ECE 250 - PROJECT 3 - DESIGN DOCUMENT

ARJUN MEHTA – [redacted]

OVERVIEW OF CLASSES

class edgeinfo

Represents the class that stores the vertex u, vertex v and the weight of the edge between them.

Member Variables :	u (int)	**stores the vertex u**
	v (int)	**stores the vertex v**
	w (double)	**stores the weight of the edge between vertex u and vertex v**
Member Functions :	None	

class graph

Represents the class that forms the basis of the graph and contains the information of its edges. It also contains the required functions.

Member Variables :	edge (vector <edgeinfo>)	**vector containing all the inserted edges**
	size (int)	**represents the size of the created graph**
	numberofedges (int)	**represents the number of edges in the graph**
	totalconnected (int)	**keeps track of total connected vertices to indicate whether mst is formed or whether the tree formed is not connected**
Member Functions :	create()	**creates a graph with m nodes**
	insert()	**inserts an edge between nodes u and v with weight w (a double type)**
	del()	**deletes the edge between nodes u and v**
	degree()	**returns the degree of vertex u**
	edge_count()	**returns total number of edges in the graph**
	clear()	**removes all the edges from the graph**
	mst()	**calculates the minimum spanning tree and its weight**

class nodelist

Represents the class that forms the linked list for the implementation of the disjoint set.

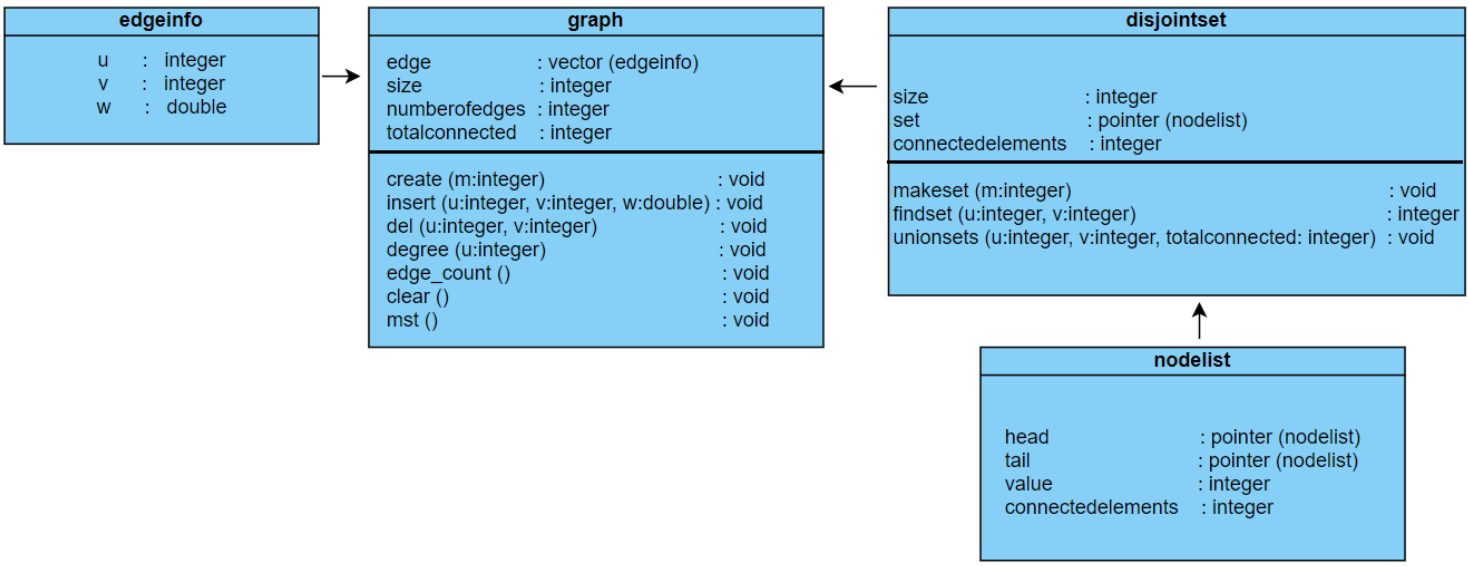
Member Variables :	head (pointer)	**pointer of type nodelist that points to the node at the head of the node**
	tail (pointer)	**pointer of type nodelist that points to the node at the tail of the node**
	value (int)	**holds the vertex present at each node**
	connectedelements (int)	**keeps track of how many vertices are connected to each other**
Member Functions :	None	

class disjointset

Represents the class that helps us form the disjoint set and perform it's various functions.

Member Variables :	size (int)	**stores the size of the disjoint set that is formed**
	set (pointer)	**pointer of type nodelist that points to a nodelist array (disjoint set)**
	connectedelements (int)	**keeps track of how many vertices are connected to each other**
Member Functions :	makeset()	**creates the disjoint set of the required size**
	findset()	**returns the parent of the passed vertex in the disjoint set**
	unionsets()	**performs union operation on two sets such that the vertices of the two sets are now connected**

UML CLASS DIAGRAM



DETAILS ON DESIGN DECISIONS

Constructors

class edgeinfo
Set default values for u, v and w to 0.

class graph
Set default values for size, totalconnected and numberofedges to 0.

EXCEPTION HANDLING

In case we want to throw an exception we use:

```
if(m<=0){
    throw "invalid argument";
}
```

and catch it by using :

```
try{ g.degree(u);
} catch(const char* warning){
    std::cout<<warning<<std::endl;
}
```

class nodelist
Set default values for value to 0 and connectedelements to 1 (as there is one element in every set by default).
Set head = nullptr and tail = nullptr so that newly created nodes always point towards a null pointer.

class disjointset
Set default value for size to 0.
Set set = nullptr so that a newly created set always point towards a null pointer.

Destructors

class nodelist
Empty destructor as the nodelist pointers formed are deleted by the disjointset destructor.

class disjointset
Delete all set[i] -> head and set them to nullptr to avoid memory leaks.
Delete the set array to completely clear it and set it to nullptr to avoid memory leaks.

TEST CASES

We implement various test cases to check if program is able to handle difficult situations. This includes running each and every function, exiting before the final statements and check for failures in cases where overlapping takes place.

n 12 i 0;1;7 i 1;2;5 i 0;3;2 i 2;3;4 i 1;3;15 edge_count mst exit	n 4 i 0;1;10 i 0;2;6 i 0;3;5 i 2;3;4 i 1;3;15 i 1;3;13 i 1;3;4 edge_count degree 1 d 1;2 d 0;2 edge_count clear edge_count exit	n 4 i 2;3;5 i 0;1;11 i 0;2;6 i 1;3;15 i 0;3;4 degree 2 degree 1 degree 0 edge_count clear mst exit clear degree 0 mst
Calls create, insert, edge_count, mst functions. checks if "not connected" is displayed in case of a incomplete mst. Tries inserting various edges and checks if edge_count returned is correct	Inserts edges. Tries inserting 1;3 with different values of weight to check if weight is updated in case same vertices are passed. Calculates degree, deletes some edges. Checks if clear() works by running edge_count afterwards.	Creates set, inserts edges. Checks degree of various edges. Counts the edges. Clears before mst to check the mst output functions correctly. Exits before clear, degree and mst to check if exit works properly.

PERFORMANCE CONSIDERATIONS

Time complexity of Kruskal's Algorithm

Let E be the number of edges, V be the number of vertices.
mst () : It takes a total time of O(E log (V)).

To calculate this we look into time taken by each function inside mst() as well as disjointset class.

Sorting the graph takes a tight bound time of O(E log (E)) which can be represented as O(E log (V)).

makeset () takes O (V) time as we create a disjoint set for each and every vertex.

findset () takes O(E) time as we recursively call findset() till we find the parent and the worst case time is equal to O(E) as there are a total of edges or we can say the worst case is when we have to run it V – 1 times.

Unionset () takes a total time of less than |V| log |V| as the number of elements in the bigger set (which absorbs the smaller one) atleast doubles.

Using above results we can conclude total running time of our mst () function (for Kruskal's algorithm) is O(E log (V)).