩阵，其中每一个格子是 `mat` 中对应位置元素到最近的 `0` 的距离。

两个相邻元素间的距离为 `1` 。

示例 1：

0	0	0
0	1	0
0	0	0

```
输入：mat = [[0,0,0],[0,1,0],[0,0,0]]
输出：[[0,0,0],[0,1,0],[0,0,0]]
```

示例 2：

0	0	0
0	1	0
1	1	1

```
输入：mat = [[0,0,0],[0,1,0],[1,1,1]]
输出：[[0,0,0],[0,1,0],[1,2,1]]
```

提示：

- `m == mat.length`
- `n == mat[i].length`
- `1 <= m, n <= 10^4`
- `1 <= m * n <= 10^4`
- `mat[i][j] is either 0 or 1.`
- `mat` 中至少有一个 `0`

542. 01 Matrix

直接思维: 对于每一个1，利用BFS找他最近的0 → O(mn * mn)

逆向思维: 对于所有的0，利用BFS填充到每一个1的距离 → O(mn)

[0, 0, 0]	[0, 0, 0]	[0, 0, 0]
[0, 1, 0]	[0, 1, 0]	[0, 1, 0]
[1, 1, 1]	[1, 1, 1]	[1, 2, 1]

542. 01 Matrix

1. Initialize a Queue with all 0 nodes, a boolean[][] to record visited nodes
2. While queue is not empty
   a. Retrieve current queue size as number of nodes in the current level
   b. for each node in current level
      i. Poll out one node
      ii. If this is the node we want, return it
      iii. Offer all its neighbor to the queue if not visited and valid
   c. Increase level

Time: O(mn) Space O(mn)

```java
int[][] dirs = {{0, 1}, {1, 0}, {-1, 0}, {0, -1}};

public int[][] updateMatrix(int[][] matrix) {
    int m = matrix.length, n = matrix[0].length;
    int[][] res = new int[m][n];

    boolean[][] visited = new boolean[m][n];
    Queue<int[]> queue = new LinkedList<>();
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] == 0) {
                queue.offer(new int[]{i, j});
                visited[i][j] = true;
            }
        }
    }

    int cost = 0;
    while (!queue.isEmpty()) {
        int size = queue.size();
        for (int s = 0; s < size; s++) {
            int[] cur = queue.poll();
            int i = cur[0], j = cur[1];
            if (matrix[i][j] == 1) {
                res[i][j] = cost;
            }

            for (int[] dir : dirs) {
                int x = i + dir[0], y = j + dir[1];
                if (x >= 0 && x < m && y >= 0 && y < n && !visited[x][y]) {
                    queue.offer(new int[]{x, y});
                    visited[x][y] = true;
                }
            }
        }
        cost++;
    }

    return res;
}
```

127. 单词接龙

已解答

困难 相关标签 相关企业

字典 `wordList` 中从单词 `beginWord` 到 `endWord` 的 **转换序列** 是一个按下述规格形成的序列 $beginWord \rightarrow s_1 \rightarrow s_2 \rightarrow ... \rightarrow s_k$：

- 每一对相邻的单词只差一个字母。
- 对于 $1 <= i <= k$ 时，每个 s_i 都在 `wordList` 中。注意，`beginWord` 不需要在 `wordList` 中。
- $s_k == endWord$

给你两个单词 `beginWord` 和 `endWord` 和一个字典 `wordList`，返回 *从 beginWord 到 endWord 的* ***最短转换序列*** *中的* ***单词数目*** 。如果不存在这样的转换序列，返回 `0` 。

示例 1：

> **输入：** beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
> **输出：** 5
> **解释：** 一个最短转换序列是 "hit" -> "hot" -> "dot" -> "dog" -> "cog", 返回它的长度 5。

示例 2：

> **输入：** beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
> **输出：** 0
> **解释：** endWord "cog" 不在字典中，所以无法进行转换。

提示：

- `1 <= beginWord.length <= 10`
- `endWord.length == beginWord.length`
- `1 <= wordList.length <= 5000`
- `wordList[i].length == beginWord.length`
- `beginWord`、`endWord` 和 `wordList[i]` 由小写英文字母组成
- `beginWord != endWord`
- `wordList` 中的所有字符串 **互不相同**

127. Word Ladder

首先要构建Graph

1. For each (word1, word2) pair
   a. if word1 and word2 are one character change away, then word1 and word2 are connected by an edge

这是个无向图

Time: $O(n^2)$ Space $O(n^2)$

```java
private Map<String, List<String>> constructGraph(List<String> wordList) {
    Map<String, List<String>> graph = new HashMap<>();
    int n = wordList.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            String w1 = wordList.get(i), w2 = wordList.get(j);
            if (oneChangeAway(w1, w2)) {
                graph.computeIfAbsent(w1, k -> new ArrayList<>()).add(w2);
                graph.computeIfAbsent(w2, k -> new ArrayList<>()).add(w1);
            }
        }
    }
    return graph;
}

private boolean oneChangeAway(String w1, String w2) {
    int diff = 0;
    for (int i = 0; i < w1.length(); i++) {
        char c1 = w1.charAt(i), c2 = w2.charAt(i);
        if (c1 != c2) {
            diff++;
        }
    }
    return diff == 1;
}
```

127. Word Ladder

构建完图之后，就变成一个利用BFS从起点找到终点的最短路径的基本问题

1. Initialize a Queue with beginWord, a HashSet to record visited words
2. While queue is not empty
   a. Retrieve current queue size as number of words in the current level
   b. for each word in current level
      i. Poll out one word
      ii. If this is the endWord, return the cost
      iii. Offer all its neighbor to the queue if not visited and valid
   c. Increase cost

Time: $O(n^2)$ Space $O(n^2)$

更多相关题目

Shortest Bridge (934)

Minimum Height Trees (310)

Shortest Path in Binary Matrix (1091)

Rotting Oranges (994)

All Nodes Distance K in Binary Tree (863)

Shortest Distance from All Buildings (317)