267. Palindrome Permutation II

Medium 640 80 Add to List Share

Given a string s, return *all the palindromic permutations (without duplicates) of it*.

You may return the answer in **any order**. If `s` has no palindromic permutation, return an empty list.

Example 1:

```
Input: s = "aabb"
Output: ["abba","baab"]
```

Example 2:

```
Input: s = "abc"
Output: []
```

Constraints:

- `1 <= s.length <= 16`
- `s` consists of only lowercase English letters.

Accepted 49,597 | Submissions 127,934

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Facebook 2

Related Topics

Hash Table String Backtracking

Similar Questions

Next Permutation Medium

Permutations II Medium

Palindrome Permutation Easy

Hide Hint 1

If a palindromic permutation exists, we just need to generate the first half of the string.

Hide Hint 2

To generate all distinct permutations of a (half of) string, use a similar approach from: Permutations II or Next Permutation.

Java Autocomplete

```java
public class Solution {
    Set < String > set = new HashSet < > ();
    public List < String > generatePalindromes(String s) {
        int[] map = new int[128];
        char[] st = new char[s.length() / 2];
        if (!canPermutePalindrome(s, map))
            return new ArrayList < > ();
        char ch = 0;
        int k = 0;
        for (int i = 0; i < map.length; i++) {
            if (map[i] % 2 == 1)
                ch = (char) i;
            for (int j = 0; j < map[i] / 2; j++) {
                st[k++] = (char) i;
            }
        }
        permute(st, 0, ch);
        return new ArrayList < String > (set);
    }
    public boolean canPermutePalindrome(String s, int[] map) {
        int count = 0;
        for (int i = 0; i < s.length(); i++) {
            map[s.charAt(i)]++;
            if (map[s.charAt(i)] % 2 == 0)
                count--;
            else
                count++;
        }
        return count <= 1;
    }
    public void swap(char[] s, int i, int j) {
        char temp = s[i];
        s[i] = s[j];
        s[j] = temp;
    }
    void permute(char[] s, int l, char ch) {
        if (l == s.length) {
            set.add(new String(s) + (ch == 0 ? "" : ch) + new StringBuffer(new String(s)).reverse());
        } else {
            for (int i = l; i < s.length; i++) {
                if (s[l] != s[i] || l == i) {
                    swap(s, l, i);
                    permute(s, l + 1, ch);
                    swap(s, l, i);
                }
            }
        }
    }
}
```