**Program description**

This program is an implementation of the Kruskal’s algorithm using disjoint set, doubly linked list, and graphs. It reads an input file and builds a graph with adjacency list and edge list. Then it makes separate disjoint set (implemented in a previous assignment) for each of the vertices. Afterwards, it will utilize Kruskal’s algorithm to find and create a minimum spanning tree.

**Data structure and algorithms**

The class Templatedoublylinkedlist is there to implement a disjointset class. The disjoint set class is to help distinguished 2 vertices that are connected either directly or indirectly from unconnected vertices. The graph data structure is to help store all the information needed, such as the edge information, the vertices, the adjacency list, and the minimum spanning tree created. Implemented algorithm includes buildGraph() that initializes a graph and its adjacency list from an imput file; insertEdge that insert an edge with its two vertices into the edgeList vector; getWeight that returns the weight between two vertices; sortEdge that sorts edgeList in increasing order for their weights; and finally MSTAlgo() that implements Kruskal’s algorithm, builds a minimum spanning tree and returns the total weight.

**Runtime analysis and implementation of algorithms**

1. buildGraph: O(V). It iterates by a counter (provided by input file), creates a new vertex each iteration, and pushes into the adjacencylist.
2. insertEdge: O(1). It runs once for every edge. It creates a new edge with vertices i and j, pushes back the edge into the list, updates the AdjacencyList and edge for the vertices.
3. getWeight: O(V). Iterates through the AdjacencyList for vertex i and try to find whether there exists an edge between i and vertex j. If so return the weight of the edge.
4. sortEdge: O(ElogE). Originally I wrote an implementation of the merge sort algorithm to sort the edges, but that requires a change to the return type. The new function utilizes standard library’s sort function with vector iterator and overridden sorting function. Although the exact sorting algorithm changes from time to time (mostly introsort, according to Stackoverflow), the stl sort also runs at O(nlogn) time.
5. MSTAlgo: It creates individual disjointset for each vertex. It then sorts the edges in increasing weight. Then it will iterate through the edgeList and union the vertices for each edge, if they do not have the same disjointset representative. It also builds a MST and a helper add\_weight() function that adds the total weight for the MST.

sortEdge is O(ElogE). The makeset part is O(V) because it runs once for every vertex. Union part is O(ElogE+VlogV). Overall the MSTAlgo is bounded by O((E+V)log(V)).

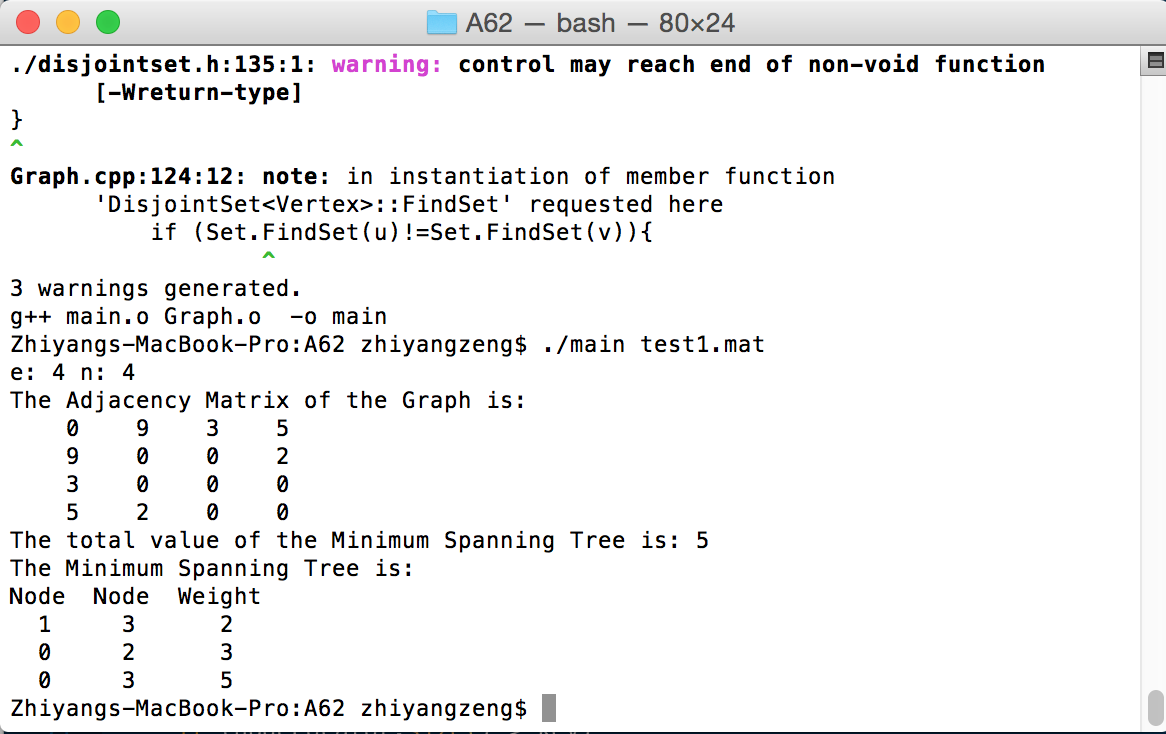
**Instruction to Compile and Run**

Make all to compile. ./main testfile to run the function with testfile as input.

**Logical Exceptions**

There are GraphExceptions such as negative edge number/vertex number, or incorrect edge counters. The disjointset also has the original exception handlings, with a few changes to accommodate the new programs. MakeSet with number greater than the size of nodeLocator no longer throws an exception, rather it resizes. This is to deal with vertices with non-consecutive numbers.

**Testing results**

****