Adina Han
Cristina Molina
Jaime del Rey

Práctica 2A

Misioneros

Algoritmo	Longitud	N. analizados	Tiempo	Heurística
Uniforme	11	15	624 µs	NO
Profundidad con grafo	11	12	558 µs	NO
Profundidad limitada (50)	49	14702	70.5 ms	NO
Profundidad con árbol	/	/	/	NO
Anchura con grafo	11	15	1.62 ms	NO
Anchura con árbol	11	11878	120 ms	NO
Profundización iterativa	11	20975	97.7 ms	NO
A*	11	14	415 µs	h
Primero el mejor	11	14	386 µs	h

*la heurística h está definida en el documento misioneros.ipynb

Análisis:

Para analizar cuan óptima es una solución deberemos tener en cuenta no sólo el tiempo que tarda si no también la profundidad que alcanza. Las dos soluciones más óptimas son el algoritmo ***A**** y ***primero el mejor***, se puede observar que son prácticamente iguales en cuanto al número de nodos que analizan y la longitud, en cuanto al tiempo se puede ver que es casi idéntico. Estos dos métodos son tan rápidos y dan una solución tan óptima por que pretenden llegar a la solución lo más rápido posible y descartan caminos por los cuales no pueden encontrar una solución óptima.

Tanto búsqueda con ***coste uniforme*** como el algoritmo de ***profundidad con grafos*** tienen también una solución óptima, al analizar los resultados obtenidos en la búsqueda de coste uniforme nos damos cuenta de que tiene bastante sentido que sea eficiente ya que es un algoritmo de búsqueda no informada en el que se van visitando los nodos según su coste, siempre empezando por el nodo que menos coste tiene hasta llegar al nodo objetivo, algo parecido pasa con la búsqueda en ***profundidad con grafos***, ya que recorre los nodos de forma ordenada aunque no uniforme.

El algoritmo de ***profundidad limitada***(algoritmo no informado), es un algoritmo para explorar los vértices de un grafo, ofrece una búsqueda en profundidad con límite de profundidad para descartar caminos en los que teóricamente no encontraríamos un nodo objetivo lo suficientemente cercano a el nodo inicial. La búsqueda con ***profundidad limitada*** garantiza completitud a los grafos ya que encuentra una solución siempre que se encuentre dentro del límite. La longitud del algoritmo es mucho mayor que el resto pues se queda con la primera respuesta que encuentra aunque esta sea más cara que la que pueda tener otro camino.

Es lógico por otra parte pensar que ***búsqueda en anchura*** pueda tardar un poco más y aunque consideramos que da una solución óptima es más costosa que por ejemplo el algoritmo ***A**** ya que recorre todas las aristas de un grafo, con lo cual toma un nodo del grafo como raíz y se exploran todos los vecinos de este nodo, a continuación para cada nodo se exploran todos los vecinos y así sucesivamente hasta que recorremos todo el grafo.

Se puede ver que hay una gran diferencia de tiempo entre ***búsqueda en anchura con árboles y con grafos***, esto se debe a que mientras que en los árboles se ve por niveles y mirando los hijos de cada elemento con los grafos esto no pasa, ya que se analizan todos los vecinos de un nodo, esto nos permite recorrer caminos diferentes en menos tiempo ya que no hace falta que lleguemos al final de un camino para poder elegir otro.

Podemos observar que el que más nodos analizados tiene es el de ***iterativo en profundidad*** por lo que el coste en memoria de mayor, esto se debe a que puede expandir nodos que provocan ciclos, analizando así un mismo nodo varias veces, por esta misma razón es también uno de los algoritmos que más tiempo tarda. Sin embargo la ***búsqueda en anchura con árbol*** aunque analiza menos nodos tarda más, esto se debe a que el algoritmo (Breadth first tree search) va analizando por niveles y a sus respectivos hijos.

Por otra parte el que menos espacio ocupa en memoria es el algoritmos de ***profundidad con grafos***, tenemos en cuenta un control de repetidos para este algoritmo por que si no, al ser un grafo y poder tener ciclos, podría visitar el mismo nodo varias veces como nos pasa en la búsqueda iterativa en profundidad.

En el problema de los misioneros hemos encontrado un algoritmo que no hemos conseguido que termine. Este algoritmo es el de ***profundidad usando tree search*** y la no terminación se debe a que al no haber un control de repetidos el algoritmo puede entrar en

un bucle al repetir una y otra vez la misma secuencia de nodos.

Conclusión:

Según los resultados obtenidos, se puede ver que el mejor algoritmo para resolver este problema es ***primero el mejor*** ya que la tanto la longitud como el tiempo del mismo dan unos resultados muy bajos y el número de nodos que analiza no difiere mucho al que menos nodos analiza, por lo tanto su coste en memoria no es muy alto.