University of Westminster
School of Computer Science and Engineering

Module code and title: 5SENG001W / Algorithms:Theory, Design and Implementation

Module leader	**Dr. Epaminondas Kapetanios**
Unit	**Coursework**
Coursework mode	**Individual coursework**
Weight	**50%**
Qualifying mark	**30 marks**
Description	**Optimising network flow constrained by maximum capacities (e.g., water or power supply, traffic network optimisation)**
Learning Outcomes	**LO2, LO3, LO4, LO5**
Handed Out:	**January, 31st, 2020**
Due Date of Coursework Submission	**Monday, 30th of March 2020, 1pm**
Expected deliverables	**Part I: Your implemented algorithm:** Your source code, as part of a zipped project or as text file, in Java, Python, or any other programming language should be submitted.Your source code shall include header comments with your student ID and name.You will be asked to demonstrate and explain your source code during the viva. The demonstrated source code must be identical with the one being submitted. A verification cross-check will be performed on a sampled set of submitted course works. **Note:**You will be zero-marked if you submit only the executable code, i.e., .class files, or source code without your student ID and name as header comments. **Part II: Performance analysis:** A report, no more than one A4 page document, in .doc, .docx, .docm, .odt, .txt, .rtf, .pdf file formats must be submitted as well. The report should discuss the performance analysis of your algorithmic solution.
Method of Submission:	**Electronically on BB**
Type of feedback and	**Feedback during in-class demonstration and vivas**
Due date of feedback	**Week commencing on the 30th of March, 2020**
Remark:	**All marks will remain provisional until formally agreed by an Assessment Board**
BCS Criteria meeting in this assignment:	**2.1.1 Knowledge and understanding of facts, concepts, principles & theories**

	2.1.3 Problem solving strategies **2.1.5 Deploy theory in design, implementation and evaluation of systems** **2.2.2 Evaluate systems in terms of quality and trade-offs** **2.3.2 Development of general transferable skills** **3.2.2 Defining problems, managing design process and evaluating outcomes** **4.1.1 Knowledge and understanding of scientific principles** **4.1.2 Knowledge and understanding of mathematical and statistical principles** **4.2.1 Use theoretical and practical methods in analysis and problem solving**

Coursework submission instructions

All coursework on this module is submitted via Blackboard only. It will automatically be scanned through a text matching system (designed to check for possible plagiarism). The system used will be either Turnitin or SafeAssign. The work you submit must be in .doc, .docx, .docm, .odt, .txt, .rtf, .pdf file formats.

To submit your assignment:

- Log on to Blackboard at http://learning.westminster.ac.uk;
- Go to the relevant module Blackboard site;
- Click on the Coursework submission link on the left-hand side as advised by the module leader;
- Click on the link to the relevant assignment;
- Follow the 'upload' and 'submit' instructions.

If you are unable to submit your work due to a finance hold you must email your work to **fst-registry@westminster.ac.uk** by the same deadline, putting on the subject line the module code, assessment number, and your name. This shows that you have completed your work by the deadline. After the finance hold is lifted you must then submit the same work as normal on Blackboard, otherwise it will not be marked and you will get a fail for that assessment.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:

http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

Assessment Rationale:

The coursework will enable students to meet the following learning outcomes:

LO2: Be able to apply the theory for the effective design and implementation of appropriate data structures and algorithms in order to resolve the problem at hand.

LO3: Be able to analyse, predict, compare and contrast the performance of designed and implemented algorithms, particularly in the context of processing data.

LO4: Be able to use a range of typical data structures and collections as part of Application Programming Interfaces (APIs) offered by programming languages.

LO5: Be able to apply the theory for the definition and implementation ofnovel algorithms.

Problem specification:

Max Flow Problem-

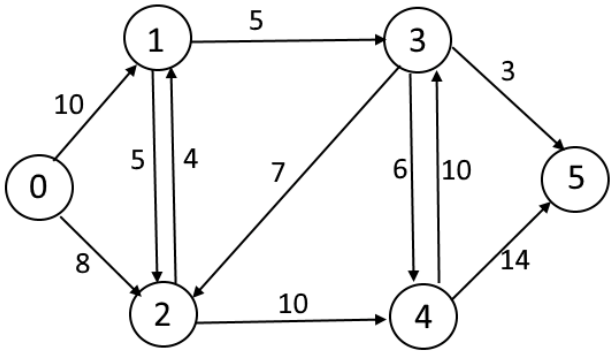
Maximum flow problems find a feasible flow through a single-source (S), single-sink (T) flow network that is maximum. This problem is useful for solving complex network flow problems such as the circulation problem.

Flow in the network has the following restrictions:

- Every link of the network has an individual capacity which is the maximum flow that link allows.
- Input flow must match to output flow for each node in the graph, except the source and sink node.
- Flow out from source node must match with the flow in to sink node.
- Flow from each edge should not exceed the capacity of that node.

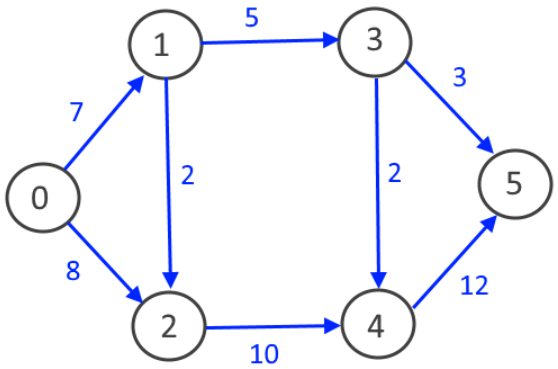
Your task: Design and implement an efficient algorithm to find the maximum flow possible from source **(S)** vertex to sink **(T)** vertex.

Example input:



Source : 0
Sink : 5

Example solution:



Maximum Flow - 15

source: 0
sink: 5

Your deliverables:

Part I: Source code and implementation: (70 marks)

Your demonstation of the implemented algorithm must comprise the following functionality:

1. A project (Java, Python, C++, etc.) has been set up by following the instructions as of the programming exercises in the tutorials.
2. A data structure, which is capable of representing a flow network, has been implemented.
3. A flow network can be initialized in terms of nodes and links with maximum capacity. The flow network must comprise at least 6 nodes, including source (S) and sink (T).
4. An algorithm has been implemented, which, upon successful execution of the code, it calculates
    a. the maximum flow possible between source (S) and sink (T),
    b. the path with the distributed flows as a justification of the maximum flow possible (see also example diagrammatic solution above).
5. Modifications:
    a. An arbitrarily chosen link can be deleted from the flow network resulting to the topology of the network being changed. Re-run the algorithm (task 4) and re-calculate its outcome (4.a and 4.b, respectively).
    b. The maximum capacity on an arbitrarily chosen link can be modified. Re-run the algorithm (task 4) and re-calculate its outcome (4.a and 4.b, respectively).
6. Scalability:
    a. Four different data sets can be imported by the implemented algorithm, each of which is double the size in terms of both nodes and links, from the previous one.
    b. Times can be measured and the ratio of changes in times being spent against all four data sets can be demonstrated. These should confirm the ratio of changes put forward in the report.

There is no need to include any GUI as part of your implementation. Demonstration with a multiple choice menu within a colsole or terminal is sufficient.

Part II: Your report about the analysis of the algorithmic performance (30 Marks)

Your report should be structured as follows:

a) Describe briefly the algorithmic approach taken in terms of (1) algorithmic strategy, (2) the chosen data structure and its traversal towards solution, (2) pseudo-code in plain English or diagrams.
b) Methodology for empirically analysing the performance of the algorithm in terms of (1) input data sizes feeding the algorithm, (2) actual measurements of times being spent to produce the outcome in accordance with, at least, four different input data sizes to be generated at wish and for the sake of this exploration.
c) Conclusions algorithmic performance in terms of Big-O *order-of-growth classifications* and justification based on (b).

Please submit your report as a separate document and NOT as part of your zipped implementation.

Coursework marking scheme:

An analytical marking scheme is provided by an Excel document and will be uploaded on BB as well.

IMPORTANT NOTE for failing to attend your viva: Marking the submitted knowledge graph will be weighted by a factor 0.3. For instance, if the quality of the submitted knowledge graph is graded 80%, your mark will be 80 * 0.3 = 24%. Hence, the maximum grade you may achieve will be 30%, the qualifying mark.

Grade	Indicative meaning
70-100 (First class)	The student demonstrated a very deep understanding of both the theory and the related practical material in the viva. (S)he is in a position to understand the problem specification and tailor data structures and algorithmic design at a high level, including analysis of algorithmic performance, which is fit for the problem solution purpose.
60-69 (Upper second class)	The student demonstrated a very good understanding of both the theory and the related practical material in the viva. (S)he is in a position to understand the problem specification and tailor data structures and algorithmic design, which may not be optimal, however, they do work out as problem solution. A basic understanding of algorithmic performance analysis analysis and justification of algorithmic design is also demonstrated.
50-59 (Lower second class)	The student demonstrated a good understanding of both the theory and the related practical material in the viva. S)he is in a position to understand the problem specification and tailor data structures and algorithmic design, which may not be optimal, however, they do work out as problem solution. A lack of understanding of algorithmic performance analysis and justification of algorithmic design is apparent.
40-49 (Third class)	The student demonstrated a basic understanding of both the theory and the related practical material in the viva. S)he is in a position to understand the problem specification and implement some data structures and algorithm, which may not be optimal, however, they do work out as an approximate problem solution. A lack of understanding of algorithmic performance analysis and justification of algorithmic design is apparent.
30-39	The student demonstrated some understanding of both the theory and the related practical material in the viva. S)he is NOT in a position to use effective data structures and design algorithms, which may work out as problem solution. A lack of understanding of algorithmic performance analysis and justification of algorithmic design is apparent.
2-29	The student could not demonstrate understanding of both the theory and the related practical material in the viva.
1	In cases of proven plagiarism to indicate attempt
0	Only in cases of non-submission or where only executable code has been submitted