DS2030 Data Structures and Algorithms for Data Science
Week-7 Practice Set

October 07th, 2025

1 Valid Anagram

Problem Statement

You are tasked with designing a feature for IIT Palakkad's plagiarism checker system that detects whether two given strings are anagrams. Two strings are anagrams if they contain the same characters with the same frequencies.

Tasks

Implement your own hash map for character counts using **linear probing**.

- Design a hash function for characters.
- Resolve collisions using linear probing.
- Use this hash table to store and compare character frequencies.

Implement the **Valid Anagram** functionality with the following requirements:

1. **Input**: Two strings `s` and `t`.
2. **Output**: Return `true` if `t` is an anagram of `s`, otherwise return `false`.
3. **Method:**
    - **Use a Hash Map:** Count the frequency of each character in string `s`. Decrease the counts while scanning string `t`. If all counts return to zero, then the strings are anagrams.
4. **Example:**
    - Input: `s = "anagram", t = "nagaram"`
    - Output: `true`
    - Input: `s = "rat", t = "car"`
    - Output: `false`
5. **Constraints:** Assume the strings consist of lowercase English letters.

2 Group Anagrams

Problem Statement

You are developing a **String Categorization System** for IIT Palakkad's digital library. The system should efficiently group words that are anagrams of each other. Two words are considered anagrams if they can be rearranged to form one another.
Given a list of words, your task is to group them into separate categories of anagrams.

Tasks

Implement your own hash map for character counts using **linear probing**.

- Define a hash function for string signatures.
- Handle collisions using linear probing.
- Store groups in this custom hash table.

Implement the **Group Anagrams** functionality with the following requirements:

1. **Input**: A list of strings `words = [w_1, w_2, ..., w_n]`.
2. **Output**: A list of groups, where each group contains words that are anagrams of each other.
3. **Method:**
    - **Use a Hash Map:** The key should be a unique signature for each anagram group (e.g., sorted string or character frequency vector). The value should be a list of words belonging to that group.
    - Insert each word into the corresponding group based on its key.
4. **Example:**
    - Input: `["eat", "tea", "tan", "ate", "nat", "bat"]`
    - Output: `[["eat","tea","ate"], ["tan","nat"], ["bat"]]`
5. **Constraints:** Assume the strings consist of lowercase English letters.
6. Ensure that the solution works efficiently for large word lists.