REPORT OF 18 POINTS PROGRAM 31/01/2019 (s246389)

DESCRIPTION OF THE IMPLEMENTATION

The data structures used in the program are a graph wrapper, which contains a pointer to a list of vertices, a list of vertices containing an id (<u>name</u>) and a pointer to a list of edges(<u>head</u>), and a list of edges for each vertex (each edge in the list points to a destination (<u>dst</u>) vertex. In addition to that two fields which are specific for this problem have been added to the vertex structure: <u>traversed</u> (an integer flag which is used by *assigncolors* function to understand if the vertex has already been traversed), <u>color</u> (which is used by *assigncolors* and *check* functions to assing a color to a vertex (using a number). (pointer <u>next</u> is used to connect vertices and edges in respectives lists).

Moreover, three main functions have been used in order to solve the proposed problem:

1. Function *graphload* reads the file and stores vertices and edges in the structures. Three minors functions are called by graphload:
   - *vertexfind* : given a vertex identifier(string) and a graph it returns a pointer to the desired vertex.
   - *newvertex* : given a vertex identifier and a graph it allocates and initialize a new vertex.
   - *newedge* : given two vertices it allocates, and initialize an edge between the first vertex and the second vertex and a an edge between the second vertex and the first (the graph is undirected).
2. Function *assigncolors,* which is called by the main, creates all possible arrangements with repetitions of k colors in n vertex. The value of k is increased by one at each call of the function in the main (it ranges from one to n in the worst case (one color for each vertex)), thus guaranteering to find the solutions with less colors involved before and to stop when the first solution (which is also the best one) is found. Another approach could have been to find all possible solutions and then select the one with less colors. However, in this way all the computed solutions with more colors would have been unuseful resulting in a waste of time (due to not needed computations). The function is recursive. The termination condition is (“pos”==0). ”pos” is an integer value which starts from n (number of vertices calculated in graphload) and is decreased by one every time a color is assigned to a vertex (every recursive call). The function checks if the vertex has already been traversed (flag <u>traversed</u>), then it executes two loops. The outer for loop ranges from one to k, assigning all possible colors to the vertex. The inner while loop allows the recursive call to head to all possible connected vertices. To conclude two backtracks operations are needed.
3. Function *check* checks whether vertices which are connected by an edge have the same color (in this case it returns 0 and the function *assigncolors* provides another coloring sequence) or not (in this case it returns 1 so that also function *assigncolors* returns 1 to the main and the procedure stops because the solution is approved). In order to perform this check, for each vertex every edge is considered and 0 is returned if two adiacent vertices present the same color.

Libraries have been used only for memory allocation (*util_malloc, uitl_strdup*) and relatives checks or for file opening (*util_fopen*). Moreover, *graphdispose* function has been added to the utility library, because considered as a normal library function when dealing with graphs.

Functions

<u>Structure fields</u>

DIFFRENCES WITH RISPECT TO MANUAL WRITTEN CODE

All improvements are related to the loading of the graph because this part has been developed more quickly in the last part of the exam.

- In the code written during the exam the allocation of the graph wrapper is not present. (lines 72,73)
- In function *newvertex* lines 184 and 192 have been added in order to take into consideration the case in which we are allocating the first vertex and g->v is = NULL.
- As also written in the exam paper the allocation of the edge in case of an already present list of edge has to be added. This part has been added at lines 230 and 254.
- The basic function *printoutgraph* (it prints the selected color for each vertex) is called in the main but its development/implementation is not present in the exam paper.