# Assignment 4 — Directed Graphs

**Due: October 30th**

In a directed graph, the existence of an edge from a vertex $u$ to a vertex $v$ does not imply the existence of an edge from $v$ back to $u$. As a result, finding an edge between two vertices does not necessarily mean that a path is free to traverse that edge; a path must also respect the edge's direction.

## Deliverables:

**GitHub Classroom:** `https://classroom.github.com/a/lKJK0oRO`

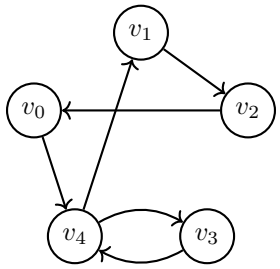
**Required Files:** `report.pdf`, `compile.sh`, `run.sh`

**Optional Files:** `*.c`, `*.h`, `*.py`, `*.java`, `*.js`, `*.json`, `*.ts`, `*.clj`, `*.kt`, `*.jl`, `*.rs`

## Part 1: Directed Eulerian Cycles

Recall that an *Eulerian cycle* is a cycle that traverses every edge exactly once. In an undirected graph, such a cycle exists if and only if every vertex has even degree. Intuitively, each vertex must have even degree so that half of its incident edges can be used to "arrive" at the vertex, and the other half, to "leave" it.

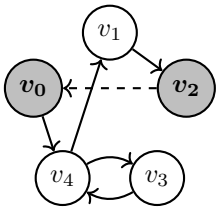
In a directed graph, however, only edges directed in to a vertex can be used to "arrive" at it, and only edges directed out from a vertex can be used to "leave" it. For example:



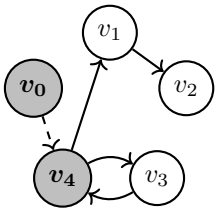

In the above directed graph, the path $(v_0, v_4, v_3, v_4, v_1, v_2, v_0)$ is one possible directed Eulerian cycle, traversing each of the 6 edges exactly once. Prove the following theorem:

> *Theorem:* Let $G = (V, E)$ be a strongly connected, directed graph. If every vertex's in-degree equals its out-degree, then there exists a directed Eulerian cycle.

To help you get started, note the following observations:




- An Eulerian cycle in a given graph is not necessarily composed of Eulerian cycles in subgraphs. However, all Eulerian cycles are also Eulerian paths, and any Eulerian path in any graph is composed of Eulerian paths in subgraphs.
- If a graph contains an Eulerian path but not an Eulerian cycle, then there exists exactly one vertex $s$ which the path "leaves" and cannot return to and exactly one vertex $t$ which the path "arrives" at and cannot leave.




- If a strongly connected, directed graph contains a directed Eulerian cycle $(s, \dots t, s)$, then removing the edge $(t, s) \in E$ creates a (potentially strongly, but certainly weakly) connected, directed graph containing a directed Eulerian path $(s, \dots t)$.
- If a weakly connected, directed graph contains a directed Eulerian path $(s, v, \dots t)$, then removing the edge $(s, v) \in E$ creates either one or two weakly connected components, both of which must have fewer edges.

## Part 2: Directed Eulerian Paths

In your programming language of choice (see Assignment 1), design and implement an algorithm to find Eulerian paths in directed graphs. Do not enumerate all possible paths — by basing your algorithm on your proof from Part 1, you should be able to do this with complexity $O(|E|)$.

Each input graph will be provided as an edge list: each edge in the graph will be represented by a comma-separated pair of vertex identifiers, indicating an edge directed from the first vertex to the second. You may assume that vertex identifiers are contiguous natural numbers — they begin at 0, and there will be no "gaps" in the identifiers used. You may further assume that the graph will be simple and weakly connected.

For example, the above graph could be represented as:

```
0, 4
1, 2
2, 0
3, 4
4, 1
4, 3
```

Your program must accept as a command line argument the name of a file containing an edge list as described above, then print to `stdout` a directed Eulerian path (if one exists) according to the following format:

- The path must appear as a single comma-separated line of vertices.
- If the path is also a cycle, it must begin (and, thus, end) at its lowest numbered vertex.

For example:

```
>$ ./compile.sh
>$ ./run.sh in1.txt
Directed Eulerian cycle:
0, 4, 3, 4, 1, 2, 0
```

You may further assume that there will exist exactly zero or one Eulerian paths, such that output as described above will be unique. Your program will be tested using `diff`, so its printed output must match *exactly*.

## Part 3: Submission

The following files are required and must be pushed to your GitHub Classroom repository by the deadline:

- `report.pdf` — Pseudocode for an efficient algorithm to find Eulerian paths in directed graphs, along with proof of the theorem in Part 1 and analysis of complexity for the pseudocode.
- `compile.sh` — A Bash script to compile your submission (even if it does nothing), as specified.
- `run.sh` — A Bash script to run your submission, as specified.

The following files are optional:

- `*.c`, `*.h`, `*.py`, `*.java`, `*.js`, `*.json`, `*.ts`, `*.clj`, `*.kt`, `*.rs` — The source code of a working program to find Eulerian paths in directed graphs, as specified.

Any files other than these will be ignored.