|  |
| --- |
| TAD <Graph> |
| Graph =List<node>, Map<node, List<node>> |
| Inv: {node different from null} |
| Primitive Operations:  AddVertex(X,Y): -> Void  RemoveVertex(X,Y): -> Void  AddEdge(V1, V2): -> Void  RemoveEdge(V1, V2): -> Void  FindVertex(x,y): -> Vertice  BFS(V): ->Void  DFS(V1, V2): -> void  Dijkstra(V1, V2, weight): ->ArrayList  Floyd(): ->double[][]  PrimL(): ->GraphAdjacencyList  PrimM(): ->GraphAdjacencyMatrix  KruskalL(): -> GraphAdjacencyList  KruskalM(): -> GraphAdjacencyMatrix |

RemoveVertex (x,y)

“Remove the selected vertex at the entered coordinates”

{pre: x,y}

{pos: Boolean}

AddEdge (V1,V2)

“Adds edge between two vertexes”

{pre: x,y}

{pos: void}

AddVertex(x,y)

“Add a new vertex at the entered coordinates”

{pre: x,y}

{pos: Void}

FindVertex (x,y)

“Find vertex”

{pre: x,y}

{pos: Void}

RemoveEdge(V1,V2)

“Remove the edge between two vertexes”

{pre: V1,V2}

{pos: Void}

DFS(V1,V2)

“Verify that there is a possible path”

{pre: V1,V2}

{pos: Void}

BFS()

“Explores arraylist from first vertice spreading level by level”

{pre:}

{pos: Void}

Floyd()

“find the minimum-weight path between any vertex”

{pre: }

{pos: double[][]}

Dijkstra(V1, V2, Weight)

“Find the shortest path between a source vertex and all other vertices in a graph with weights on each edge. Weight must not be negative”

{pre: V1,V2, Weight}

{pos: Void}

Prim()

“Find the minimum spanning tree”

{pre: V1,V2, Weight}

{pos: Void}

FindVertex(x,y)

“Remove the edge between two vertexes”

{pre: V1,V2}

{pos: Void}

FindVertex(x,y)

“Remove the edge between two vertexes”

{pre: V1,V2}

{pos: Void}