CHROMATIC NUMBERS

CAN IDIL, ANDREEA-VIOLETA LAZAR, LAURENCE MANOUKIAN, NILS STOLK, GIACOMO TERRAGNI, ABHINANDAN VASUDEVAN, VIKRAM VENKAT

Group 27

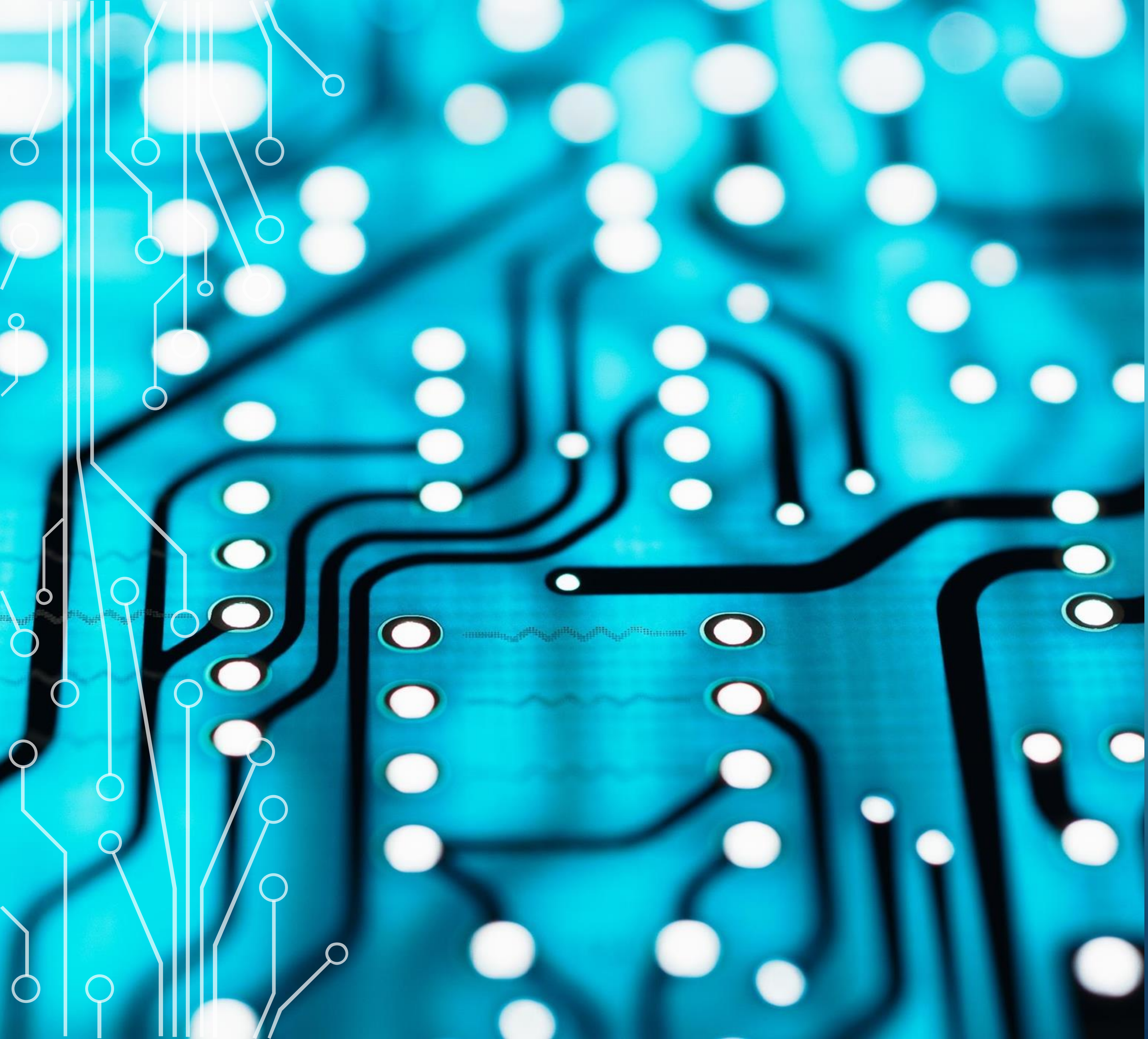


TABLE OF CONTENTS

To what extent can computer algorithms be developed and employed to efficiently compute the so-called chromatic number of a given graph?

To what extent does the execution environment of the code affect the results obtained?

Recap of Phase 1 and 2

Phase 1

- *Task: create an algorithm to compute the chromatic number for a graph G with number of vertices V and connecting edges E. The approach was to create 2 different algorithms to solve the graph, and ultimately testing each algorithm type for reliability and accuracy.*

Phase 2

- *Task: create single-player game with 3 game modes, using graph colouring games. Operated through a custom GUI. Game must be fully interactive, with hints to be displayed either on demand or based on the player's progress.*

Breakdown of Phase 3

- *The Third phase is conceptually similar to the First Phase. This time, the test graphs provided by the examiners are larger and more computationally challenging, necessitating the use of certain algorithm(s).*

- *Additionally, some graphs have special structures which makes the computation easier, assuming we can write an algorithm to detect the special structures.*

- *For example, if a graph has no cycles(i.e, it is a tree), then it's chromatic number will be at most 2 no matter how many edges it has. Hence, there are many more special cases like this.*

- *We have found that the lower bound approach is the most effective, and have implemented the Bron Kerbosch method*

SPECIAL GRAPH STRUCTURES

Graph	Vertices (V)	Edges (E)	Unique element
A	6	(V-1)	Everything is connected
B	5	2	Uneven circle
C	6	2	Even circle
D	6	5 or 1	Star
E	5	4 or 3	Even wheel
F	6	5 or 3	Uneven wheel

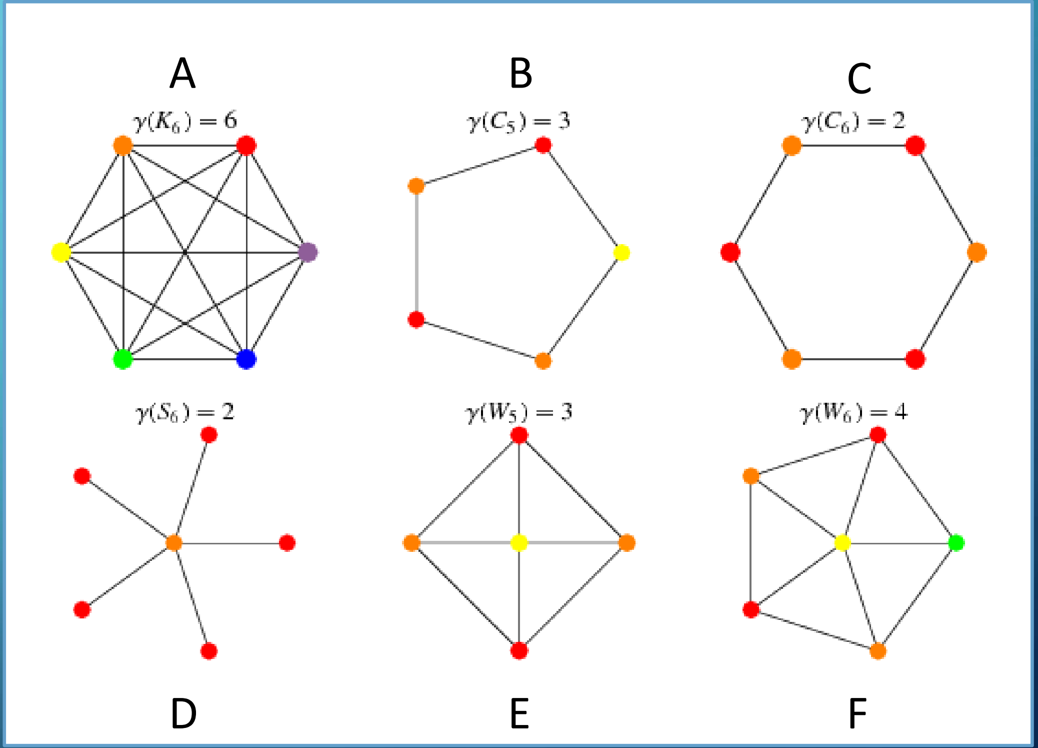
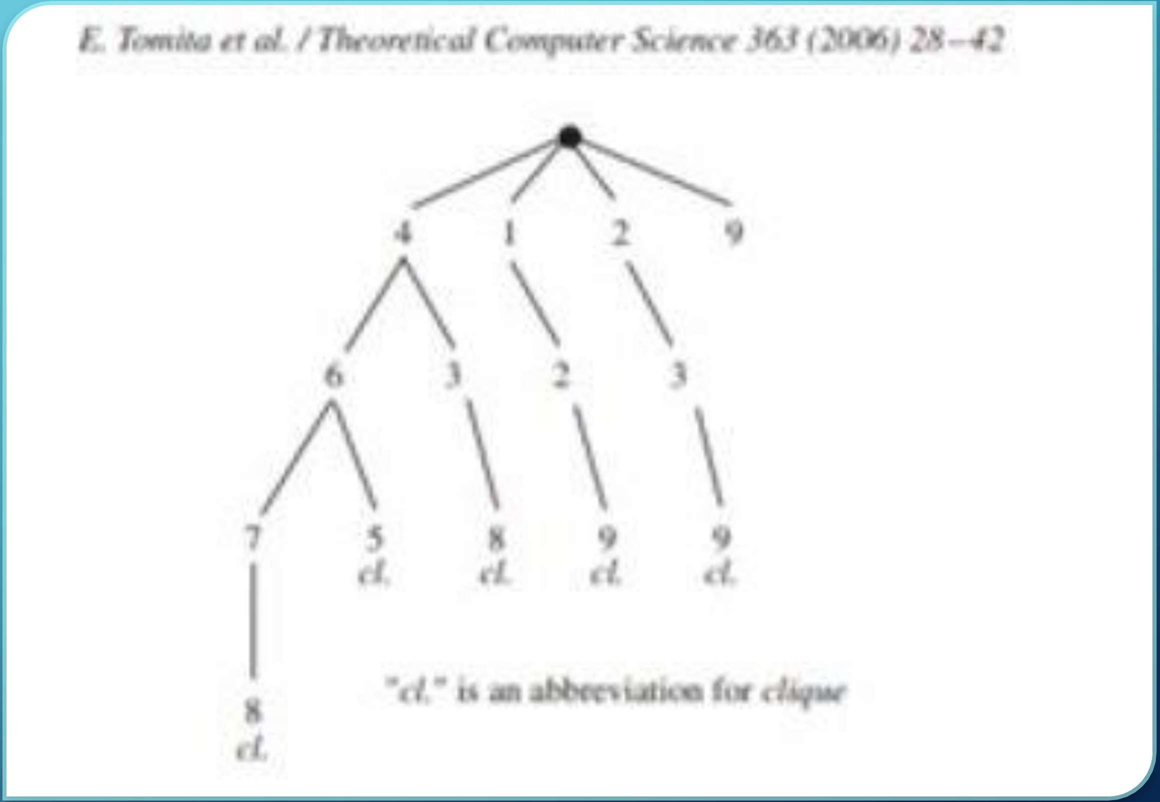


Image: *Weisstein, E* https://mathworld.wolfram.com/ChromaticNumber.html

GENERAL APPROACH – IMPLEMENTATION OF THE ALGORITHMS

- *Bron-Kerbosch algorithm-It is an algorithm for finding the max cliques in an undirected graph*

  (Bron & Kerbosch, 1973).
- It lists all subsets of vertices with the two properties that each pair of vertices in one of the subsets is connected by an edge, and no subset can have any additional vertices added to it.



With pivoting

- The basic form of the Bron–Kerbosch algorithm is a recursive backtracking algorithm that searches for all maximal cliques in the Test Graph G.
- However, this approach is not efficient in the case of graphs with many non-maximal cliques. To allow the algorithm to backtrack quicker, a variant of the algorithm, involving a pivot vertex, u, chosen from P U X.
- Any max clique should involve either u or one of its non-neighboring sets.
- Hence, we can see that the main reason the with pivots version is more effective is that it only tests u and it's non-neighbors as the choices of v(the vertex) that is then added to R in each recursive call.

UPPER BOUND AND LOWER BOUND

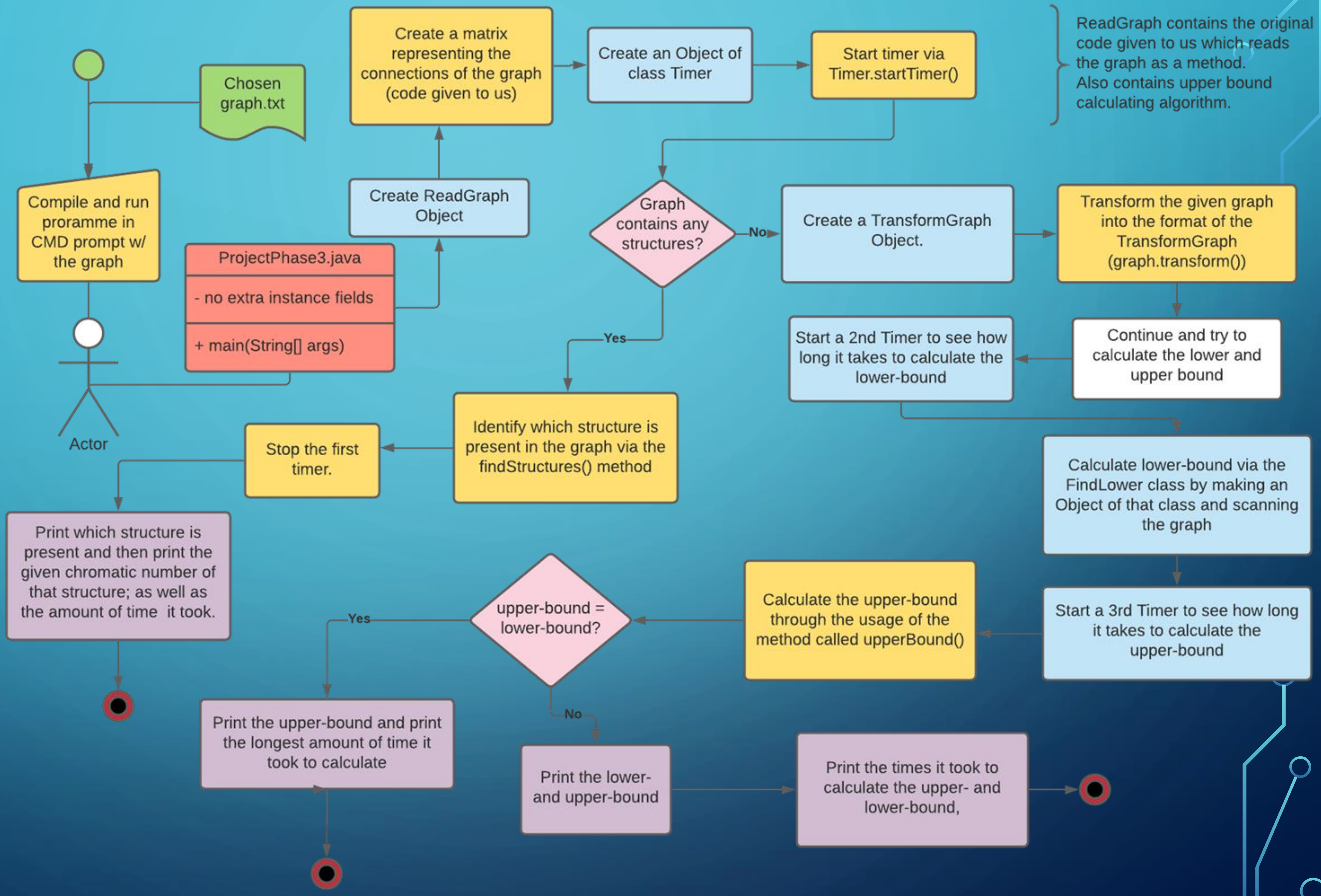
- Lower bound

The Lower Bound computation is important for any algorithm. Once we calculated it, then we can compare it with the actual complexity of the algorithm and if their order are same then we can declare our algorithm as optimal. So in this section we will be discussing about techniques for finding the lower bound of an algorithm.

- Upper bound

According to the upper bound theory, for an upper bound U(n) of an algorithm, we can always solve the problem in at most U(n) time. Time taken by a known algorithm to solve a problem with worse case input gives us the upper bound.

Final flowchart for this product:



Explanation of experiments conducted to answer the research question(s):

Experiment 1:

- Test product on example graphs which were given to us in the third phase (Windows vs. Linux).
- Test each individual graph 10 times
- Take average values
- Why this experiment, you ask?

Experiment 2:

- Test product on example graphs which were provided to us in the first phase (Windows vs. Linux).
- Test each individual graph 10 times
- Take average values
- Why this experiment, you ask?

A live demonstration of how the product works:

Run in PowerShell on Windows 10 using the following script:

```
For ($i=0; $i -le 20; $i++) {
   java ProjectPhase3 .\example_graphs\phase3_2020_graph$i.txt;
  }
```

Results and Conclusion: Linux vs. Windows 10 Performance Comparison

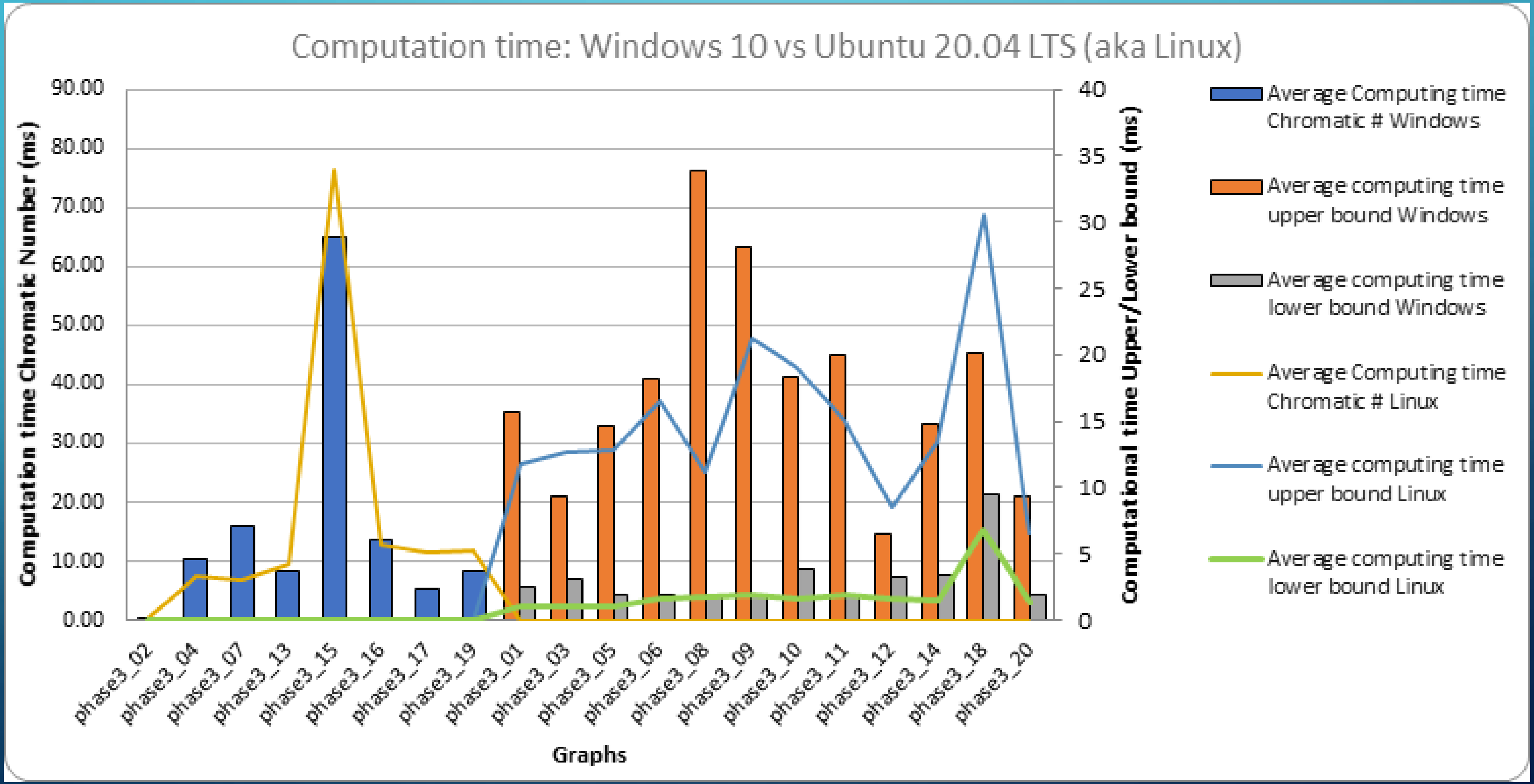
	Computer 1	**Computer 2**
Processor Model and Clock Speed	Intel Core i7-8750H @ 2.2 GHz	Intel Core i7-8750H @ 2.2 GHz
Installed RAM	16 GB	16 GB
Operating System and Version	Windows 10.0.18363 Build 18363	Linux Ubuntu 20.04 LTS

- Both machines had the exact same hardware specifications, the only difference was the operating system.
- We obtained different runtime results on each OS, with Linux on avg. being consistently faster than Windows

Results and Conclusion: Linux vs. Windows 10 Performance Comparison (Experiment 1)

Graph	Chromatic #	Upper bound	Lower bound	Average Computing time Chromatic # Windows	Average computing time upper bound Windows	Average computing time lower bound Windows	Average Computing time Chromatic # Linux	Average computing time upper bound Linux	Average computing time lower bound Linux	Vertices	Edges
phase3_02	2 (bipartite)			0,20			0,40			529	271
phase3_04	2 (bipartite)			10,20			7,50			744	744
phase3_07	3			16,10			6,90			212	252
phase3_13	11			8,20			9,50			143	498
phase3_15	2 (bipartite)			64,90			76,30			4007	1198933
phase3_16	98			13,70			12,80			107	4955
phase3_17	15			5,30			11,60			164	889
phase3_19	8			8,40			12,00			106	196
phase3_01		6	8		15,70	2,50		11,70	1,10	218	1267
phase3_03		3	7		9,30	3,10		12,60	1,10	206	961
phase3_05		5	10		14,70	1,90		12,80	1,10	215	1642
phase3_06		10	13		18,20	2,00		16,50	1,60	131	1116
phase3_08		4	7		33,90	2,00		11,20	1,80	107	516
phase3_09		8	12		28,10	2,00		21,30	2,00	43	529
phase3_10		8	9		18,40	3,80		19,00	1,70	387	2502
phase3_11		9	14		20,00	1,80		15,00	2,00	85	1060
phase3_12		2	5		6,50	3,20		8,50	1,70	164	323
phase3_14		3	6		14,80	3,40		13,40	1,50	456	1028
phase3_18		3	5		20,10	9,40		30,50	6,80	907	1808
phase3_20		2	4		9,30	2,00		6,60	1,40	166	197

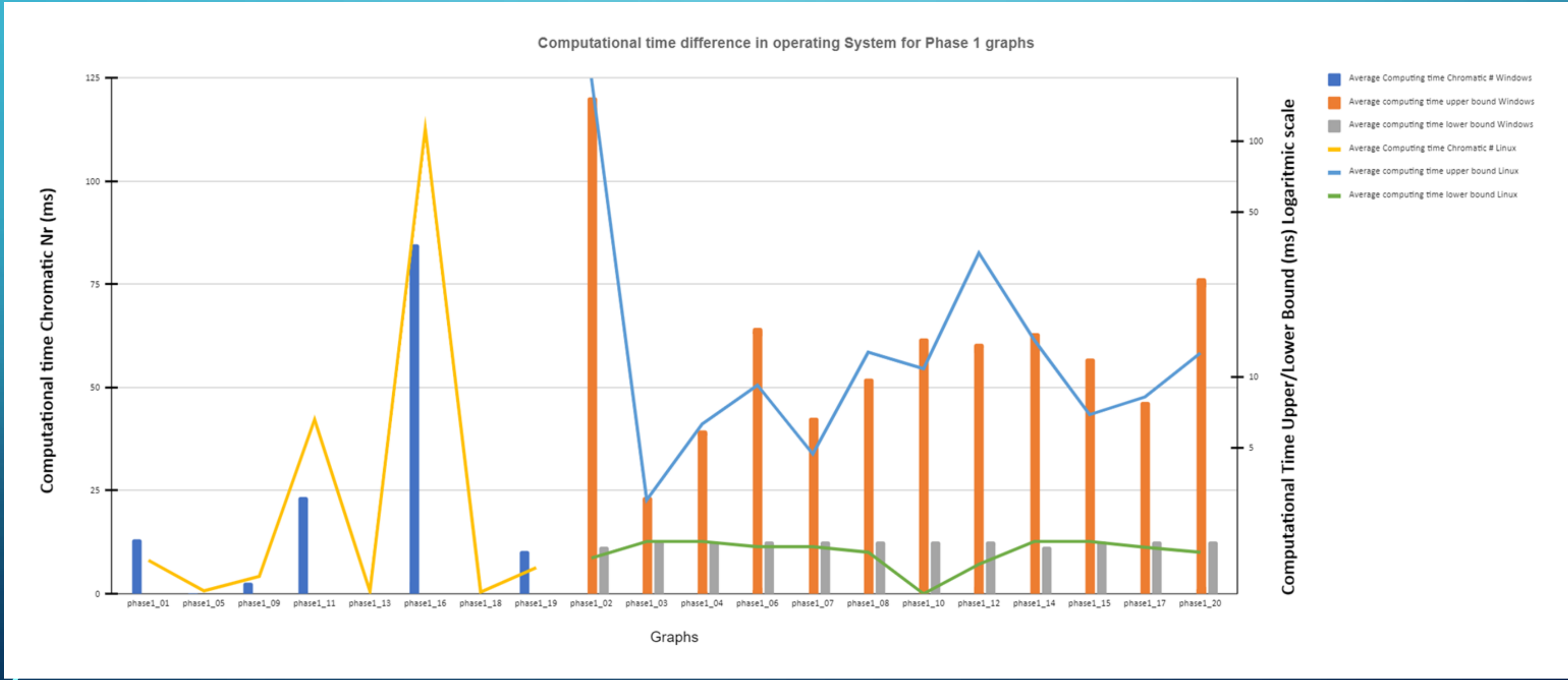
Results and Conclusion: Linux vs. Windows 10 Performance Comparison (Experiment 1)



Results and Conclusion: Linux vs. Windows 10 Performance Comparison (Experiment 2)

Graph	Chromatic #	Upper bound	Lower bound	Average Computing time Chromatic # Windows	Average computing time upper bound Windows	Average computing time lower bound Windows	Average Computing time Chromatic # Linux	Average computing time upper bound Linux	Average computing time lower bound Linux	Vertices	Edges
phase1_01	11			13,10			8,10			76	303
phase1_05	2 (bipartite)			0,20			0,70			93	92
phase1_09	10			2,80			4,20			41	184
phase1_11	31			23,40			42,20			191	3888
phase1_13	2			0,00			0,60			61	448
phase1_16	54			84,80			112,60			867	18711
phase1_18	2 (bipartite)			0,00			0,40			52	52
phase1_19	31			10,50			6,30			32	466
phase1_02		15	25		153,30	1,90		184,60	1,70	61	1390
phase1_03		3	5		3,10	2,00		3,00	2,00	18	45
phase1_04		2	3		5,90	2,00		6,30	2,00	11	16
phase1_06		3	7		16,10	2,00		9,20	1,90	69	409
phase1_07		2	5		6,70	2,00		4,70	1,90	43	127
phase1_08		7	12		9,80	2,00		12,70	1,80	56	483
phase1_10		3	8		14,50	2,00		10,80	1,20	209	1249
phase1_12		3	9		13,80	2,00		33,50	1,60	195	2365
phase1_14		16	17		15,30	1,90		14,30	2,00	31	385
phase1_15		4	8		11,90	2,00		6,90	2,00	41	205
phase1_17		4	5		7,80	2,00		8,20	1,89	81	293
phase1_20		3	10		26,20	2,00		12,60	1,80	82	811

Results and Conclusion: Linux vs. Windows 10 Performance Comparison (Experiment 2)



Results & Conclusion

From these results which we have compiled, we can say that it is very possible to create a computer algorithm which is capable of finding the chromatic number of a given undirected graph.

This of course can be seen through our results from our first and second experiment, as 7 out of 20 of the graphs which we tested (phase 3) gave an exact chromatic number to an undirected graph, and all of the ranges that our product gave for the phase 1 graphs were accurate (according to results provided by Katharina Schüller).

As for our secondary research question, we can see that the environment of execution does affect the results, although, it is only in terms of computational speed. The results, in terms of the chromatic numbers we acquire, from running the product on 2 separate operating systems do not change.

References

Bron, C., & Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. Communications Of The ACM, 16(9), 575-577. doi: 10.1145/362342.362367

Weisstein, E. *Chromatic Number.* Retrieved January 10, 2021, from https://mathworld.wolfram.com/ChromaticNumber.html

Thank you!

-Group 27