The following exercises have been made to be solved in a terminal on a Linux system, e.g. Ubuntu, Ubuntu for Windows Subsystem for Linux or similar. Make sure that **git, sha1sum, tar, wget** are installed.

1. If you want to download an Ubuntu disk image, the disto makers provide MD5, SHA1 and SHA256 checksums of the disk image (.iso) files, so that you can verify the integrity of your downloaded disk image. Go to http://releases.ubuntu.com/ and find the MD5, SHA1 and SHA256 checksums for the disk images of the latest version of Ubuntu (18.04.1 LTS).

2. Download crimeandpunishment.tar.gz from https://umuzi.gitlab.io/files/crimeandpunishment.tar.gz. This is a tar archive, which is similar to a zip-file. You can extract the files in it by running **tar xzvf crimeandpunishment.tar.gz** in the directory where you downloaded the file. Ten large text files will be extracted. All of them contain versions of Dostoyevski's Crime and Punishment which differ in tiny, miniscule ways. Some of the files are identical, some are not. Hash the files using e.g. **sha1sum**, **git hash-object** or **md5** and compare the hashes to find out which files are identical and which are different. Once you have solved the problem using one hashing algorithm, check the results by running one of the others as well.

3. Draw directed acyclic graphs which fulfil the following criteria:

- 3 nodes, 2 edges
- 3 nodes, 3 edges
- 4 nodes, 2 edges

4. Explain *why* all directed acyclic graphs can be topologically sorted. You can argue as follows:

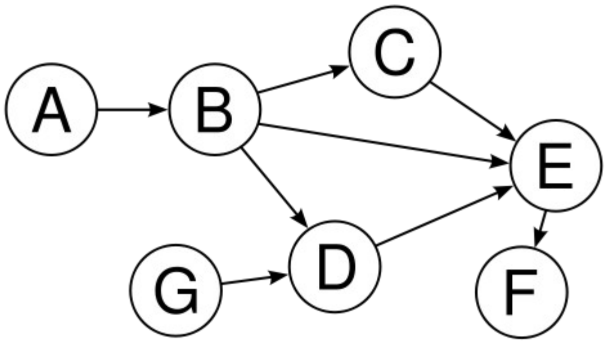
Let's say you have a directed acyclic graph on n nodes. We start with an empty list of nodes, which eventually will contain the nodes in topologically sorted order.

First we show that you can find a node without any ancestors: Pick a node. If it doesn't have any ancestors, you are done. Otherwise proceed to its ancestor. If this one has an ancestor, proceed there. Repeat. This process must terminate with a node without any ancestors in fewer than n steps, otherwise we've produced a cycle in the graph (why?).

When you have found a node without any ancestors, append it to the list. Then remove the node and all incident edges from the graph. Now repeat the process on the remaining graph.

Why does the list remain topologically sorted at every step of this procedure? Why will the procedure eventually produce a topological ordering of the entire graph?

5. Topologically sort the following directed acyclic graph. I.e. write out the names of the nodes in a list such that if x occurs before y in the list, x is an ancestor of y.



6. Open a git repository of your choice. Run **`git log --oneline --graph --all`**. Identify commits which

- Have no ancestors
- Have multiple parents (merge commits)