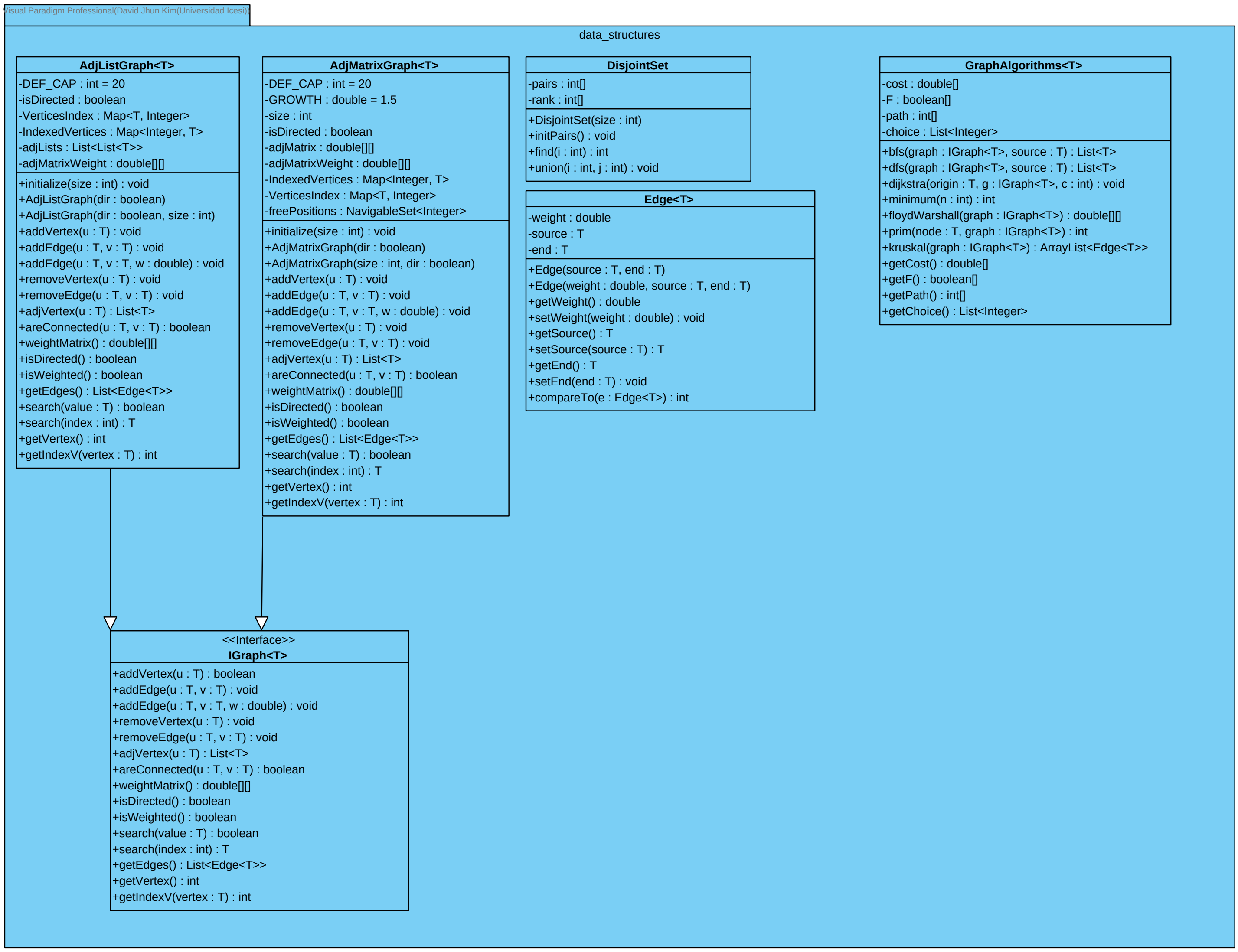
Visual Paradigm Professional(David Jhun Kim(Universidad Icesi))



data_structures

AdjListGraph<T>

-DEF_CAP : int = 20
-isDirected : boolean
-VerticesIndex : Map<T, Integer>
-IndexedVertices : Map<Integer, T>
-adjLists : List<List<T>>
-adjMatrixWeight : double[][]

+initialize(size : int) : void
+AdjListGraph(dir : boolean)
+AdjListGraph(dir : boolean, size : int)
+addVertex(u : T) : void
+addEdge(u : T, v : T) : void
+addEdge(u : T, v : T, w : double) : void
+removeVertex(u : T) : void
+removeEdge(u : T, v : T) : void
+adjVertex(u : T) : List<T>
+areConnected(u : T, v : T) : boolean
+weightMatrix() : double[][]
+isDirected() : boolean
+isWeighted() : boolean
+getEdges() : List<Edge<T>>
+search(value : T) : boolean
+search(index : int) : T
+getVertex() : int
+getIndexV(vertex : T) : int

AdjMatrixGraph<T>

-DEF_CAP : int = 20
-GROWTH : double = 1.5
-size : int
-isDirected : boolean
-adjMatrix : double[][]
-adjMatrixWeight : double[][]
-IndexedVertices : Map<Integer, T>
-VerticesIndex : Map<T, Integer>
-freePositions : NavigableSet<Integer>

+initialize(size : int) : void
+AdjMatrixGraph(dir : boolean)
+AdjMatrixGraph(size : int, dir : boolean)
+addVertex(u : T) : void
+addEdge(u : T, v : T) : void
+addEdge(u : T, v : T, w : double) : void
+removeVertex(u : T) : void
+removeEdge(u : T, v : T) : void
+adjVertex(u : T) : List<T>
+areConnected(u : T, v : T) : boolean
+weightMatrix() : double[][]
+isDirected() : boolean
+isWeighted() : boolean
+getEdges() : List<Edge<T>>
+search(value : T) : boolean
+search(index : int) : T
+getVertex() : int
+getIndexV(vertex : T) : int

DisjointSet

-pairs : int[]
-rank : int[]

+DisjointSet(size : int)
+initPairs() : void
+find(i : int) : int
+union(i : int, j : int) : void

Edge<T>

-weight : double
-source : T
-end : T

+Edge(source : T, end : T)
+Edge(weight : double, source : T, end : T)
+getWeight() : double
+setWeight(weight : double) : void
+getSource() : T
+setSource(source : T) : T
+getEnd() : T
+setEnd(end : T) : void
+compareTo(e : Edge<T>) : int

GraphAlgorithms<T>

-cost : double[]
-F : boolean[]
-path : int[]
-choice : List<Integer>

+bfs(graph : IGraph<T>, source : T) : List<T>
+dfs(graph : IGraph<T>, source : T) : List<T>
+dijkstra(origin : T, g : IGraph<T>, c : int) : void
+minimum(n : int) : int
+floydWarshall(graph : IGraph<T>) : double[][]
+prim(node : T, graph : IGraph<T>) : int
+kruskal(graph : IGraph<T>) : ArrayList<Edge<T>>
+getCost() : double[]
+getF() : boolean[]
+getPath() : int[]
+getChoice() : List<Integer>

<<Interface>>
IGraph<T>

+addVertex(u : T) : boolean
+addEdge(u : T, v : T) : void
+addEdge(u : T, v : T, w : double) : void
+removeVertex(u : T) : void
+removeEdge(u : T, v : T) : void
+adjVertex(u : T) : List<T>
+areConnected(u : T, v : T) : boolean
+weightMatrix() : double[][]
+isDirected() : boolean
+isWeighted() : boolean
+search(value : T) : boolean
+search(index : int) : T
+getEdges() : List<Edge<T>>
+getVertex() : int
+getIndexV(vertex : T) : int