README

Class Descriptions.

DSAGraph class and DSAGraphVertex class - The DSAGraph class contains a list a vertex which will be used to create the graph. Methods to add, remove and search for vertexes are present. The graph traversal methods such as Depth first search, breadth first search and dijkastras algorithm are implemented in this class.

The DSAGraphVertex class will be used to create vertex's which contain a label, a DSALinked list for its adjacent vertexes, a boolean , temperature value, humidty value and wind speed value.

DSALinkedList - The DSALinkedList class is an implementation of a doubly linked double ended linked list. Methods to insert, remove , search ,traverse the list and display values are present in this class. DSAListNode is a private inner class of DSALinkedList.

DSAHeap and DSAHeapEntry - The DSAHeap class is an implementation of a max heap. Methods to add, remove, sort, extract the minimum value and display the heap are present.

DSAHashTable and DSAHashEntry - This code is based on the practical 7 submisson. Methods to put, get , remove entries, resize and display the hash table code is implemented in DSAHashTable. Two hashing methods are implemented.

Pair - This code is used when need to pair two values together. There are two constructors implemented. The first constructor " public Pair(DSAGraphVertex name, double value) {}" will create a Pair instance which can hold a DSAGraphVertex object and a double value. This is used to store a vertex and the distance from to it from the source vertex.

The second constructor " public Pair(String vertexName, String parent) " is used to store a vertex's name and its parent. This is used in Breadth First Search and Dijkstra's Algorithm.

UAVSimulator Class - This class contains the main function which will be used to run the program.

Program Functionalities

- The user must enter a location file name (File with the edges) first.
- The file name with the UAV data must be entered second.

Menu functions

0) A user can exit the simulator by inputting 0
1) A user can insert a location as follows by inputting 1
2) User can insert an edge as follows by inputting 2. Two vertexes needed to be added first. Once they are added enter the first location, the second and the distance between them.
3) User can remove a location by inputting 3. Insert the name of the location to be deleted.
4) User can display a display a location's data by inputting 4. Insert the name of the location to be displayed.

5) User can display graph as an adjacency list by inserting 5.
6) User can carry out Depth First Search by inputting 6.
   0)By inputting 0 a user can go back to the main menu
   1) By inputting 1 a user can display the hash table entries.
   2) By inputting 2 a user can display the heap.
   3) By inputting 3 a user can display the high-risk areas
   4) By inputting 4 a user can display a location from the hash table

7) By inputting 7 a user can find the shortest path between the two locations using Breadth first search. This implementation of BFS does not consider the distance between the two locations.

8) By inputting 8 a user can find the shortest path between the two locations using Dijkstra's Algorithm. This implementation does take into consideration the distance between the two locations. The user is given the option to determine the number of UAVs that will be traversing.

   Please find a more detailed description in the report.