Primer entrega Proyecto Final

Problema: Una empresa móvil que realiza visitas para mostrar sus productos y lleva productos a domicilio por varios puntos de la ciudad. Esta empresa al analizar los costos que se lleva realizando las rutas de visita y entrega se dio que el valor es alto, entonces está buscando una solución para reducir el costo sin dejar de realizar las rutas completa, y también que si se añade un nuevo domicilio no genere un gasto muy grande. Entonces requieren una aplicación que les indiquen la ruta más corta y con menos costo visitando todas las personas que solicitaron servicio de visita o domicilio.

TAD:

|  |
| --- |
| TAD GRAPH |
| Representación:    3  2  1  5  1  45  1  1  5  1  3  4  10 |
| Invariante:  {E1 (Vj,Vk); 0<=j<=k} |
| Operations:  +CreateGraph: -----🡪 Graph  +addVertex:Graph x Vertex ----🡪 Graph  +addEgde: Graph x Vertex x Vertex  +removeVertex: Graph x Vertex  +removeEgde: Graph x Egde  +vertexAdjacent: Graph x Vertex  +areConnected: Graph x Vertex x Vertex |

|  |
| --- |
| creatGraph()  “Crea un nuevo grafo vacio” |

|  |
| --- |
| addVertex(Graph,Vertex)  añade un vertice al grafo  pre: Graph!=null^V1!=null |

|  |
| --- |
| addEgde(Graph,Vertex,Vertex)  añade una arista entre los dos vertices  pre: Graph!=null^V1!=null^V2!=null |

|  |
| --- |
| removesVertex(Graph,Vertex)  se eleimina un vertice del grafo  pre: V1!=null^V2!=null |

|  |
| --- |
| removesEgde(Graph,Egde)  se elimina la arista así desconectando dos vertices |

|  |
| --- |
| vertexAdjacent(Graph,Vertex)  retorna una lista de vértices que contiene los vértices adyacentes a este vertice  pre: Graph!=null^V1!=null |

|  |
| --- |
| areConected(Graph,Vertex,Vertex)  indica si dos vértices esta conectados  pre: Graph!=null^V1!=null ^V2!=null |

Diseño de casos de prueba

Graph

Escenario 1: se crea un grafo simple con los siguiente vértices y arista:

(a,b),(c,d),(a,e),(b,d),(a,c),(e,d)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba no.1 | Objetivo: Comprobar si se crea un grafo vacio | | | |
| Clase | Método | Escenario | Valores de Entrada | Valore de Salida |
| Graph | createGraph() | 1 | 0 | Graph vacio |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba no.2 | Objetivo: Comprobar si se añade un vertice al grafo | | | |
| Clase | Método | Escenario | Valores de Entrada | Valore de Salida |
| Graph | addVertex () | 1 | f | Graph con el nuevo vertice F |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prueba no.3 | Objetivo: Comprobar si se añade una arista entre dos vertices | | | |
| Clase | Método | Escenario | Valores de Entrada | Valore de Salida |
| Graph | addEgde () | 1 | D,f | Graph con la arista entre d y f |