This Bushfire Monitoring program that I have created first gives an option of which test data would like to be used and then opens a menu of options for the user to choose from. The program is able successfully execute each task given in the assignment brief and acts as a looping menu so when an option is chosen and the task is executed, the user is redirected to the menu of options. This program’s Main.java file was used as test harness for every class in the program.

This program uses graphs to store the locations and edges of a given area and acts as a map which allows to find the shortest path between locations when accounting for distance. Furthermore, this program uses a hash table for the information of each location, a heap to give priority to areas with higher risk levels and a linked list to store locations for the UAV itineraries.

A hash table is more efficient with storage of information than a list or array since it has an average complexity of adding, removing and finding of O(1) which is generally faster than that of a list or an array. Furthermore, the hash table aims to use less memory with a resizing method which is more efficient than the fixed size of an array.

This program uses a linked list for to create a list that can be iterated through and easily modified. The linked list acts as a list of nodes that each store information and has the ability to insert and remove from both the front and end as well as remove specific nodes in the list. The DSAQueue class uses the linked list to create an ADT that behaves as a first-in-first-out queue.

The graph is an ADT that stores vertices and edges and allows for traversal techniques to show paths from one vertex to another. This program’s DSAGraph class uses a linked list to stores its vertices and edges and uses breadth first search and depth first search traversals to find the shortest path between vertices and show the whole graph. For this assignment I have assumed that in Task 1, where it says there is a “starting vertex” and an “ending vertex”, the edges are directed and so the traversal methods all take this into account.

This program uses a hash table to easily store information and easily retrieve this information without worrying how big the hash table gets. The DSAHashTable class has an array of hash entries. This class when adding a hash entry places the entry into a certain placement according to its key and when the placement is not available it makes sure to place it in an available placement. Furthermore, when the hash table becomes too small or too big it resizes so that the hash table can fit as many hash entries with as little memory being used.

The max heap acts as a priority queue with the heap entry with the highest priority being at the top of the heap. The DSAHeap class has an array of heap entries and when a new entry is added or removed it uses a trickle up or trickle down method to reorganise the heap according to the priorities of the entries.

The program’s looping menu feature acted as a test harness for each class and can help find many errors and eventually help correcting them. After every relevant method was made, the looping menu could be used to test the method. For example, it was found out through testing that if a location was removed more locations could not be added after that to the graph which made meant that the remove() method in the DSALinkedList class was not functioning properly. This allowed for the correction of the code in the remove() method to be possible.

For this assignment, many assumptions had to be made for some of the tasks. For task 1, it was assumed that the edges were directed. For task 3, it was assumed each operation would be available for the already created graphs that use given information from the files. For task 6, it was assumed a location with high risk of bushfires would mean the location has at least one risk factor a high level of risk, furthermore, the UAVs will deploy for each possible trip to these high risk locations meaning that if a UAV cannot make a path to the next location a new UAV will be dispatched since the edges are assumed to be directed.

The program successfully complies with each task given with the added assumptions, although the testing may have been less difficult with multiple unit test harness frameworks for each class, however, the menu was sufficient. Also, if the edges were undirected then a more accurate representation of reality may have been presentable through this program. The exception handling in this program could also have been more thoroughly looked into.

