2020-04-18 - Handout – String Algorithms

Q1.Longest Substring Without Repeating Characters

Link: https://leetcode.com/problems/longest-substring-without-repeating-characters/

Given a string, find the length of the **longest substring** without repeating characters.

Example 1:

```
Input: "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
```

Example 2:

```
Input: "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
```

Q2. Generate Parentheses

Link: https://leetcode.com/problems/generate-parentheses/

Given *n* pairs of parentheses, write a function to generate all combinations of well-formed parentheses.For example, given *n* = 3, a solution set is:

For example, given *n* = 3, a solution set is:

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
]
```

Q3. Word Ladder

Link: https://leetcode.com/problems/word-ladder/

Given two words (*beginWord* and *endWord*), and a dictionary's word list, find the length of shortest transformation sequence from *beginWord* to *endWord*, such that:

1. Only one letter can be changed at a time.
2. Each transformed word must exist in the word list.

Note:

- Return 0 if there is no such transformation sequence.
- All words have the same length.
- All words contain only lowercase alphabetic characters.
- You may assume no duplicates in the word list.
- You may assume *beginWord* and *endWord* are non-empty and are not the same.

Example 1:

```
Input:
beginWord = "hit",
endWord = "cog",
wordList = ["hot","dot","dog","lot","log","cog"]

Output: 5

Explanation: As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" ->
"cog",
return its length 5.
```

Q4. Similar String Groups

Link: https://leetcode.com/problems/similar-string-groups/

Two strings X and Y are similar if we can swap two letters (in different positions) of X, so that it equals Y. Also two strings X and Y are similar if they are equal.

For example, "tars" and "rats" are similar (swapping at positions 0 and 2), and "rats" and "arts" are similar, but "star" is not similar to "tars", "rats", or "arts".

Together, these form two connected groups by similarity: {"tars", "rats", "arts"} and {"star"}. Notice that "tars" and "arts" are in the same group even though they are not similar. Formally, each group is such that a word is in the group if and only if it is similar to at least one other word in the group.

We are given a list A of strings. Every string in A is an anagram of every other string in A. How many groups are there?

Example 1:

```
Input: A = ["tars","rats","arts","star"]
Output: 2
```

Constraints:

- `1 <= A.length <= 2000`
- `1 <= A[i].length <= 1000`
- `A.length * A[i].length <= 20000`
- All words in `A` consist of lowercase letters only.
- All words in `A` have the same length and are anagrams of each other.

- The judging time limit has been increased for this question.