UNIVERSITY TIME-TABLE SCHEDULING

Project Report

Submitted By-

KUSHAGRA KESERWANI (2K19/IT/070)

AND

ISHAAN JAGGI (2K19/IT/062)

DISCRETE STRUCTURES

Project 2020

—

IT - 205

—

Under the supervision of–

Ms. SWATI SHARDA

CERTIFICATE and ACKNOWLEDGEMENT

We, Kushagra Keserwani (Roll No – 2K19/IT/070) and Ishaan Jaggi (Roll No – 2K19/IT/062) , students of B.Tech. (INFORMATION TECHNOLOGY), hereby declare that the project Dissertation titled *"UNIVERSITY TIME-TABLE SCHEDULING"* which is submitted by us to the Department of INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfilment of the requirement for the completion and evaluation of marks of course IT-205 , is original and not copied from any source.

We would like to convey our heartfelt thanks to our teacher Ms. Swati Sharda for her ingenious ideas, tremendous help, suggestions and cooperation.

Kushagra Keserwani
Ishaan Jaggi

Date: 1-12-2020

ABSTRACT

EVERY SEMESTER OR YEAR, THE UNIVERSITIES AND COLLEGES ARE REQUIRED TO GENERATE A TIME TABLE FOR THE SMOOTH FUNCTIONING OF CLASSES AND FOR CONDUCTING THE INTERNAL AND THE FINAL SEMESTER EXAMS. THE PRESENCE OF A LARGE NUMBER OF STUDENTS AND A LARGE NUMBER OF OFFERED COURSES SOMETIMES MAKES IT DIFFICULT TO SCHEDULE THE EXAM WITHOUT HAVING ANY CONFLICT.

Time table generation is tedious job for educationalist with respect to time and man power. Providing a automatic time table generator will help to generate time table automatically. Proposed system of our project will help to generate it automatically also helps to save time. It avoids the complexity of setting and managing Timetable manually*. In our project* we are going to use the algorithm *NGC* to reduce these difficulties of generating timetable. This algorithm incorporate a numeral of strategy, aimed to improve the operation-speed of the search operation.

The system will take various inputs like :-

- In which year.
- Number of subjects.
- Name of each subject.
- Subject code.
- Number of credits per subject.
- Which subjects should not be scheduled at the same time.

By relying on these inputs, it will generate possible time tables for working days of the week. This will integrate by making optimal use of all resources in a way that will best suit the constraints.

TABLE OF CONTENTS

INTRODUCTION

Graph colouring is a simple way of labelling graph components such as vertices, edges, and regions under some constraints like, no two adjacent vertices, adjacent edges, or adjacent regions are coloured with ***minimum*** number of colours. The minimum number of colours needed for colouring a graph is known as ***chromatic number***.

PROBLEM STATEMENT :-

WE HAVE TO GENERATE THE TIME-TABLE FOR STUDENTS STUDYING IN THE UNIVERSITY FOR EACH YEAR.

1) The timetabling is a multi-dimensional assignment problem, in which students are assigned to courses and events are assigned to classrooms and timeslots.

2) The events are usually meeting between people at a particular location i.e. classroom.

3) NGC algorithm helps to generate Time-Table by using minimum no. of colours .

Firstly, the compiler asks the user to "***enter how many year's Time-Table user wants?*** " , then it asks to enter the ***no. of subjects***, ***name*** of subjects with subject ***codes and credits***, and then user asks "***whether the subjects enter can be scheduled in the same slot ?*** ". After entering these ***Inputs***, ***Output*** is generated as a Time-Table having Subjects with ***allotted room*** number.

CONSTRAINT = If more than one subjects are scheduled at the same time slot then the different room number is allotted to each subject. Another constraint is no. of slots available in a day and no. of credits per subject.

We have to take certain ***Inputs*** for generating Time-Table for each year in University :-

1. Adjacency matrix of student-course enrolment year wise.
2. No. of Subjects with their codes.
3. Subject Credits.
4. No. of slots per day.
5. No. of working days per week 5.
6. No. of rooms.

NOTE - Vertices in the graph represent the courses and the edges between them represent that both the courses connected with the edge can't be scheduled at the same time.

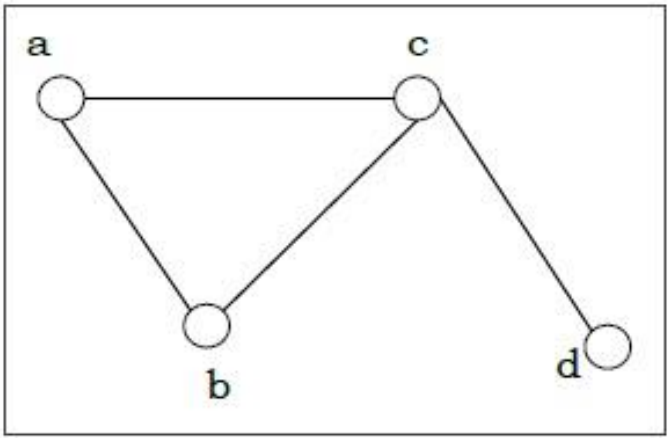
PROGRAMMING LANGUAGE USED IN PROJECT = C++

ABOUT GRAPHS

Definition – A graph (denoted as G=(V,E)) consists of a non-empty set of vertices or nodes V and a set of edges E.

Let us consider a graph G(V,E) where,

V={a,b,c,d} and E={{a,b},{a,c},{b,c},{c,d}}



Degree of a Vertex – The degree of a vertex V of a graph G (denoted by deg (V)) is the number of edges incident with the vertex V.

Vertex	Degree	Even / Odd
a	2	even
b	2	even
c	3	odd
d	1	odd

Degree of a Graph – The degree of a graph is the largest vertex degree of that graph. For the above graph the degree of the graph is 3.

GRAPH COLOURING

Graph colouring is the procedure of assignment of colours to each ***vertex*** of a graph G such that ***no adjacent vertices*** get same colour. The objective is to minimize the number of colours while colouring a graph. The smallest number of colours required to colour a graph G is called its ***chromatic number*** of that graph.

Method to Colour a Graph :-

The steps required to colour a graph G with n number of vertices are as follows :−

Step 1 − Arrange the vertices of the graph in some order.

Step 2 − Choose the first vertex and colour it with the first colour.

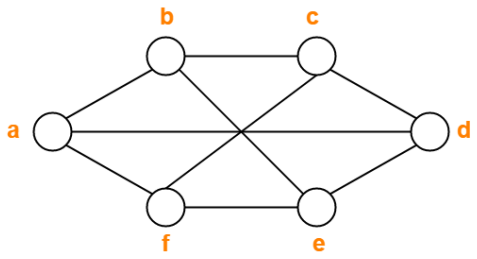
Step 3 − Choose the next vertex and colour it with the lowest numbered colour that has not been coloured on any vertices adjacent to it. If all the adjacent vertices are coloured with this colour, assign a new colour to it. Repeat this step until all the vertices are coloured.

Applications of Graph Colouring :-

- It is used for Register Allocation.
- Map Colouring.
- It is used foe Bipartite Graph Checking.
- Mobile Radio frequency Assignment.
- Making Time-Table, etc.

EXAMPLE FOR COLOURING THE GRAPH :-

Q) Find the chromatic no. of the following graph,



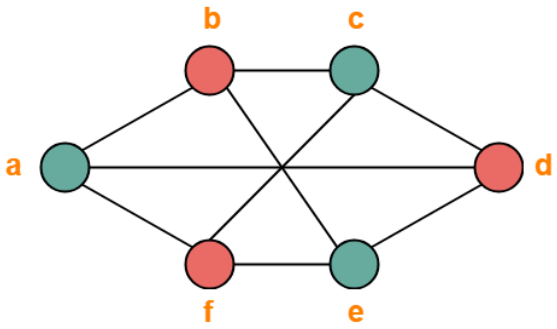
Sol.) Applying Greedy Algorithm we have,

Vertex	a	b	c	d	e	f
Colour	C1	C2	C1	C2	C1	C2

From here,

- Minimum number of colours used to colour the given graph are 2.
- Therefore, **Chromatic Number** of the given graph ***= 2.***

The given graph may be properly coloured using 2 colours as shown below :-



NGC ALGORITHM

The basic idea of the algorithm is as follows.

- We maintain a table having the four attributes namely, vertex, color, flag and color-assigned.
- The initial configuration of this table is shown in Fig. 1 for the augmented cube shown in Fig. 2. Here, we assume that the number of colors available is equal to the number of the nodes of the graph.
- At any iteration, we choose one vertex, say ‘x’ of the graph to color it. At the beginning of the iteration, we reset all the flag bits to 0.
- We now test every vertex adjacent to ‘x’ to examine whether it is colored. If it is so, we check its color and put the flag bit 1 corresponding to its color.
- We now scan the flag bit vector from the beginning to search for the first flag bit as 0. This indicates that the corresponding color has not yet been used.
- So we assign this color for the vertex ‘x’. We consider the next vertex and follow the same procedure to color it.
- The method is repeated until all the vertices of the graph are colored..

vertex	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	0	0	0	0	0	0	0	0	0
color-assigned									

Fig. 1. Initial table configuration

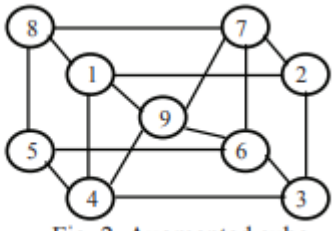


Fig. 2. Augmented cube

*TO UNDERSTAND NGC ALGORITHM EASILY LET US TAKE AN **EXAMPLE*** :-

1. We maintain a table having the four attributes namely, vertex, color, flag and color-assigned. The initial configuration of this table is shown in Fig. 1. We assume that the number of colors available is equal to the number of the nodes of the graph.

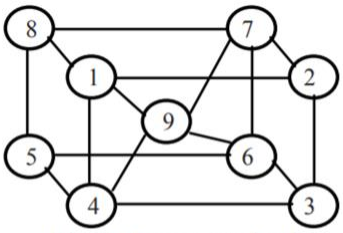


Fig. 2. Augmented cube

vertex	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	0	0	0	0	0	0	0	0	0
color-assigned									

Fig. 1. Initial table configuration

2. We randomly choose any one vertex from the graph. Without any loss of generality, we start coloring with the vertex 1. Initially all the flag bits are zero. This indicates that no color has been used so far. Therefore, we assign color 1 to the vertex 1 and set the corresponding flag bit 1. The situation is shown in Fig. 3.

node	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	1	0	0	0	0	0	0	0	0
color-assigned	1								

Fig. 3. Table configuration after 1st iteration

3. Now we move to the vertex 2. We first reset all the flag bits to 0 as shown in Fig. 4. The adjacent vertices of the vertex 2 are 1, 3 and 7 .

node	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	0	0	0	0	0	0	0	0	0
color-assigned	1								

Fig. 4. Table configuration after resetting flag bits to 0

We now consult the color-assigned vector in Fig. 4 and observe that out of these adjacent vertices the vertex 1 is colored with color 1, therefore we put 1 in the flag bit corresponding to color 1 (see Fig. 5). As the vertices 3 and 7 are not colored so far, their flag bits remain same.

4. Now to color vertex 2, we traverse flag vector from left to right to encounter the 1st zero flag bit, (which means that the corresponding color is not used). It is seen from Fig. 5 that flag bit corresponding to color 2 is 0. Therefore, we can assign color 2 to the vertex 2 and we update color-assigned vector as shown in Fig. 5.

node	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	1	0	0	0	0	0	0	0	0
color-assigned	1	2							

Fig. 5. Table configuration after 2nd iteration

5. Move to next vertex 3 and reset all the flag bits to 0 as shown in Fig. 6.

node	1	2	3	4	5	6	7	8	9
colors	1	2	3	4	5	6	7	8	9
flag	0	0	0	0	0	0	0	0	0
color-assigned	1	2							

Fig. 6. Table configuration after resetting flag bits to 0

Vertex 2, 6, 4 are adjacent to the vertex 3. Out of these, the vertex 2 is colored with color 2, therefore we put 1 in the corresponding flag bit (see Fig. 7) and search for the 1st zero flag bit from left to right fashion and find that the 1st flag bit is zero. Therefore, we can assign the color 1 to the vertex 3 as shown in Fig. 7.

node	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	0	1	0	0	0	0	0	0	0
color-assigned	1	2	1						

Fig. 7. Table configuration after 3rd iteration

6. The above method is continued and at the end of the final iteration we obtain the final solution in Fig. 8.

node	1	2	3	4	5	6	7	8	9
color	1	2	3	4	5	6	7	8	9
flag	1	0	0	0	0	0	0	0	0
color-assigned	1	2	1	2	1	2	1	2	3

Fig. 8. Table configuration after final iteration

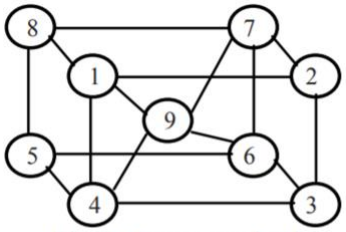


Fig. 2. Augmented cube

PROJECT ANALYSIS

→ STEP-BY-STEP METHOD FOR FINDING TIME-TABLE BY NGC :-

a. Take the input from the user the number of courses (subjects) for each year along with their names, codes and credits.

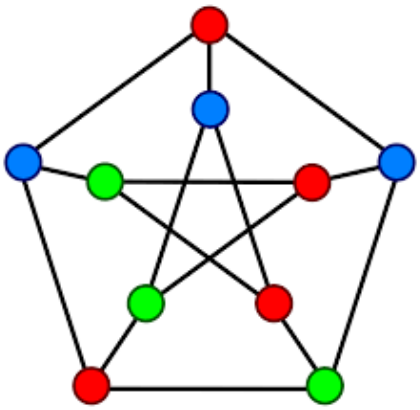
b. Also ask user which courses cannot be scheduled in the same slot.

c. Make a graph for every year with courses as vertices and an edge between courses which cannot be scheduled at the same time.

d. Now color the graph using NGC algorithm. The courses having same color can be scheduled in the same slot.

e. Courses are sorted in a way such that one's belonging to the same slot are separated for every slot and room numbers are allotted accordingly.

f. Build a user-friendly user interface to take input and display output.



→ *PROJECT CODE :-*

```
#include<iostream>
#include<string>
#include<vector>
#include<cstdio>
using namespace std;


int slotwise[6][15] = { {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, 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}
};
// slot-wise list of subjects slotwise[0] conins list of subjects in slot 1


int room[6][15] = { {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
};

int slw[6] = { 0, 0, 0, 0, 0, 0 };      // Number of subjects in each slot e.g
slot[0] tells number of subjects in slot number 1;

int islw[6] = { 0, 0, 0, 0, 0, 0 };      // Number of subjects in each slot till
previous year e.g slot[0] tells number of subjects in slot number 1;







void NGC (int **ar, int n, int *Codes, int *Credits)
{

int V = n;

int i = 0, k = 0, j = 0, l = 0;

int *a = new int (n);                          // colour-assigned

a[0] = 1;                  // start from 1 and assign colour 1 to vertex 1
```

```
for (int x = 1; x < n; x++)
    {

a[x] = 0;

}
int *c[2];

*c = new int (n);                          // colour

*(c + 1) = new int (n);                    // flag

for (int x = 0; x < n; x++)
    {

c[0][x] = x + 1;

c[1][x] = 0;

}
int count = 1;

int p = 0;

//
//
    for (i = 1; i < V; i++)
    {

for (j = 0; j < V; j++)
    {

                // checks for adjacent vertices
        if ((ar[i][j] == 1) && (a[j] >= 0) && (i != j))
        {

for (k = 0; k < V; k++)
        {       // mark the vertices with the colours of adjacent vertices as 1
          if (c[0][k] == a[j])
            {

c[1][k] = 1;

break;

}

}

}
```

```
}           //
      // checks from the left for the first non-zero flagged vertex
      for (l = 0; l < V; l++)
    {

if ((c[1][l] != 1) && (p == 0))
        {

a[i] = c[0][l];            // then assign the colour when first  flag=0

p = 1;

if (count < c[0][l])
        {

count = c[0][l];

}

}


      else                 // makes other vertices flagged as 0
        c[1][l] = 0;

}

p = 0;



}

                                  // Adding the subject codes
    for (i = 0; i < V; i++)
    {

c[0][i] = Codes[i];

}

printf ("\n\n\n");

printf ("Vertex --> Color");

printf ("\n");

for (i = 0; i < V; i++)
    {
```

```
printf ("%d --> %d\n", c[0][i], a[i]);


}


    for (int q = 0; q < 10; q++)
    {

if (a[q] == 1)
    {


slotwise[0][slw[0]] = c[0][q];

slw[0]++;

}

      else if (a[q] == 2)
    {


slotwise[1][slw[1]] = c[0][q];

slw[1]++;

}

      else if (a[q] == 3)
    {


slotwise[2][slw[2]] = c[0][q];

slw[2]++;

}

      else if (a[q] == 4)
    {


slotwise[3][slw[3]] = c[0][q];

slw[3]++;

}

      else if (a[q] == 5)
    {
```

```
slotwise[4][slw[4]] = c[0][q];

slw[4]++;

}


}


printf ("\n");

printf ("-----------Time Table---------------");

printf ("\n");


printf ("           M       T       W       Th       F");

printf ("\n");

for (int k = 0; k < count; k++)
    {


int s = 0;

s = k;


int D = 5;

for (int z = islw[k]; z < slw[k]; z++)

    {

if (D - Credits[k] == 0)

        {

printf ("slot : %d ", s + 1);

printf ("  %d R%d  %d R%d  %d R%d  %d R%d  %d R%d\n",

slotwise[k][z], room[k][z], slotwise[k][z],
               room[k][z],
slotwise[k][z], room[k][z],
               slotwise[k][z], room[k][z],
slotwise[k][z],
```

```
                room[k][z]);

}


if (D - Credits[k] == 1)

        {

printf ("slot : %d ", s + 1);

printf ("  -     %d R%d  %d R%d  %d R%d  %d R%d\n",

slotwise[k][z], room[k][z], slotwise[k][z],
                room[k][z],
slotwise[k][z], room[k][z],
                slotwise[k][z], room[k][z]);

}

if (D - Credits[k] == 2)

        {

printf ("slot : %d ", s + 1);

printf ("  %d R%d  -     %d R%d  -    %d R%d\n",
                slotwise[k][z],
room[k][z], slotwise[k][z],
                room[k][z], slotwise[k][z],
room[k][z]);

}

if (D - Credits[k] == 3)

        {

printf ("slot : %d ", s + 1);

printf ("  -     %d R%d  -     %d R%d  -  \n", slotwise[k][z],

room[k][z], slotwise[k][z], room[k][z]);


}
    }


}
```

```
for (int al = 0; al < 6; al++)

    {

islw[al] = slw[al];

}

printf ("Numbers of color used: %d\n\n", count);



}



int main ()
{


int years;

cout << "Enter Number of years you want timetable - ";

cin >> years;


int n;


for (int y = 1; y <= years; y++)
    {

cout << "\nFor Year " << y << "\n\n";



cout << "Enter the number of subjects - ";

cin >> n;

vector < string > Names;

string word;

int *Codes = new int (n);

int *Credits = new int (n);
```

```
int **a = new int *[n];


int eachYear;



for (int i = 0; i < n; ++i)

a[i] = new int[n];


for (int i = 0; i < n; i++)
    {

cout << "Enter the name of the subject number " << i + 1 << " - ";

cin >> word;

Names.push_back (word);

cout << "Enter the subject code of " << word << " - ";

cin >> Codes[i];

cout << "Enter the number of credits of " << word << " - ";

cin >> Credits[i];

cout << "\n";


}

cout <<
    "\n\n Will each student study each subject. Enter 1 for no and 0 for yes.\n";

cin >> eachYear;

if (eachYear == 0)
    {

for (int i = 0; i < n; i++)
        {

for (int j = 0; j < n; j++)
        {

a[i][j] = 1;

}
```

```
}
}
      else
    {


for (int i = 0; i < n; i++)
        {

                        cout << "Can " << Names[i] << " be scheduled with ";

                        for (int k = 0; k < n; k++)
                                {

                        if (k == n - 1)
                                    {

                        cout << Names[k] << " ";

                        }
                                else

                        cout << Names[k] << ", ";

                        }

                        cout << ". Enter 1 for no and 0 for yes separated by
space.\n";

for (int j = 0; j < n; j++)
        {

cin >> a[i][j];

}
}}




printf ("\n\n\n\n-------------Year %d-------------", y);

NGC (a, n, Codes, Credits);

cout << "Subject Code                Subject Name\n\n";

for (int r = 0; r < n; r++)
    {

cout << Codes[r] << "                          " << Names[r] <<
        endl;
```

```
}
}


return 0;


}
```

NOTE – All *Comments* in Code are in **Dark Yellow** colour.

CODE RESULTS & OUTPUT

1) *INPUT* :-

```
Enter Number of years you want timetable - 1

For Year 1

Enter the number of subjects - 4
Enter the name of the subject number 1 - Maths
Enter the subject code of Maths - 101
Enter the number of credits of Maths - 5

Enter the name of the subject number 2 - English
Enter the subject code of English - 103
Enter the number of credits of English - 4

Enter the name of the subject number 3 - Hindi
Enter the subject code of Hindi - 104
Enter the number of credits of Hindi - 4

Enter the name of the subject number 4 - SST
Enter the subject code of SST - 106
Enter the number of credits of SST - 3



Will each student study each subject. Enter 1 for no and 0 for yes.
1
Can Maths be scheduled with Maths, English, Hindi, SST . Enter 1 for no and 0 for yes separated by space.
0 1 1 1
Can English be scheduled with Maths, English, Hindi, SST . Enter 1 for no and 0 for yes separated by space.
1 0 1 1
Can Hindi be scheduled with Maths, English, Hindi, SST . Enter 1 for no and 0 for yes separated by space.
1 1 0 1
Can SST be scheduled with Maths, English, Hindi, SST . Enter 1 for no and 0 for yes separated by space.
1 1 1 0
```

HERE WE GIVE INPUT AS :-

- HOW MANY NO. OF YEAR'S TIME-TABLE IS NEEDED.
- ENTER NO. OF SUBJECTS.
- NAME, CODE AND CREDITS OF SUBJECTS.
- THEN IT ASKS THAT WILL EACH STUDENT STUDY EACH SUBJECT.
- IF WE ENTER 1, THEN WE INPUT THE ADJACENCY MATRIX.
- AFTER THIS WE GET THE OUTPUT TIME-TABLE GENERATED.

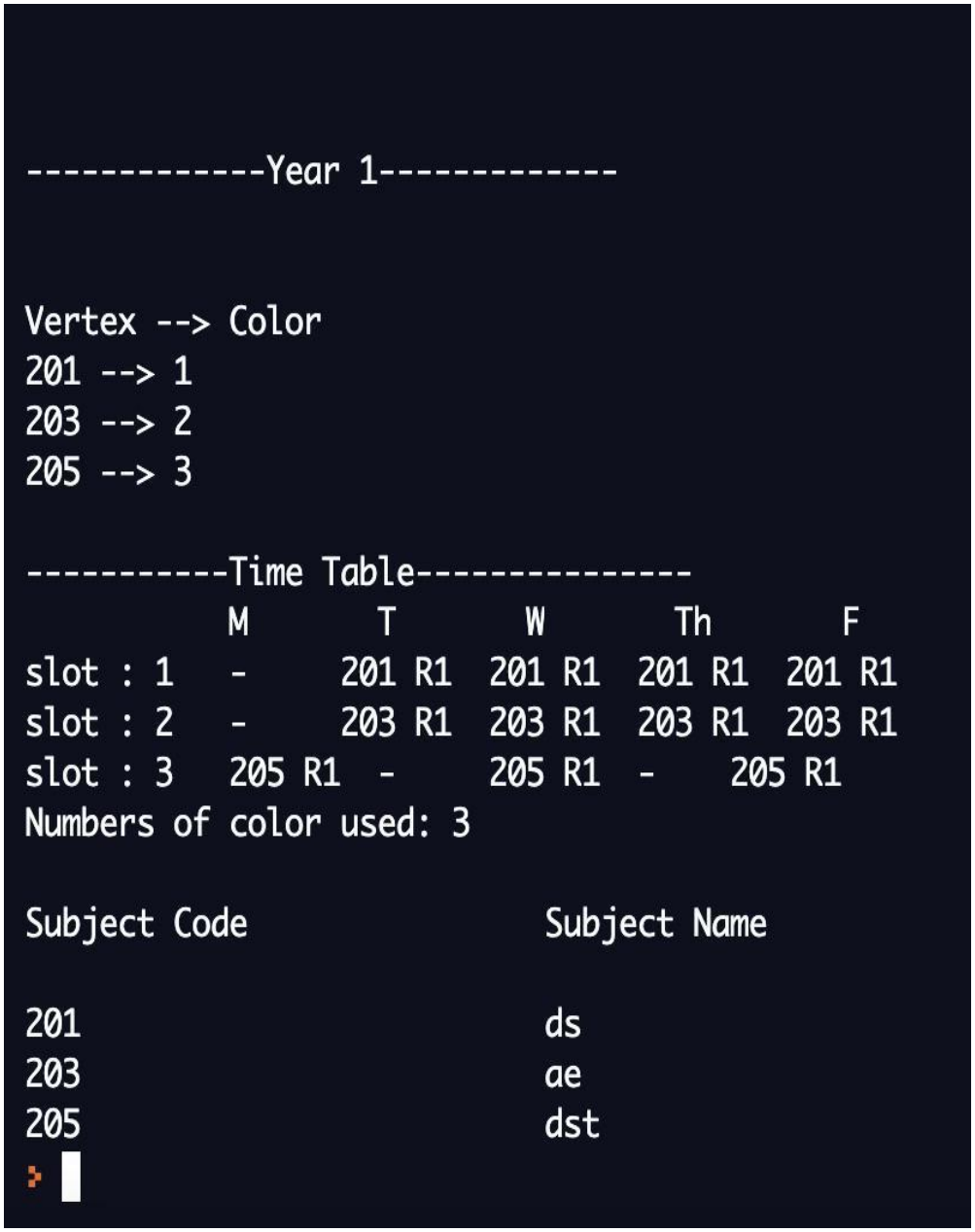
OUTPUT :-

```
------------Year 1------------


Vertex --> Color
101 --> 1
103 --> 2
104 --> 3
106 --> 4

-----------Time Table---------------
           M       T       W       Th       F
slot : 1   101 R1  101 R1  101 R1  101 R1  101 R1
slot : 2   -     103 R1  103 R1  103 R1  103 R1
slot : 3   -     104 R1  104 R1  104 R1  104 R1
slot : 4   106 R1  -     106 R1  -    106 R1
Numbers of color used: 4

Subject Code                Subject Name

101                         Maths
103                         English
104                         Hindi
106                         SST
```

2) *INPUT* :-

```
clang++-7 -pthread -std=c++17 -o main main.cpp
./main
Enter Number of years you want timetable - 1

For Year 1

Enter the number of subjects - 3
Enter the name of the subject number 1 - ds
Enter the subject code of ds - 201
Enter the number of credits of ds - 4

Enter the name of the subject number 2 - ae
Enter the subject code of ae - 203
Enter the number of credits of ae - 4

Enter the name of the subject number 3 - dst
Enter the subject code of dst - 205
Enter the number of credits of dst - 3



Will each student study each subject. Enter 1 for no and 0 f
or yes.
0
```

OUTPUT :-


------------Year 1------------
Vertex --> Color
201 --> 1
203 --> 2
205 --> 3
----------Time Table---------------
M T W Th F
slot : 1 - 201 R1 201 R1 201 R1 201 R1
slot : 2 - 203 R1 203 R1 203 R1 203 R1
slot : 3 205 R1 - 205 R1 - 205 R1
Numbers of color used: 3
Subject Code Subject Name
201 ds
203 ae
205 dst

CONCLUSION & FUTURE WORK

Our project is on ***University Time-Table Scheduling***. We successfully completed it. We take this opportunity to express our sense of indebtedness and gratitude to all those people who helped us in completing this project. We are immensely grateful to our esteemed project-guide ***Ms. Swati Sharda*** and other faculties for their supervision and guidance without which this work would not have been possible. This project has contributed a lot to our knowledge that has proved to be a value addition for us.

The complexity of a scheduling problem (NGC) is directly proportional to the number of constraints involved. Here we have studied a typical major and general course combination scheduling problem under university curriculum. Uniqueness and optimality are the main concerns in this scheduling. For the same ***chromatic number***, there are many alternative solutions, and thus it is not unique. Although all the solutions can be claimed optimal when solved using ***minimum number of colours***, a better schedule is one which ***maximizes*** satisfaction of soft constraints among its alternative solutions.

Our ***future scope*** is to prepare a well- designed and prosperous ***Graphic User Interface*** (GUI) which makes the front-end more fruit-full. This GUI makes the User more interested towards the application.

PROJECT LINK TO GITHUB :-

→ LINK =

REFERENCES :-

- https://www.researchgate.net/publication/257743106_A_Novel_Scheme_for_Graph_Coloring/link/026793370cf2946d9a21f810/download
- https://www.sciencedirect.com/science/article/pii/S2212017312003192
- https://www.ripublication.com/ijcam17/ijcamv12n2_26.pdf
- https://www.tutorialspoint.com/discrete_mathematics/graph_and_graph_models.htm#:~:text=The%20two%20discrete%20structures%20that,set%20of%20lines%20called%20edges.
- https://www.scribd.com/presentation/435618941/Graph-Coloring-Algorithm

THE END ..