TAD Grafo (Matriz de adyacencia).

|  |
| --- |
| TAD: Grafo. |
| Representacion   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | A | B | C | D | E | F | G | | A | 0 | 12 | 0 | 0 | 0 | 2 | 4 | | B | 12 | 0 | 6 | 0 | 0 | 0 | 0 | | C | 0 | 6 | 0 | 9 | 0 | 0 | 0 | | D | 0 | 0 | 9 | 0 | 8 | 0 | 10 | | E | 0 | 0 | 0 | 8 | 0 | 10 | 0 | | F | 2 | 0 | 0 | 0 | 10 | 0 | 0 | | G | 4 | 0 | 0 | 10 | 0 | 0 | 0 |   9  8  10  10  4  6  2  12 |
| Invariantes:   * Un grafo se define como G = (V, E) donde V es el conjunto de vértices y E el conjunto de las aristas * Cada arista conecta dos vértices. * Dos vértices son adyacentes si tienen una arista que los conecta. * Una arista es incidente a un vértice si esta lo une a otro. * Ponderación es una función que a cada arista se le asocia un valor. |
| Operaciones   * <ContruirGrafo>: Grafo * <AgregarVertice>: Grafo, Valor Grafo * <AgregarArista>: Grafo, Valor1, Valor2, Etiqueta Grafo * <EliminarVertice>: Grafo, Valor Grafo * <RecorrerGrafo>: Grafo, Valor Grafo * <CaminoMasCorto>: Grafo, Valor1 Entero * <EliminarArista>: Grafo, Valor1, Valor2 Grafo |

|  |
| --- |
| ConstruirGrafo() |
| “Crea un nuevo grafo sin ningún vértice”  {Pre: True}  {Post: Grafo = {Grafo} |

|  |
| --- |
| AgregarVertice() |
| “Agrega un nuevo vértice al grafo, sin ninguna conexión ya que no vértices adyacentes”  {Pre: Grafo = {Grafo}, Valor; Valor = dato}  {Post: Grafo = Grafo con un vértice nuevo agregado} |

|  |
| --- |
| AgregarArista() |
| “Agrega una nueva arista al grafo, esta conecta dos vértices especificados, esto se agrega en la matriz de adyacencia que representa la conexión entre los vértices”  {Pre: Grafo = {Grafo}, Valor1 = dato del vértice; Valor2 = dato del vértice 2; Etiqueta = Entero que representa ponderación en la arista}  {Post: Grafo con unja nueva arista} |

|  |
| --- |
| EliminarVertice() |
| “Elimina el vértice que se requiere de acuerdo con el valor especificado, además si este tenía una arista esta también se eliminara”  {Pre: Grafo = {Grafo}, Valor = dato del vértice}  {Post: Grafo sin el nodo con la llave} |

|  |
| --- |
| RecorrerGrafo() |
| “Recorre todo el grafo sin repetir los vértices”  {Pre: Grafo = {Grafo}, Valor = dato del vértice desde donde inicia el recorrido}  {Post: Grafo} |

|  |
| --- |
| CaminoMasCorto() |
| “Retorna el valor de la distancia más corta entre dos vértices, esto de acuerdo con la etiqueta de las aristas”  {Pre: Grafo= {Grafo}, Valor1 = dato del vértice; Valor2 = dato del vértice al cual se quiere llegar, valor}  {Post: Entero que representa la distancia más corta} |