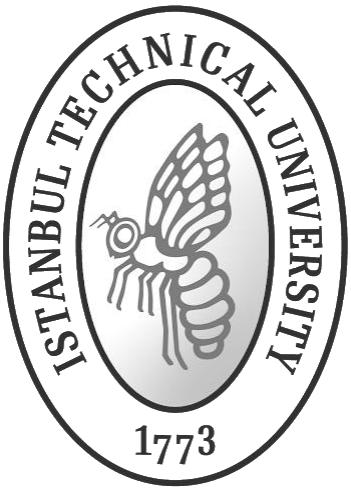
ISTANBUL TECHNICAL UNIVERSITY
FACULTY OF COMPUTER AND INFORMATICS ENGINEERING



BLG336E – Analysis of Algorithms II
Spring 2022

Lecturer: Assoc. Prof. Dr. İlkay Öksüz
T.As: Res. Asst. Uğur Ayvaz,
Alperen Kantarcı,
Caner Özer,
Muhammed Raşit Erol

Project 2

Mustafa İzzet Muştu
504211564

Development and Environment

The project is coded on Microsoft Visual Studio Code in a Microsoft Windows 10 system. It is compiled by using g++ both on local computer and ITU SSH server without any warning or error. Chrono library is used in the report and time library is used in the submit.

Questions

1) Give the adjacency matrix representation for the G2.

Regions	1	2	3	4	5	6	7
1	0	1	0	1	1	0	0
2	1	0	1	1	0	0	0
3	0	1	0	1	0	1	1
4	1	1	1	0	1	1	0
5	1	0	0	1	0	1	0
6	0	0	1	1	1	0	1
7	0	0	1	0	0	1	0

2) Write a program to implement GA1 given below.

```
PS C:\ninova\MSc\BLG 336E  -  Analysis of Algorithms 2\Projects\2> .\GA1_504211564_Q2.exe
Vertex 1 --> Color 1
Vertex 2 --> Color 2
Vertex 3 --> Color 1
Vertex 4 --> Color 3
Vertex 5 --> Color 2
Vertex 6 --> Color 4
Vertex 7 --> Color 2

Number of different colors:4
Time in ms. 1.8087
```

3.a) Write the program to implement GA2 given below.

```
PS C:\ninova\MSc\BLG 336E  -  Analysis of Algorithms 2\Projects\2> .\GA2_504211564_Q3.exe
Vertex 4 --> Color 1
Checking 3 --> false
Checking 6 --> false
Checking 1 --> false
Checking 2 --> false
Checking 5 --> false
Checking 7 --> true
Vertex 7 --> Color 1
Vertices 4, 7 are dropped!!

Vertex 3 --> Color 2
Checking 6 --> false
Checking 1 --> true
Vertex 1 --> Color 2
Checking 2 --> false
Checking 5 --> false (since it is connected to 1)
Vertices 3, 1 are dropped!!

Vertex 6 --> Color 3
Checking 2 --> true
Vertex 2 --> Color 3
Checking 5 --> false
Vertices 6, 2 are dropped!!

Vertex 5 --> Color 4
Vertices 5 are dropped!!

Well done!! All the vertices are colored.
Min color num:4

Time in ms. 7.8437
```

3.b) What is the complexity of the algorithm? Explain with pseudo code of your program.

Let's say V represents vertices, E for edges, n is number of vertices and m is number of edges in the graph. Finding degrees of each vertex takes n^2 time. Sorting degrees takes nlogn time because I used merge sort algorithm. For the rest, worst case is n^2 if all the vertices are connected together. So, time complexity is O(n^2 + nlogn + n^2) = O(n^2).

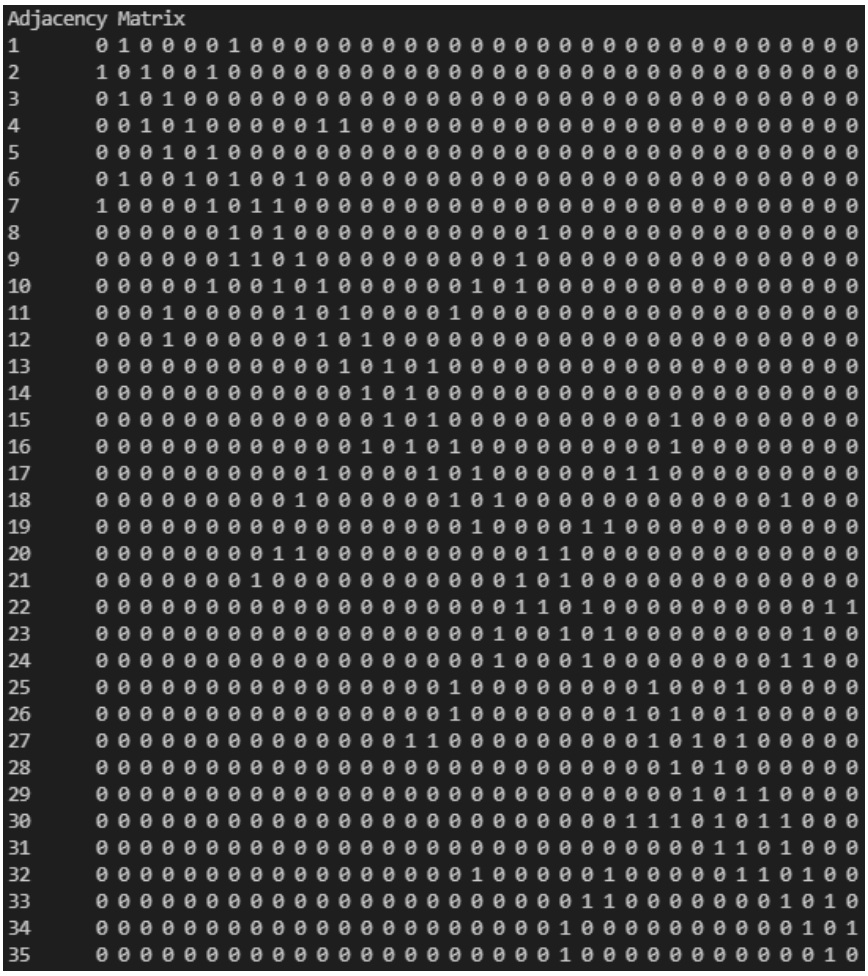
```
GA2(Graph(V,E)):
        create vertex_color_list
        create vertex_degree_list
        for(v in V)
                set degree = 0
                for(u in V)
                        if(v and u are neighbors)
                                degree += 1
                vertex_degree_list[v] = degree
        sort vertex_degree_list by descending order // mergesort is used (nlogn)
        set color = 0
        while(vertex_degree_list is not empty)
                current_vertex = vertex_degree_list.begin()
                color += 1
                vertex_color_list[current_vertex] = color
                create popped_vertex_list
                add current_vertex to popped_vertex_list
                list_iterator = vertex_degree_list.begin()
                delete first vertex from vertex_degree_list
                increment list_iterator
                while(list_iterator is not end at end of the vertex_degree_list)
                        if(current_vertex and list_iterator are neighbours)
                                increment list_iterator
                        else
                                is_connected = false
                                for(u in popped_vertex_list)
                                        if(list_iterator and u are neighbours)
                                                is_connected = true
                                                increment list_iterator
                                                break
                                if(is_connected == false)
                                        vertrex_color_list[list_iterator] = color
                                        add list_iterator to popped_vertex_list
                                        delete list_iterator from vertex_degree_list
                                        increment list_iterator
                clear popped_vertex_list
```

4) Analyze the following greedy algorithm (GA3) and show whether the given GA3 performs well for every possible scenario?

Firstly, if we do not count the colored vertices and do not repeat picking randomly a vertex, the algorithm may not color all the vertices since the algorithm starts from a vertex and goes

to one of its neighbor vertices and then to one of the neighbors of last chosen vertex repeatedly. At some point, there may not be left not colored neighbor vertex and we should start over and pick randomly a vertex. For example, from the Graph 1:
4 (Color 1) -> 6 (Color 2) -> 3 (Color 3) -> 7 (Color 1)
There are no uncolored neighbor vertices of vertex 7. So, algorithm will fail.

5.a) Give the adjacency matrix for SNG.



```
Adjacency Matrix
1       0 1 0 0 0 0 1 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
2       1 0 1 0 0 1 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
3       0 1 0 1 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
4       0 0 1 0 1 0 0 0 0 1 1 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
5       0 0 0 1 0 1 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
6       0 1 0 0 1 0 1 0 0 1 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
7       1 0 0 0 0 1 0 1 1 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
8       0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9       0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10      0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11      0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12      0 0 0 1 0 0 0 0 0 0 1 0 1 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
13      0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14      0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15      0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
16      0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
17      0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
18      0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
19      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20      0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21      0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
22      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0
23      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0
24      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
25      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0
26      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0
27      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
28      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 1 0 1 0 0 0 0 0 0 0 0 0 0
29      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 1 0 1 1 0 0 0 0 0 0 0 0
30      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 1 1 1 0 1 0 1 1 0 0 0 0 0 0
31      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 1 1 0 1 0 0 0 0 0 0 0 0
32      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0
33      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0
34      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0
35      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
```

5.b) Code two new programs to implement GA1 and GA2 on SNG.

```
PS C:\ninova\MSc\BLG 336E  -  Analysis of Algorithms 2\Projects\2> .\GA1_504211564_Q5.exe
Vertex 1 --> Color 1
Vertex 2 --> Color 2
Vertex 3 --> Color 1
Vertex 4 --> Color 2
Vertex 5 --> Color 1
Vertex 6 --> Color 3
Vertex 7 --> Color 2
Vertex 8 --> Color 1
Vertex 9 --> Color 3
Vertex 10 --> Color 1
Vertex 11 --> Color 3
Vertex 12 --> Color 1
Vertex 13 --> Color 2
Vertex 14 --> Color 1
Vertex 15 --> Color 2
Vertex 16 --> Color 1
Vertex 17 --> Color 2
Vertex 18 --> Color 3
Vertex 19 --> Color 1
Vertex 20 --> Color 2
Vertex 21 --> Color 3
Vertex 22 --> Color 1
Vertex 23 --> Color 2
Vertex 24 --> Color 3
Vertex 25 --> Color 1
Vertex 26 --> Color 3
Vertex 27 --> Color 4
Vertex 28 --> Color 1
Vertex 29 --> Color 2
Vertex 30 --> Color 5
Vertex 31 --> Color 1
Vertex 32 --> Color 2
Vertex 33 --> Color 1
Vertex 34 --> Color 2
Vertex 35 --> Color 3
Number of different colors:5
Time in ms. 8.594
```

```
PS C:\ninova\MSc\BLG 336E  -  Analysis of Algorithms 2\Projects\2> .\GA2_504211564_Q5.exe
Vertex 30 --> Color 1
Checking 10 --> true
Vertex 10 --> Color 1
Checking 17 --> true
Vertex 17 --> Color 1
Checking 22 --> true
Vertex 22 --> Color 1
Checking 27 --> false
Checking 32 --> false
Checking 4 --> true
Vertex 4 --> Color 1
Checking 6 --> false (since it is connected to 10)
Checking 7 --> true
Vertex 7 --> Color 1
Checking 9 --> false (since it is connected to 10)
Checking 11 --> false (since it is connected to 10)
Checking 16 --> false (since it is connected to 17)
Checking 18 --> false (since it is connected to 10)
Checking 20 --> false (since it is connected to 10)
Checking 23 --> false (since it is connected to 22)
Checking 24 --> true
Vertex 24 --> Color 1
Checking 26 --> false
Checking 33 --> false (since it is connected to 24)
Checking 2 --> true
Vertex 2 --> Color 1
Checking 8 --> false (since it is connected to 7)
Checking 12 --> false (since it is connected to 4)
Checking 13 --> true
Vertex 13 --> Color 1
Checking 15 --> true
Vertex 15 --> Color 1
Checking 19 --> false (since it is connected to 24)
Checking 21 --> false (since it is connected to 22)
Checking 25 --> false
Checking 29 --> false
Checking 31 --> false
Checking 34 --> false (since it is connected to 22)
Checking 1 --> false (since it is connected to 7)
Checking 3 --> false (since it is connected to 4)
Checking 5 --> false (since it is connected to 4)
Checking 14 --> false (since it is connected to 13)
Checking 28 --> true
Vertex 28 --> Color 1
Checking 35 --> false (since it is connected to 22)
Vertices 30, 10, 17, 22, 4, 7, 24, 2, 13, 15, 28 are dropped!!
```

```
Vertex 6 --> Color 2
Checking 9 --> true
Vertex 9 --> Color 2
Checking 11 --> true
Vertex 11 --> Color 2
Checking 16 --> false
Checking 18 --> false (since it is connected to 32)
Checking 20 --> false (since it is connected to 9)
Checking 23 --> true
Vertex 23 --> Color 2
Checking 26 --> false
Checking 33 --> false (since it is connected to 32)
Checking 8 --> false (since it is connected to 9)
Checking 12 --> false (since it is connected to 11)
Checking 19 --> false (since it is connected to 23)
Checking 21 --> true
Vertex 21 --> Color 2
Checking 25 --> true
Vertex 25 --> Color 2
Checking 29 --> true
Vertex 29 --> Color 2
Checking 31 --> false (since it is connected to 32)
Checking 34 --> true
Vertex 34 --> Color 2
Checking 1 --> true
Vertex 1 --> Color 2
Checking 3 --> true
Vertex 3 --> Color 2
Checking 5 --> false (since it is connected to 6)
Checking 14 --> true
Vertex 14 --> Color 2
Checking 35 --> false (since it is connected to 34)
Vertices 27, 32, 6, 9, 11, 23, 21, 25, 29, 34, 1, 3, 14 are dropped!!
```

```
Vertex 16 --> Color 3
Checking 18 --> true
Vertex 18 --> Color 3
Checking 20 --> true
Vertex 20 --> Color 3
Checking 26 --> true
Vertex 26 --> Color 3
Checking 33 --> true
Vertex 33 --> Color 3
Checking 8 --> true
Vertex 8 --> Color 3
Checking 12 --> true
Vertex 12 --> Color 3
Checking 19 --> false (since it is connected to 18)
Checking 31 --> true
Vertex 31 --> Color 3
Checking 5 --> true
Vertex 5 --> Color 3
Checking 35 --> true
Vertex 35 --> Color 3
Vertices 16, 18, 20, 26, 33, 8, 12, 31, 5, 35 are dropped!!

Vertex 19 --> Color 4
Vertices 19 are dropped!!

Well done!! All the vertices are colored.
Min color num:4

Time in ms. 36.2234
```

5.c) Which greedy algorithm performed better in terms of execution time? Output the execution times in your program.

GA1 is better in terms of execution time. Because we do not implement sorting and we iterate a row in adjacency matrix once for every vertex.

GA1)

```
Number of different colors:5
Time in ms. 8.594
```

GA2)

```
Well done!! All the vertices are colored.
Min color num:4

Time in ms. 36.2234
```

5.d) Which greedy algorithm performed better in terms of finding the minimum number of colors? Show it in your program output.

GA2 performed better in terms of finding the minimum number of colors.

GA1)

```
Number of different colors:5
Time in ms. 8.594
```

GA2)

```
Well done!! All the vertices are colored.
Min color num:4

Time in ms. 36.2234
```