AlphabetSoup-Using-Django

Everyone loves alphabet soup. And of course, you want to know if you can construct a message from the letters found in your bowl.

```
Task:
```

Write a function that takes as input two strings: the message you want to write all the letters found in your bowl of alphabet soup.

```
Assumptions:
```

It may be a very large bowl of soup containing many letters. There is no guarantee that each letter occurs a similar number of times - indeed some letters might be missing entirely. The letters are ordered randomly.

The function should determine if you can write your message with the letters found in your bowl of soup. The function should return True or False accordingly.

Try to make your function efficient. Please use Big-O notation to explain how long it takes your function to run in terms of the length of your message (m) and the number of letters in your bowl of soup (s).

Solution is Contained inside the Script folder

How to Use

```
git clone https://github.com/kennedyCzar/AlphabetSoup-Using-Django

Unpack the zip file into any dirctory of choice on your local drive.

You can run the script file directly from the testBB.py file.

To use the Django application
------------------------------------------
>conda install django
OR
>pip install django

>Open anaconda propmt
Navigate to the dirctory where you unpacked the zipped file

>Enter into the Djnago project folder
>cd /DJANGO/AlphabetSoup
>dir
(This command checks the file in the AlphabetSoup directory.
Ensure you are in the folder that contains the manage.py file)
```

```
then run the command
>python manage.py runserver
(This should start django on the localhost and ip 8000 or 8050
depending on which is open.) OR specify your own ip and port
>python manage.py runserver ip:port
>Enter the ip:port in any browser of choice(preferably chrome)
```

Algorithm core

```
self.listA = list(self.message)
self.listB = list(self.alphabet)
self.final = []
self.start_time = time.clock()
while self.listA != []:
    for msg in self.listA:
        if msg in self.listB:
            self.final.append(msg)
            self.listA.remove(msg)
            self.listB.remove(msg)
        elif msg not in self.listB:
            self.listA.remove(msg)
    continue
if sorted(self.final) == sorted(list(self.message)):
    return True, self.final, self.listA, self.listB, time.clock() - self.start_time
else:
    return False, self.final, self.listA, self.listB, time.clock() -
self.start_time
```

Mathematical background

The intuition behind this algorithm is quite straight forward. 1. We convert our message into a list for we we wan easily iterate 2. Do the same for the alphabet soup and ensure they are structured

We then create a while loop for which: if A <-- Alphabet & B is message we iterate over the range of B then we do same for A if the element of B can be found in A Then we can form a message using this we repeat this process and store the content in another list we call final finally we ckeck the final list with the message list if this condition satisfies the we return a boolean True otherwise we return a False

Advantage of algorithm

```
1. Time Efficient: Could be time consuming also considering it loops in N
2. Time Complexity: O(N**2) for Worst Case
3. Space complexity: O(1) space efficient.
4. Running in O(N**2) is bar far not the most efficient but it gets the job don
however the inputs come. I had earlier demonstrated
    how the algorithm can run in O(M) O(N), this however wasnt the best solution as
some inputs exhibited strange behaviours.
```

```
 See Tim Wilson's sort algorithm for further Read
https://en.wikipedia.org/wiki/Timsort
 ```

 ## DjangoApp for AlphabetSoup

 ###### Before input
 ![Image of Django
App](https://github.com/kennedyCzar/AlphabetSoup-Using-Django/blob/master/IMAGES/dj
angoApp.PNG)

  ###### True Output
  ![Image of Django
App](https://github.com/kennedyCzar/AlphabetSoup-Using-Django/blob/master/IMAGES/tr
ue%20output.PNG)
  ###### False output
  ![Image of Django
App](https://github.com/kennedyCzar/AlphabetSoup-Using-Django/blob/master/IMAGES/fa
lse%20output.PNG)

Final words
```

The AlphaSoup algorithm is sort-like algorithm which is capable of telling if a message can me form from a wordcloud or soup. In addition to this i have demostrated how you can use this in a django application. It would be great to have a database but is for dmonstration purpose and a database is not required at this point. Perhaps you could consider a database when hosting and trying to check users. Again this is license opensource and contributions to improve the algoritm are welcomed.
```

contributions welcom