Find All Anagrams in a List

Problem Statement:

Given a string `s` and a list of strings `words`, return all the strings from the list that are anagrams of the given string `s`. Two strings are considered anagrams if they contain the same characters with the same frequency, but possibly in a different order.

An **anagram** is a word or phrase formed by rearranging the letters of another, using all the original letters exactly once.
Example of anagram: `"listen"` and `"silent"` are anagram because both the words use the same letters: `l,i,s,t,e,n`.

Constraints:

- `1 <= len(s) <= 100`
- `1 <= len(words) <= 10^4`
- `1 <= len(words[i]) <= 100`
- Both `s` and `words[i]` consist of lowercase English letters only.

Example 1:

Input:

```
s = "listen"
words = ["enlists", "google", "inlets", "banana"]
```

Output:

```
["inlets"]
```

Explanation
The string `"inlets"` is an anagram of "listen". The other strings do not match the character frequency of `"listen"`.

Example 2:

Input:

```
s = "rat"
```

```
words = ["art", "tar", "rat", "car"]
```

Output:

```
["art", "tar", "rat"]
```

Explanation
All three strings `"art"`, `"tar"`, and `"rat"` are anagrams of `"rat"`.

Example 3:

Input:

```
s = "a"
words = ["a", "b", "c", "d"]
```

Output:

```
["a"]
```

Explanation:
Only `"a"` is an anagram of `"a"`.

Hint:

1. Think about how you can count the frequency of characters in both the given string `s` and each word in the list `words`. Comparing these frequencies can help identify anagrams efficiently.
2. One approach might be sorting the strings, and comparing them, but can you think of more efficient approach?