02_Strings

Counting Characters in a String

String of unique characters

A neat trick to count the characters in a string of unique characters is to use a 26-bit bitmask to indicate which lower case latin characters are inside the string.

```
mask = 0
for c in word:
  mask |= (1 << (ord(c) - ord('a')))
```


To determine if two strings have common characters, perform `&` on the two bitmasks. If the result is non-zero, ie. `mask_a & mask_b > 0`, then the two strings have common characters.

String of Unique Characters

Counting characters

Often you will need to count the frequency of characters in a string. The most common way of doing that is by using a hash table/map in your language of choice.

If you need to keep a counter of characters, a common mistake is to say that the space complexity required for the counter is O(n). The space required for a counter of a string of latin characters is O(1) not O(n). This is because the upper bound is the range of characters, which is usually a fixed constant of 26. The input set is just lowercase Latin characters.

Anagram

An anagram is word switch or word play. It is the result of rearranging the letters of a word or phrase to produce a new word or phrase, while using all the original letters only once.

In interviews, usually we are only bothered with words without spaces in them.

To determine if two strings are anagrams, there are a few approaches:

- Sorting both strings should produce the same resulting string.
  - This takes O(n.log(n)) time and O(log(n)) space.
- If we map each character to a prime number and we multiply each mapped number together, anagrams should have the same multiple (prime factor decomposition).
  - This takes O(n) time and O(1) space. Examples: Group Anagram
- Frequency counting of characters will help to determine if two strings are anagrams.
  - This also takes O(n) time and O(1) space.

Palindrome

Palindrome

A palindrome is a word, phrase, number, or other sequence of characters which reads the same backward as forward, such as `madam` or `racecar`.

Here are ways to determine if a string is a palindrome:

- Reverse the string and it should be equal to itself.
- Have two pointers at the start and end of the string. Move the pointers inward till they meet. At every point in time, the characters at both pointers should match.

The order of characters within the string matters, so hash tables are usually not helpful.

When a question is about counting the number of palindromes, a common trick is to have two pointers that move outward, away from the middle.

- Note that palindromes can be even or odd length. For each middle pivot position, you need to check it twice - once that includes the character and once without the character.
- For substrings, you can terminate early once there is no match