| **Graph** | | | |
| --- | --- | --- | --- |
| Graph G = (V, E)  V= {v1,v2,...,vn} E= {e1, e2,..., en}  vertices edges | | | |
| Invariants:   * An already existing vertex can not be added | | | |
| Operation | Input | Output | Type |
| Graph | none | G = (V=null) (E=null) | Builder |
| addNode | G vnew | V = {...,vnew} | Modifier |
| addRelation | G v1 v2  weight | enew={v1,v2,weight}  E={...,enew} | Modifier |
| dijkstra | G vStar vEnd | int  v  {vstart,...,vEnd} | Analyzer |
| prim | G vStar | {vstart,...,vEnd} | Analyzer |
| BFS | G vstart | boolean | Analyzer |
| DFS | G | int=Timestamp | Analyzer |
| DFSvisit | G nu int | int = time | Analyzer |
| floydWarshall | G | String=weightMatrix | Analyzer |
| Kruskal | G | int=pathWeight | Analyzer |

| Graph | |
| --- | --- |
| Created the Graph without any vertices and edges | |
| **PRE** | **POST** |
| none | G= (V=null) (E=null) |

| addNode | |
| --- | --- |
| Created and add a new node to the graph | |
| **PRE** | **POST** |
| G must be initialized | G= (V={vnew,...},E) |

| addRelation | |
| --- | --- |
| Created and add a new edge to the graph | |
| **PRE** | **POST** |
| G must be initialized and V must have the vertices that are going to be related | G= (V, E={enew,...}) |

| dijkstra | |
| --- | --- |
| Takes two vertices and finds the shortest path between the vertices | |
| **PRE** | **POST** |
| G must be initialized and V must have the vertices that are going to be used | {vstart,...,vEnd}  v  int = weighPath |

| prim | |
| --- | --- |
| Take a initial vertex and finds the shortest path with that vertex | |
| **PRE** | **POST** |
| G must be initialized and V must have the vertex that are going to be used | {vstart,...,vEnd}  v  int = weighPath |

| BFS | |
| --- | --- |
| Take initial vertex and check if the graph is related since that point | |
| **PRE** | **POST** |
| G must be initialized and the root vertex must exist | Reachable vertices from root are colored |

| DFS | |
| --- | --- |
| Runs DFSvisit in every node that has not been visited | |
| **PRE** | **POST** |
| G must be initialized | timestamp |

| DFSvisit | |
| --- | --- |
|  | |
| **PRE** | **POST** |
| G must be initialized and vertex must be initialized and has not been visited | timestamp |

| floydWarshall | |
| --- | --- |
| Take the graph and finds the shortest path between the vertices in the graph if it’s possible to connect the vertices | |
| **PRE** | **POST** |
| G must be initialized and V must have the vertices that are going to be used | distances [][] |

| Kruskal | |
| --- | --- |
| Returns the set of edges that includes all vertices and has the minimum weight possible | |
| **PRE** | **POST** |
| G must be initialized | minimum weight  v  set of edges |