Project Feedback

Graph Theory 2015/2016

The following document provides some feedback on the project you recently completed for this module. In order to set a context for the feedback, first consider the following.

You should be aware that the standard required for submissions at level 7 (this year) is higher than at level 6 (first and second year). Significant effort is made to ensure that the standard is fair and consistent across third level institutes, both nationally and internationally. The standard we set for modules in computing is informed by Quality and Qualifications Ireland's Award Standard for Computing [2]. Below is a particularly relevant selection of the learning outcomes contained in that document.

Level 7 (Year 3)

The learner will be able to:

- *integrate concepts learned across a variety of subject areas.*
- *identify relevant material on a given topic from available information sources.*
- *succinctly present rational and reasoned arguments to a range of audiences.*
- *test and confirm the extent to which a computerbased system meets the criteria defined for its current use.*
- *develop innovative solutions to pragmatic situations.*
- *recognise the suitability of a given solution to a problem.*
- *apply knowledge learned in new situations.*

Level 6 (Years 1 and 2)

The learner will be able to:

- *describe best practices in computing.*

- *recognise and apply common best practices.*
- *apply knowledge in a practical setting under supervision.*
- *demonstrate the capacity to learn new knowledge and skills.*
- *use troubleshooting strategies and techniques in correcting a variety of computer hardware and software problems.*
- *implement computer based systems solutions to well-defined problems.*

General feedback

This project was not just about writing Cypher queries. You were also required to design a database, acquire data to populate the database, and write a report on the whole process. If your write-up does not contain a few sections detailing how data is stored in your database, such as explaining that election constituencies are stored as nodes with the label *Constituency*, then it probably doesn't explain why you chose to store them that way either. If you spent time making that choice, then you should discuss it in the report. Otherwise, I can only guess why you made those design choices and, given the problem, that's not obvious.

When it came to populating your database, you should have developed a strategy based on the skills you have learned in your course to date. Manually populating your database is the worst way you can do it. I heard some grumbles about how time consuming that was in the presentation and feedback sessions. You are training to be a software developer, and the name of the game is to use a computer to avoid manual work. One-off processes for isolating and cleaning data are usually done using a programming language or regular expressions in Notepad++ or similar. It's generally not a good idea to high-light how much manual work you've done with a dataset when you're a programmer. You should be looking to utilise your skills learned to date to prevent such labour.

Showing that you considered a number of alternative ways of proceeding at various stages of your project is essential to receive a high mark at this level. Generally, in years one and two, we accept solutions to problems from students as long as they give the correct answer. In third year, we expect that you are aware that there may be more than one algorithm or way to solve a problem, and have considered the trade-offs.

Cypher queries

As part of your project you were asked to submit three interesting Cypher queries based on your dataset. I can think of two ways in which a Cypher query could be interesting. The first, most straight-forward way, is that the query is interesting because it provides insight into the real world. Of course, the choice of query would be informed by what data is contained in the database. You can't say much interesting about the real world unless you have interesting data. You can think about this the other way too: perhaps you could have decided what queries you wanted to perform before populating your database, and then you could have filled your database with enough data to run those queries.

A lot of submissions, for instance, focused on looking at the gender breakdown of election candidates. Some submissions were very straight-forward. It would be a push to call them interesting as the information is freely available from a number of established news websites. To really excel with such a query, you would have to find something slightly off the beaten path. One suggestion would be to see if Simpson's paradox had occurred in this election [1]. That's only if you really wanted to push your marks into the top category.

The other way your queries could have been interesting would be from a technical point-of-view. One of the reasons for using Neo4j, as discussed in class, is that relational databases are inefficient at performing certain types of analyses. These include queries which result in the joining of a large, variable number of tables. This is the case with the Bacon number analysis covered in class.

Of course, two people might disagree about what is interesting. The good news is that, in your project you had the opportunity to explain why your queries were interesting in your write-up. If you didn't have a few clear sentences explaining how you selected each query and why it was interesting, then you left it up to the reader to decide for themselves, if at all. In the real-world, nobody will buy your software if you don't explain why they might want to, and in this project your queries won't seem interesting unless you explain why they are.

Project management

Git is an excellent source code management tool, and GitHub is an excellent platform built on top of git. As part of this project you were required to use both. At a minimum, you were required to make regular commits to your git repository over the full course of your project. This would involve initialising your repository within a couple of days of receiving the project specification, and then making a number of commits each week for the project duration. The commits should contain reasonably sized units of work, and should have meaningful commit messages. That would be worth four out of ten marks for the *Commits* category on the marking sheet. This is clear from the project marking scheme.

To really achieve a significant mark in this category, git should have been fundamental to your approach to the project. Your project attempt should have been organised in the way git works. This would involve keeping your master branch as a production copy of your project. This means that at any point in the duration of your project, what was in master would be a working, submission-ready copy of the project. This is in the ethos of git, and helps you as the developer to think of your project as layers of functionality. You create the initial, minimal working example of your project and commit it to master. Then you build in other functionality, in other branches with appropriate names. Once another branch is completed and tested, it is merged into master.

Another way to impress with your use of source-code management tools is to use some of the extra features provided by GitHub. These include easy-to-use issue tracking features, where you can even set milestones for your project. The GitHub website has an easy-to-read help section on these topics, but really, if you can use Facebook you can use GitHub's issue tracker.

Aside from being useful tools, git and GitHub help you to think like a software developer. The point isn't to just think of them as tools, but to adopt the work flow that they are built around. We regularly receive feedback from potential employers of our graduates that the ability to use them is essential for potential recruits.

Summary

Please take all of the above in the spirit that it is intended: to guide you in reflecting on this project, so that you can apply what you've learned to future projects. If I have one piece of advice to give regarding these projects, it is to learn to use the tools, like git, correctly. They're not just useful tools, they suggest a way of managing projects that has been informed by leaders in the software development field over a number of decades.

References

[1] Brad Hershbein. When average isn't good enough: Simpson's paradox in education and earnings. `http://www.brookings.edu/blogs/social-mobility-memos/posts/2015/07/29-simpsons-paradox-education-earnings-hershbein`.

[2] Quality and Qualifications Ireland. Award standards – computing. `http://www.qqi.ie/Publications/Computing%20-%20QQI%20Awards%20Standards.pdf`.