

ADTs Graph			
$G = (V, E)$, where V is a set of vertices, and E is a set of edges			
{ <i>inv</i> : There cannot be two vertexes with the same value on the Graph.}			
Primitive Operations:			
- Graph	Graph	→	Graph
- addVertex	Graph x Value	→	Graph
- addEdge	Graph x Value x Value x String x Int	→	Graph
- deleteVertex	Graph x Value	→	Graph
- deleteEdge	Graph x Value x Value x String	→	Graph
- search Vertex Index	Graph x Value	→	Int
- searchEdge	Graph x Value x Value x String	→	boolean
- dijkstra	Graph x Value x Value	→	Int []
- DFS	Graph	→	Graph
- DFSVALIDATOR	Graph x Values[]	→	Boolean
- getEdgesWeightList	Graph x Vertexes []	→	Int[]
- subGraphDistance	Graph x Int[]	→	Int
-			

Graph()
"Create a new Graph"
{ pre: TRUE }
{ post: a Graph is created }

addVertex(Graph, v)
"Adds vertex "v" to the graph G"
{pre: True}
{ post: The vertex is added to the Graph }

addEdge(a, b, w)
"Adds a edge from vertex "a" to vertex "b" of weight "w" "
{pre: pre: a and b must belong to the set of vertices of the graph}
{post: : An edge connecting a with a is created in the graph }

deleteVertex(v)
"Deletes the vertex "v" from the Graph "
{pre: u must belong to the set of vertices of the graph G }

{ post: : The vertex is removed from the graph G }

deleteEdge(u, v, id)

"Removes edge(u,v) with identification "id" from the graph "

{pre: *There must be an edge between u and v*}

{ post: The edge is removed from the graph G}

searchVertexIndex(a)

"Searches the index in the graph of vertex "v" "

{pre: *vertex "u" must be part of the graph*}

post: $i \in \mathbb{N} \wedge i \geq 0$

searchEdge(a, b, id)

"Checks if there is an edge between two nodes in the graph "

{pre: $[(a \wedge b) \text{ must be part of the graph}] \wedge [\text{there must be an edge between } (a \wedge b)]$ }

post: $(\text{true if there is an edge between } a \wedge b) \vee (\text{false if there is not an edge between } a \wedge b)$

Dijkstra(a, b)

"Finds the shortest path from vertex a to b in the graph. "

{pre: $[(a \wedge b) \text{ are part of the Graph}] \wedge [\text{there is an edge between } (a \wedge b)]$ }

{ post: String chain with values in ascending order}

DFS()

"Explore in depth by visiting all the neighbors of a vertex in the graph G. "

{pre: *True*}

{ post: The distance and time of discovery of each vertex in the graph G is determined.}

DFSVALIDATOR(Values [])

“Explore in depth by visiting all vertices with a pipe, assuring that the pipes path is corrected connected ”

{pre: Values: all vertices has its state different than CONNECTOR}

{ post: (True if path is correctly connected v False: if path is not corrected connected.)}

getEdgesWeightList(Vertexes[])

“Obtain the edges that connect various vertices ”

{pre: All vertices are correctly connected with at least one edge }

{ post: A parallel int array,with the weight of the edges that are connected, to the vertices array}

subGraphDistance(int[])

“Calculates the distance from the start and end vertex in a subgraph ”

{pre: the int array correspond in parallel with the edges connection of the subgraph}

{ post: distance (int) summed from the start vertex to the end vertex}