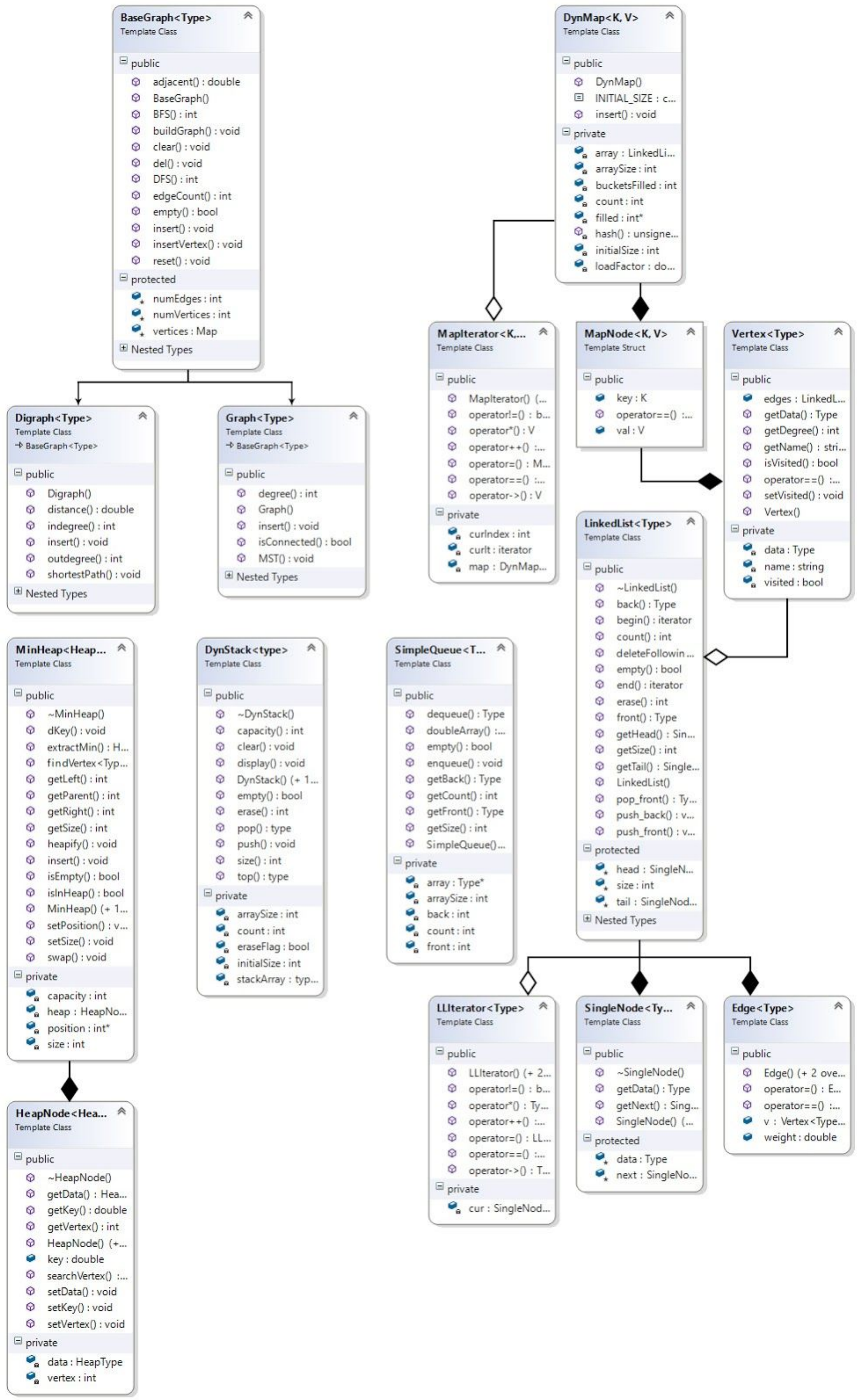
Project 4 Report

Brad Daniels, Matthew Kramer, Sterling Price

July 18th, 2016

1. UML Diagram



2. Graph

a. Design

A graph contains vertices and edges. Our implementation of a graph has vertex nodes which each contain a linked list of edges. These vertices are accessed via a hash table. The edges of each vertex contain pointers to another vertex. In the undirected graph, when an edge is inserted between two vertices, for example vertex A and vertex B, vertex A will have an edge within its linked list that points to vertex B, and vertex B will have an edge within its linked list that points to vertex A. When an edge is inserted into the directed graph, only the “initial” vertex contains the edge within its linked list, with the linked list of the “destination” vertex remaining untouched.

b. Algorithm Analysis

Many of the graph’s methods operate in O(1) or O(n) time, leveraging the optimized hash map to make accessing vertices and their data members immediate, except that of edges which - in worst case - may require O(n) to reach the edge at the end of a linked list.

3. Dijkstra’s Algorithm

a. Design

Dijkstra’s Algorithm is used for accomplishing the *shortPath* method in a directed graph. A minimum heap is used to traverse the graph finding the smallest amount of cumulative weight needed to go from one node to another. The smallest weight is then accessed from the top when the iterations are finished. The indices of the vertices are tied to this path in the minimum heap, in consecutive order, to give us the final path needed to go from one node to the other specifically chosen node.

b. Algorithm Analysis

Dijkstra’s Algorithm worst case scenario operates in O((E+V) log V) time, where E is the total number of edges, and V is the total number of vertices in a graph. The algorithm traverses all edges of each vertex until a path is found that reaches the destination vertex. This path’s total weight is then preserved and compared against when iterating through the other paths in order to find the optimal route.

4. Prim’s Algorithm

a. Design

Prim’s algorithm is used for calculating the *minimum spanning tree*, or, *MST* of an undirected graph. In this algorithm, two sets are maintained: one containing the list of vertices already included in the minimum spanning tree and the other set containing the vertices not yet included. Using a breadth-first search to traverse the graph, a minimum heap tree is then used to store the not-yet-included vertices. This heap is implemented as a priority queue which maintains the vertices ‘closer’ to the root nearest the root of the heap tree.

b. Algorithm Analysis

Prim’s algorithm operates in O(E log V), where E is the number of edges in the graph and V is the number of vertices in the graph. Using the minimum heap allows for a more optimized running time than an adjacency matrix, which requires O(V^2).