884. Uncommon Words from Two Sentences

A **sentence** is a string of single-space separated words where each word consists only of lowercase letters.

A word is **uncommon** if it appears exactly once in one of the sentences, and **does not appear** in the other sentence.

Given two **sentences** `s1` and `s2`, return *a list of all the* ***uncommon words***. You may return the answer in **any order**.

Example 1:

Input: s1 = "this apple is sweet", s2 = "this apple is sour"

Output: ["sweet","sour"]

Explanation:

The word `"sweet"` appears only in `s1`, while the word `"sour"` appears only in `s2`.

Example 2:

Input: s1 = "apple apple", s2 = "banana"

Output: ["banana"]

Constraints:

- `1 <= s1.length, s2.length <= 200`
- `s1` and `s2` consist of lowercase English letters and spaces.
- `s1` and `s2` do not have leading or trailing spaces.
- All the words in `s1` and `s2` are separated by a single space.

884. Uncommon Words from Two Sentences

```
/*
   Custom algo to split words + map
   Time complexity: O(nlog w1 +m log (w1+w2)+log (w1+w2))
   Space complexity: O(w1+w2)
   n: size of sentence 1
   m: size of sentence 2
   w1: #words in sentence 1
   w2: #words in sentence 2
*/
class Solution {
   public:
      void count(std::string& s,std::map<std::string,int>& freq){
         s+=" ";
         int n=s.size();
         std::string word="";
         for(auto& letter: s){
            if(letter!=' ') word+=letter;
            else{
               freq[word]++;
               word="";
            }
         }
      }
      std::vector<std::string> uncommonFromSentences(std::string s1, std::string s2) {
         std::map<std::string,int> freq;
         count(s1,freq);
         count(s2,freq);
         std::vector<std::string> ans;
         for(auto& [word,f]: freq){
            if(f==1) ans.push_back(word);
         }
         return ans;
      }
};
```

884. Uncommon Words from Two Sentences

```
/*
   stringsteam to split words + map
   Time complexity: O(w1 log w1 +w2 log (w1+w2)+log (w1+w2))
   Space complexity: O(w1+w2)
   n: size of sentence 1
   m: size of sentence 2
   w1: #words in sentence 1
   w2: #words in sentence 2
*/
class Solution {
   public:
      void count(std::string& s,std::unordered_map<std::string,int>& freq){
         std::stringstream ss(s);
         int n=s.size();
         while(!ss.eof()){
            std::string word;
            ss>>word;
            freq[word]++;
         }
      }
      std::vector<std::string> uncommonFromSentences(std::string s1, std::string s2) {
         std::unordered_map<std::string,int> freq;
         count(s1,freq);
         count(s2,freq);
         std::vector<std::string> ans;
         for(auto& [word,f]: freq){
            if(f==1) ans.push_back(word);
         }
         return ans;
      }
};
```