TAD de estructuras de datos:

|  |  |  |
| --- | --- | --- |
| **TAD:** | <WeightedGraph> | |
| **Description:**  A Graph is a non-linear collection containing vertices and edges connecting vertices. This ADT does not specify if the edges are directed, leaving that to an implementation. The edges have non-negative weights. | | |
| **Invariantes:** | * 1. Empty graph: number of vertices is 0; number of edges is 0. * 2. Self-loops are not allowed. * 3. Edge weights must be >= 1 | |
| **Operaciones Primitivas:** | | **Graph**()  **addEdge**( Vertex v1 , double w, Vertex v2 )  **addEdge**( Vertex v1 , Vertex v2 )  **addVertex**( Vertex v )  **getEdgeWeight**( Vertex v1 , Vertex v2 )  **setEdgeWeight**( Vertex v1 , double newWeight, Vertex v2 )  **removeVertex**( Vertex v )  **removeEdge**(Vertex v1 , Vertex v2 )  **getNeighbors**( Vertex v )  **getNumberOfVertices**()  **getNumberOfEdges**() |

|  |
| --- |
| **graph**() |
| “Initialize the WeightedGraph.” |
| **Pre-condition:** none  **Responsibilities:** initializes the graph attributes.  **Post-condition:** number of vertices is 0.  number of edges is 0 (1.0). |

|  |
| --- |
| **addEdge**( Vertex v1 , double w, Vertex v2 ) |
| “Add edge in the WeightedGraph.” |
| **Pre-condition:** v1 and v2 are Vertices in this graph and aren’t already connected  by an edge; w is >= 0.  **Responsibilities:** connect Vertices v1 to v2 with weight w; if this is an undirected  graph, this edge also connects v2 to v1.  **Post-condition**: an edge connecting v1 and v2 with weight w is added to this  Graph.  *number of edges* is incremented by 1  **Exception:** if v1 or v2 are not in the graph, are already connected by an edge,  or w < 0.  **Returns:** nothing. |

|  |
| --- |
| **addEdge**( Vertex v1 , Vertex v2 ) |
| “Add edge in the WeightedGraph.” |
| **Pre-condition:** v1 and v2 are Vertices in this graph and aren’t already connected  by an Edge.  **Responsibilities:** connect Vertices v1 to v2; if this is an undirected graph, this edge  also connects v2 to v1.  **Post-condition:** an edge connecting v1 and v2 is added to this graph *number of edges* is incremented by 1.  **Exception:** if v1 or v2 are not in the graph or are already connected by an  edge  **Returns:** nothing. |

|  |
| --- |
| **addVertex**( Vertex v ) |
| “Add the vertex in WeightedGraph.” |
| **Pre-condition:** v is not already in the graph.  **Responsibilities:** insert a Vertex into this graph.  **Post-condition:** a Vertex is added to this graph *number of vertices* is incremented by 1.  **Exception:** if Vertex v is already in this graph.  **Returns:** nothing. |

|  |
| --- |
| **getEdgeWeight**( Vertex v1 , Vertex v2 ) |
| “show the weight.” |
| **Pre-condition:** v1 and v2 are Vertices in this graph and are connected by an Edge.  **Responsibilities:** get the weight of the edge connecting Vertices v1 to v2.  **Post-condition:** the graph is unchanged.  **Exception:** if v1 or v2 are not in the graph or are not connected by an Edge.  **Returns:** the weight of the edge connecting v1 to v2. |

|  |
| --- |
| **setEdgeWeight**( Vertex v1 , double newWeight, Vertex v2 ) |
| “Change the weight.” |
| **Pre-condition:** v1 and v2 are Vertices in this graph and are connected by an edge; newWeight is >= 0.  **Responsibilities:** set the weight of the edge connecting Vertices v1 to v2 to newWeight.  **Post-condition:** the graph is unchanged.  **Exception:** if v1 or v2 are not in the graph, are not connected by an edge, or newWeight < 0.  **Returns:** nothing. |

|  |
| --- |
| **removeVertex**( Vertex v ) |
| “remove the vertex of WeightedGraph. ” |
| **Pre-condition:** v is a Vertex in this graph  **Responsibilities:** remove Vertex v from this graph.  **Post-condition:** Vertex v is removed from this graph, All edges incident on v are removed  *number of vertices* is decremented by 1  *number of edges* is decremented by degree( v ).  **Exception:** if v is not in this graph.  **Returns:** nothing. |

|  |
| --- |
| **removeEdge**(Vertex v1 , Vertex v2 ) |
| “remove the edge of WeightedGraph.” |
| **Pre-condition:** v1 and v2 are vertices in this graph and an edge exists from v1 to v2.  **Responsibilities:** remove from this graph the edge connecting v1 to v2; if this is an undirected graph, there is no longer an edge from v2 to v1.  **Post-condition:** the edge connecting v1 and v2 is removed from this graph *number of edges* is decremented by 1.  **Exception:** if v1 or v2 are not in this graph, or if no edge from v1 to v2 exists.  **Returns:** nothing. |

|  |
| --- |
| **getNeighbors**( Vertex v ) |
| “return all terms adjacent to v.” |
| **Pre-condition:** v is a Vertex in this graph.  **Responsibilities:** get the neighbors of Vertex v from this graph.  **Post-condition:** the graph is unchanged.  **Exception:** if v is not in this graph.  **Returns:** a collection containing the Vertices incident on v. |

|  |
| --- |
| **getNumberOfVertices**() |
| “Allows get number of vertex.” |
| **Pre-condition:** none.  **Responsibilities:** get the number of vertices in this graph.  **Post-condition:** the graph is unchanged.  **Returns:** the number of vertices in this graph. |

|  |
| --- |
| **getNumberOfEdges**() |
| “Allows get number of edges.” |
| **Pre-condition:** none.  **Responsibilities:** get the number of edges in this graph.  **Post-condition:** the graph is unchanged.  **Returns:** the number of edges in this graph. |