2020-21
School of Computing, Engineering and Mathematics

5003CEM
Advanced Algorithms

Assessed Advanced Task 3/3

This Week's Task

This week, **there is one advanced assessed task (3/3)**. Between Weeks 4 and 9, over 6 weeks, you will be set 5 standard tasks and 3 advanced tasks, which need to be submitted, along with a report, on 9 April 2021 at 18:00.

Here are the **assessed tasks so far:**

Week 4	Assessed standard task 1	implement selection sort
Week 5	Assessed standard task 2	implement BST search
	Assessed advanced task 1	implement BST remove node method
Week 6	Assessed standard task 3	implement graph as adjacency matrix
Week 7	Assessed standard task 4	implement Prim's algorithm
	Assessed standard task 5	implement insert method for linked list (see below)
Week 8	Assessed advanced task 2	implement Dijkstra's algorithm (see below)
Week 9 (this week)	Assessed advanced task 3	Implement a concurrent headline scraper

So, this week sees the final assessed task. No further assessed tasks will be set.

It is an excellent idea to write up the tasks as you go and get feedback in the Lab sessions.

For more information, see COURSEWORK under the Assessment section on the Aula page.

1 Advanced Assessed Task 3/3: **Implement a concurrent headline scraper** **Advanced**

In the lecture this week there is an example of a concurrent application that reports the size of the data at different URLs.

If you look at the folder ADVANCED-TASK-3-CODE, you will find a program which goes to a set of URLs and gets the first 5 headlines back. However, it does not do this concurrently.

Your task this week is the 3rd and final advanced viva task. All 5 standard tasks have been set, so this is your final task.

The task is to implement a concurrent version of the code in ADVANCED-TASK-3-CODE, which should do the same thing, but faster.

To do this, you should continue to use concurrent.futures, as well as the Python newspaper module. The major work is in integrating these two things so that they work properly.

You should check that the headlines are being retrieved correctly (both number and content). NB, it does not matter if some headlines turn out to be a section heading or other non-news content (which can happen, depending on how the news site has been organised).

You should use timeit (there's an example in the code given) to compare and test the non-concurrent and concurrent versions. If the concurrent version is working properly, it should be faster than the non-concurrent version. The bigger the test number, the better the effect.

It may be useful to look at the documentation on concurrent.futures as well as newspaper.