GUIA 2 ESTRUCTURA DE DATOS 3.

**Gráficas**

* Estructura de datos no-lineal compuesta de vértices y aristas.
* Cada nodo tiene un valor.
* Cada arista puede tener un valor (distancia o peso).

**Tipos de gráficas**

* Dirigidas y no dirigidas. Las aristas tienen o no dirección.
* Conexas y disconexas. En la conexa desde cualquier nodo se puede llegar a otro.
* Ponderadas y no ponderadas. Las aristas tienen un peso.

**Gráficas vs Árboles**

* Una gráfica no tiene raíz.
* Una gráfica puede tener ciclos.

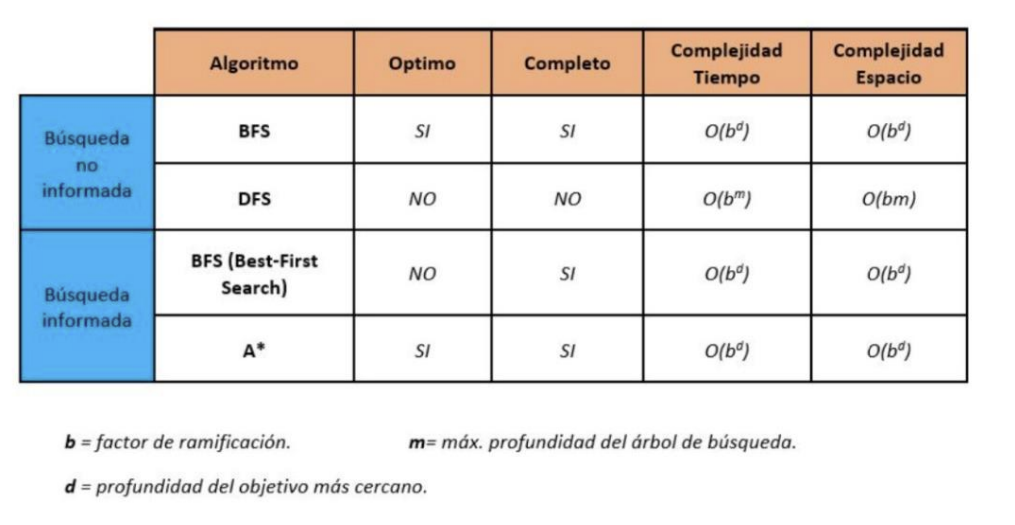
**Representación Gráficas**

* Matriz de adyacencia.
  + Mejor para gráficas densas. (la mayoría de los nodos están conectados entre sí, V^2)
  + Matriz VxV.
  + Si se van a realizar muchas consultas sobre la existencia de aristas en el grafo, entonces una matriz de adyacencia puede ser más eficiente ya que permite verificar rápidamente si dos nodos están conectados en tiempo constante (O(1)).
* Lista de adyacencia.
  + Mejor para gráficas dispersas (el número de aristas es mucho menor que el número máximo posible de aristas, lo que significa que muchos nodos no están conectados entre sí. En una gráfica dispersa, la mayoría de las entradas de la matriz de adyacencia serán nulas, lo que hace que su uso no sea eficiente en términos de memoria)
  + V listas de listas.
  + Si el grafo es ponderado, entonces una lista de adyacencia es más adecuada ya que se puede almacenar el peso de cada arista en la lista.
  + La inserción y eliminación de aristas son más eficientes

**Algoritmos de búsqueda**

Criterios:

* **Completitud**. Capacidad de encontrar una solución si existe una. Un algoritmo completo es aquel que SIEMPRE encuentra una solución si existe una.
* **Optimalidad**. Capacidad de encontrar la solución de mayor calidad (con la menor cantidad de pasos o con la mejor calidad.
* **Complejidad** **de tiempo**.
* **Complejidad** **de espacio**.



**DFS (Búsqueda en profundidad)**

**BFS (Búsqueda en amplitud)**

* Cola para los vértices.
* Mientras no esté vacía, sacamos el siguiente.
* Para cada nodo conexo y no visitado, lo marcamos y lo metemos a la cola.

**Heurísticas**

* Encontrar aproximaciones a soluciones a problemas con complejidad super polinomial.
* Guiar al algoritmo para elegir buenas opciones.
* Sacrifican optimalidad por completitud.
* Nearest-Neighbour, Ant-Colony.

**Búsqueda Informada**

* Tienen información acerca del objetivo de la búsqueda.
* Información a través de heurísticas: distancia Euclidiana, Manhattan
* Cálculo que da una estimación rápida.

**Best-First-Search (Búsqueda primero el mejor)**

* Función de evaluación para decidir cual vértice adyacente es el más prometedor.
* Única guía es la heurística.
* Requiere heap o priority queue.
* BFS con prioridad.
* Problemas:
  + Greedy.
  + No minimiza la distancia de la ruta.

**A\* Search**

* Escoge el siguiente nodo a explorar basado en el valor de la función f(n) = g(n) + h(n).
  + g(n) = cuanto ha costado moverse desde el origen hasta el nodo actual.
  + h(n) = costo estimado de moverse desde el nodo actual hasta el final.
* Un nodo no debe considerarse como opción hasta que se salga de la cola. En ese momento se marca como visitado

**Búsqueda informada vs no informada.**

Informed Search (en comparación con Uninformed Search, Blind Search):

* Heuristic Search
* Usa conocimiento para el proceso de búsqueda
* Encuentra las soluciones más rápido
* Costo es bajo
* Consume menos tiempo
* Es menos largo mientras se implementa.
* Es más eficiente

En general, una búsqueda informada es más eficiente que una búsqueda no informada, ya que utiliza información adicional para guiar el proceso de búsqueda y puede llegar más rápidamente a la solución óptima. Sin embargo, esto depende del problema en cuestión y de la calidad de la heurística utilizada en la búsqueda informada. En algunos casos, una búsqueda no informada puede ser más eficiente si la complejidad de la heurística utilizada es alta y no proporciona una buena guía para el algoritmo de búsqueda.