318. Maximum Product of Word Lengths

Medium 437 43 Favorite Share

Given a string array `words`, find the maximum value of `length(word[i]) * length(word[j])` where the two words do not share common letters. You may assume that each word will contain only lower case letters. If no such two words exist, return 0.

Example 1:

```
Input: ["abcw","baz","foo","bar","xtfn","abcdef"]
Output: 16
Explanation: The two words can be "abcw", "xtfn".
```

Example 2:

```
Input: ["a","ab","abc","d","cd","bcd","abcd"]
Output: 4
Explanation: The two words can be "ab", "cd".
```

Example 3:

```
Input: ["a","aa","aaa","aaaa"]
Output: 0
Explanation: No such pair of words.
```

```java
public class L318 {
    /* 主要难点在于如何简单的判断两个word是否存在相同字母，方法如下：小写字母一共有26个，
     * 一个int是32位，那么可以用int的每个位来代表一个字母
例如：abcd  = 0000 0000 0000 0000 0000 0000 0000 1111;
例如：abcg  = 0000 0000 0000 0000 0000 0000 0100 0111;
     */
    public int maxProduct(String[] words) {
        if(words.length <= 1)
            return 0;

        int [] preProcessed = new int [words.length];

        for(int i = 0; i < words.length; i ++) {
            for(int j = 0; j < words[i].length(); j ++) {
                // a |= b 的意思是a与b按位或，然后给a。另外word只需要看有那几个字母，不需要看字母的个数
                preProcessed[i] |= 1 << (words[i].charAt(j) - 'a');
            }
        }
        int maxL = 0;
        for(int i = 0; i < words.length; i ++) {
            for(int j = i + 1; j < words.length; j ++) {
                if((preProcessed[i] & preProcessed[j]) == 0) {
                    int product = words[i].length() * words[j].length();
                    if(maxL < product) {
                        maxL = product;
                    }
                }
            }
        }
        return maxL;
    }
}
```