Shortest Word Distance III

This is a **follow up** of Shortest Word Distance. The only difference is now *word1* could be the same as *word2*.

Given a list of words and two words *word1* and *word2*, return the shortest distance between these two words in the list.

word1 and *word2* may be the same and they represent two individual words in the list.

For example,
Assume that words = `["practice", "makes", "perfect", "coding", "makes"]`.

Given *word1* = `“makes”`, *word2* = `“coding”`, return 1.
Given *word1* = `"makes"`, *word2* = `"makes"`, return 3.

Note:
You may assume *word1* and *word2* are both in the list.

Solution 1

Solution 1 ... Java "short"

`i1` and `i2` are the indexes where word1 and word2 were last seen. Except if they're the same word, then `i1` is the previous index.

```
public int shortestWordDistance(String[] words, String word1, String word2) {
    long dist = Integer.MAX_VALUE, i1 = dist, i2 = -dist;
    for (int i=0; i<words.length; i++) {
        if (words[i].equals(word1))
            i1 = i;
        if (words[i].equals(word2)) {
            if (word1.equals(word2))
                i1 = i2;
            i2 = i;
        }
        dist = Math.min(dist, Math.abs(i1 - i2));
    }
    return (int) dist;
}
```

Solution 2 ... Java "fast"

Same as solution 1, but minimizing the number of string comparisons.

```
public int shortestWordDistance(String[] words, String word1, String word2) {
    long dist = Integer.MAX_VALUE, i1 = dist, i2 = -dist;
    boolean same = word1.equals(word2);
    for (int i=0; i<words.length; i++) {
        if (words[i].equals(word1)) {
            if (same) {
                i1 = i2;
                i2 = i;
            } else {
                i1 = i;
            }
        } else if (words[i].equals(word2)) {
            i2 = i;
        }
        dist = Math.min(dist, Math.abs(i1 - i2));
    }
    return (int) dist;
}
```

Solution 3 ... C++ "short"

C++ version of solution 1.

```
int shortestWordDistance(vector<string>& words, string word1, string word2) {
    long long dist = INT_MAX, i1 = dist, i2 = -dist;
    for (int i=0; i<words.size(); i++) {
        if (words[i] == word1)
            i1 = i;
        if (words[i] == word2) {
            if (word1 == word2)
                i1 = i2;
            i2 = i;
        }
        dist = min(dist, abs(i1 - i2));
    }
    return dist;
}
```

Solution 4 ... C++ "fast"

C++ version of solution 2.

```
int shortestWordDistance(vector<string>& words, string word1, string word2) {
    long long dist = INT_MAX, i1 = dist, i2 = -dist;
    bool same = word1 == word2;
    for (int i=0; i<words.size(); i++) {
        if (words[i] == word1) {
            i1 = i;
            if (same)
                swap(i1, i2);
        } else if (words[i] == word2) {
            i2 = i;
        }
        dist = min(dist, abs(i1 - i2));
    }
    return dist;
}
```

written by StefanPochmann original link here

Solution 2

```
public class Solution {
    public int shortestWordDistance(String[] words, String word1, String word2) {
        int p1 = -1;
        int p2 = -1;
        int min = Integer.MAX_VALUE;
        for (int i = 0; i < words.length; i++) {
            if (words[i].equals(word1)) {
                if (word1.equals(word2)) {
                    if (p1 != -1 && i - p1 < min) {
                        min = i - p1;
                    }
                    p1 = i;
                } else {
                    p1 = i;
                    if (p2 != -1 && p1 - p2 < min) {
                        min = p1 - p2;
                    }
                }
            } else if (words[i].equals(word2)) {
                p2 = i;
                if (p1 != -1 && p2 - p1 < min) {
                    min = p2 - p1;
                }
            }
        }
        return min;
    }
}
```

written by stevenye original link here

Solution 3

Shortest Word Distance I:

```
public class Solution {
    public int shortestWordDistance(String[] words, String word1, String word2) {
        int index = -1;
        int min = words.length;
        for (int i = 0; i < words.length; i++) {
            if (words[i].equals(word1) || words[i].equals(word2)) {
                if (index != -1 && !words[index].equals(words[i])) {
                    min = Math.min(i - index, min);
                }
                index = i;
            }
        }
        return min;
    }
}
```

Shortest Word Distance III:

```
public class Solution {
        public int shortestWordDistance(String[] words, String word1, String word
2) {
            int index = -1;
            int min = words.length;
            for (int i = 0; i < words.length; i++) {
                if (words[i].equals(word1) || words[i].equals(word2)) {
                    if (index != -1 && (word1.equals(word2) || !words[index].equa
ls(words[i]))) {
                        min = Math.min(i - index, min);
                    }
                    index = i;
                }
            }
            return min;
        }
    }
```

written by blackminimi original link here