**Approach to PROJ 5, LRU Buffer Pool**

Riley Dorough / Chandler Whitley

18ARP20

This project was completed through the collaborative efforts of the genius Chandler Whitley & the innovative Riley Dorough.

Source code taken from pages 389-390 in our text.

**Minimum Spanning Tree**

**Graphm.h**

The main header where implementation is hidden from the user

**Note, Smart pointers are used in the mark bit array**

Graphm variables:

* Integer for number of vertexes
* Integer for number of edges
* Integer pointer to the adjacency matrix
* **Smart pointer** integer for mark bit array
* Constant integers for visited and unvisited tracking to help ensure ease of use and reading.

Graphm class (construct & destruct):

* Constructor calls the Init function and passes through the number of vertexes given from the driver.
* Destructor enters a for loop the length of the number of vertexes and clears each value in the matrix, then when the matrix is empty the matrix itself can be deleted. The smart pointers with the mark array should take care of themselves.

Graphm methods:

* Init() Should initialize the graph with an integer input. Should create a new mark array the length of the number of vertices, also set every vertex to unvisited. Each slot in the matrix array should be the head of its own array the length of the number of vertexes. Finally, set each index to a weight of zero.
* Verticies() Returns the number of vertices
* Edges() Returns the number of edges
* First() Seeks the first neighbor of a given vertex starting from any position
* Next() Seeks the first neighbor of a given vertex starting from any position it ended on, allowing the loop to continue without resetting.
* SetEdge() Assign an edge weight given two vertices and add 1 to the number of edges variable. Modify the authors code to check to ensure that the weight is never less than 0.
* DeleteEdge() Decrement the number of edges variable by 1 and set the given array position edge weight to 0.
* IsEdge() given two positions in the matrix, if an edge exists between them, return true.
* Weight() Returns the value of the weight of an edge given two potions in the matrix
* GetMark() Checks the mark array at a given index to find if It’s been visited
* SetMark() Sets the mark array to a 1 or 0 given an index and value
* InputGraph() Should open the text file and assign its values to a matrix then runs through for loops to set values and edges where dictated by the input file.
* PrintGraph() runs through the matrix printing out the columns and rows.
* OutputGraph() Should work the same as printgraph(), but instead of only printing to the screen, also output to a text file.

**Main.cpp**

* Set up 3 different graphs with the 3 given starting vertices.
* Each iteration should create an pointer to an array with length the number of vertices.
* Using a loop set all indexes to a large value.
* Set the starting vertex to 0.
* Create a pointer to a new object of class Graphm and call the Prim() function with inputs (the first graph, the starting location array, and the starting vertex’s number).
* Modify the author’s minVertex and Prim algorithms to work with our code’s variable, object, and class names.