|  |
| --- |
| TAD(Grafo genérico)  **Grafo<T>** |
| **Descripción:** Un grafo no dirigido es un tipo de grafo en el que los nodos (vértices) están conectados por aristas sin dirección. Un grafo dirigido es un tipo de grafo en el que los nodos están conectados por aristas con dirección, por lo cual A podría ir a B, pero no necesariamente B a A. Además, cada arista tiene un peso asociado(En nuestra implementación). |
| **Invariantes:**   1. Cada nodo en el grafo debe tener un identificador único, como un número entero. 2. Las aristas no pueden conectar un nodo consigo mismo. 3. No puede haber múltiples aristas entre el mismo par de nodos(A menos que sea uno dirigido, y una arista tenga un peso y la otra otro, y direcciones contrarias). 4. Cada arista tiene un peso asociado, que es un valor numérico. |
| **Operaciones:**   1. **addVertex(T vertex1, T vertex2, int weight):**    * **Descripción:** Agrega un nuevo nodo al grafo.    * **Input:** Los dos nodos que serán conectados por una arista y el peso de la arista.    * **Output:** NA. 2. **addVertexDirigido(T vertex1, T vertex2, int weight):**  * **Descripción:** Agrega dos nodos y una arista al grafo. * **Input:** Los nodos a ser conectados. * **Output:** NA.  1. **deleteVertex(T vertex):**    * **Descripción:** Elimina un vértice y todas las aristas asociadas a ese vértice del grafo.    * **Input:** El vértice.    * **Output:** NA. 2. **deleteEdge(T vertex1, T vertex2):**    * **Descripción:** Elimina una arista específica entre dos vértices del grafo.    * **Input:** Los dos vértices conectados por la arista a eliminar.    * **Output:** NA. 3. **getEdgeWeight(T vertex1, T vertex2):**    * **Descripción:** Obtiene el peso de una arista específica entre dos vértices.    * **Input:** Los dos vértices conectados por la arista.    * **Output:** El peso de la arista (int). 4. **checkConnect(T vertex1, T vertex2):**    * **Descripción:** Verifica si existe una arista que conecta dos vértices específicos.    * **Input:** Los dos vértices conectados por la arista.    * **Output:** Retorna un valor booleano que indica si los dos nodos están conectados por una arista. 5. **fillMatrix():**  * **Descripción:** Toma los valores de la lista de adyacencia y los usa para llenar la matriz. * **Input:** NA * **Output:** NA  1. **floydWarshall()**  * **Description:** Acomoda la matriz para que tenga la información del camino más corto de un camino a otro. * **Input:** NA. * **Output:** NA.  1. **Dijkstra(T start, T destination):**  * **Description:** Encuentra el camino más corto entre un vértice salida dado, un vértice llegada dado. * **Input:** T start, T destination. (Los nodos de comienzo y final, respectivamente) * **Output:** int minDistance. (Valor del camino más corto) |