**Tablas para medir los algoritmos**

**Métodos sin Información**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algoritmo** | **Cantidad de nodos** | **Complejidad temporal (segundos)** | **Complejidad especial (bytes)** | **Completitud** | **Optimalidad** |
| Depth First Search | 10 - 100 (44) | 0.008 | 3202 | No | No |
| 10 – 100 (66) | 0.008 | 5504 |
| 10000 – 1000000 (178633) | 118.861 | 849852 |
| 10000 – 1000000 (11863) | 0.172 | 64136 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Depth Limited Search | 10 – 100 (97) | 0.205 | 8450 | No | No |
| 10 - 100 (44) | 0.007 | 3234 |
| 10000 – 1000000 (178633) | 136.07 | 866156 |
| 10000 – 1000000 (11863) | 0.145 | 65288 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Iterative Deepening Search | 10 – 100 (81) | 0.232 | 2612 | No | Sí |
| 10 - 100 (44) | 0.02 | 2992 |
| 10000 – 1000000 (178633) | 1637.666 | 2031410 |
| 10000 – 1000000 (11863) | 7.107 | 351178 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Breadth First Search | 10 – 100 (98) | 0.362 | 9874 | Sí | Sí |
| 10 - 100 (44) | 0.003 | 3338 |
| 10000 – 1000000 (178633) | 133.307 | 1572670 |
| 10000 – 1000000 (11863) | 12.239 | 595528 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Uniform Cost Search | 10 – 100 (50) | 0.029 | 4844 | Sí | Si |
| 10 – 100 (90) | 0.025 | 6714 |
| 10000 – 1000000 (11863) | 3346.071 | 731220 |
| 10000 – 1000000 (27665) | 5008.341 | 873008 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Bidirectional Search | 10 – 100 (40) | 0.005 | 2294 | Sí | Sí |
| 10 - 100 (89) | 0.015 | 3480 |
| 10000 – 1000000 (12730) | 2.234 | 146134 |
| 10000 – 1000000 (11863) | 0.903 | 127224 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |

**Métodos con Información**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algoritmo** | **Cantidad de nodos** | **Complejidad temporal (segundos)** | **Complejidad especial (bytes)** | **Completitud** | **Optimalidad** |
| Simulated Annealing | 10-100 (18) | 0.018 | 1962 | No | No |
| 10-100  (73) | 0.077 | 5552 |
| 10000-1000000  (12643) | 112.847 | 2669948 |
| 10000-1000000  (18147) | 191.261 | 3448370 |
| 1000000 o más (2054842) | -- | -- |
| 1000000 o más  (1401204) | -- | -- |
| Tabu Search | 10-100  (64) | 0.014 | 2572 | No | No |
| 10-100  (88) | 0.02 | 4696 |
| 10000-1000000  (12584) | 37.518 | 1597132 |
| 10000-1000000  (14572) | 24.018 | 1146994 |
| 1000000 o más  () | -- | -- |
| 1000000 o más () | -- | -- |
| Best First Search | 10 – 100 (98) | 0.009 | 4652 | Si | No |
| 10 - 100 (82) | 0.006 | 3450 |
| 10000 – 1000000 (10500) | 131.595 | 890884 |
| 10000 – 1000000 (12100) | 70.577 | 798626 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| A\* Search | 10 – 100 (64) | 0.009 | 3384 | Si | Si |
| 10 - 100 (98) | 0.01 | 4708 |
| 10000 – 1000000 (10500) | 134.595 | 932770 |
| 10000 – 1000000 (12100) | 51.691 | 702460 |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |
| Hill Climbing | 10 – 100 (60) | 0.022 | 3152 | No | No |
| 10 - 100 (57) | 0.006 | 1484 |
| 10000 – 1000000 () | -- | -- |
| 10000 – 1000000 () | -- | -- |
| 1000000 o más () | -- | -- |
| 1000000 o más () | -- | -- |

**Métodos sin información**

* 10 – 100 nodos
  + En esta categoría, basados en los datos que nos generaron los algoritmos, podemos observar que el método Bidirectional Search es el más eficiente con respecto al tiempo, ya que en los casos de prueba fue el que duró menos, pero lo que es referente a la memoria, se observa que el mejor algoritmo fue el Iterative Deepening Search.
  + En el caso del tiempo el algoritmo de Bidirectional Search aumenta la rapidez de búsqueda al realizar dos búsquedas simultáneas, una desde el nodo inicio dado hasta el destino y otra desde el destino al nodo inicio. De esta manera se reduce el tiempo de búsqueda ya que se acaba cuando ambos agentes se encuentran a mitad del camino.
* 10000 – 1000000 nodos
  + En esta categoría, basados en los datos que nos generaron los algoritmos, podemos observar que el método Bidirectional Search es el más eficiente con respecto al tiempo, ya que en los casos de prueba fue el que duró relativamente menos, pero lo que es referente a la memoria, se observa que el mejor algoritmo fue el Depth First Search, el cual consumió mucho menos.
  + En el caso del tiempo el algoritmo de Bidirectional Search aumenta la rapidez de búsqueda al realizar dos búsquedas simultáneas, una desde el nodo inicio dado hasta el destino y otra desde el destino al nodo inicio. De esta manera se reduce el tiempo de búsqueda ya que se acaba cuando ambos agentes se encuentran a mitad del camino.

**Métodos con información**

* 10 - 100 nodos
  + En esta categoría, basados en los datos que nos generaron los algoritmos, podemos observar que el método Best First Search es el más eficiente con respecto al tiempo, ya que en los casos de prueba fue el que duró menos, pero lo que es referente a la memoria, se observa que el mejor algoritmo fue el Hill Climbing.
  + En el caso de la memoria, el algoritmo Hill Climbing reduce mucho el espacio de búsqueda ya que la lista de nodos abiertos se vacía cuando se elige el camino a tomar (se elige el mejor en ese estado), con esto utiliza mucha menos memoria que los otros algoritmos
* 10000 - 1000000 nodos
  + En esta categoría, basados en los datos que nos generaron los algoritmos, podemos observar que el método Tabu es el más eficiente con respecto al tiempo y a la memoria, ya que en los casos de prueba fue el que duró menos y e que menos consumió memoria.
  + Basados en los resultados, se puede observar que el algoritmo Tabu utiliza menos memoria y tiempo para encontrar una solución, no obstante esta solución no siempre es la de menor costo ya que el algoritmo selecciona la primera solución que encuentre y le realiza alteraciones para encontrar nuevas soluciones (crea vecinos), y aunque devuelve la solución más corta entre esos vecinos pueden que existan soluciones más óptimas que no se están encontrando debido a que solo se está utilizando de base una solución inicial que puede que sea muy costosa.

En el caso de los algoritmos SA y Tabu, esta prueba con 1000000 o más nodos resultó imposible de realizar, esto debido a que ambos algoritmos poseen una función recursiva que se invoca tantas veces que provoca que la pila de llamadas de Javascript llegue a su límite, cancelando así la continuación del algoritmo y llamando la siguiente excepción “Maximum call stack size exceeded”.

El algoritmo Hill Climbing falla con grafos de gran cantidad de nodos, por ello no se hicieron pruebas de más de 10000 nodos. Esto sucede porque el algoritmo evita el backtracking al elegir un solo nodo para avanzar con cada paso que hace en la búsqueda, ignorando las demás opciones en una profundidad dada. Al hacer esto se cortan muchos posibles caminos y resulta en que la búsqueda termine pronto al acabarse los posibles caminos a seguir sin haber encontrado solución.

La evaluación de los algoritmos con diferentes tipos de grafos fue complicada ya que los estados iniciales y finales eran seleccionados al azar y no había forma de seleccionarlos a gusto de uno por la gran cantidad de nodos que habían. Por esta razón, muchas pruebas daban error indicando que no había una ruta y ya con grafos de un millón o más nodos, todos los algoritmos estuvieron a prueba por varias horas sin dar resultado alguno, por lo que se decidió parar la ejecución de estas pruebas. El factor de hardware y software también influyó en esta decisión.