Qn link : <https://leetcode.com/problems/longest-palindrome-by-concatenating-two-letter-words/>

Observation :

There are only two kind of string is present

* One is both the chars are same. 🡪 **gg**
* Another one is both the chars are not same. 🡪 **lc**

We make cases for these two strings

Step 1 : Init an empty Hashmap and put all words inside the map.

Step 2 : Run a for loop from 0 to n

Step 3 : For each iteration get the reverse of that word

Step 4 : Check wthter the string is belong to **type 1** or **type 2** .

Step 5 : if it is type 1 , then the reverse is equal to the current string else it is not equal

Step 6 : if the palindromic string present odd number of times then we can also take it as the palindrome

Step 7 : If it is type 2 , then get the min of original and reversed

Step 8 : For each cnt we multiply it by 4 , because the palindrome is formed by four words .

class Solution {

    public int longestPalindrome(String[] words) {

        Map<String, Integer> wordCount = new HashMap<>();

        for (String word : words) {

            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);

        }

        int center = 0;

        int length = 0;

        for (String word : words) {

            String reversedWord = new StringBuilder(word).reverse().toString();

            if (word.equals(reversedWord)) {

                int count = wordCount.get(word);

                length += 4 \* (count / 2);

                if (count % 2 == 1) {

                    center = 1;

                }

                wordCount.put(word, 0);

            } else if (wordCount.containsKey(reversedWord)) {

                int countWord = wordCount.get(word);

                int countReversedWord = wordCount.get(reversedWord);

                length += 4 \* Math.min(countWord, countReversedWord);

                wordCount.put(word, 0);

                wordCount.put(reversedWord, 0);

            }

        }

        return center \* 2 + length;

    }

}