|  |  |  |
| --- | --- | --- |
| **TAD Graph** | | |
| Data structure used to represent relationships and connections between elements. This consists of a set of nodes called vertices and a set of connections called edges. These connections can have either the property of connecting multiple times among themselves or also connecting among themselves. The same addresses may or may not have an address. | | |
| *inv*:   * The edge set (E) must consist of valid pairs of vertices, ensuring that all edges connect existing vertices in the vertex set. * In an undirected graph, if an edge {vi, vj} is in E, it must also be {vj, vi} in E to maintain symmetry in the edge representation. * In a directed graph, the presence of an edge (vi, vj) in E does not imply that (vj, vi) must be in E, since the edges have a specific direction. | | |
| create |  | * Graph |
| addVertex | Graph x element | * Graph |
| addEdge | Graph x element x element | * Graph |
| deleteVertex | Graph x element | * Graph |
| deleteEdge | Graph x element | * Graph |
| DFS | Graph | * Graph |
| BFS | Graph x element | * Graph |
| dijkstraAlgorithm | Graph x element | * Graph |
| primAlgorithm | Graph | * Graph |

|  |
| --- |
| create |
| Constructor |
| Build a new graph without elements |
| precondition: |
| postcondition: A new empty graph is created |

|  |
| --- |
| addVertex |
| Modifier |
| Add an element to a graph |
| precondition: There must be a graph previously created |
| postcondition: A new element is add/save in the graph |

|  |
| --- |
| addEdge |
| Modifier |
| Link an element with it self or another element in a graph |
| precondition: There must be a graph previously created and at least one element |
| postcondition: One or two elements are connected |
| deleteVertex |
| Modifier |
| Delete an element from a graph |
| precondition: There must be at least one element in the graph |
| postcondition: A new element is delete from the graph |

|  |
| --- |
| deleteEdge |
| Modifier |
| Delete the connection between an element with it self or another element |
| precondition: There must be at least one element |
| postcondition: The connection is deleted |

|  |
| --- |
| DFS |
| Reviewer |
| It runs through the graph using indicators, verifying the depth between adjacent vertices |
| precondition: There must be at least one element |
| postcondition: Returns a route indicator |

|  |
| --- |
| BFS |
| Reviewer |
| It runs through the graph using indicators, verifying the width between adjacent vertices. |
| precondition: There must be at least one element |
| postcondition: Returns a route indicator |

|  |
| --- |
| dijkstraAlgorithm |
| Reviewer |
| It runs through the graph using indicators, finding the path of minimum length taking into account the weighting of its edges |
| precondition: The graph must have been created and it must contain vertices connected by edges with different weights |
| postcondition: Returns the path of minimum length or an indicator of this |

|  |
| --- |
| primAlgorithm |
| Reviewer |
| It runs through the graph using indicators, finding minimum generating tree taking into account the weighting of the edges |
| precondition: The graph must have been created and it must contain vertices connected by edges with different weights |
| postcondition: Returns the minimum spawning tree or an indicator of this |