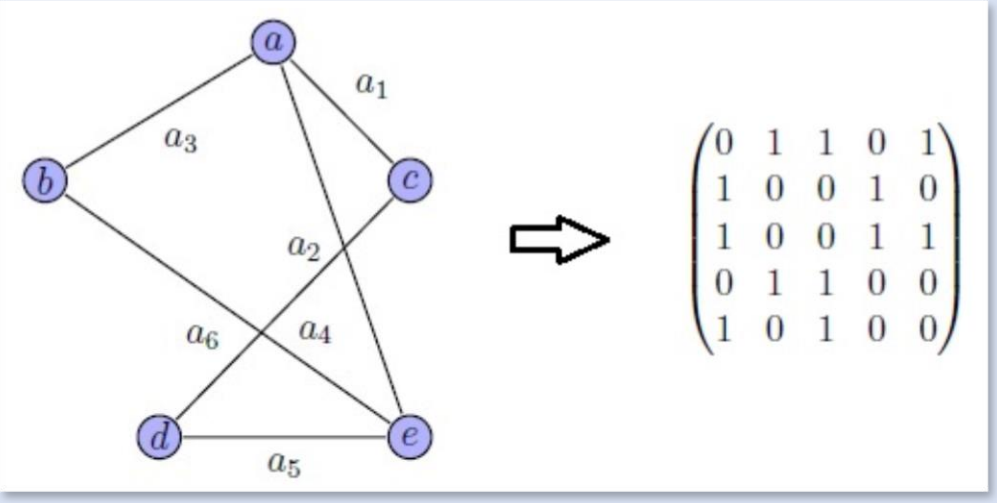
TAD GRAFO



{*inv.: Un grafo consta de un conjunto V de vértices y conjunto E de aristas (V, E). El grafo puede ser: (1) dirigido o no dirigido. (2) Multígrafo o grafo simple. (2) conexo o no conexo. (3) Sus aristas pueden tener o no peso y (4) puede existir o no una jerarquía. Por otro lado, se puede representar por medio de una matriz de adyacencia en la cual se da una relación entre los nodos y la posición fila columna (i, j) con un número 1 o de lo contrario 0. También se puede representar por medio de una lista de adyacencia la cual se asocia a cada nodo del grafo una lista que contenga todos los nodos adyacentes a él.*}

Operaciones Primitivas:

• CrearGrafo:		-> GRAFO
• AgregarArista:	*GRAFOxLlavexLlave*	-> GRAFO
• AgregarVertice:	*GRAFOxLlavexVertice*	-> GRAFO
• EstaVacío:	*GRAFO*	-> Booleano
• BuscarVertice:	*GRAFOxLlave*	-> Vertice
• EliminarVertice:	*GRAFOxLlave*	-> GRAFO
• EliminarArista:	*GRAFOxLlavexLlave*	-> GRAFO
• Dijkstra	*GRAFOxVertice*	-> Booleano
• BellmanFord	*GRAFOxVertice*	-> Booleano

CrearGrafo()

"Crea un grafo"

{*pre:* TRUE}

{post: *Se crea un grafo vacío*}

AgregarArista(*Llave, Llave***)**

"Agrega la arista entre dos vértices buscados de acuerdo con las llaves dadas y de acuerdo a sí es dirigido o no dirigido"

{*pre:* GRAFO!=NIL ^ ∀ (Vertice.Llave==Llave)!=NIL}

{post: *Se conectan los dos vértices con las llaves dadas*}

AgregarVertice(*Llave, Vertice***)**

"Se agrega un nuevo vértice conectándolo con el vértice al que pertenece la llave"

{*pre:* GRAFO!=NIL ˆ (Vertice.Llave == Llave)!=NIL}

{post: *Se agrega un nuevo vértice conectando con (Vertice.Llave == Llave)*}

EstaVacio()

"Devuelve un valor de verdadero o falso si el GRAFO no tiene ningún vertice o arista"

{*pre: GRAFO!=NIL*}

{post: TRUE si no existe ningún Vertice y Arista}

BuscarVertice(*Llave***)**

"Devuelve un Vertice que su llave coincide con la llave del parámetro"

{*pre: GRAFO!=NIL ^ Llave!=NIL*}

{post: Vertice donde Vertice.Llave==Llave o NIL si ∀ (Vertice.Llave!=Llave)}

EliminarVertice(*Llave***)**

"Elimina un Vertice donde Vertice.Llave==Llave"

{*pre: GRAFO!=NIL ^ Llave!=NIL*}

{post: GRAFO con Vertice.Llave==Llave eliminado}

EliminarArista(*Llave, Llave***)**

“Elimina la arista que une Vertice.Llave==Llave con la otra llave Vertice.Llave==Llave”

{*pre: GRAFO!=NILL* ^ ∀ Llave!=NILL}

{post: Arista eliminada entre los dos nodos dados de acuerdo con las llaves dadas}

Dijkstra(Vertice**)**

“Identifica si hay ciclos en el grafo a partir del Vertice dado. Este es el vértice de inicio del grafo”

{*pre: GRAFO!=NILL* ^ Vertice!=NILL}

{post: TRUE si existe un ciclo en el grafo. De lo contrario, FALSE}

BellmanFord(Vertice**)**

“Identifica si hay ciclos -tanto positivos como negativos- en el grafo a partir del Vertice dado. Este es el vértice de inicio del grafo”

{*pre: GRAFO!=NILL* ^ Vertice!=NILL}

{post: TRUE si existe un ciclo en el grafo. De lo contrario, FALSE}