Journal of Computer Science

Original Research Paper

Graph Coloring Program of Exam Scheduling Modeling Based on Bitwise Coloring Algorithm Using Python

[1]Samsul Arifin, [2]Indra Bayu Muktyas, [3]Wikky Fawwaz Al Maki and [4]Mohd Khairul Bazli Mohd Aziz

[1]*Department of Statistics, School of Computer Science, Bina Nusantara University, Jakarta, 11480, Indonesia*
[2]*Mathematics Education Department, STKIP Surya, Tangerang, 15115, Indonesia*
[3]*Department of Informatics, Telkom University, Bandung, 40257, Indonesia*
[4]*Faculty of Industrial Sciences and Technology, University Malaysia Pahang, 26300 Gambang, Pahang, Malaysia*

Article history
Received: 23-09-2021
Revised: 28-12-2021
Accepted: 19-01-2022

Corresponding Author:
Samsul Arifin
Department of Statistics,
School of Computer Science,
Bina Nusantara University,
Jakarta, 11480, Indonesia
Email: VanDerMerwe@nwu.ac.za

Abstract: A graph coloring is the process of assigning labels to the vertices of a graph in such a way that no two adjacent vertices have the same color. The chromatic number of a graph G is the smallest number of colors that can be assigned to it. Graph coloring has a wide range of applications and is commonly used to solve scheduling issues. In this article, the researchers design an algorithm and apply it to a computer program (Python) to solve graph coloring and to visualize the variation of exam scheduling modeling at Binus University in graphs based on the Bitwise Graph Coloring Algorithm. The researchers develop a graph coloring algorithm by considering some of the graph vertices to be binary numbers. Bitwise operations make this algorithm run very fast. The algorithm constructed by the researcher is a modification of Komosko, etc.'s algorithm in 2015 and it is the key result of this research. The researchers try to offer an alternative method in the process of making the final semester exam schedule. Next, the researcher tested the program on the data of subjects and students who took it at the Study program of TI-Stat-Math in Binus University. Our results show that from the program created and the simulations performed, 8 schedule slots are generated in about 0.675 sec.

Keywords: Graph Coloring, Bitwise Graph Coloring, Scheduling, Python

Introduction

In the world of education, exams are very important. With the examination, it will be known to what extent the level of understanding of the _eld of study is being studied. For elementary, junior high, or high school levels, determining the exam schedule is not too di_cult, because each student gets the same subject. It's di_er-ent at university. Every student has the right to take the courses they want, as long as they have taken the prerequisite courses. This makes it possible for students to take several courses that can conflict with their exam schedules. Academic office staff must determine an exact exam schedule and avoid all exam schedules colliding. It is possible that he can make a schedule with each course having a different exam time, but it will take a very long examination period. This will be detrimental to many parties, both lecturers, students and all university employees. For this reason, the officer certainly wants to make the exam period as short as possible without a collision schedule (Budiman, 2007).

In the Binus University, TI-Stat department, scheduling semester exams sometimes still have problems. Students with a collision schedule are allowed to contact academic officers. Then the officer changed the schedule to a revised schedule and announced it again through the Binusmaya website. If there are still reports of collisions from students, then another revision will be held and so on until the final schedule where there is no collision exam schedule. This was done in a relatively long interval. However, this problem can be solved easily if using the graph coloring concept. One of the advantages of graph coloring is that the existing schedules do not collide with each other (Rosen, 2019). Besides, by using the concept of Graph Coloring, the researcher believes that the time required to compile a schedule is also much shorter when compared to the usual method (Komosko *et al*., 2016).

Several studies that have used graph coloring for scheduling have been carried out by Muktyas (2010; Firdaus, 2020; Bustan and Salim, 2019). Furthermore, there is a very good method used in making schedules using the concept of graph coloring, which the researchers call the bitwise graph coloring method (Komosko *et al*., 2016). On the other hand, the Python programming language has supported several functions and libraries that the researcher can use to make scheduling using the concept of graph coloring with the help of the bitwise graph coloring method (Oliphant, 2007). For that, from some of the literature, the researchers want to study the Graph coloring program of exam scheduling modeling at Binus University based on a bitwise coloring algorithm using Python. Furthermore, the researcher would like to show an alternative algorithm in the application of graph coloring techniques, one of which is the process of making an exam schedule at Binus University.

Materials and Methods

The research style employed here is one of investigation and application of previously obtained results, as evidenced by a review of the literature (Gravetter and Forzano, 2012). This section examines the theories that were utilize in this investigation.

Notion of Graph

The researcher will start by looking at the concept of graphs (Rosen, 2019).

Definition

A graph $G = (V, E)$ is a system consisting of a set of objects $V = \{v_1,v_2,v_3,v_4,v_5,...,v_n\}$ which is called a set of vertex/points and a collection of $E = \{e_1,e_2,e_3,e_4,e_5,e_6,...,e_n\}$ which is a collection of edge such that each edge of the e_k is associated with an unordered pair (v_i, v_j).

The points v_i, v_j concerning ek are called the endpoints of the edge e_k. If a point v_i is the endpoint of an edge e_k, then v_i and e_k are called incidences, or the point v_i is related/attached/incident to the edge e_k. Two nonparallel edges are called neighboring/adjacent if both sides are attached to the same point. Two points are called neighboring if the two points are endpoints of the same side (Rosen, 2019). The most common way to present a graph is with a diagram. Points are represented by dots and sides are represented by a simple curve connecting each of the two points. G is called a simple graph if G contains no loops and doubles. The number of edges associated with the point v is called the degree of v, denoted by $d(v)$. Any graph G corresponds to a matrix of size $n \times n$ which is called the adjacency matrix of G. Adjacency matrix $A(G) = [a_{ij}]$, where a_{ij} is the number of edges associated with the pair of points v_i and v_j (Widodo, 2016).

Notion of Coloring Graph

Vertex coloring on a graph is the giving of colors to the vertex of a graph so that no two adjacent vertexes on the graph have the same color (Malkawi *et al*., 2008). Definition 2.2 is the definition of graph coloring (Klotz, 2002).

Definition

A vertex coloring of graph G = (V, E) *is an* $F: V \rightarrow N$ mapping where neighboring points are different colors in N, i.e., if uv is in E, *then* $F(u)$ is not equal to $F(v)$.

A graph G can be colored with a different color at each point, however, in its application, coloring the graph with as little color as possible is very necessary. The minimum number required by graph G so that all points are colored without a pair of neighboring points of the same color is called the chromatic number of graph G, denoted $\chi(G)$. A graph G is called k - chromatic if $\chi(G) = k$ (Klotz, 2002).

A well-known combinatorial optimization problem is the Vertex Colouring Problem (VCP). It's one of Karp's twenty-one NP-complete issues. It is impossible to discover an exact solution in a reasonable amount of time for this class of problems because the solution space grows exponentially in proportion to the size of the input data. The VCP entails determining the smallest number of colors required to color all of a graph's vertices in such a way that any two neighboring vertices have distinct colors. The VCP is useful in a variety of situations. provide an excellent survey. It can be used to solve a variety of scheduling issues. The graph vertices in this case represent courses that should be included in the timetable and the colors represent time gaps during which many courses are held simultaneously. If the related courses cannot be held at the same time, an edge is put between them (Komosko *et al*., 2016).

The Bitwise graph coloring algorithm is one of the algorithms used to color graphs. Komosko *et al*., (2016) developed a graph coloring technique based on bitwise operations on the adjacency matrix (A) and a special matrix known as the forbidden matrix C A fast greedy sequential heuristic for the vertex coloring problem is described in their work. The proposed approach colors the graph in the same way as the well-known greedy sequential heuristic, which colors the current vertex in the smallest possible color at each step. The development of a special prohibited colors matrix and the use of efficient bitwise operations on bit representations of the adjacency and forbidden colors matrices are two of their primary contributions. (Komosko *et al*., 2016).

The researchers see that there is a deficiency in the algorithm proposed by Komosko, there is no order of degrees from the point. Note that in graph theory, the order of the degree of points will affect the number of coloring results. If not sorted, it will usually result in a large staining result. Furthermore, the number of coloring results corresponds to the number of schedule slots formed. To overcome this weakness, in this study the degrees of points are ordered in descending order. By ordering the vertices in descending order, the resulting graph color is equal to the upper limit of the chromatic number, which will ensure that the number of colors in the graph is small. Thus, the resulting schedule slots are also few. The researcher feel that this weakness needs to be addressed, it is related to the optimization of schedule results. The researchers believe that if the degrees of the points are not ordered, then the scheduling results obtained are not the best results. Theorem 2.3 is very useful in making the main program.

Theorem

The forbidden matrix *C* is a square matrix of size $n \times n$ with $C_{ij} = 1$ if the color i is forbidden by point *j*, because one of its neighbors is already colored by *i*.

One row of the adjacency matrix is directly operated with a bitwise OR operation with one of the rows of the forbidden matrix *C*. In its implementation using Python, one row of the adjacency matrix *A* is considered a binary number. This number is then operated with another binary number obtained from one of the rows in the forbidden matrix *C*. The result of the coloring is the matrix *C*. Each point *j* in matrix *C* is defined as the minimum color *i* which is not forbidden for point *j* ($C_{ij} = 0$) (Komosko *et al*., 2016). Thus, this algorithm is more effective because if the Greedy algorithm is usually done one by one, but in this algorithm, it is done at once for each row. The researchers make some modifications by sorting the adjacency matrix A by degrees in descending order. The complete algorithm is presented in Algorithm 1. Definition 2.4 is the notion of an operation of bit (Rosen and Krithivasan, 2012).

Definition

A bit string is a sequence of zero or more bits. The length of this string is the number of bits in the string. Bit operations can be extended to bit string operations, i.e., bitwise. Bitwise OR and of two strings of the same length can be obtained by operating OR and on each corresponding bit.

From Definition of 2.4, the researcher will get a perspective on graph coloring contained in the 2.5, which describes one of the graph coloring algorithms, namely Greedy coloring. Please see Table 1 for more details.

Theorem

Greedy coloring can be viewed as a series of bitwise operations OR and SHIFT operators.

Generate Code

From Theorem 2.5, a graph coloring algorithm can be constructed which is contained in Algorithm 1.

Algorithm 1.

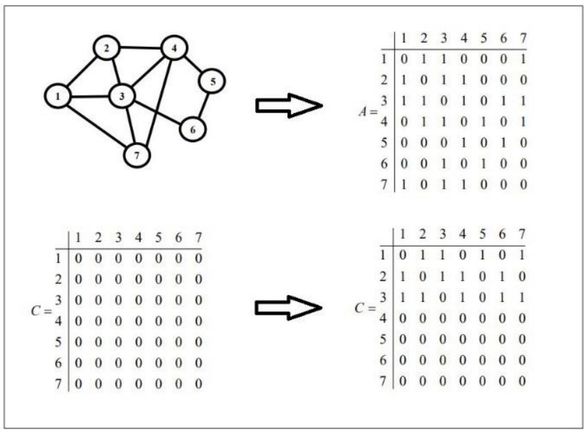
Bitwise Graph Coloring

```
1.  Input: A_nxn
2.  Sort the matrix A by degrees in descending order, also
    save the order index
3.  Assume that each row in matrix A is a binary.
    Convert each row in matrix A to a decimal number.
    We get an Aint matrix of size 1xn
4.  C = zero matrix of size 1xn
5.  topeng = 1 << (n - 1)    # << is left shift
6.  n_warna = 0
7.  for i in range(n):           # i -> row of A
8.      for j in range(n):       # j -> row of C
9.          cek = C[j] AND topeng # is the bit alive?
10.         if cek == 0:
11.             C[j] = C[j] OR A[i] # have colored?
12.             topeng = topeng >> 1 # next column
13.             break
```

From Algorithm 1, it will be applied to a small graph as stated in the Example 2.6 (Komosko *et al*., 2016).

Example 2.6

Look at the picture below. A graph with 7 vertices is represented in the adjacency matrix *A*. The initial forbidden matrix *C* is a zero matrix of size 7×7.



$$A = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 4 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 6 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 7 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

$$C = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \Rightarrow C = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Matrix C on the right is the final result of the graph coloring. The coloring of each point *i* is determined by looking at the zero values in column *i* of matrix *C*. Based on the forbidden matrix *C*, the graph coloring from points 1,2,3,4,5,6,7 is 1,2,3,1,2,1,2.

The algorithm above is proven to be able to run well in graph coloring and is able to produce an efficient scheduling process. On the other hand, the concept of bitwise graph coloring has also been applied in several ways, namely in the topic TDMA schedule and Congestion Game Analysis (Xiaoying, 2019; Jioudi *et al*., 2019). Algorithm 2 is about converting some dataset into an adjacency matrix.

Algorithm 2.

Convert The Dataset into Adjacency Matrix

1. Arrange the raw data into two columns, the first column is for the course id, the second column is for the student id or student name
2. mk = the list containing unique course ids. Also create a separator index from the data already in the group

```
3.  n = len(mk)
4.  adj = zero-matriks of size n x n
5.  mk_pisah = list of second column data separatedby
    separator index
6.  for i in range(0,n):
7.      for j in range(i+1, n):
8.          if there exist intersection of mk_pisah[i] and
            mk_pisah[j]:
9.              adj[i,j] = 1
10.             adj[j,i] = 1
```

Python

After studying graph theory and the concept of coloring a graph, the researcher will discuss the programming language that the researcher use. The researcher will go over the Python scripts that employed in the next few paragraphs. The researcher used Python 2.7.14 to generate some code. The main outcome of this article is the Python program code. The things mentioned in Algorithm 1 become the foundation for creating the software. However, it should be emphasized that the Python program in question must have the ”NumPy” plugin installed. The procedures for installing NumPy in Windows OS are simple (Project Teams, 2020). Python is a widely used programming language. Python is a computer language that is considerably easier to learn and use than other programming languages (Chollet, 2021). Because the idea of python itself promotes readability, the syntax is straightforward, easy to comprehend and remember. Python code is simple to develop and read, making it simple to debug and maintain (Muktyas *et al*., 2021).

Python is not only easier to read, but it is also more efficient than languages like C, C++ and Java. With another language, 5 lines of code may be required; but, in Python, only one line of code may be required. As a result, Python programs are more concise and faster to write than programs written in other languages (Rahman *et al*., 2019). Python is a multipurpose programming language. Python can be used for a wide range of tasks, including text processing, website development, network programming, robotics, data mining, artificial intelligence and more (Arifin and Muktyas, 2021). We can make desktop and smartphone applications with Python. Python has a lot of built-in library support. You can utilize a variety of modules and software extensions to customize the application to meet your specific requirements. Python’s community is very active in developing the language so that it becomes a very trustworthy language. Python has the ability to communicate with other languages. Python code can be called from C, C++ and other programming languages and vice versa (Arifin *et al*., 2012).

In short, the reason the researcher use Python is that Python is a powerful language and can be run on multiple platforms, but it’s also very easy to understand. After that, the researcher will do a simple simulation using huge data (2803 rows) located in the TI-Mat-Stat Study Program, Bina Nusantara University. The first thing to do is create a table containing the courses and NIM students who take them. Table 3 is a table view of courses and NIM students. Furthermore, the complete code of our Python program is as follows, which will close this session: https://github.com/muktyas/bit-graph-coloring.

Results and Discussion

The specifications of the computer used are Intel Core i3-2310M CPU @ 2.10GHz x 2 ram 7.7 GiB with Linux Mint 20.2 Cinnamon Operating System. The raw data obtained from the academic section in the form of course id and student names are arranged into two columns. The first column contains course id, while the second column contains the student’s name. The table is sorted by the first column. After that, the raw data is converted into an adjacency matrix. The raw data obtained from the academic section of Binus University at TI-Math-Stat program in all batches consisted of 59 different subjects. If it is arranged into two columns, then 2802 rows are formed, which can be seen in the Fig. 1. This is our main program, which is the main result of this study:

The Main Program

```
import numpy as np
import time

waktu_mulai = time.time()
adj = np.genfromtxt(’adjku-binus.txt’, dtype=np.uint8)
n_makul = len(adj)
print "....."

#urutkan berdasarkan derajat tiap titiknya

def UrutAdj(adj): n_makul = len(adj)

.
.
.

keluaran = open(’keluaran-perwarna-binus.txt’,’w’) print
warna
 print ’Okay, jadi bisa menggunakan’,len(warna),
’warna saja’
 keluaran.write(’Okay, jadi bisa menggunakan %s warna
saja\n
’ %(len(warna)))

i = 0
for baris in warna:
print ’warna’,i,’bisa untuk titik:’,baris
keluaran. write (’warna %s bisa untuk titik: %s\n’
%(i, baris))
i = i + 1


keluaran.close ()
 print ’The Coloring Bitwise time is
 ’, time. time() - waktu_mulai,
 ’ seconds.’
```

The next process, Algorithm 2 is applied to Python program to convert raw data into adjacency matrix. The result is on Table 2. Moreover, the average time required is 1.198271608352 seconds. After sorting the adjacency matrix based on the degree of the point in descending order, then graph coloring is carried out using Algorithm 1. This program produces a matrix *C* in the form of decimal numbers as follows.

The Output of Matrix C

```
c = [287878360632130526, 396312094275534822,
484135860426505643, 562378202004053948,
575194045616614992, 576407408675319809,
571950546240339968, 408587686408880128, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

From the Result 1, there are 8 non-zero numbers in matrix *C*. This means that only 8 colors are needed to color the graph. Each number in the *C* matrix is then interpreted into a binary number so that it can be read easily. So the new matrix *C* can be viewed as a matrix with a size of $i \times j$. The zero bits in column *i,j* in the new matrix C indicate that the *j* points can be colored with the color *i*. There are two outputs from this program, the first is output based on courses, while the second is output based on color. This is one of the advantages of coloring using bitwise. The resulting output is two columns with 59 rows. The number of rows indicates the number of courses. The first column shows the sequence of dots (the course code), while the second column shows the color (time slot), as follow.

Result 2. The Course Code and Time Slot

```
0 0
1 2
2 0
3 5
4 5
.
.
.
55 4
56 1
57 5
58 6
```

From the Result 2, the researcher can immediately see the time slots for each course code index. If the researchers want more slot variations, then the researcher can see it in the second output, namely the output based on color.

The Result 3 is the output of the program that has been made.

Result 3. The Output of the Program

```
Okay, we can use 8 color only:


Color 0: [32, 0, 10, 2, 47, 25, 39, 35]
Color 1: [45, 6, 33, 21, 56, 14, 38, 40, 35]
Color 2: [1, 30, 5, 18, 20, 23, 9, 38, 41]
Color 3: [48, 27, 24, 29, 34, 50, 46, 52, 9, 44, 35]
Color 4: [22, 36, 15, 49, 12, 31, 55, 54, 8, 37, 43, 39, 40,
         41, 44, 35]
Color 5: [11, 26, 19, 28, 3, 57, 4, 25, 23, 37, 43, 9, 39, 38,
         40, 41, 44]
Color 6: [27, 33, 13, 58, 49, 3, 7, 12, 20, 51, 53, 52, 8, 4,
         25, 23, 37, 43, 9, 39, 38, 40, 41, 44, 35]
Color 7: [45, 30, 27, 22, 10, 11, 33, 13, 18, 19, 2, 47, 34,
         16, 50, 49, 17, 3, 12, 42, 56, 20, 46, 31, 55, 14,
         54, 57, 51, 53, 52, 8, 4, 25, 23, 37, 43, 9, 39, 38,
         40, 41, 44, 35]
```

From Result 3 of color output, the researcher can choose certain subjects to be colored with any color. For example, courses with a sequence of 35 can be for colors 0, 1, 3, 4 and 6. This means that semester exams can be grouped into 8 time slots only. Moreover, Table 2 is also telling us about the time needed of our code from experiment 1 until 5.

Note that in this paper the researchers try to offer an alternative method in the process of making the final semester exam schedule. The algorithm that the researcher proposed is the development of Komosko *et al.* (2016), which still has some weaknesses. The researcher think that this method has more benefits if applied on a smaller scale at Binus University or other universities. However, this method still needs to be developed so that it can generate schedule variations for users, so that they can choose one of the schedule variations provided by the algorithm.

Table 1: Bit operation of OR (∨) and (∧)

x	y	$x \vee y$	$x \wedge y$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

Table 2: Time needed converting raw data into adjacency matrix and time needed bitwise graph coloring

Experiment	Time	Average time	Experiment	Time	Average time
1	1.149		1	0.812	
2	1.247		2	0.685	
3	1.056	1.199	3	0.599	0.675
4	1.238		4	0.653	
5	1.301		5	0.619	

Table 3: Raw data of TI Mat-Stat lecture

Academic program	Campus	Binusian ID	Student ID	Official name	Semester	Course code	Long title	Class section	Class Nbr	SKS theory	SKS peacticum
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	LANG6027	Indonesian	LA06	12845	2	0
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	STAT6175	Statistical methods For data science	LA06	15904	2	0
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	COMP6048	Data structures	LA06	11413	4	2
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	COMP6048	Data structures	LA06	11413	4	2
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	COMP6048	Data structures	TA06	11415	4	2
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	MATH6039	Calculus ll	LA06	13110	4	0
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	MATH6039	Calculus ll	LA06	13110	4	0
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	EESE2	EESE2	LA06	15039	0	0
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	COMP6048	Data structures	BA06	12302	4	2
DBCST	CSKMG	BN12394421	2440115956	NICHOLAS LAVENDO TANERI	2	ENGL6129	English savvy	LA06	12513	0	0

Conclusion and Open Problem

The conclusion of this research is as follows. Bitwise coloring can be used for scheduling semester exams. From the program, 8 time slots are needed. The time used to run this program is 0.675 sec. With this short time, these results can be used as consideration for the scheduling officer at Binus University in preparing the semester exam schedule. The results in this study can also be used and developed in other institutions. The open problem of this research is how to produce output in the form of schedule variations according to what the researchers want. The researchers believe that if the researcher can generate permutations of vertices of the same degree, then our problem can be shed light on. These two pieces of information will close this study.

Acknowledgement

The researchers are very grateful to experts for their appropriate and constructive suggestions to improve this study. The researchers also wish to thank Bina Nusantara University. Binus international research grant facilities, funds and infrastructure were used to support this endeavor. Thank you very much for your help during this difficult time.

Author's Contributions

Indra Bayu Muktyas and Samsul Arifin: Coding the python program,

writing and finalize the manuscript.

Wikky Fawwaz Al Maki and Mohd Khairul Bazli Mohd Aziz: Simulating the data, tidy up the theoretical basis and the methods we use

Ethics

This article is original and contains unpublished material. The corresponding author confirms that there is no conflict of interest in this paper and no ethical issues involved.

References

Budiman, H. (2007). Penerapan Graph Colouring untuk Merencanakan Jadwal. Tersedia: http://www. informatika. org/rinaldi/Matdis/2007/2008/Makalah/MakalahIF2153-0708-025. pdf.

Arifin, S., & Muktyas, I. B. (2021, April). Generate a system of linear equation through unimodular matrix using Python and Latex. In AIP Conference Proceedings (Vol. 2331, No. 1, p. 020005). AIP Publishing LLC. doi.org/10.1063/5.0041651

Arifin, S., Muktyas, I. B., & Sukmawati, K. I. (2021, February). Product of two groups integers modulo m, n and their factor groups using python. In Journal of Physics: Conference Series (Vol. 1778, No. 1, p. 012026). IOP Publishing. doi.org/0.1088/1742-6596/1778/1/012026

Bustan, A. W., & Salim, M. R. (2019). Penerapan Pewarnaan Graf Menggunakan Algoritma Welch-Powell Untuk Menentukan Jadwal Bimbingan Mahasiswa. Jurnal THEOREMS (The Original Research of Mathematics), 4(1), 79-86. https://core.ac.uk/download/pdf/228883523.pdf

Chollet, F. (2021). Deep Learning with Python, Second Edition. United States: Manning.

Firdaus, M. A. (2020). Aplikasi pewarnaan Graf menggunakan Algoritma Welch-Powell pada penyusunan jadwal mata kuliah prodi matematika UIN Sunan Ampel Surabaya (Doctoral dissertation, Uin Sunan Ampel Surabaya). http://digilib.uinsby.ac.id/42168/

Gravetter, F. J., & Forzano, L. B. (2012). Research methods for the behavioral sciences 4th edition. Belmont: Cengage Learning.

Muktyas, I. B. (2010). Program Pewarnaan Graf untuk Pemodelan Penjadwalan Ujian Semester di Jurusan Matematika Unnes (Doctoral dissertation, Universitas Negeri Semarang). lib.unnes.ac.id/13117/

Jioudi, B., Sabir, E., Moutaouakkil, F., & Medromi, H. (2019, November). A Congestion Game Analysis for Route-Parking Selection with Dynamic Pricing Policies. In International Symposium on Ubiquitous Networking (pp. 171-181). Springer, Cham.

Rosen, K. H. (2019). Discrete Mathematics and its Applications, 8th ed. New York, USA: McGraw-Hill Education, 2019.

Klotz, W. (2002). Graph coloring algorithms (pp. 1-9). Verlag nicht ermittelbar. https://www.mathematik.tu-clausthal.de/fileadmin/AG-GraphentheorieKombinatorik/papers/gca.pdf

Komosko, L., Batsyn, M., San Segundo, P., & Pardalos, P. M. (2016). A fast greedy sequential heuristic for the vertex colouring problem based on bitwise operations. Journal of Combinatorial Optimization, 31(4), 1665-1677. doi.org/10.1007/s10878-015-9862-1

Malkawi, M., Hassan, M. A. H., & Hassan, O. A. H. (2008). A New Exam Scheduling Algorithm Using Graph Coloring. International Arab Journal of Information Technology (IAJIT), 5(1). https://iajit.org/PDF/vol.5,no.1/11-68.pdf

Muktyas, I. B., Sulistiawati, & Arifin, S. (2021, April). Digital image encryption algorithm through unimodular matrix and logistic map using Python. In AIP Conference Proceedings (Vol. 2331, No. 1, p. 020006). AIP Publishing LLC. doi.org/10.1063/5.0041653

Oliphant, T. E. (2007). Python for scientific computing. Computing in science & engineering, 9(3), 10-20. doi.org/10.1109/MCSE.2007.58

Rahman, B., Arifin, S., & Muktyas, I. B. (2019). All Cyclic Subgroups In Group (ZmxZn,+) Using Python. Int. J. Sci. Technol. Res., 8, 2282-5.

Rosen, K. H., & Krithivasan, K. (2012). Discrete mathematics and its applications: With combinatorics and graph theory. Tata McGraw-Hill Education.

Project Teams, T. N. (2020). INSTALLING NUMPY, Community, the NumPy and wider scientific Python. https://numpy.org/install/

Widodo, A. (2016). Teori Graf. Universitas Brawijaya Press.

Xiaoying, S. (2019, October). A greedy approach for TDMA based on matrix operation. In 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 284-287). IEEE. doi.org/10.1109/CyberC.2019.00055