**EX0 - OOP**

* code installation: https://github.com/chenoh1990/EX0.git

**author: chen ohayon**

**summary:**

In this project I am developing an infrastructure of unintentional and unweighted graph.

I develop the methods of these graphs according to interfaces pre-defined for me by the lecturer.

After realizing the graph properties, I will implement an algorithm to calculate the shortest path, and another algorithm for finding the shortest trajectory using the graph properties I implemented.

**NodeDate class methods:**

* getKey(): return the key value of node\_data.
* getNi(): return a collection of all neighbors of this node\_data.
* hasNi(): return true if the node\_data of this given key is a neighbor of this node\_data.
* getInfo(): return the info of this node.
* setInfo(): get string and set it as a info of this node.
* getTag(): return the tag of this node.
* setTag(): the tag help us to mark the node according to our needs.
* removeNode(): remove node from this node neighbors collection.
* addNi(): adding a neighbor for this node neighbors collection , node cannot be a neighbor of himself.

**Graph\_DS class methods:**

* getNode(): return the node\_data by given a key value.
* hasEdge(): return true if there is an edge between those two nodes , and false if not exist.
* connect(): connect between two nodes if they exist in this graph.
* getV(): return collection of this graph nodes.
* removeNode(): remove node data from graph by given key.
* removeEdge(): by given two keys of nodes, this method disconnect between those two nodes if they connected.
* nodeSize(): return the number of the nodes in this graph.
* edgeSize(): return the number of the edges in this grpah.
* getMC(): return the number of all changes that nade in ths graph.
* addNode(): add a new node to the graph.

**Graph\_Algo algorithms & methods:**

* isConnected(): check if the graph is connected graph or not-connected grpah. return true if graph connected, if not - return false.
* shortestPathDist(): return the distance of the shortest path between two nodes.
* BFS(breadth first search) (): return number of vertices connected to the start vertex.
* shortestPath(): build the list of this path by the tag information,if start or end nodes are not exist in the graph , the method return a null list.