PyNEMO 3: SE PERDIERON TODOS.
Una aplicación con Algoritmos Búsqueda informada y no informada

Jefferson Peña Torres, Cristian Alexander Valencia Torres, Domenico Di Mambro Cortes
1425590, 1329454, 1038452
(jefferson.amado.pena, cristian.a.valencia, domenico.di)@correounivalle.edu.co
Escuela de Ingeniería de Sistemas y Computación (EISC)
Universidad del Valle
Santiago de Cali, Colombia

Abstract __PyNEMO 3 es una implementación que pone a prueba varios algoritmos de búsqueda aplicadas en IA, estos algoritmos son evaluados de acuerdo a términos de tiempo, espacio y costo de la solución obtenida. Alguno de estos algoritmos de estos búsqueda no informada, búsqueda por anchura (BFS), búsqueda por profundidad (DFS), búsqueda por costo uniforme (UCS) y algunos de búsqueda informada como Avara (Heuristic) y A estrella (A star), construyen un grafo a partir de un ambiente en el que debe encontrar en un orden específico tres de los personajes del films Buscando a Nemo.

Este informe describe como Nemo 3 permitió analizar estos algoritmos, comparando el tiempo de ejecución de todos sobre una misma entrada y la forma en que procede cada uno para hallar la solución. Este informe hace parte del curso de Inteligencia Artificial (IA) de la Universidad del valle descrito por el ingeniero Carlos Delgado.

1. INTRODUCCIÓN

Muchos de los problemas de ingeniería pueden ser relacionados o modelados con una búsqueda a través de grafos, la búsqueda es uno de estos y que ha sido ampliamente estudiado generando dos grupos de acuerdo a la forma en que hallan la solución.

Estos grupos son denominados búsqueda informada y no-informada, en los que se encuentran diferentes algoritmos. En la búsqueda no-informada Amplitud, Profundidad y Costo Uniforme y en la búsqueda informada Avara o heurística y A* (A estrella), que son discutidos y analizados en este informe.

El problema presentado en este informe es la pérdida de tres de los personajes de la película Buscando a Nemo ``Nemo, Marlin y Dori'' en un arrecife y para ayudar a resolver este problema se han implementado diferentes estrategias de búsqueda aplicando IA, a través de un nuevo personaje Robot, este es guiado por cada uno de los algoritmos y debe hallar los tres personajes en un orden específico. El arrecife se carga desde un archivo de texto plano y a partir de este el robot debe ``construir'' la ruta pasando por una serie de obstáculos y ayudas que este posee.

Por esta razón las pruebas consisten en un conjunto de archivos que moldean arrecifes de diferentes tamaños y en los que se ejecuta cada uno de las implementaciones de algoritmos y de acuerdo a las salidas se obtienen las rutas, el costo de la ruta y los tiempo de ejecución, que son comparados y finalmente permiten concluir que una ``buena'' heurística, en concordancia a lo visto en clase, puede conducir a un mejor resultado en cuanto a tiempo de ejecución, espacio en memoria y costo de la búsqueda.

Este informe primero describe cada uno de los algoritmos, la forma de implementación, en la segunda sección se indica el tamaño de algunas entradas de prueba, en la tercera sección el uso de la aplicación y la codificación, en la cuarta sección ejemplos de los resultados obtenidos y finalmente la discusión de los resultados y las conclusiones.

II. ALGORITMOS DE BÚSQUEDA

Los algoritmos de búsqueda realizan una exploración limitada por el espacio en los sistemas de cómputo, por esta razón un aspecto importante es la estructura de datos que modela la estrategia de búsqueda, aunque algunos ya poseen estructuras definidas, en esta sección se hace una descripción de la implementación para Nemo3 y algunas características que permiten que la ejecución sea un poco más rápida sin modificar el algoritmo.

Inicio

El escenario se encuentra en un archivo txt, para evitar la consulta y carga innecesaria de una matriz completa, se cargan 5 colecciones de coordenadas (Tiburones, Rocas, Tortugas,

Hombres con arpón y Metas), cada una me permite verificar una posición versus un elemento de las colección.

Generalidades para la implementación de los algoritmos y operadores:
Operadores:

↑	↓	←	→

operadores de movimientos posibles

Objeto Posición:
Coordenada: compuesta por las componente x, y.
Metas: arreglo que contiene las 3 banderas, una por cada meta que encuentre.
Padre (Posición): es la referencia al padre (posición anterior).

Objeto Cola o Pila:
Posiciones:

P1	P2	P3	P4	P5	P6	P7	P8	P9	P10

Coordenadas

(X1,Y1)	(X2,Y2)	(X3,Y3)	(X4,Y4)	(X5,Y5)	(X6,Y6)	(X7,Y7)	(X8,Y8)	(X9,Y9)	(X10,Y10)

La cola se implementa con 2 colecciones, la primera colección de posiciones se

utiliza para almacenar la información (estado de las metas, ruta de los padres, etc) y la segunda colección de coordenadas se utiliza para comparar con las 5 colecciones de elementos que se cargaron al principio. Esta implementación separada permite un menor costo de ejecución al comparar posiciones de un arreglo con otro arreglo en vez de referenciar objetos y sacar la información de sus atributos para hacer las comparaciones.

A BÚSQUEDA NO-INFORMADA

1. **Búsqueda por amplitud en Nemo 3:**

Resumen: utiliza una cola almacenar las coordenadas de la búsqueda que realiza, iniciando en la posición del robot y con un orden de operadores (Por ejemplo: Izquierda, Derecha, Arriba y Abajo), con la posibilidad de ir a cuatro coordenadas nuevas, como el arrecife cuenta con algunos obstáculos, algunas coordenadas no se podrán tomar.

Implementación

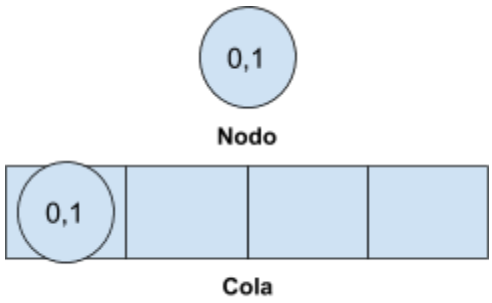


Gráfico 1 [Estado Inicial de cola]

En el gráfico 1 se puede observar que el algoritmos siempre la cola empieza con el nodo raíz(posición inicial robot). Este se verifica si es meta o no y luego se expanden sus hijos (nuevas posiciones donde puede ir el robot).

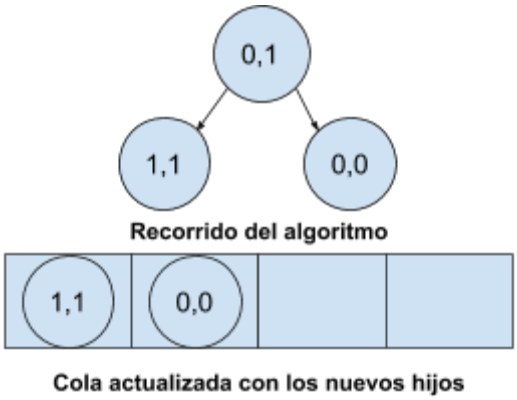


Gráfico 2 [Forma de expansión]

En el gráfico 2 se explica que el nodo que se expande se elimina de la cola y adiciona sus posibles posiciones siguientes y estos se insertan en la cola en el mismo orden de operadores.

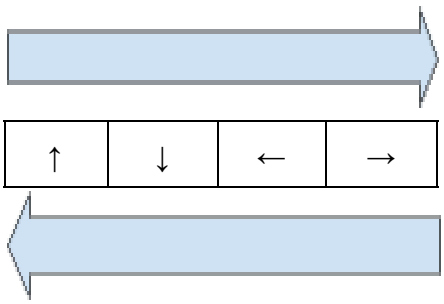
2. **Búsqueda por profundidad en Nemo 3:**

Resumen: Utiliza una pila para almacenar las coordenadas de la búsqueda que realiza, iniciando en la posición del robot y con un orden de operadores (Por ejemplo: Izquierda, Derecha, Arriba y Abajo), pero en este caso para la inserción en la pila el recorrido de los operadores se invierte debido a que la inserción es al principio los movimientos estarían invertidos (Abajo, Arriba, Derecha, Izquierda) .

Nota: *Se considero realizar la búsqueda con la implementación de NO DEVOLVERSE y Evitas Ciclos.*

Implementación

Operadores: Orden en que se deben aplicar los operadores.



Orden en el que se aplican los operadores para la inserción en la pila.

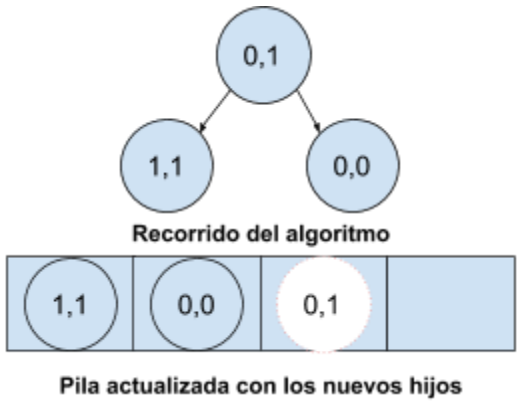


Gráfico 3 [Forma de expansión por profundidad]

En el gráfico 3 se puede visualizar que la inserción de los siguientes pasos en la pila se haría de una forma inversa

Nota: *Se considero realizar la búsqueda con la implementación de NO DEVOLVERSE y Evitas Ciclos.*

3. **Búsqueda con costo uniforme en Nemo 3:**

Resumen**:** Utiliza una cola de prioridad para almacenar las coordenadas de la búsqueda que realiza, iniciando en la posición del robot y con un orden de operadores (Por ejemplo: Izquierda, Derecha, Arriba y Abajo), la inserción es la de una cola normal, pero la forma en que se sacan las posiciones para realizar la búsqueda depende del costo (valor que debe pagar el robot por pasar por esa posición) que tenga cada una, es decir a menor costo mayor prioridad. Para esta implementación el objeto posición tiene un atributo llamado g (función de costo):

$$G(n) = G(n)propio \ + \ G(n)acumulado\ de\ los\ padres$$

Costos:

Objeto	Costo
Espacio Disponible	1
Tiburón	10
Hombre con Arpón	Infinito
Posición con ayuda de tortuga	(costo propio)*0.5

Tabla 1 Costos de los objetos del escenario.

La Tabla 1 muestra los costos que tiene cada uno de los objetos del escenario, para el Hombre con Arpón se utilizó un costo infinito para asegurar que el robot nunca pase por el.

Implementación

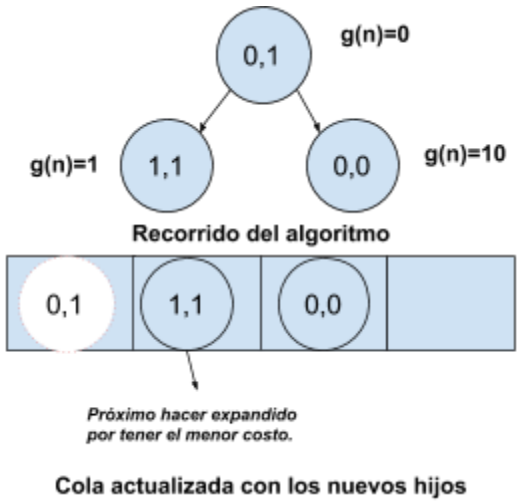


Gráfico 4 [Forma de expansión]

En el gráfico 4 se puede visualizar que la inserción de los siguientes pasos en la cola se realiza normal primero siempre al final, pero el que se atiende primero es el que tiene mayor prioridad, es decir menor costo.

Nota: *Se considero realizar la búsqueda con la implementación de NO DEVOLVERSE y Evitas Ciclos.*

B BÚSQUEDA INFORMADA

1. **Búsqueda heurística 1 en Nemo 3:**

$$H(n) = Distancia\ Manhattan * 0.5$$

La distancia manhattan es una de las heurísticas más conocidas e implementadas, pero para convertirla en una heurística mínima admisible, debe ser multiplicada por el menor costo que tiene el robot para realizar sus pasos, este costo es de 0.5 el cual se aplica a las siguiente 4 posiciones que se recorran después de que ha encontrado una tortuga, en el mejor de los casos el robot podría encontrarse en el camino con varias tortugas y llegar a la meta con la ayuda de las mísmas.

Ejemplo:

Para el mejor de los casos con el costo real G(n), si el robot encuentra espacios disponibles (con costo 1) sin ningún obstáculo y con ayuda de las tortugas en el camino el mínimo costo que tendría es este:

$$G(n) = (Número\ de\ posiciones\ recorrida - tortugas\ tomadas) * 0.5 + tortugas\ tomadas$$

Nota: las tortugas se cuentan como espacios disponibles.

$$Distancia\ Manhattan\ es\ aproximadamente\ el\ Número\ de\ posiciones\ recorrida$$

$$H(n) \leq G(n)$$

La distancia Manhattan es calculada en el algoritmo de la siguiente forma:

$$Posición = (x1, y1)$$
$$Meta = (x2, y2)$$
$$Distancia\ Manhattan = |\Delta x| + |\Delta y|$$

2. **Búsqueda heurística 2 en Nemo 3:**

$$H(n) = Distancia\ Euclidiana * 0.5$$

La distancia euclidiana o más conocida como distancia entre puntos, pero para convertirla en una heurística mínima admisible, debe ser multiplicada por el menor costo que tiene el robot para realizar sus pasos, este costo es de 0.5 el cual se aplica a las siguiente 4 posiciones que se recorran después de que ha encontrado una tortuga, en el mejor de los casos el robot podría encontrarse en el camino con varias tortugas y llegar a la meta con la ayuda de las mísmas.

Ejemplo:

Para una distancia en línea recta, el robot en la posición (0,0) y una meta en la posición (0,5), con una tortuga en la posición (0,1) tenemos:

$$Distancia\ Euclidiana = \sqrt{(0-0)^2 + (0-5)^2} = 5$$
$$H(n) = 5 * 0.5 = 2$$

$G(n) = (Pasos - número\ de\ tortugas\ tomadas) * 0.5 + número\ de\ tortugas\ tomadas = (4-1) * 0.5 + 1 = 2.5$

Se cumple que $H(n) \leq G(n)$

$G(n) = (Número\ de\ posiciones\ recorrida - tortugas\ tomadas) * 0.5 + tortugas\ tomadas$

Nota: las tortugas se cuentan como espacios disponibles.

La distancia Euclidiana es calculada en el algoritmo de la siguiente forma:

$Posición = (x1, y1)$

$Meta = (x2, y2)$

$Distancia\ Euclidiana = \sqrt{(x2-x1)^2 + (y2-y1)^2}$

Implementación :

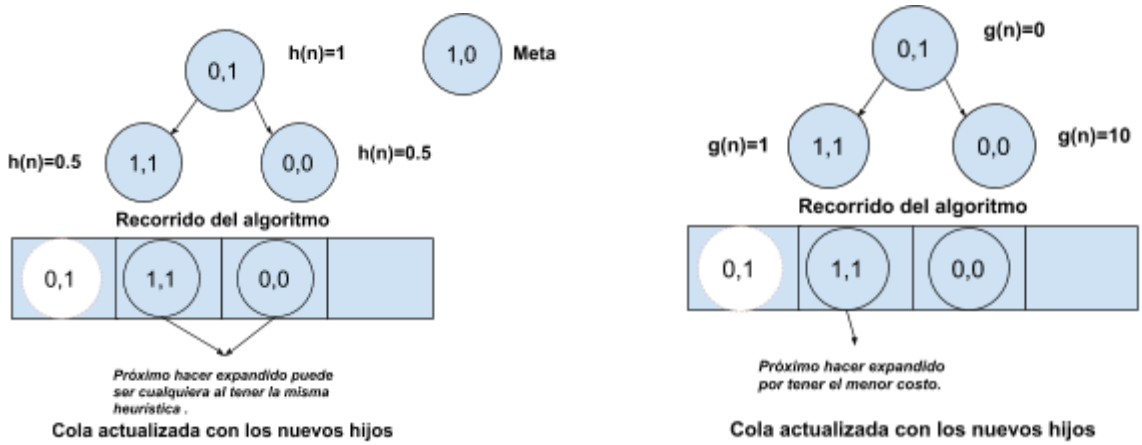


Gráfico 5 [Avara y Costo Uniforme separados]

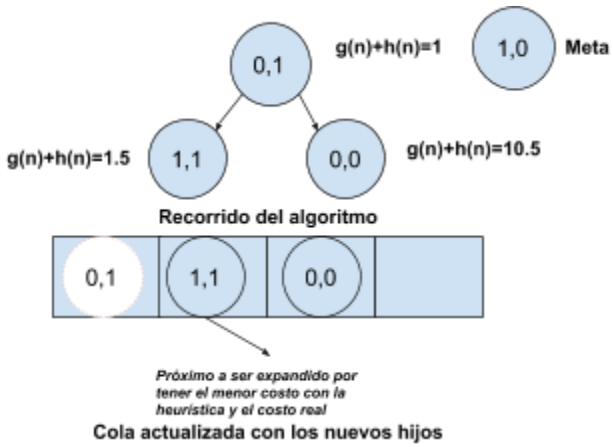


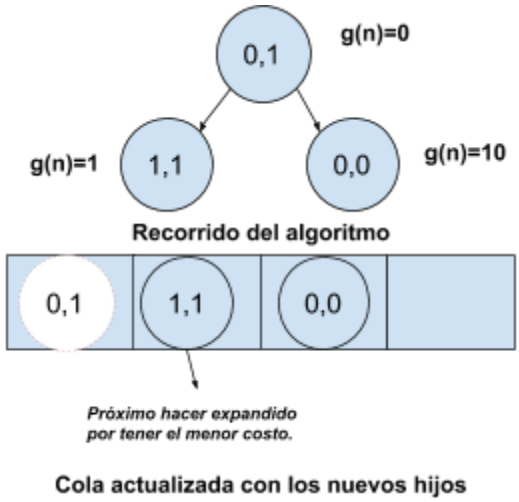
Gráfico 5 [Avara y Costo Uniforme separados]

En el gráfico 5 se puede visualizar que la inserción de los siguientes pasos en la cola se realiza normal primero siempre al final, pero el que se atiende primero es el que tiene mayor prioridad, es decir menor heurística

Nota: *Se considero realizar la búsqueda con la implementación de NO DEVOLVERSE y Evitas Ciclos.*

3. **Búsqueda con a estrella en Nemo 3:**

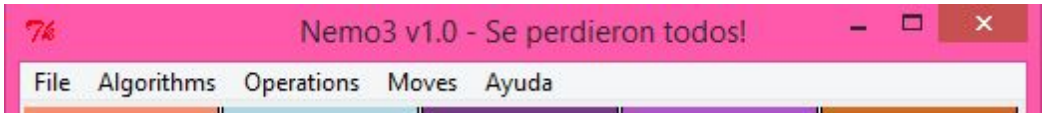
Resumen: este algoritmo tiene el mismo funcionamiento de Avara y Costo uniforme, la diferencia es que esté implementa a ambos es decir, que implementa una heurística y el costo real, al sumar ambos obtiene un valor más óptimo para realizar la búsqueda, por eso es que este algoritmo es uno de los mejores



Ejecución de un ejemplo de entrada y análisis de los resultados con los diferentes algoritmos:

- ➔ Ejecutamos en la carpeta src
- ➔ python nemo.py
- ➔ Elegimos el menú bar file y elegimos el archivo a usar como matriz de juego en este caso in_5x5_star.txt.

Mostrara un menu bar nuevo con todas las posibles características que se pueden realizar:



En algoritmos existen todos los posibles algoritmos que se pueden realizar y puestos en el proyecto
En operaciones todas las posibles permutaciones de órdenes de todos los movimientos (Arriba,Abajo,Derecha,Izquierda)

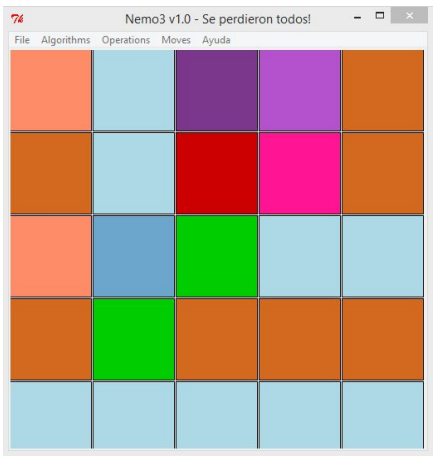
En operaciones:

Iniciar: tomando en cuenta el algoritmo y movimientos previamente elegidos, se empezará a visualizar en la matriz como procede a expandir los posibles movimientos.

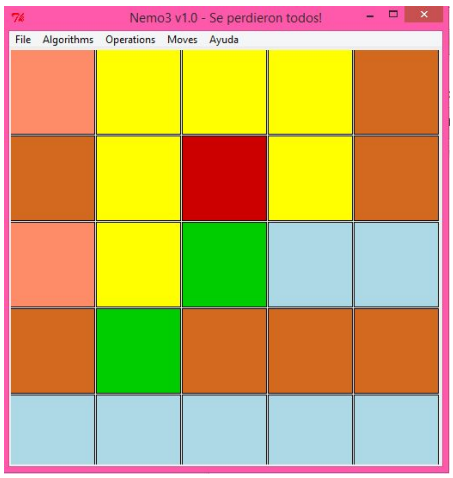
- Detener: si se quiere detener el algoritmo y reiniciar la matriz
- Lento: para que el algoritmo corra más lento
- Normal: para que el algoritmo corra según la potencia del procesador
- Rápido: para que el algoritmo aumente en su velocidad
- Archivo:
- Salvar: se guarda el archivo todo los datos pedidos en el proyecto y las secuencia de pasos que tuvo para llegar a la meta.

Ejemplo:

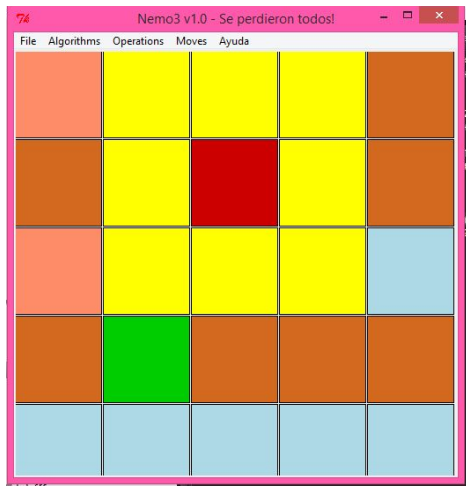
Matriz 5x5



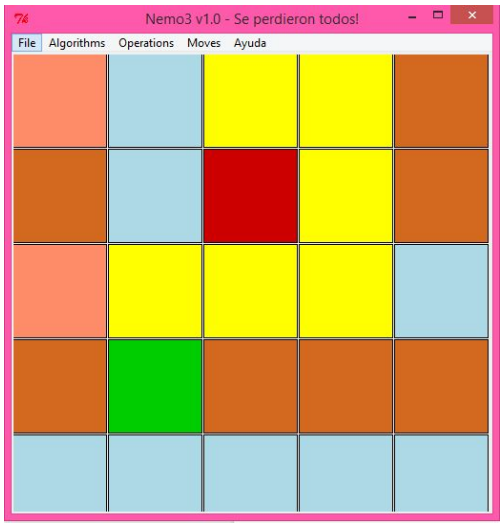
.



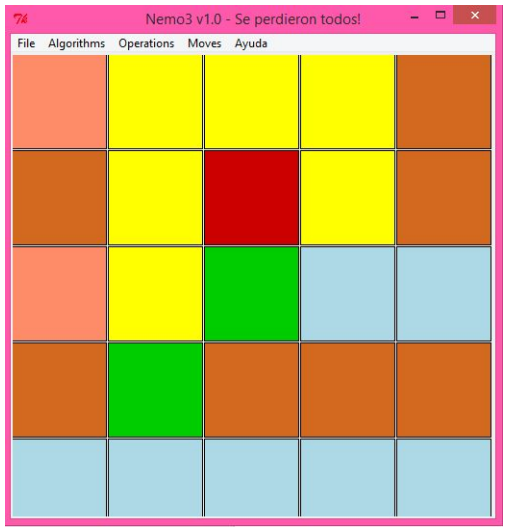
Amplitud



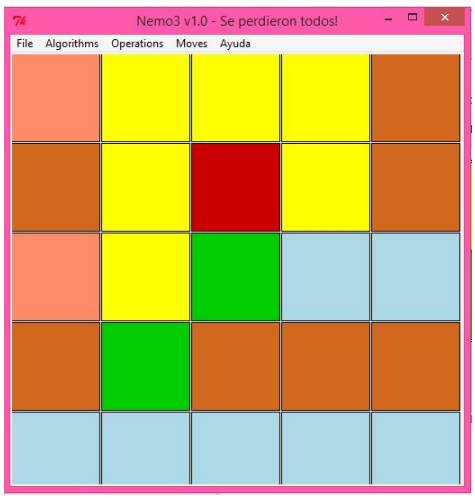
Profundidad



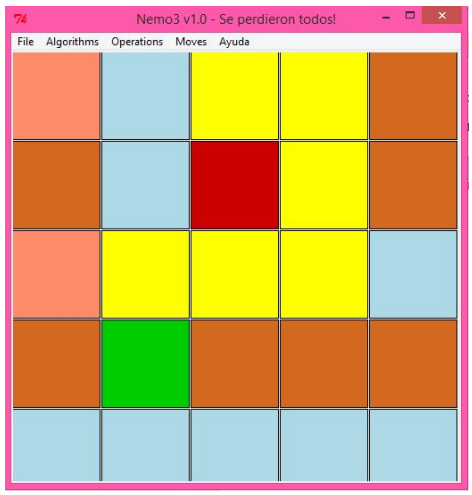
Costo



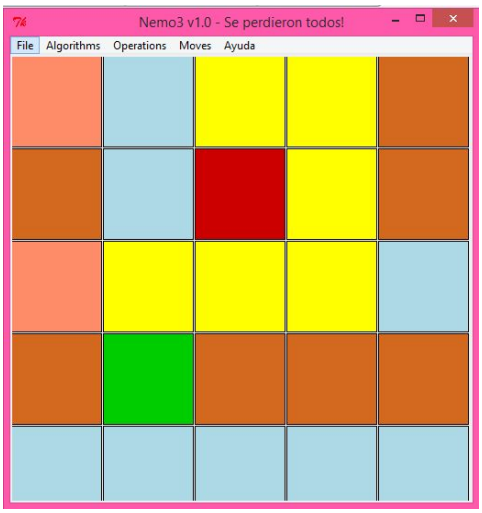
Avara 1



Avara 2



Star 1



Star 2

Nota: cada archivo se encuentra en el output con sus respectivos datos y tiempos de ejecución.

Para mostrar un análisis más detallado se procedió a sacar los datos para comparar los algoritmos

Algoritmos	tiempos (s)	número pasos	costo total	nodos expandidos	nodos creados
Amplitud	2.738	5	5	21	42
Profundidad	2.655	11	11	20	27
Costo Uniforme	3.079	7	4	22	37
Avara 1	1.319	5	5	9	22
Avara 2	0.937	5	5	6	14
A* 1	1.898	7	4	14	27
A* 2	2.053	7	4	15	29

Gráfica 10 [Datos de la ejecución de cada algoritmo]

Conclusiones y análisis

Los algoritmos de búsqueda informada gracias a la heurística, dan mejores resultados en cuanto al tiempo de ejecución para encontrar la respuesta, pero en el caso de creación y expansión de nodos, al tener un ejemplo con costos tan parejos, las heurística permite abrir muchos nodos con costos parecidos pero con mayor promesa de ser solución. Por el contrario los algoritmos de búsqueda no informada si presentan tiempos más altos con creación y expansión mayores, ya que estos implementan un solución voraz de un camino según un orden de operadores.