Technical Test

Task Description

“Please write a program to count the number of instances of each word in a text file. The solution should list the top 10 words along with the number of occurrences.“

The ‘example’ provided is a large block of text from The Lord of The Rings - The Fellowship Of The Ring.

Analysis

Breaking down this ‘story’ after a discussion …

- *‘a program’* - implies a simple piece of code, something executable.
- *‘count each word’* - this seems like solutionizing early by ‘the business’ and describes a possible implementation. This is open to options.
- *‘text file’* - this can be a legitimate technical constraint of the business and should be considered a requirement but should be checked.
- *‘list the top 10 words with the number of occurrences’* - this is the outcome, the value this story will provide. Output of result is expected.

The Story

Rewording this into a story

- In order to generate revenue based on words counts in documents
- As a ‘customer’
- I want know the top 10 words and their number of occurrences

Acceptance criteria

The correct list of top 10 words provided with counts of occurrences.

The Behaviour

Describing the story in terms of behaviours

- Given a text document
- And this document is large
- When we process ‘words’ of this document
- Then the top 10 words are listed
- And the number of occurrences are shown with each word

more ...

Assumptions

A initial list of assumptions (ideally to be checked or discovered during TDD implementation) are …

- From the phone interview and job specification
    - Use of Api’s
    - .NET Core
    - Language is C#
    - Performant
        - Consider stream processing file/url and processing in parallel.
        - The file could be chunked up or processed line by line.
    - Readable/Maintainable etc.
- Domain knowledge of problem
    - The provided text is known and the words will be mainly English words even through the author used his own made up spoken languages these will be fewer in number.
    - Words can include hyphens, accents and be surrounded by punctuation. Numbers can also be seen in text. These will all be ‘initially’ assumed to be fewer than most words which will be whole unbroken words without any special characters. Later this can be improved upon.
- Technical
    - A large text file should be able to fit into memory nowadays but what if this is a service that processes many so streaming should be considered.
    - Identify the encoding and parse accordingly
    - Future consideration could be the language used in the document or simply consider all forms of ‘letter’ characters.
    - Whitespace should be reduced to single space if preprocessing or stepping through.
    - Preprocessing could make the processing take longer so could consider iterating through string.
    - When checking a word is a word the code should escape early in the presence of a non letter character in preference to a positive assertion that all characters in the word are letters.
    - Assuming that most common words will be shorter in length so could try sorting or grouping by word length. This may also help with performance because the combination of string length check and matching may be quicker than matching alone. So we could consider partitioning in the solution.
    - There are a number of ways to process this and we could build in a heuristic approach to pick the winning algorithms.
    - Reading from a stream should be considered instead of in memory processing.

more ...

Approach

The steps I will likely take to achieve this (and potentially estimable)

1. Iterative to release value early (and show workings). Git (or separate stages in folders)
2. Benchmark progress by recording timings.
3. KISS (keep it simple stupid or stupid stupid simple)
4. Serial approach
5. Parallel approach
6. In memory approach
7. Streaming approach
8. Early real-time feedback to user.
    a. It could be possible to provide early feedback to user of the top 10 by keeping a separate enumerable of the top 10 and sorting, replacing the last entry if a new higher number appears.

End.
Jeremy Byford-Rew 18 April 2019. jeremybyfordrew@gmail.com