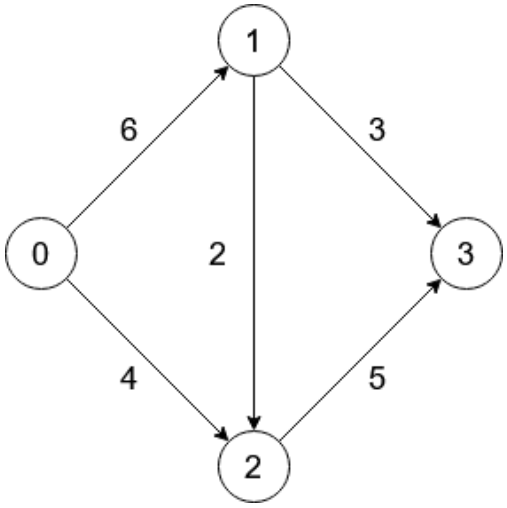
Coursework Description: Network Flow

One often uses graphs to model transportation networks—networks whose edges carry some sort of traffic and whose nodes act as "switches" passing traffic between different edges. Consider, for example, a highway system in which the edges are highways and the nodes are interchanges; or a computer network in which the edges are links that can carry packets and the nodes are switches; or a fluid network in which edges are pipes that carry liquid, and the nodes are junctures where pipes are plugged together.
Network models of this type are directed graphs with some additional ingredients:

(1) A **capacity** $\boldsymbol{c(e)}$ on each edge $\boldsymbol{e}$, indicating how much flow it can carry
(2) **Source** nodes in the graph, which can generate flow
(3) **Target** (aka sink or destination) nodes in the graph, which can absorb flow

An example of such a graph is given below. The nodes are numbered 0,1,2,3; node 0 is the source and node 3 the target. Each edge $\boldsymbol{e}$ is labelled with its capacity $\boldsymbol{c(e)}$.



Defining Flows

Suppose our network contains one source node s and one target node t. We say that a **flow from s to t** is a function $\boldsymbol{f}$ that maps each edge $\boldsymbol{e}$ to a nonnegative real number $\boldsymbol{f(e)}$**,** the amount of flow along edge $\boldsymbol{e}$. A flow $\boldsymbol{f}$ must satisfy two properties:

(1) **Capacity conditions:** For each edge $\boldsymbol{e} \in \boldsymbol{E}$, we have $\boldsymbol{0 \leq f(e) \leq c(e)}$.
(2) **Conservation conditions:** For each node v except s and t, we have $\sum_{\boldsymbol{e \in in(v)}} \boldsymbol{f(e)} = \sum_{\boldsymbol{e \in out(v)}} \boldsymbol{f(e)}$ where $\boldsymbol{in(v)}$ and $\boldsymbol{out(v)}$ are the sets in edges into and out of $\boldsymbol{v}$, respectively.

In other words, the flow on an edge cannot exceed its capacity. For every node except the source and the sink, the amounts of flow entering and leaving must be equal. The source has no entering edges but may have flow going out; in other words, it can generate flow. The target may have flow coming in but has no edges leaving it; it consumes flow.
The **value** of a flow $\boldsymbol{f}$ is defined to be the amount of flow generated at the source, which is the sum $\sum_{\boldsymbol{e \in out(s)}} \boldsymbol{f(e)}$. The goal of this coursework is to implement an algorithm for maximising this value, i.e. finding a **maximum** generated flow from s to t.
For the example above, a solution would be:
f(0,1) = 4, f(0,2) = 4, f(1,2) = 1, f(1,3) = 3, f(2,3) = 5 with a value of 8.

Tasks to be performed:

Task 1 (10 marks). Set up a project (Java or C++).
Task 2 (20 marks). Choose and implement a data structure which can represent a flow network. Assume that all capacities are integers, and you therefore only need to consider integer-valued flows. Explain the chosen data structure in the report accompanying your code.
Task 3 (20 marks). Add a simple parser which can read a description of a network from an input file. The structure of the files will look like the following example, representing the network in the above image (the comments are just added for clarification, they will not be in the actual input file and your parser does not have to be able to handle them):

```
4          // 4 nodes, numbered 0...3; node 0 is the source, node 3 the target.
0 1 6      // Edge from node 0 to node 1 with capacity 6
0 2 4      // Edge from node 0 to node 2 with capacity 4
1 2 2      // Edge from node 1 to node 2 with capacity 2
1 3 3      // Edge from node 1 to node 3 with capacity 3
2 3 5      // Edge from node 2 to node 3 with capacity 5
```

The first line contains the number **n** of nodes. Nodes are numbered starting from 0; node 0 is the source and node n-1 is the target. Each following line contains a triple of numbers **a b c** meaning that there is an edge from node a to node b with capacity c. There are no other edges than these given ones.
Your parser should be able to handle all input files which have this format. We will provide benchmark examples for your performance analysis, but make sure to also create some yourself to test your implementation.
Task 4 (20 marks). Choose and implement an algorithm which computes a maximum flow for a network and outputs it along with additional information explaining how it was obtained (such as incremental improvements for iterative algorithms like Ford-Fulkerson).
Task 5 (30 marks). Write a brief report (no more than 2 A4 pages) containing the following:

a) A short explanation of your choice of data structure and algorithm.
b) A run of your algorithm on the smallest benchmark example. This should include the supporting information as described in Task 4.
c) A performance analysis of your algorithmic design and implementation. This can be based either on an empirical study, e.g., doubling hypothesis, or on purely theoretical considerations, as discussed in the lectures and tutorials. It should include a suggested order-of-growth classification (Big-O notation).

To be submitted:

- Your zipped source code (for Tasks 1 to 4) in Java or C++. Make sure to include all .java or .h/.cpp files, **not** just an executable or .sln file. Your source code shall include header comments with your student ID and name.
- The report about the algorithmic performance analysis (Task 5).

Coursework marking scheme:

Criterion and range	Indicative mark	Comments
Task 1 (0-10 marks)	8-10	A compilable and executable project has been created and follows programming guidelines for writing clear such as https://introcs.cs.princeton.edu/java/11style
	4-7	A compilable and executable project has been created but does not follow programming guidelines such as of the *Java* related example above.
	0-3	A project has been created, but it is prone to compilation or runtime errors.
Task 2 (0-20 marks)	11-20	A data structure has been implemented, which satisfies the following principles of abstraction: - Builds on top of already existing programming language specific data structures, e.g., array. - Is equipped with basic data operations such as INSERT, DELETE and SEARCH. - Fits the purpose of the intended algorithm and nature of the problem.
	0-10	A data structure has been implemented but fails to follow the principles of abstraction as stated above.
Task 3 (0-20 marks)	11-20	A parser has been implemented, and is able to create a flow network from a given input file. It can handle any input file which has the format given in the problem description.
	0-10	A parser has been implemented, and is able to create flow networks from the benchmark examples provided with the coursework specification, but not others.
Task 4 (0-20 marks)	11-20	The correct maximum flow, i.e., no overloading of nodes or less than optimal flow through the network, can be calculated, for any given network. The implementation outputs enough information (such as improvement steps in incremental solutions) to fully justify the calculated maximum flow.
	6-10	The correct maximum flow can be calculated, but the implementation does not work for all possible networks, **or** does not provide enough information to justify its result.
	0-5	The correct maximum flow can be calculated, but the implementation does not work for all possible

		networks, **and** does not provide enough information to justify its result.
Task 5 (0-30 marks)	21-30	The student has submitted a full report explaining their solution and its algorithmic performance in accordance with the following principles: - A methodology has been put forward to justify the estimated asymptotic analysis; - The suggested order-of-growth classification in Big-O notation is fully justifiable from the methodological approach.
	11-20	The student has submitted a report discussing the algorithmic performance, but the methodological approach and suggested order-of-growth classification are not well justified and connected.
	0-10	The student has submitted a report suggesting a methodological approach for the algorithmic performance analysis, but the methodology has not been applied, and no order-of-growth classification has been suggested.