Assignment 5 Graphs Report
BHMQAI001

Object-Oriented Design

I have created the classes Program and Generator. These classes work in conjunction with the Graph class provided. Below, I will explain the function of the classes I created and how they interact.

Generator
The Generator class generates data as per the requirements given. It takes in V and E as parameters and outputs a set of data to the dataset text file. This text file is then called in the Program class and the data is read and used.

Program
This class reads the generated data that the Generator class produces and sends it to the Graph class to be put through Dijkstra's algorithm. This class also updates the dataset text file with the new data set as well as updating the csv file.

Goal of Experiment

The goal of this experiment is to compare the performance of Dijkstra's shortest paths algorithm with the theoretical performance bounds, through code implementation.

Execution

I executed this experiment through utilising the code we were given that creates graphs and implements Dijkstra's algorithm. Firstly, I added additional code to the Graph program to count the number of times vertices and edges were being processed on each iteration. I created variables to track the counter, and implement it when there are operations that need to be counted. This data is stored in an internal data structure and reported to the screen in a clear format.

Secondly, I created the Generator program to generate random data sets for each pair of V and E values. The vertex names were named "NodeXXX" as per the assignment instructions and the edge costs vary in the range of 1-10. I tested this program and made sure it worked effectively before proceeded onto measuring performance.

The main Program measures the performance. The class readData is an adaptation of the main class of Graph, which I removed. I created it to simply read the generated data, send it to Graph to be processed and then print the final data to the terminal and update the csv file.

Experiment Design Decisions

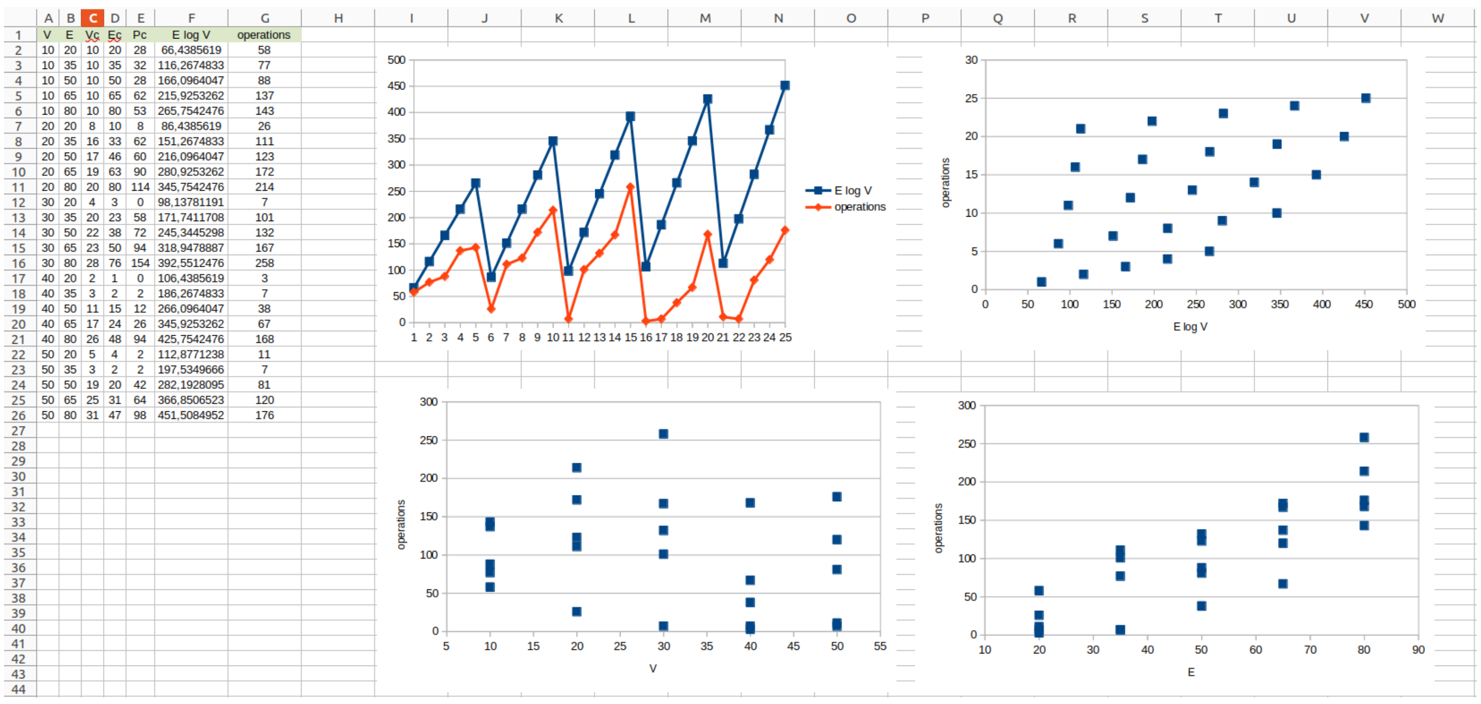
The Generator program sends each new set of data to the text file dataset, overwriting it with the new data each time. Initially, I created multiple datasets named with a convention of "dataset.10.20". These are saved and stored in the data folder. I decided to change my code to then overwrite a single file "dataset". This made it easier to work with the data and kept my project folder clean. The csv file works in a similar manner, making it quick and efficient to use when plotting mathematical graphs in LibreOffice Calc.

Creativity

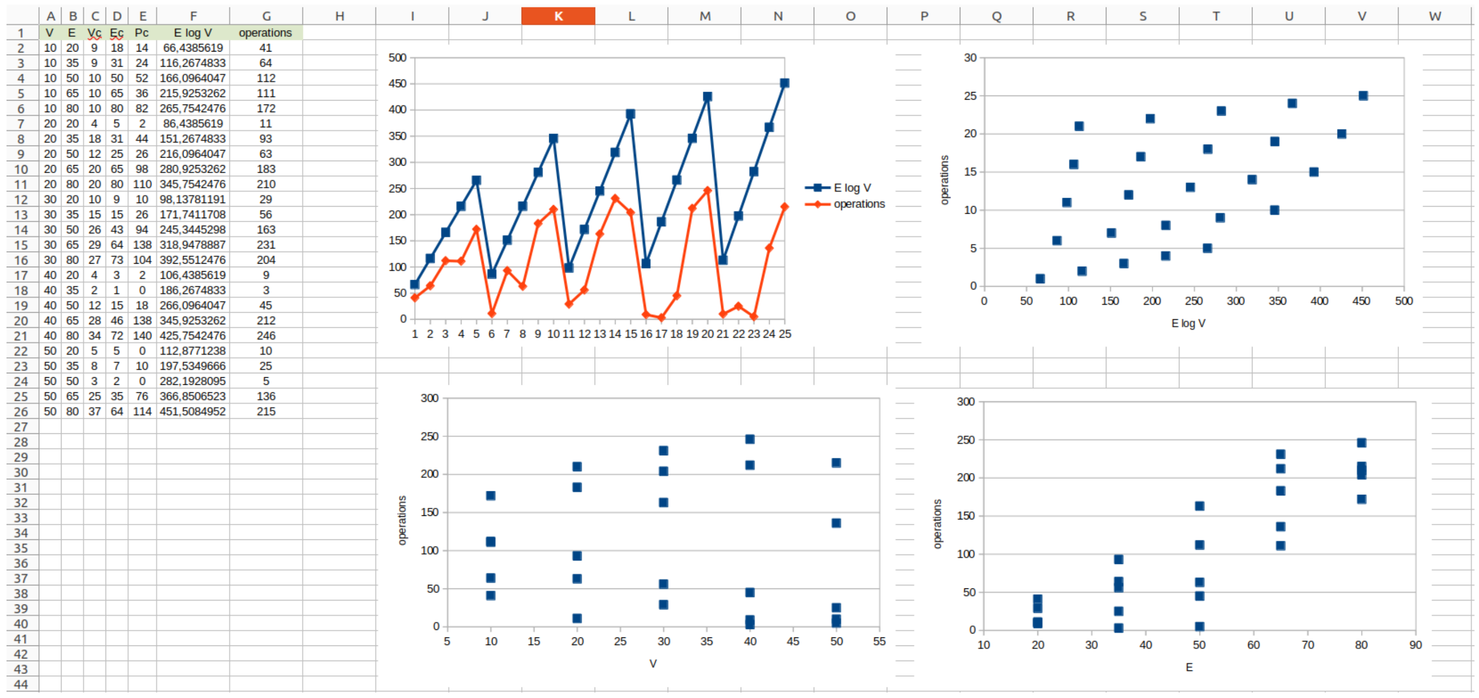
The continuously regenerating csv file constitutes as my creativity portion of the assignment. Initially I printed out the values in the form of integers separated with commas, but by implementing writing to a csv file, the process of creating mathematical graphs became much easier. The csv file can simply be imported and every time that the program runs and it is overwritten, the imported values are updated and the mathematical graphs utilise the new values. This makes it quicker and easier to notice trends in the mathematical graphs and draw conclusions.

Final Results and Discussion

I used LibreOffice Calc to plot different mathematical graphs showing the algorithm performs for different V and E values. Below are two examples of sets of graphs generated with different datasets. The dataset information is shown in the tables on the left-hand side of the figures.



V	E	Vc	Ec	Pc	E log V	operations
10	20	10	20	28	66,4385619	58
10	35	10	35	32	116,2674833	77
10	50	10	50	28	166,0964047	88
10	65	10	65	62	215,9253262	137
10	80	10	80	53	265,7542476	143
20	20	8	10	8	86,4385619	26
20	35	16	33	62	151,2674833	111
20	50	17	46	60	216,0964047	123
20	65	19	63	90	280,9253262	172
20	80	20	80	114	345,7542476	214
30	20	4	3	0	98,13781191	7
30	35	20	23	58	171,7411708	101
30	50	22	38	72	245,3445298	132
30	65	23	50	94	318,9478887	167
30	80	28	76	154	392,5512476	258
40	20	2	1	0	106,4385619	3
40	35	3	2	2	186,2674833	7
40	50	11	15	12	266,0964047	38
40	65	17	24	26	345,9253262	67
40	80	26	48	94	425,7542476	168
50	20	5	4	2	112,8771238	11
50	35	3	2	2	197,5349666	7
50	50	19	20	42	282,1928095	81
50	65	25	31	64	366,8506523	120
50	80	31	47	98	451,5084952	176



V	E	Vc	Ec	Pc	E log V	operations
10	20	9	18	14	66,4385619	41
10	35	9	31	24	116,2674833	64
10	50	10	50	52	166,0964047	112
10	65	10	65	36	215,9253262	111
10	80	10	80	82	265,7542476	172
20	20	4	5	2	86,4385619	11
20	35	18	31	44	151,2674833	93
20	50	12	25	26	216,0964047	63
20	65	20	65	98	280,9253262	183
20	80	20	80	110	345,7542476	210
30	20	10	9	10	98,13781191	29
30	35	15	15	26	171,7411708	56
30	50	26	43	94	245,3445298	163
30	65	29	64	138	318,9478887	231
30	80	27	73	104	392,5512476	204
40	20	4	3	2	106,4385619	9
40	35	2	1	0	186,2674833	3
40	50	12	15	18	266,0964047	45
40	65	28	46	138	345,9253262	212
40	80	34	72	140	425,7542476	246
50	20	5	5	0	112,8771238	10
50	35	8	7	10	197,5349666	25
50	50	3	2	0	282,1928095	5
50	65	25	35	76	366,8506523	136
50	80	37	64	114	451,5084952	215

Operations is a sum of v_count, e_count and pq_count. This is denoted in the figures above as Vc, Ec and Pc. These values indicate the number of times nodes and edges are being processed, and their positions in the priority queue.

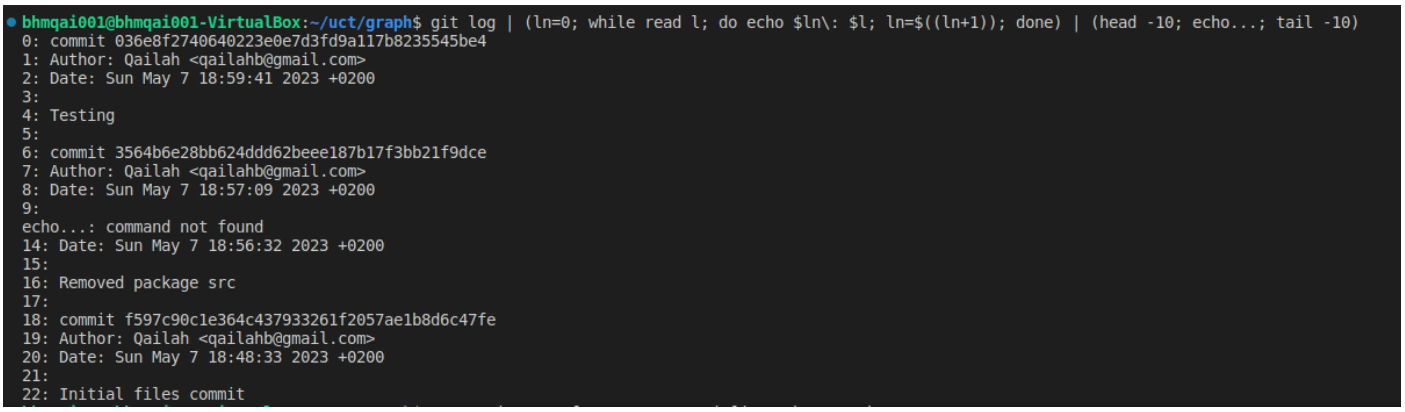
It can be seen above in the first graph that E log V is a bound for operations. This theoretical bound is derived from the overall time complexity of Dijkstra's algorithm of O(ElogV). The operations do not cross this bound at all, as O(ElogV) is the worst case, but maintains a similar pattern as expected.

The V versus operations graph is quite scattered, not showing much pattern or trend in its values. The E versus operations graph however shows a clear upwards trend, similar to that of the E log V versus operations scatter plot. E is directly proportional to operations.

Git Usage Log

```
0: commit 036e8f2740640223e0e7d3fd9a117b8235545be4
1: Author: Qailah <qailahb@gmail.com>
2: Date: Sun May 7 18:59:41 2023 +0200
3:
4: Testing
5:
6: commit 3564b6e28bb624ddd62beee187b17f3bb21f9dce
7: Author: Qailah <qailahb@gmail.com>
8: Date: Sun May 7 18:57:09 2023 +0200
9:

14: Date: Sun May 7 18:56:32 2023 +0200
15:
16: Removed package src
17:
18: commit f597c90c1e364c437933261f2057ae1b8d6c47fe
19: Author: Qailah <qailahb@gmail.com>
20: Date: Sun May 7 18:48:33 2023 +0200
21:
22: Initial files commit
```



```
bhmqai001@bhmqai001-VirtualBox:~/uct/graph$ git log | (ln=0; while read l; do echo $ln\: $l; ln=$((ln+1)); done) | (head -10; echo...; tail -10)
0: commit 036e8f2740640223e0e7d3fd9a117b8235545be4
1: Author: Qailah <qailahb@gmail.com>
2: Date: Sun May 7 18:59:41 2023 +0200
3:
4: Testing
5:
6: commit 3564b6e28bb624ddd62beee187b17f3bb21f9dce
7: Author: Qailah <qailahb@gmail.com>
8: Date: Sun May 7 18:57:09 2023 +0200
9:
echo...: command not found
14: Date: Sun May 7 18:56:32 2023 +0200
15:
16: Removed package src
17:
18: commit f597c90c1e364c437933261f2057ae1b8d6c47fe
19: Author: Qailah <qailahb@gmail.com>
20: Date: Sun May 7 18:48:33 2023 +0200
21:
22: Initial files commit
```