TAD Grafo Objeto abstracto: Grafo Grafo: {ArrayList> = vertices HashMap> = vertexes int time = 0 int white = 1 int grey = 2 int black = 3 } Invariante: Un grafo es un conjunto de vértices y aristas, no vacío Operaciones primitivas addVertex: value T, key int -----> void deleteVertex: key int -----> void deleteAllReference: key int -----> void BFS: keyRoot int -----> void DFS: ----> void dfsVisit: ----> void getHashSize: ----> int proveConex: ----> int añadirAdyacentes: vertice int, padre int -----> void addArista: keyFrom int, keyTo int, peso int -----> void Dijkstra: source int -----> String Floyd-Warshall: grafo Grafo[][] -----> String Prim: grafo Grafo -----> String

Kruskal: grafo Grafo -----> String

```
addVertex(T value, int key) -----> void

"Crates an especific Vertex and add it into the vertexes array list"

{pre : The vertex to add is not into the vertexes array list}

{pos : Vertex added}
```

```
deleteAllReference(int key) -----> void
"Deletes all vertexes"

{pre : none}

{pos : Vertexes array list = null}
```

```
deleteVertex(int key) -----> void

"Deletes the vertex with the especific key from the vertexes array list"

{pre: The vertex to delete is into the vertexes array list}

{pos: Vertex deleted}
```

```
BFS(int keyRoot) -----> void

"Verify connectivity from the root vertex to its neighbors"

{pre : Graph ≠ null}

{pos : BF tree}
```

DFS() -----> void

"Cover all the graph vertexes"

{pre : Graph ≠ null}

{pos : DF forest}

getHashSize() -----> int

"Returns the vertexes array size"

proveConex() -----> int

"Check if the graph is strongly connected"

{pre : edge ≠ null, vertex ≠ null}

{pos:true}

añadirAdyacentes(int vertice, int padre) ----> void

"Add to the vertex padre an adjacent vertex"

addArista(int keyFrom, int keyTot, int peso) -----> void

"Add a certain edge"

(pre : The vertexes connected by the edge exist at the

vertexes array list}

{pos : true}

Dijkstra(int source) -----> String

"Returns the path with less weight from the source to a certain Vertex"

{pre : Graph ≠ null}

{pos : path with less weight}

Floyd-Warshall(Grafo[][] grafo) -----> String

"Find the shortest path between all the pairs of vertices in a weighted graph"

{pre : Graph ≠ null}

{pos : shortest path between all the pairs of vertices}

Prim(Grafo grafo) -----> String

"Find the minimum spanning tree from a graph"

{pre : Graph ≠ null}

{pos : minimum spanning tree}

Kruskal(Grafo grafo) -----> String

"Find the minimum spanning tree from a graph"

{pre : Graph ≠ null}

{pos : minimum spanning tree}