Available online at ijci.wcci-international.org

International Journal of Curriculum and Instruction 14(2) (2022) 1166–1188

IJCI
International Journal of Curriculum and Instruction

Using graph coloring for effective timetable scheduling at ordinary secondary level

Emmanuel Deogratias [a] *

[a] *Department of Mathematics and Statistics, University of Dodoma, P.O.Box 259, Dodoma, Tanzania*

Abstract

The purpose of this study was to assess the effectiveness of timetable scheduling that was developed using graph coloring for the class period time tabling. This study presents a study of using graph coloring for effective timetable scheduling at ordinary secondary level, a case study of Dodoma central secondary school in Dodoma city. Algorithms for timetable scheduling using graph coloring were developed, the training for academic teachers on how to design a class periods timetable using graph coloring was offered, and the developed class timetable through teachers' opinions on the effectiveness of the new timetable was assessed. It was found that the new timetable was effective because it eliminated collisions among teachers while using the timetable. This finding has implications on teaching and learning process by using an effective timetable preparation and implementation.

Keywords: Graph colouring; Python programming language; Timetabling; secondary school

1. Introduction

Teachers at schools in Tanzania use timetable scheduling to attend classes and for invigilation of examinations (mid-term, terminal and annual examinations). However, there have been a challenge of preparing this timetable (Carter, Laporte & Lee, 1996). One of the challenges is overlapping of teaching class periods. The same teacher is allocated at two or more class periods at the same time (Burke, Elliman, & Weare, 1994). Another challenge is that some schools in Tanzania still using traditional approach of preparing timetable such as using excel and MS word. This study used graph coloring to eliminate these challenges during teaching and examination invigilation by formulating an algorithm using python programming language. Academic teachers at one of

* Corresponding author name. Phone.: +0-000-000-0000
E-mail address: author@institution.xxx

secondary schools in Dodoma were also trained on how to use graph coloring and python programming language to prepare timetable.

Main objective of this study was to assess effectiveness of a new class timetable that was developed using graph coloring after being used at ordinary secondary school level. The following were specific objectives:

- To formulate an algorithm using graph coloring for effective timetable scheduling at ordinary secondary school level.
- To develop timetable scheduling of the class periods using graph coloring.
- To assess how the developed class timetable using graph coloring is effective in teaching and learning process.
- To provide training to teachers on how to prepare class timetable using graph coloring.

The intention of this study was to provide knowledge and skills to teachers on how to prepare class timetable that will be friendly for teaching and learning. Also, this study informs teachers in secondary schools on how they can design class timetable using graph coloring to avoid collision of the class periods among teachers as well as to avoid conflicts among teachers while using class timetable scheduling.

The study focused on exploring the following research questions:

- How an algorithm can be formulated using graph coloring and python programming language for effective timetable scheduling at ordinary secondary school level?
- How can timetable scheduling of the class periods be developed using graph coloring?
- How can the developed timetable using graph coloring be assessed for effective teaching and learning process?
- How can teachers be trained on how to prepare class timetable using graph coloring?

2. An informative literature reviews

Discuss Graph coloring is the process of assigning colors to the vertices of the graph so that no two adjacent vertices are assigned the same color (Deo, 1990). While assigning colors to an object, the focus is on its vertices, edges and faces. However, the key object is the vertex coloring because other objects can be transformed into vertex version graph coloring problems.

While coloring the graphs, the assumption is that graphs are connected in the sense that all components of the graphs are connected and can be colored dependently.

Graph coloring is applicable in various complex problems, including optimization (Miner, Elmohamed & Yau,1995). For example, we can use graph coloring to optimize partition of mutually exclusive events. The same approach can be used to focus on

developing and exploring time tabling and class time tabling at ordinary secondary level. This approach can be used because graph coloring is a heuristic algorithm which deals with timetable scheduling to satisfy the teaching and learning requirements involving subjects demand and their combination. Subject conflict graph is constructed in such a way that subjects represent nodes while edges represent conflicting subjects having common students (Welsh & Powell, 1967).

Various work has been done by different scholars on the problem of subject scheduling by using graph coloring (Carter, 1986; Kiaer & Yellen, 1992). The authors used the relationship between time tabling and graph coloring to solve (or approximately solve) the minimum coloring problem more efficiently. They were also successfully in coloring graphs that arise from time tabling problem, more specifically examination time tabling problems.

Welsh and Powell (1967) illustrated the relationship between time tabling and graph coloring and developed a new general graph coloring algorithm to solve the minimum coloring problem more efficiently.

Dutton and Bingham (1981) also introduced two of the most popular heuristic graph coloring algorithms. Considering each color one by one, a clique is formed by continually merging the two vertices with the most common adjacent vertices. On completion, identical coloring is applied to all the vertices which are merged into the same.

Through developing color algorithms, other scholars were able to use this concept to solve problem of time tabling. For instance, Carter (1986) conducted an examination time tabling survey. The developed examination time tabling schedule using graph coloring was accepted and used by many educational institutions to solve their examination timetabling problems.

According to Carter in his survey, his work is significant as it has the objective of obtaining “conflict-free” schedules, given a fixed number of time periods turned out to be one of the most complex timetabling applications.

Kiaer and Yellen (1992) also described heuristic algorithm using graph coloring approach to find approximate solutions for a University course timetabling problem. The algorithm using a weighted graph to model the problem aimed at finding a least cost K-coloring of the graph. K represents the number of available time slots while minimizing conflicts.

From the literature review, it was found that all researchers focused on timetable scheduling for colleges and for universities, but this project focused on assessing how graph coloring can be effective for timetable scheduling at ordinary secondary school level. Also, no research that has been conducted in Tanzanian ordinary secondary schools to investigate the effectiveness of timetable scheduling using graph coloring. Furthermore, there have been software used in preparing timetable in Tanzania including using python. This is usually done to schools in urban areas where there is availability of electricity and internet access. However, most of secondary school teachers

in rural areas use manual, MS word and excel to prepare class timetable. For example, teachers in Dodoma Central secondary school still use excel to prepare class timetable. This study used graph coloring for preparing class timetable for ordinary secondary school level.

3. Method

This was a qualitative study in nature (Merriam, 1988, 1998, 2009). In this study qualitative case study was used because the study focused on opinions of the teachers after using the new developed timetable using graph coloring and python programming language.

Two academic teachers at Dodoma central secondary school was trained on how to use graph coloring and python programming language to develop class timetable. A new class timetable was developed using graph coloring and python programming language and implemented in the class for a duration of two weeks. After that, five teachers were asked to bring their opinions on how the new timetable informed their teaching in the classrooms.

In this study the data were collected by using prepared reflective questions (Appendix A) and using tape recorders to gather opinions from the teachers after using the new timetable.

Content analysis was used to analyze data gathered using tape recorders. Content analysis involved three steps: preparing data, organizing data, and reporting results (Elo et al., 2014). Also, python programming language was used to develop algorithms for timetable construction by generating codes (see Appendix B).

The reliability of this study was checked and found that since the same procedures and process for preparing timetable using graph coloring were used by academic teachers after training, then the results are reliable. Also, the validity of the study was checked and found that the academic teachers were able to develop a class timetable after training using graph coloring and python programming language.

Ethical considerations revealed in this study including applying a letter of permission for data collection at Dodoma central secondary school, teachers at Dodoma central school were asked to volunteer participating in this study, and teachers were asked for the consent before conducting this study.

4. Results

4.1 Developed algorithms for class timetabling using graph coloring

According to this study of using graph coloring for effective timetable scheduling, the general codes were constructed that were used to develop the general timetable for

Dodoma central secondary school. The following are the algorithms that were developed using python programming language and later were used to construct class timetable.

i. Import the necessary library
   - Pathlib
   - Csv (Comma separated value)

ii. Creating empty list for
   - Subjects
   - Class list
   - Starting hour for first subject-7
   - Defining next subject hour-8
   - Defining school days
     - Monday
     - Tuesday
     - Wednesday
     - Thursday
     - Friday

iii. Function to fill the subjects
   // Ask user the subject and fill in subject list
   // Accept user input (in string format)
   // Separate the subject list by comma
   // Loop the entire list {
   Loop the subject list and transform in capital letter
   If (not a subject in list) {
   Append the subject

iv. Function for planning time () {
   // Ask hour to user
   // Output the subject list first
   // Specify time for subject
   Let user to enter the subject at the specified time
   Return user answer
   }

v. Function to fill the subjects
   // Ask user the subject and fill in subject list
   // Accept user input (in string format)
   // Separate the subject list by comma
   // Loop the entire list {
   Loop the subject list and transform in capital letter
   If (not a subject in list) {
   Append the subject

vi. Function for planning time () {

// Ask hour to user
// Output the subject list first
// Specify time for subject
Let user to enter the subject at the specified time
Return user answer
}

vii. Function for filling timetable () {
// Allocate the school days
// Loop when time is less to 4
Specify format
//If time is midday, then it is **break time**
// If not then continue to fill subject for that day
}

viii. Save data to excel format
Function for saving and writing records () {
//Recall the list
// Fill into specified file path
//Notify user if data is already created
}

4.2 Developed class timetable using graph coloring

From the developed algorithms using python programming language, a new class timetable was developed as described below.

Table 1. Timetable for Dodoma central secondary school

Day	Time	Period	Form 1	Form II	Form III	Form IV
Monday	8h-9h	1	Chemistry	English	Mathematics	Kiswahili
	9h-10h	2	Chemistry	English	Mathematics	Kiswahili
	10h-11h	3	Break Time	Break Time	Break Time	Break Time
	11h-12h	4	Civics	Kiswahili	Geography	Biology
Tuesday	8h-9h	1	Mathematics	Kiswahili	English	Geography
	9h-10h	2	Mathematics	Biology	English	Geography
	10h-11h	3	Break Time	Break Time	Break Time	Break Time
	11h-12h	4	English	Mathematics	History	Mathematics
Wednesday	8h-9h	1	Physics	English	History	Civics
	9h-10h	2	History	English	Kiswahili	History
	10h-11h	3	Break Time	Break Time	Break Time	Break Time
	11h-12h	4	Kiswahili	Geography	Physics	History

Thursday	8h-9h	1	Kiswahili	Geography	Geography	Kiswahili
	9h-10h	2	physics	geography	biology	Kiswahili
	10h-11h	3	Break Time	Break Time	Break Time	Break Time
	11h-12h	4	English	Civics	Biology	Mathematics
Friday	8h-9h	1	English	Physics	Geography	History
	9h-10h	2	Geography	Chemistry	Mathematics	Physics
	10h-11h	3	Break Time	Break Time	Break Time	Break Time
	11h-12h	4	Geography	Biology	Chemistry	Chemistry

4.3 The effectiveness of the class timetable developed using graph coloring in teaching and learning process

The new class timetable was developed and sent to the Dodoma Central Secondary School. After that, teachers at school started to use this timetable immediately, and this class timetable was implemented for two weeks. This is because students were approaching to start terminal examinations.

At the end of the study, opinions from teachers were collected using reflective questions to assess effectiveness of our timetable. The following are examples of the participants' reflections:

- "The new timetable is good as compared to the old one" (Teacher 1)
- "There is new change since the new timetable has tried to resolve the problem of collisions of class periods." (Teacher 2)
- The new timetable has helped teachers to resolve the issue of collisions of class periods "since all subjects are arranged clearly in the new timetable through following the class periods properly as well as subject teachers." (Teacher 3)

Based on the above participants' opinions, all five teachers responded that the new class timetable was good and friendly to them, this is because it helped to minimize collision of class periods.

4.4 Training academic teachers on how to prepare class timetable using graph coloring

At the beginning of this study, two academic teachers at Dodoma central school were asked if they had heard about preparing class timetable using graph coloring and python programming language. The academic teachers said no, they used only excel to prepare school timetable. See appendix C which is the old class timetable for teaching students in Form I to Form IV classes.

After that, the academic teachers were taught on how to prepare class timetable using graph coloring while using python programming language in developing codes. At the end of the training, python programming software was installed in teachers' computers so that they can practice and be competent to prepare the timetable by using graph coloring in the future. Also, teachers reflected on their learning after being trained on how to prepare timetable by using graph coloring and python programming language.

Some of their reflections are:

- "I have learned and get new knowledge on how to prepare class timetable by using python software, and I am able to use this approach for class timetable preparation" (Academic teacher 1)
- "I have surprised to see new and interesting approach to timetable construction whereby you enter the subjects into the program, the timetable is generated automatically" (Academic teacher 2)
- "Also, your timetable developed using graph coloring and implemented by python programming language has helped to reduce collision of class periods." (Academic teacher 2)
- "I didn't know how to use python programming language and graph coloring in constructing a new timetable and for now I know how to prepare it." (Academic teacher 1)

From these reflections, we can see that using graph coloring to prepare class timetable was potential for effective teaching and learning process at Dodoma Central secondary school.

5. Conclusions

This study gives a lesson that nothing is impossible, what matter is provision of knowledge and skills. Academic teachers were not aware of using graph coloring to develop class timetable. However, after giving a training, the teachers were able to do so and promise to practice it in preparing school timetabling using graph coloring and python programming language. It was found that the developed class timetable using graph coloring was effective in teaching and learning process because there was no collision of class periods.

In closing, this study recommends that teachers at schools should use graph coloring for preparing class period timetabling because of avoiding collision. Also, further study should be conducted at Tanzanian's schools on using graph coloring for examination timetabling.

Acknowledgements

The author acknowledges seven teachers at Dodoma Central Secondary School for active participation in this study.

References

Burke, K., Elliman, G., & Weare, R. (1994). A university timetabling system based on graph coloring and constraint manipulation. *Journal of Research on Computing in Education, 27*(1), 1-18.

Carter, M. (1986). Or practice—a survey of practical applications of examination timetabling algorithms. *Computers and Operations Research, 34*(2), 193—202.

Carter, W., Laporte, G., & Lee, Y. (1996). Examination timetabling: Algorithmic strategies and applications. Journal of the Operational Research Society 47(3), 373–383.

Deo, N. (1990). Graph theory with applications to engineering and computer Science. Prentice Hall of India.

Dutton, R., & Brigham, R. (1981). *A new graph coloring algorithm. Computer Journal, 24*(1), 85–86.

Eto, S., Kaäriaïnen, M., Kanste, O., Poölkki, T., Utriainen, K., & Kyngaös, H. (2014). Qualitative content analysis: A focus on trustworthiness. SAGE Open, 1—10. doi: 10.1177/2158244014522633

Kiaer, L., & Yellen, J. (1992). Weighted graphs and university timetabling. *Computers and Operations Research, 19*(1), 59-67.

Merriam, S. (1988). Case study research in education: A qualitative approach. San Francisco, CA: Jossey-Bass Publishers.

Merriam, S. (1998). Qualitative research and case study applications in education. San Francisco, NY: California.

Merriam, S. B. (2009). Qualitative research a guide to design and implementation. San Francisco, CA: Jossey-Bass.

Miner, K., Elmohamed, S., & Yau, W. (1995). Optimizing timetabling solutions using graph coloring. Syracuse University.

Welsh, A., & Powell, B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal,10*(1), 85-86.

Appendix A: Reflective questions

1. What can you say about this new class timetable?
2. Is there any change you have noticed after using a new class timetable? If yes, please explain for me.
3. What can you say after using a new class timetable from the previous one? Please explain for me.
4. How do you think the new timetable has helped you to achieve your teaching goal(s)? Please explain for me.
5. What surprised you after using this new timetable. Please explain for me.
6. What have you learned that you did not know before using this new timetable? Please explain for me.
7. Do you think this new timetable has helped you and other teachers to resolve the issue of collision of class periods? How?
8. 8.How would you recommend for a new class timetable to other teachers?

Appendix B: Codes generated while developing algorithms for class timetabling using python programming language

Code1: constructing form one timetable

```
import pathlib
import csv

subjects_list = []
class_list=[]
start_hour = 7
next_hour = 8
school_days = [
   'monday',
   'tuesday',
   'wednesday',
   'thursday',
   'friday'
]
time_slot_list = []
subject_per_slot = {}
MAX_HOUR_PER_SUBJECT = 6
subject_hour_count = {}
# # ==================enter the list of classes contained to the school==============
# def fill_out_class_list():PYT
#   """Ask user classes and fill in classes list"""

#   classes = input('Type all class you want
# and separate them by comma: ')

#   the_classes = classes.replace(', ', ',')
#   the_classes = the_classes.split(',')

#   for clas in the_classes:
```

```
#      clas = classes.capitalize()
#   for x in the_classes:
#      print(x)

#      if not clas in class_list:
#         class_list.append(clas)
#         subject_hour_count[clas] = MAX_HOUR_PER_SUBJECT

# ==============================end of
class=======================================

def fill_out_subjects_list():
   print("==============SCHOOL TIME TABLE FOR FORM
ONE======================")
   """Ask user subjects and fill in subjects list"""
   subjects = input('Type all subjects you want add in subjects list\
and separate them by comma: ')
   the_subjects = subjects.replace(', ', ',')
   the_subjects = the_subjects.split(',')

   for subject in the_subjects:
      subject = subject.capitalize()

      if not subject in subjects_list:
         subjects_list.append(subject)
         subject_hour_count[subject] = MAX_HOUR_PER_SUBJECT

def ask_hour():
   """Ask hour to user"""
   print(f'Subjects list: {subjects_list}')
   print(f'class list: {class_list}')
   # form_one=print('Enter form one class to continue')

   print(f'Planning time: {start_hour}h-{next_hour}h')
   user_answer = input('What\'s subject do you want put here? ')

   return user_answer

def fill_in_timetable():
    # print("Enter the subjects for  Form ome (1) Students")
   """Display an hour & ask user which subject he want to put there"""
   global start_hour
   global next_hour

   for day in school_days:

      the_hour = {}
      time = 0
      start_hour = 8
      next_hour = 9
```

```
        print('\n---------------------------')
        print(f'{day.capitalize()} timetable')
        print('---------------------------\n')

        while time < 4:

            hour_format = f'{start_hour}h-{next_hour}h'
            # it's represent 8 hours/per day for school
            if time == 2: # if it's a midday (12.am), make a break
                # Add a break in timetable with 'Break time' as inscription
                subject_per_slot[hour_format] = ['Break time']

                # Add hour format while making sure we avoid duplicate
                if not hour_format in time_slot_list:
                    time_slot_list.append('hour_format')

            else:
                chosen_subject = ask_hour().capitalize()
                print(f'start_hour: {start_hour}')
                print(f'next_hour: {next_hour}')

                # Check that subject chosen by user is in subjects list
                while not chosen_subject in subjects_list:
                    print(f'{chosen_subject} is not in subjects list.')
                    print('Choose another subject.')
                    chosen_subject = ask_hour().capitalize()

                # Add hour format while making sure we avoid duplicate
                if not hour_format in time_slot_list:
                    time_slot_list.append(hour_format)
                    subject_per_slot[hour_format] = [chosen_subject]
                else:
                    subject_per_slot[hour_format] += [chosen_subject]

                # Check that chosen subject max hours didn't reached
                for subject, max_hour in subject_hour_count.items():
                    if chosen_subject == subject:
                        # remove one hour on subject max hour
                        subject_hour_count[chosen_subject] = max_hour - 1

            # go to next hour
            start_hour += 1
            next_hour += 1
            time += 1


# fill_out_class_list()
fill_out_subjects_list()
fill_in_timetable()
```

```
print(f'Subject per slot: {subject_per_slot}')

timetable_path = pathlib.Path.cwd() / 'form_one_timetable.csv'

with open(timetable_path, 'w') as timetable_file:
    timetable_writing = csv.writer(timetable_file)

    # Write headers into csv file
    csv_kichwa_kikuuu=['DODOMA CENTRAL SECONDARY SCHOOL - O LEVEL TIME
TABLE 2021']
    csv_kichwa_kikuuu=['FORM ONE']
    csv_headers = ['Hours']
    csv_headers.extend(school_days)
    timetable_writing.writerow(csv_kichwa_kikuuu)
    timetable_writing.writerow(csv_headers)

    # Write content into csv file
    for time_slot, concerned_subjects in subject_per_slot.items():
        time_line = [time_slot]
        concerned_subjects_list = []

        if concerned_subjects == ['Break time']:
            for x in range(0, len(school_days)):
                concerned_subjects_list.append('Break time')
        else:
            concerned_subjects_list = concerned_subjects

        final_line = time_line + concerned_subjects_list
        timetable_writing.writerow(final_line)
    print('Your form one timetable is ready')

Code for constructing Form two timetable
import pathlib
import csv

subjects_list = []
class_list=[]
start_hour = 7
next_hour = 8
school_days = [
    'monday',
    'tuesday',
    'wednesday',
    'thursday',
    'friday'
]
time_slot_list = []
subject_per_slot = {}
MAX_HOUR_PER_SUBJECT = 6
subject_hour_count = {}
```

```
# =================enter the list of classes contained to the school==============
def fill_out_class_list():
    """Ask user classes and fill in classes list"""

    classes = input('Type all class you want \
and separate them by comma: ')

    the_classes = classes.replace(', ', ',')
    the_classes = the_classes.split(',')

    for clas in the_classes:
        clas = classes.capitalize()
    for x in the_classes:
        print(x)

        if not clas in class_list:
            class_list.append(clas)
            subject_hour_count[clas] = MAX_HOUR_PER_SUBJECT

# ==============================end of
class=======================================

def fill_out_subjects_list():
    print("==============SCHOOL TIME TABLE FOR FORM TWO
2021======================")
    """Ask user subjects and fill in subjects list"""
    subjects = input('Type all subjects you want add in subjects list and separate them by
comma: ')
    the_subjects = subjects.replace(', ', ',')
    the_subjects = the_subjects.split(',')

    for subject in the_subjects:
        subject = subject.capitalize()

        if not subject in subjects_list:
            subjects_list.append(subject)
            subject_hour_count[subject] = MAX_HOUR_PER_SUBJECT

def ask_hour():
    """Ask hour to user"""
    print(f'Subjects list: {subjects_list}')
    print(f'class list: {class_list}')
    # form_one=print('Enter form one class to continue')

    print(f'Planning time: {start_hour}h-{next_hour}h')
    user_answer = input('What\'s subject do you want put here? ')

    return user_answer

def fill_in_timetable():
```

```
 # print("Enter the subjects for  Form ome (1) Students")
"""Display an hour & ask user which subject he want to put there"""
global start_hour
global next_hour

for day in school_days:

    the_hour = {}
    time = 0
    start_hour = 8
    next_hour = 9

    print('\n---------------------------')
    print(f'{day.capitalize()} timetable')
    print('--------------------------\n')

    while time < 4:

        hour_format = f'{start_hour}h-{next_hour}h'
        # it's represent 8 hours/per day for school
        if time == 2: # if it's a midday (12.am), make a break
            # Add a break in timetable with 'Break time' as inscription
            subject_per_slot[hour_format] = ['Break time']

            # Add hour format while making sure we avoid duplicate
            if not hour_format in time_slot_list:
                time_slot_list.append('hour_format')

        else:
            chosen_subject = ask_hour().capitalize()
            print(f'start_hour: {start_hour}')
            print(f'next_hour: {next_hour}')

            # Check that subject chosen by user is in subjects list
            while not chosen_subject in subjects_list:
                print(f'{chosen_subject} is not in subjects list.')
                print('Choose another subject.')
                chosen_subject = ask_hour().capitalize()

            # Add hour format while making sure we avoid duplicate
            if not hour_format in time_slot_list:
                time_slot_list.append(hour_format)
                subject_per_slot[hour_format] = [chosen_subject]
            else:
                subject_per_slot[hour_format] += [chosen_subject]

            # Check that chosen subject max hours didn't reached
            for subject, max_hour in subject_hour_count.items():
                if chosen_subject == subject:
                    # remove one hour on subject max hour
```

```
                subject_hour_count[chosen_subject] = max_hour - 1

            # go to next hour
            start_hour += 1
            next_hour += 1
            time += 1

# fill_out_class_list()
fill_out_subjects_list()
fill_in_timetable()
print(f'Subject per slot: {subject_per_slot}')

timetable_path = pathlib.Path.cwd() / 'form_two_timetable.csv'

with open(timetable_path, 'w') as timetable_file:
    timetable_writing = csv.writer(timetable_file)

    # Write headers into csv file
    csv_kichwa_kikuuu=['DODOMA CENTRAL SECONDARY SCHOOL - O LEVEL TIME
TABLE 2021']
    csv_kichwa_kikuuu=['FORM TWO']
    csv_headers = ['Hours']
    csv_headers.extend(school_days)
    timetable_writing.writerow(csv_kichwa_kikuuu)
    timetable_writing.writerow(csv_headers)

    # Write content into csv file
    for time_slot, concerned_subjects in subject_per_slot.items():
        time_line = [time_slot]
        concerned_subjects_list = []

        if concerned_subjects == ['Break time']:
            for x in range(0, len(school_days)):
                concerned_subjects_list.append('Break time')
        else:
            concerned_subjects_list = concerned_subjects

        final_line = time_line + concerned_subjects_list
        timetable_writing.writerow(final_line)
    print('Your form two timetable is ready')


Code for constructing Form three timetable
import pathlib
import csv

subjects_list = []
class_list=[]
start_hour = 7
next_hour = 8
```

```
school_days = [
   'monday',
   'tuesday'
]
time_slot_list = []
subject_per_slot = {}
MAX_HOUR_PER_SUBJECT = 6
subject_hour_count = {}
# ================enter the list of classes contained to the school============
def fill_out_class_list():
   """Ask user classes and fill in classes list"""

   classes = input('Type all class you want \
and separate them by comma: ')

   the_classes = classes.replace(', ', ',')
   the_classes = the_classes.split(',')

   for clas in the_classes:
      clas = classes.capitalize()
   for x in the_classes:
      print(x)

      if not clas in class_list:
         class_list.append(clas)
         subject_hour_count[clas] = MAX_HOUR_PER_SUBJECT

# ============================end of
class=======================================

def fill_out_subjects_list():
   """Ask user subjects and fill in subjects list"""

   subjects = input('Type all subjects you want add in subjects list\
and separate them by comma: ')

   the_subjects = subjects.replace(', ', ',')
   the_subjects = the_subjects.split(',')

   for subject in the_subjects:
      subject = subject.capitalize()

      if not subject in subjects_list:
         subjects_list.append(subject)
         subject_hour_count[subject] = MAX_HOUR_PER_SUBJECT

def ask_hour():
   """Ask hour to user"""
   print(f'Subjects list: {subjects_list}')
   print(f'class list: {class_list}')
```

```
    # form_one=print('Enter form one class to continue')

    print(f'Planning time: {start_hour}h-{next_hour}h')
    user_answer = input('What\'s subject do you want put here? ')

    return user_answer

def fill_in_timetable():
     # print("Enter the subjects for  Form ome (1) Students")
    """Display an hour & ask user which subject he want to put there"""
    global start_hour
    global next_hour

    for day in school_days:

        the_hour = {}
        time = 0
        start_hour = 8
        next_hour = 9

        print('\n---------------------------')
        print(f'{day.capitalize()} timetable')
        print('---------------------------\n')

        while time < 4:

            hour_format = f'{start_hour}h-{next_hour}h'
            # it's represent 8 hours/per day for school
            if time == 2: # if it's a midday (12.am), make a break
                # Add a break in timetable with 'Break time' as inscription
                subject_per_slot[hour_format] = ['Break time']

                # Add hour format while making sure we avoid duplicate
                if not hour_format in time_slot_list:
                    time_slot_list.append('hour_format')

            else:
                chosen_subject = ask_hour().capitalize()
                print(f'start_hour: {start_hour}')
                print(f'next_hour: {next_hour}')

                # Check that subject chosen by user is in subjects list
                while not chosen_subject in subjects_list:
                    print(f'{chosen_subject} is not in subjects list.')
                    print('Choose another subject.')
                    chosen_subject = ask_hour().capitalize()

                # Add hour format while making sure we avoid duplicate
                if not hour_format in time_slot_list:
                    time_slot_list.append(hour_format)
```

```
                subject_per_slot[hour_format] = [chosen_subject]
            else:
                subject_per_slot[hour_format] += [chosen_subject]

            # Check that chosen subject max hours didn't reached
            for subject, max_hour in subject_hour_count.items():
                if chosen_subject == subject:
                    # remove one hour on subject max hour
                    subject_hour_count[chosen_subject] = max_hour - 1

        # go to next hour
        start_hour += 1
        next_hour += 1
        time += 1

# fill_out_class_list()
fill_out_subjects_list()
fill_in_timetable()
print(f'Subject per slot: {subject_per_slot}')

timetable_path = pathlib.Path.cwd() / 'form_three_timetable.csv'

with open(timetable_path, 'w') as timetable_file:
    timetable_writing = csv.writer(timetable_file)

    # Write headers into csv file
    csv_kichwa_kikuuu=['DODOMA CENTRAL SECONDARY SCHOOL - O LEVEL TIME
TABLE 2021']
    csv_kichwa_kikuuu=['FORM THREE']
    csv_headers = ['Hours']
    csv_headers.extend(school_days)
    timetable_writing.writerow(csv_kichwa_kikuuu)
    timetable_writing.writerow(csv_headers)

    # Write content into csv file
    for time_slot, concerned_subjects in subject_per_slot.items():
        time_line = [time_slot]
        concerned_subjects_list = []

        if concerned_subjects == ['Break time']:
            for x in range(0, len(school_days)):
                concerned_subjects_list.append('Break time')
        else:
            concerned_subjects_list = concerned_subjects

        final_line = time_line + concerned_subjects_list
        timetable_writing.writerow(final_line)
    print('Your form three timetable is ready')
```

```
Code for constructing Form four timetable
import pathlib
import csv

subjects_list = []
class_list=[]
start_hour = 7
next_hour = 8
school_days = [
    'monday',
    'tuesday',
    'wednesday',
    'thursday',
    'friday'
]
time_slot_list = []
subject_per_slot = {}
MAX_HOUR_PER_SUBJECT = 6
subject_hour_count = {}
# ==================enter the list of classes contained to the school==============
def fill_out_class_list():
    """Ask user classes and fill in classes list"""

    classes = input('Type all class you want \
and separate them by comma: ')

    the_classes = classes.replace(', ', ',')
    the_classes = the_classes.split(',')

    for clas in the_classes:
        clas = classes.capitalize()
    for x in the_classes:
        print(x)

        if not clas in class_list:
            class_list.append(clas)
            subject_hour_count[clas] = MAX_HOUR_PER_SUBJECT

# ==============================end of
class=======================================

def fill_out_subjects_list():
    print("==============SCHOOL TIME TABLE FOR FORM FOUR
2021======================")
    """Ask user subjects and fill in subjects list"""
    subjects = input('Type all subjects you want add in subjects list\
and separate them by comma: ')
    the_subjects = subjects.replace(', ', ',')
    the_subjects = the_subjects.split(',')
```

```
    for subject in the_subjects:
        subject = subject.capitalize()

        if not subject in subjects_list:
            subjects_list.append(subject)
            subject_hour_count[subject] = MAX_HOUR_PER_SUBJECT

def ask_hour():
    """Ask hour to user"""
    print(f'Subjects list: {subjects_list}')
    print(f'class list: {class_list}')
    # form_one=print('Enter form one class to continue')

    print(f'Planning time: {start_hour}h-{next_hour}h')
    user_answer = input('What\'s subject do you want put here? ')

    return user_answer

def fill_in_timetable():
     # print("Enter the subjects for  Form ome (1) Students")
    """Display an hour & ask user which subject he want to put there"""
    global start_hour
    global next_hour

    for day in school_days:

        the_hour = {}
        time = 0
        start_hour = 8
        next_hour = 9

        print('\n---------------------------')
        print(f'{day.capitalize()} timetable')
        print('---------------------------\n')

        while time < 4:

            hour_format = f'{start_hour}h-{next_hour}h'
            # it's represent 8 hours/per day for school
            if time == 2: # if it's a midday (12.am), make a break
                # Add a break in timetable with 'Break time' as inscription
                subject_per_slot[hour_format] = ['Break time']

                # Add hour format while making sure we avoid duplicate
                if not hour_format in time_slot_list:
                    time_slot_list.append('hour_format')
            else:
                chosen_subject = ask_hour().capitalize()
                print(f'start_hour: {start_hour}')
                print(f'next_hour: {next_hour}')
```

```
            # Check that subject chosen by user is in subjects list
            while not chosen_subject in subjects_list:
                print(f'{chosen_subject} is not in subjects list.')
                print('Choose another subject.')
                chosen_subject = ask_hour().capitalize()

            # Add hour format while making sure we avoid duplicate
            if not hour_format in time_slot_list:
                time_slot_list.append(hour_format)
                subject_per_slot[hour_format] = [chosen_subject]
            else:
                subject_per_slot[hour_format] += [chosen_subject]

            # Check that chosen subject max hours didn't reached
            for subject, max_hour in subject_hour_count.items():
                if chosen_subject == subject:
                    # remove one hour on subject max hour
                    subject_hour_count[chosen_subject] = max_hour - 1

        # go to next hour
        start_hour += 1
        next_hour += 1
        time += 1

# fill_out_class_list()
fill_out_subjects_list()
fill_in_timetable()
print(f'Subject per slot: {subject_per_slot}')

timetable_path = pathlib.Path.cwd() / 'form_four_timetable.csv'

with open(timetable_path, 'w') as timetable_file:
    timetable_writing = csv.writer(timetable_file)

    # Write headers into csv file
    csv_kichwa_kikuuu=['DODOMA CENTRAL SECONDARY SCHOOL - O LEVEL TIME
TABLE 2021']
    csv_kichwa_kikuuu=['FORM FOUR']
    csv_headers = ['Hours']
    csv_headers.extend(school_days)
    timetable_writing.writerow(csv_kichwa_kikuuu)
    timetable_writing.writerow(csv_headers)

    # Write content into csv file
    for time_slot, concerned_subjects in subject_per_slot.items():
        time_line = [time_slot]
        concerned_subjects_list = []

        if concerned_subjects == ['Break time']:
```

```
            for x in range(0, len(school_days)):
                concerned_subjects_list.append('Break time')
        else:
            concerned_subjects_list = concerned_subjects

        final_line = time_line + concerned_subjects_list
        timetable_writing.writerow(final_line)
    print('Your form four timetable is ready')
```

Appendix C: Old timetable used by Dodoma Central secondary school before implementing this study

Time	Period	Subject	Form	Day
7:00-8:10	1	Test	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
8:10-8:50		Mathematics	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
8:50-9:30	2	Mathematics	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
9:30-10:10	3	Kiswahili	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
10:10-10:50	4	Chemistry	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
10:50-11:10	5	Break	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
11:10-11:50	6	English	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
11:50-12:30	7	English	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday
12:30-13:10	8	Civics	I, II, III, IV	Monday, Tuesday, Wednesday, Thursday, Friday