UNIVERSITY OF LEICESTER

CO3090 CW1

Module code:	**CO3090**
Module name:	**Module name**

24 February 2019

Author(s):	**Nishat Tarannum**
Student ID(s):	**169018358**
Degree:	**MEng Software and Electronic Engineering**
Module Convenor:	**Dr. Yi Hong**

SUPERVISOR'S COPY/EXAMINER'S COPY [For 3rd/4th year projects only - delete as appropriate]

1 Task 1

1.1 Explain why a multi-threaded letter frequency counter has better performance than the single-threaded version.

A multithreaded letter frequency counter aims to increase utilization of multiple cores. Multithreading can speed up the counting because each thread can work concurrently on a different core, parallelly running the program on webpages, thus making the process finish twice as fast or even faster. Hence, there is a greater program responsiveness with multithreading because it allows a program to run even if part of it is blocked. On the other hand, multithreading on a single core may lead to the process taking longer depending on the scheduling policy (as there is overhead to context switching between threads) [1].

A single threaded version would take longer because the process of counting the letters could not be divided and allocated to separate threads, thus only one core would execute the process.

Creating separate processes would be more expensive than using multithreading because it would require more memory and resources, hence a greater overhead than for managing threads as they share process resources (includes memory and data etc). Different threads running an application can share the same address space due to sharing resources [1].

1.2 Explain the strategy you used for your implementation and justify your choice.

After researching both algorithms of Depth First Search (DFS) and Breadth First Search (BFS), it was concluded that the DFS algorithm would not be the best approach as it would iterate as far as possible (in terms of tree traversal) and this could take longer especially for a web crawler.

In terms of tree traversal, for BSF, nodes are searched at each level from the root node. After that, the search moves to the children and then children's children etc until the search finds the nodes that meet the condition. However, DFS goes through all the nodes that are on the leftmost path of the tree from the root to the leaf and then moves to the next left most path going through all the nodes downwards. As shown below in the diagram [3].

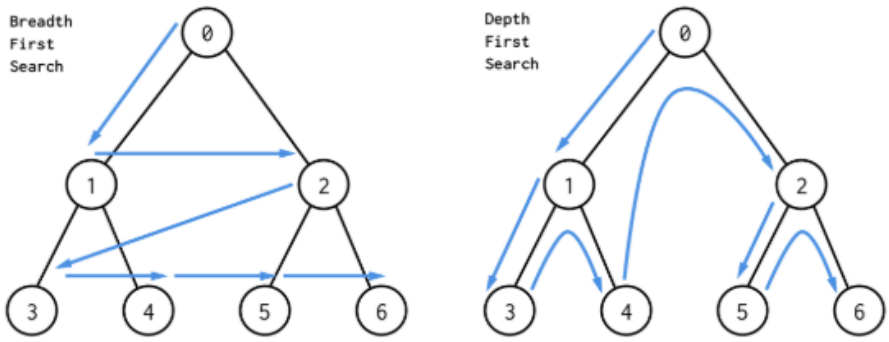

Figure 1 - Diagram showing how BSF and DFS are implemented in a binary tree[3]

A website generally places hyperlinks in the first few layers, thus it isn't of much depth, hence using a DFS search for web crawling would mean traversing through the webpage for longer through things that are not necessary in finding the links. Thus, it would be time consuming. Using a DFS implementation may however be useful if it was for a website like Wikipedia, where there are links for references on every page [2].

1.3 How many threads will be started according to your implementation?

According to the code that has been written, whatever the value of the constant MAX_THREAD_NUM is, that will be the number of threads that will be started.

2 References

1. Tutorialspoint.com. (2018). *The Benefits of Multithreaded Programming*. [online] Available at: https://www.tutorialspoint.com/the-benefits-of-multithreaded-programming [Accessed 18 Feb. 2019].
2. Quora.com. (2016). *Which traversal is better for Web Crawling? DFS or BFS and Why? - Quora*. [online] Available at: https://www.quora.com/Which-traversal-is-better-for-Web-Crawling-DFS-or-BFS-and-Why [Accessed 18 Feb. 2019].
3. Mishadoff.com. (2017). *DFS on Binary Tree Array*. [online] Available at: http://mishadoff.com/blog/dfs-on-binary-tree-array/ [Accessed 23 Feb. 2019].