Word Search

四联通遍历, 上下左右
八联通遍历, 上下左右 斜方向
X

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

1. 创建 Trie
2. 遍历 Board 每一个元素, DFS + Back tracking 来生成所有可能性
   - 剪枝: 如果当前生成的字符不在 Trie 中, 则当前字符串终止 DFS

```java
class Solution {
    private int m;
    private int n;
    private int[] dx = {-1, 1, 0, 0};
    private int[] dy = {0, 0, -1, 1};

    public List<String> findWords(char[][] board, String[] words) {
        if (words == null || words.length == 0) return new ArrayList<String>();
        Trie root = new Trie();
        for (String word : words)
            root.insert(word);
        m = board.length;
        n = board[0].length;
        HashSet<String> set = new HashSet<>();
        for (int i = 0; i < m; ++i)
            for (int j = 0; j < n; ++j)
                dfs(board, set, i, j, "", root);
        return new ArrayList<String>(set);
    }

    public void dfs(char[][] board, HashSet set, int i, int j, String currWord, Trie root) {
        if (i < 0 || j < 0 || i > m - 1 || j > n - 1 || board[i][j] == '@') return;
        currWord += board[i][j];
        TrieNode search = root.searchPrefix(currWord);
        if (search == null) return;
        if (root.search(currWord)) {
            set.add(currWord);
            return;
        }
        char temp = board[i][j];
        board[i][j] = '@';
        for (int k = 0; k < 4; ++k) {
            dfs(board, set, i+dx[k], j+dy[k], currWord, root);
        }
        board[i][j] = temp;
    }
}
```

Friend Circles ----- UnionFind

DFS 访问二维数组顺序

https://algorithms.tutorialhorizon.com/depth-first-search-dfs-in-2d-matrix-2d-array-iterative-solution/

```
int [][] grid = new int[][] {
                {1, 2, 3, 4},
                {5, 6, 7, 8},
                {9, 10, 11, 12},
                {13, 14, 15, 16}}
Output:
Depth-First Traversal:
1 5 9 13 14 10 6 2 3 7 11 15 16 12 8 4
```