Fomentando el Car-Pooling

Juan Camilo Jiménez Rojas
Universidad
Colombia
jcjimenezr@eafit.edu.co

Santiago Espinosa Valderrama
Universidad
Colombia
sespinosav@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

Palabras Claves.

A*, Árbol, Algoritmos, Búsqueda, Complejidad, Datos, Estructura, Heurística, Profundidad, *Node, Graph, Dijkstra.*

Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths

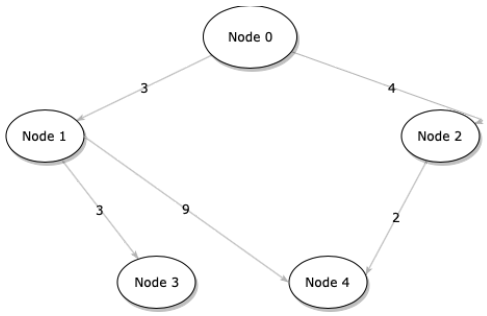
4. Algoritmo A*

A* está catalogado con unos de los mejores algoritmos de búsquedas en grafos o árboles binarios, debido a la forma de cómo maneja sus costes.

Su función de evaluación que etiquetar los nodos de la red con un valor heurístico y los arcos con el valor de para dirigirse a los nodos conexos.

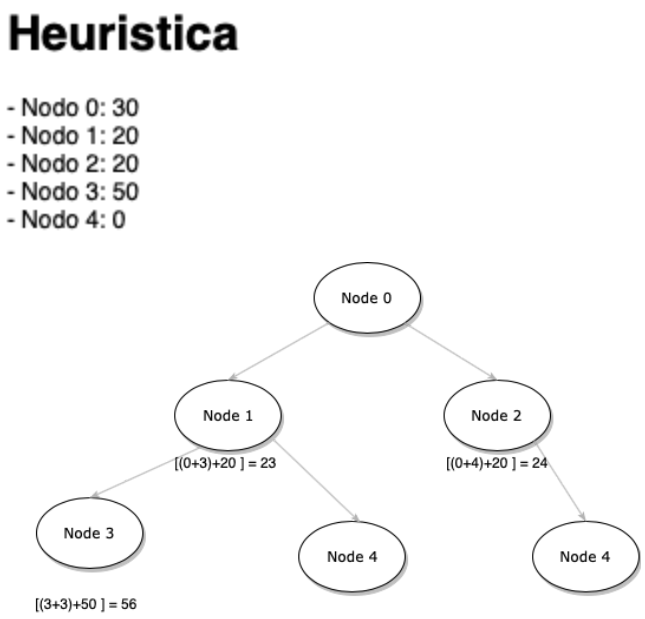
El algoritmo tiene dos funciones, una de ellas indicará la distancia actual desde el nodo origen hasta el nodo a etiquetar, y la otra expresa la distancia estimada desde este nodo a etiquetar hasta el nodo destino, comenzando a evaluar cada camino existente, siendo así no genera uno solo camino sino varios camino y tomará el más óptimo

4.1 Estructura de datos



Gráfica 1: Grafo con diferentes distancias, cada nodo representa un lugar.

4.2 Operaciones de la estructura de datos



Gráfica 2: Árbol del grafo con las fórmulas realizadas para llegar a los diferentes nodos.

En el gráfico anterior podemos ver que cada no tiene un valor heurístico asignado y la sima entre la distancia de el nodo anterior más el valor heurístico podemos calcular la ruta más corta entre el Node 0 y el Node 4 que es el destino.

4.3 Criterios de diseño de la estructura de datos

- Es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad:

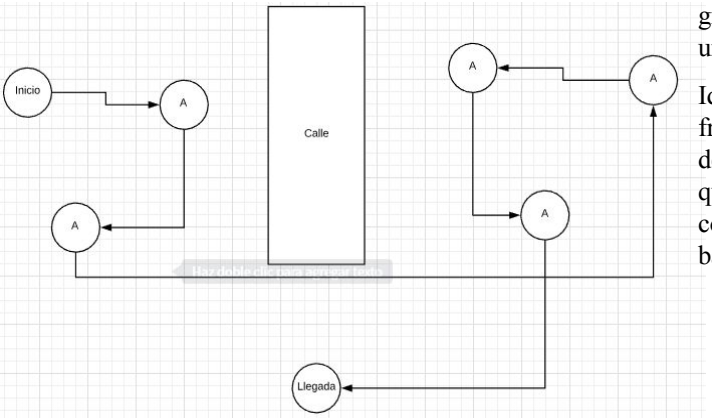
mientras que h(n) tiende a primero en profundidad, g(n) tiende a primero en anchura. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores.

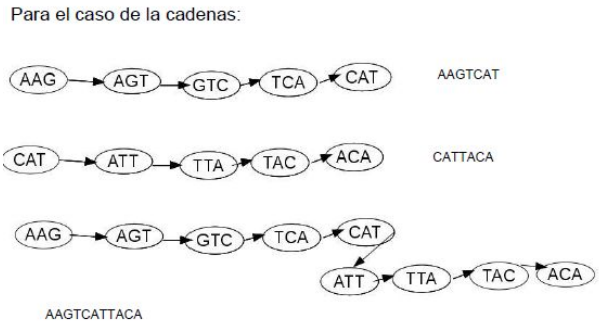
- Su velocidad puede llegar ser mucho mayor que su algoritmo padre que es el Dijkstra y esa es la prioridad en nuestro proyecto
- La facilidad de implementación.
- Tiene algunas buenas referencias acerca elementos de ruda de programación en la industria como Google y Waze implementa este algoritmo para encontrar las rutas más cortas entre dos puntos.

4.4 Análisis de complejidad

Mètodo	Complejidad
Bùsqueda Fonètica	O(1)
Imprimir búsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Búsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir bùsqueda	O(s)

4.5 Gráfico de solución.





4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear el grafo de *Bruijn* con las secuencias	O(S)
Actualizar el grafo de *Bruijn* con las secuencias	$O(A.V^2)$
Encontrar los genes	O(S)
Complejidad Total	$O(A.S^2 + V)$

4.7 Criterios de diseño del algoritmo.

El problema requiere del análisis más efectivo para elegir el camino más corto para recoger a los usuarios.

Tras investigar las diferentes opciones de búsqueda en grafos y reconocer las necesidades del problema, se llega a una conclusión:

Identificando que la conglomeración de los clientes es frecuente en diferentes situaciones, lo que facilita el hecho de que el algoritmo A* sea una solución viable, debido a que en grafos donde los nodos tienen una cantidad considerable de vértices adyacentes, este algoritmo trabaja bastante bien.

4.8 Tiempos de Ejecución

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Mejor caso	10 sg	20 sg	5 sg
Caso promedio	12 sg	10 sg	35 sg
Peor caso	15 sg	21 sg	35 sg

Tiempos de ejecución con diferentes conjuntos de datos:

n	$\log_2 n$
1	0
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024	10
1,048,576	20
2,097,152	21

4.9 Memoria

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	10 MB	20 MB	5 MB

4.10 Análisis de los resultados

Tabla de valores durante la ejecución

Estructuras de autocompletado	LinkedList	Arrays	HashMap
Espacio en el Heap	60MB	175MB	384MB
Tiempo creación	1.16 - 1.34 s	0.82 - 1.1 s	2.23 - 2.6 s
Tiempo búsqueda ("a")	0.31 - 0.39 s	0.37 - 0.7 s	0.22 - 0.28 s
Tiempo búsqueda ("zyzzyvas")	0.088 ms	0.038 ms	0.06 ms
Búsqueda ("aerobacteriologically")	0.077 ms	0.041 ms	0.058 ms
Tiempo búsqueda todas las palabras	6.1 - 8.02 s	4.07 - 5.19 s	4.79 - 5.8 s

REFERENCIAS

Referenciar las fuentes usando el formato para referencias de la ACM. Léase en http://bit.ly/2pZnE5g Vean un ejemplo:

1.Adobe Acrobat Reader 7, Be sure that the references sections text is Ragged Right, Not Justified. http://www.adobe.com/products/acrobat/.

2. Fischer, G. and Nakakoji, K. Amplifying designers' creativity with domainoriented design environments. in Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-36

3.https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/running-time-of-binary-search