Project 1

The theme of this project is to create an implementation of a cluster concept called k-core, and experiment with it.

The definition of the k-core is given in Lecture Note 13 *Clusters in Graphs,* within the Linear Programming learning module. The definition is copied below:

Definition: *Given an integer $k \geq 1$, the k-core of a graph is the largest subset of nodes, such that each node in the subset has at least k neighbors within the subset.*

The referred lecture note also outlines a simple algorithm to find the k-core. The following figure, copied from the lecture note, illustrates the concept:

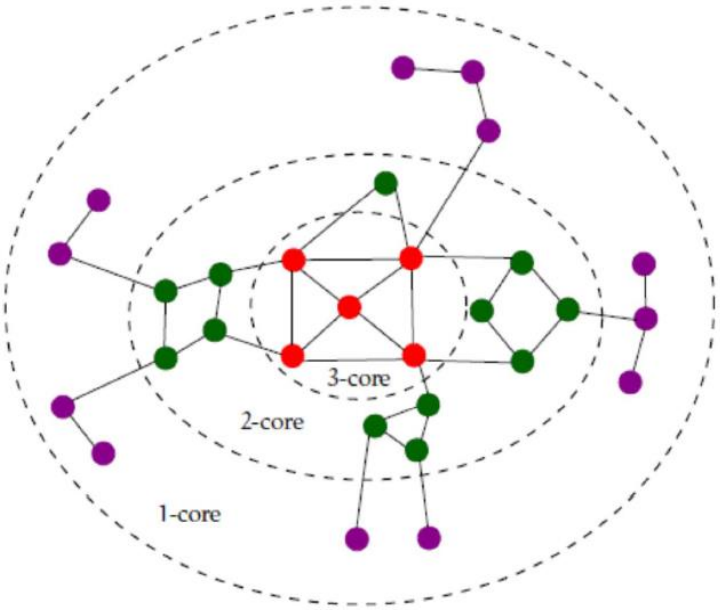

Figure 1

Explanation: all nodes belong to the 1-core, since they all have at least 1 neighbor among all nodes. The green and red nodes belong to the 2-core, as they all have at least 2 red or green neighbors, that is, at least 2 neighbors in the same set. Finally, the red nodes form the 3-core, since they all have at least 3 red neighbors. Generally, the k-core is the largest subset of nodes, such that each node in it must have at least k neighbors from the same subset. Note that it is not enough to have just any k neighbors, they all must be in the same subset.

Tasks:

Task 1. Implement an algorithm, which receives as input the following:

- A positive integer k.
- An undirected graph given by its adjacency matrix. Assume that the graph has no self-loops, parallel edges, and isolated nodes.

As output, the program produces the list of nodes that are in the k-core of the graph. The nodes are numbered according to the corresponding rows in the adjacency matrix.

Task 2. Generate an input graph using your 10-digit UTD student ID, which is unique for everybody. The graph will have 27 nodes. Generate the adjacency matrix of the graph, as follows:

- In the student ID replace the even digits by the 0 bit, and the odd digits by the 1 bit. For example, if your ID is 1736520839, then after this replacement you get the bit sequence 1110100011.
- Then repeat this bit sequence 73 times, so that you get a 730-bit long sequence.
- Put the first 27 bits from the obtained sequence in the first row of the adjacency matrix. Put the second 27 bits in the second row, and so on, each subsequent 27-bit sub-sequence in the next row, until you fill the whole 27x27-sized matrix. This way you will use 27x27 = 729 bits from the 730-bit sequence.
- Since the graph is assumed undirected with no self-loops, no parallel edges and no isolated nodes, carry out the following transformations to reflect these properties in the adjacency matrix:
  - Eliminate the isolated nodes such that whenever a node is isolated, put a 1 in the first and last position both in its row and column.
  - Replace each entry in the main diagonal with 0.
  - Make the matrix symmetric such that for each entry in position (i,j) with i<j, replace the entry with the bit that is in position (j,i).

Task 3. Run the algorithm implemented in Task 1 on the graph generated in Task 2. Run it on all k=1,2,3,… values, as long as it produces non-empty k-cores.

Task 4. Illustrate the results by a figure, similar to Figure 1 in the previous page. Note that the figure must be drawn by software, hand-drawn figures are not acceptable as professional quality. You may use any existing software module for drawing.

Task 5. Experiment with the program to find out how the k-cores change if an edge is deleted from the graph.

For example, with k=3, in Figure 2 below, the graph in (a) has all the nodes in the 3-core, marked with blue color. If, however, edge e_1 is removed, then the 3-core becomes empty: the graph in (b) has an empty 3-core. Even though some nodes have degree 3 in (b), there is no

subgraph in which all nodes have degree 3. If, however, instead of e_1, we remove edge e_2 from the original graph, then we obtain (c), where 4 nodes remain in the 3-core, marked by blue color.

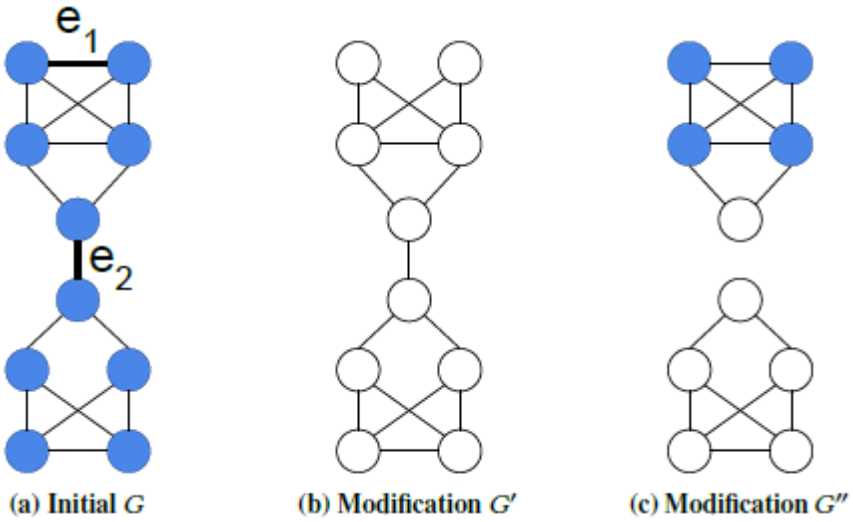

Figure 2

For the specific experimentation, select randomly an edge from the graph generated in Task 2, delete this edge, modifying the adjacency matrix accordingly. Then repeat Tasks 3 and 4 for the new graph, to illustrate how the k-core structure changes with the deleted edge.

Repeat the above with 2 further randomly selected edges. Illustrate all the results graphically. . You may draw each graph in a separate figure, as they are larger (27 nodes), so they likely will not all fit in a single figure.

Notes

- You may use any programming language, operating system, and drawing module. It is of your choice, use whatever is most convenient to you.
- If something is not specified in this project description, it automatically means that it is of your choice.
- There will be a separate posting about submission guidelines and formatting requirements.