Interview questions

1. Write a program that checks whether the given two strings are anagrams or not

An anagram is direct word switch or word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once

cinema iceman

William Shakespeare I am a weakish speller

Madam Curie Radium came

Input: str1, str2

- all punctuation symbols and spaces are ignored
- no difference between lower/upper cases

Interview questions

1. Write a program that checks whether the given two strings are anagrams or not

cinema iceman

Input: str1, str2

An idea:
- convert all letters to lower case
- alphabetically sort both strings (use **sorted** method)
- compare letters ignoring punctuation symbols
  as soon as two different letters are encountered, we stop.

Try it now!

Interview questions

1. Write a program that checks whether the given two strings are anagrams or not

cinema iceman

Input: str1, str2

An idea:
- convert all letters to lower case
- alphabetically sort both strings (use **sorted** method)
- compare letters ignoring punctuation symbols
  as soon as two different letters are encountered, we stop.

Try it now!

See my code: anagramCheck.py

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Ideas:

needs additional data structure
Time: n^2

- recall list comprehension (unique elements)
  we can create a string with unique elements from the original one and then compare their lengths.

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Ideas:

needs additional data structure
Time: n^2

- recall list comprehension (unique elements)
  we can create a string with unique elements from the original one and then compare their lengths.

- sort the string alphabetically (method **sorted**);
  this way same characters will be together!

needs additional data structure
Time: nlogn at best

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Ideas:

- write our own sorting procedure that will sort in place!

doesn't needs additional data structure
Time: nlogn

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Ideas:

- write our own sorting procedure that will sort in place!

doesn't needs additional data structure
Time: nlogn

- use a binary array (takes a "little" space) of size 26 filled with 0's initially.

slot 0: for *a*, … slot 25 for *z*.
If we meet a letter, we change corresponding 0 to 1, but if the slot already has 1, it means that we already met that letter! Stop.

needs additional data structure (small size)
Time: n

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Let's use the last idea.

- binary array (takes a “little” space) of size 26, filled with 0's initially.

slot 0: for *a*, … slot 25 for *z*.

If we meet a letter, we change corresponding 0 to 1, but if the slot already has 1, it means that we already met that letter! Stop.

needs additional data structure (small size)
Time: n

Interview questions

2. Implement an algorithm to determine if a string has all unique characters.
What if you cannot use any additional data structures?

Assumption: string has only letters from the alphabet.

Let's use the last idea.

- binary array (takes a “little” space) of size 26, filled with 0's initially.

slot 0: for *a*, … slot 25 for *z*.

If we meet a letter, we change corresponding 0 to 1, but if the slot already has 1, it means that we already met that letter! Stop.

needs additional data structure (small size)
Time: n

See my code: checkUnique.py

Interview questions

3. Given a phrase that contains letters of the alphabet and spaces only rearrange it so that all letters are alphabetically ordered, and the spaces are in the original places.

Example: Hello Mark → aeHkl lMor

Interview questions

3. Given a phrase that contains letters of the alphabet and spaces only rearrange it so that all letters are alphabetically ordered, and the spaces are in the original places.

Example: Hello Mark → aeHkl lMor

An idea:
- split the string by spaces ["Hello","Mark"] O(n)
- count the number of characters in each word [5,4] O(n)

Interview questions

3. Given a phrase that contains letters of the alphabet and spaces only rearrange it so that all letters are alphabetically ordered, and the spaces are in the original places.

Example: Hello Mark → aeHkl lMor

An idea:
- split the string by spaces ["Hello","Mark"] O(n)
- count the number of characters in each word [5,4] O(n)
- use the original string to order the elements alphabetically
  _ _ a e H k l l M o r O(n logn)

Interview questions

3. Given a phrase that contains letters of the alphabet and spaces only rearrange it so that all letters are alphabetically ordered, and the spaces are in the original places.

Example: Hello Mark → aeHkl lMor

An idea:
- split the string by spaces ["Hello","Mark"] O(n)
- count the number of characters in each word [5,4] O(n)
- use the original string to order the elements alphabetically
  _ _ a e H k l l M o r O(n logn)
- ignore the preceding spaces (you can use length of list/array of lengths) and start populating a new string with letters, using the lengths of the original words ([5,4]) O(n)

_ _ _ _ _ _ _ _ _

Interview questions

3. Given a phrase that contains letters of the alphabet and spaces only rearrange it so that all letters are alphabetically ordered, and the spaces are in the original places.

Example: Hello Mark → aeHkl lMor

An idea:
- split the string by spaces ["Hello","Mark"] O(n)
- count the number of characters in each word [5,4] O(n)
- use the original string to order the elements alphabetically
  _ _ a e H k l l M o r O(n logn)
- ignore the preceding spaces (you can use length of list/array of lengths) and start populating a new string with letters, using the lengths of the original words ([5,4]) O(n)

See my code: reArrange.py