**[Longest String Chain](https://leetcode.com/problems/longest-string-chain/)**

You are given an array of words where each word consists of lowercase English letters.

wordA is a **predecessor** of wordB if and only if we can insert **exactly one** letter anywhere in wordA **without changing the order of the other characters** to make it equal to wordB.

* For example, "abc" is a **predecessor** of "abac", while "cba" is not a **predecessor** of "bcad".

A **word chain**is a sequence of words [word1, word2, ..., wordk] with k >= 1, where word1 is a **predecessor** of word2, word2 is a **predecessor** of word3, and so on. A single word is trivially a **word chain** with k == 1.

Return *the****length****of the****longest possible word chain****with words chosen from the given list of*words.

**Example 1:**

**Input:** words = ["a","b","ba","bca","bda","bdca"]

**Output:** 4

**Explanation**: One of the longest word chains is ["a","ba","bda","bdca"].

**Example 2:**

**Input:** words = ["xbc","pcxbcf","xb","cxbc","pcxbc"]

**Output:** 5

**Explanation:** All the words can be put in a word chain ["xb", "xbc", "cxbc", "pcxbc", "pcxbcf"].

**Example 3:**

**Input:** words = ["abcd","dbqca"]

**Output:** 1

**Explanation:** The trivial word chain ["abcd"] is one of the longest word chains.

["abcd","dbqca"] is not a valid word chain because the ordering of the letters is changed.

**Constraints:**

* 1 <= words.length <= 1000
* 1 <= words[i].length <= 16
* words[i] only consists of lowercase English letters.

Code :

class Solution {

public:

bool check(string& s1, string& s2){

    if(s1.size() != s2.size() + 1) return false;

    int first = 0;

    int second = 0;

    while(first < s1.size()){

        if(second < s2.size() && s1[first] == s2[second]){

            first ++;

            second ++;

        }

        else first ++;

    }

    if(first == s1.size() && second == s2.size()) return true;

    else return false;

}

    static bool comp(string& s1, string& s2){

        return s1.size() < s2.size();

    }

    int longestStrChain(vector<string>& words) {   // based on LIS if observed carefully!

        int n = words.size();

        //sorting accordin to the size of each word using comparator!

  //IMP -> SORT

  sort( words.begin(), words.end(), comp );

        vector<int> dp(n,1);

        int maxi = 1;

        for(int i=0; i<n; i++){

            for(int prev=0; prev<i; prev++){

                if(check(words[i],words[prev]) && dp[i]<dp[prev]+1){

                    dp[i]=dp[prev]+1;

                }

            }

            maxi = max(maxi,dp[i]);

        }

        return maxi;

    }

};

Link: <https://leetcode.com/problems/longest-string-chain/?envType=daily-question&envId=2024-02-19>