Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey

Implementación de algoritmo A*
Rompecabezas 4x4

Sistemas Inteligentes

José Francisco Murillo Lozano
A01374561
Roberto Emmanuel González Muñoz
A01376803

17/08/2022

Contenido

Introducción

Utilizando el algoritmo A* se implementó un programa de búsqueda informada en Python capaz de resolver el problema de rompecabezas 4x4. El tablero de entrada o estado inicial del rompecabezas, además del estado meta son proporcionados en un archivo “Datos.txt”.

En el cual:

- ➔ Las primeras 4 líneas articulan el estado inicial
- ➔ Las últimas 4 líneas describen el estado final
- ➔ Todos los elementos se encuentran separados por comas
- ➔ El último elemento de cada línea incluye un salto de línea
- ➔ El espacio correspondiente al rompecabezas se representa mediante un cero

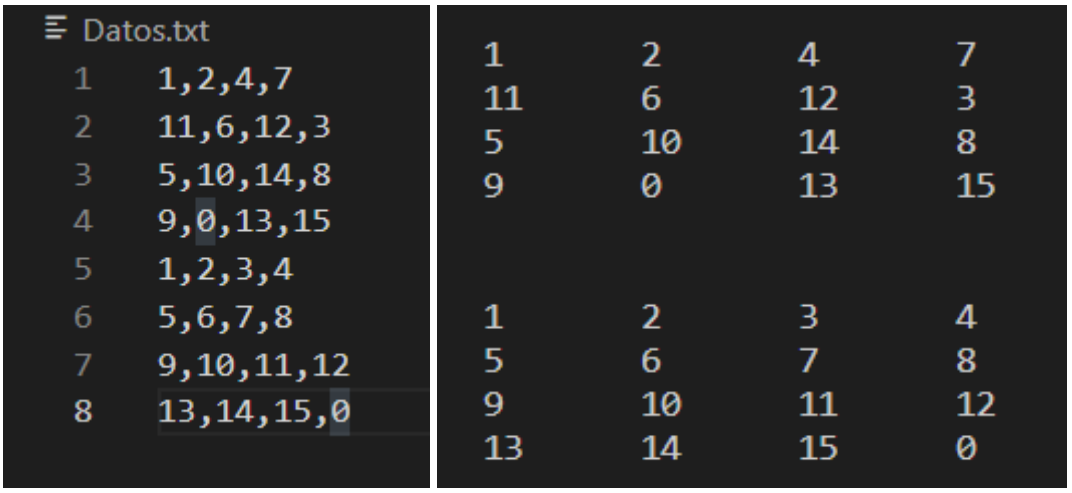

Figura 1. **Contenido del archivo “Datos.txt” e impresión en la terminal de la representación de la matriz 4x4**

Por otro lado, el sistema después procesar el estado inicial y final regresa la secuencia de movimientos que debe realizar el espacio correspondiente al rompecabezas (cero) como una cadena separadas por comas, siendo:

Código	**Descripción**
U	Arriba (Up)
D	Abajo (Down)
L	Izquierda (Left)
R	Derecha (Right)

El resultado de la implementación se muestra siempre en la consola, por ejemplo:

```
Solution Path: [<NodeMoves.RIGHT: 4>, <NodeMoves.UP: 1>, <NodeMoves.UP: 1>, <NodeMoves.RIGHT: 4>, <NodeMoves.UP: 1>, <NodeMoves.LEFT: 3>, <NodeMoves.DOWN: 2>,
 <NodeMoves.LEFT: 3>, <NodeMoves.LEFT: 3>, <NodeMoves.DOWN: 2>, <NodeMoves.RIGHT: 4>, <NodeMoves.UP: 1>, <NodeMoves.RIGHT: 4>, <NodeMoves.RIGHT: 4>, <NodeMove
s.DOWN: 2>, <NodeMoves.LEFT: 3>, <NodeMoves.LEFT: 3>, <NodeMoves.LEFT: 3>, <NodeMoves.DOWN: 2>, <NodeMoves.RIGHT: 4>, <NodeMoves.RIGHT: 4>, <NodeMoves.RIGHT:
4>]
Solution Path: ['E0', 'E2', 'E6', 'E10', 'E35', 'E39', 'E151', 'E153', 'E154', 'E760', 'E1043', 'E1044', 'E8232', 'E8365', 'E8367', 'E8371', 'E8372', 'E8374',
 'E8377', 'E8381', 'E8382', 'E8383', 'E8385']
```

Figura 2. **Resultado de la búsqueda informada generada a partir del algoritmo A*, describe los pasos a seguir para llegar al estado inicial, al igual que los estados intermedios.**

Función de evaluación

Utilizando el algoritmo A* se implementaron dos funciones de evaluación una simple donde se usa un recuento de saltos y celdas mal colocadas y otra haciendo uso de la métrica de evaluación Manhattan con el objetivo de mejorar los resultados producidos por el sistema.

Simple

Esta función de evaluación considera la suma de los costos desde la raíz hasta n como la suma acumulada de los arcos o movimientos desde el estado final hasta el estado evaluado n. Por otro lado, también estima el costo mínimo desde n, hasta una solución como el número de celdas mal colocadas en el estado evaluado.

$$f(n) = g(n) + h(n)$$
$$g(n) = 1\ por\ arco\ recorrido$$
$$h(n) = número\ de\ celdas\ mal\ colocadas$$

Si se evalúa un estado actual inmediatamente después del estado inicial el resultado de esta forma de evaluación se describe en la siguiente figura.

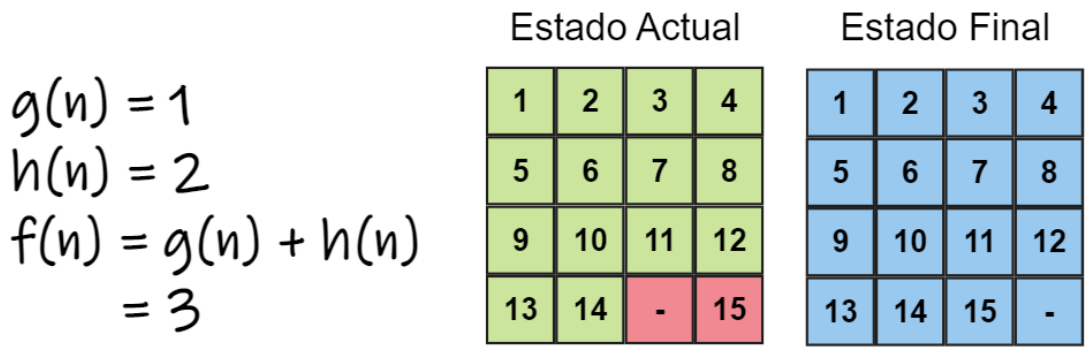

Figura 3. **Representación gráfica de la evaluación simple de la función g(n) , h(n) y f(n).**

El objetivo de esta función es minimizar el número de arcos y celdas mal colocadas, por lo tanto, entre más lejos se encuentre el estado actual del estado inicial mayor será h(n).

Distancia Manhattan

Esta función de evaluación considera la suma de los costos desde la raíz hasta n como la suma acumulada de los arcos o movimientos desde el estado final hasta el estado evaluado n. Por otro lado, también estima el costo mínimo desde n, hasta una solución como la distancia Manhattan más pequeña del estado evaluado hasta el estado final. Para lograr esto se identifica la distancia Manhattan más pequeña de entre todos los números.

$$f(n) = g(n) + h(n)$$
$$g(n) = 1\ por\ arco\ recorrido$$
$$h(n) = distancia\ manhattan\ más\ pequeña$$

Si se evalúa un estado actual inmediatamente después del estado inicial el resultado de esta forma de evaluación se describe en la siguiente figura. En este caso el espacio y 15 tienen la misma distancia Manhattan por lo cual el valor de h(n) corresponde al número 15 (1). Si 2 o más números tienen la misma distancia Manhattan se selecciona uno de ellos de forma aleatoria.

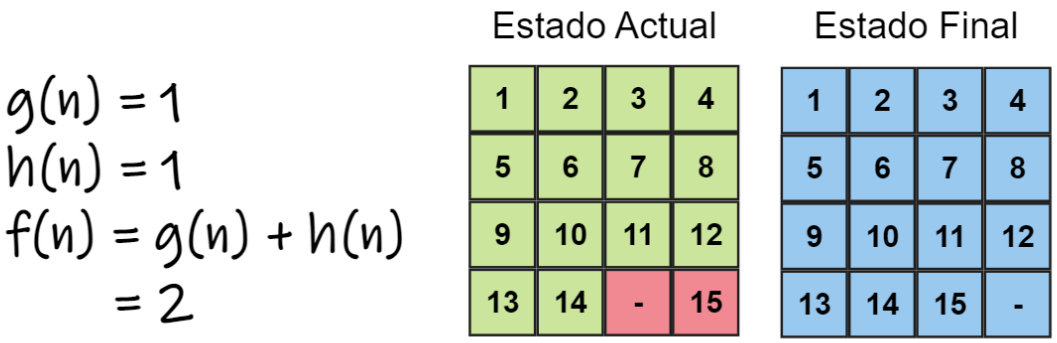

Figura 4. **Representación gráfica de la evaluación utilizando la distancia Manhattan de la función g(n) , h(n) y f(n).**

El objetivo de esta función es minimizar el número de arcos y celdas lejanas, por lo tanto, entre más lejos se encuentre el estado actual en términos de desplazamientos lineales del estado inicial mayor será h(n). Esta función es buena porque considera los movimientos con menor distancia al estado final con mayor prioridad.

Algoritmo

El algoritmo A* se hace una búsqueda seleccionando primero a los mejores candidatos, para ello hace uso de una función de evaluación. La función de evaluación se divide en dos; h(n) y g(n). La suma de los costos desde la raíz hasta n se representa como g(n), mientras que la estimación de costo mínimo desde n hasta una solución se describe como h(n).

$$f(n) = g(n) + h(n)$$
$$g(n) = costo\ desde\ la\ raíz\ hasta\ n$$

$$h(n) = \ costo\ mínimo\ desde\ n\ hasta\ solución$$

Una de las ventajas más importantes de este algoritmo es que llega a una solución sin expandir completamente el árbol de candidatos. En el caso particular de un rompecabezas 4x4, implementar este algoritmo nos permite escoger los movimientos más prometedores mediante la función de evaluación f(n).

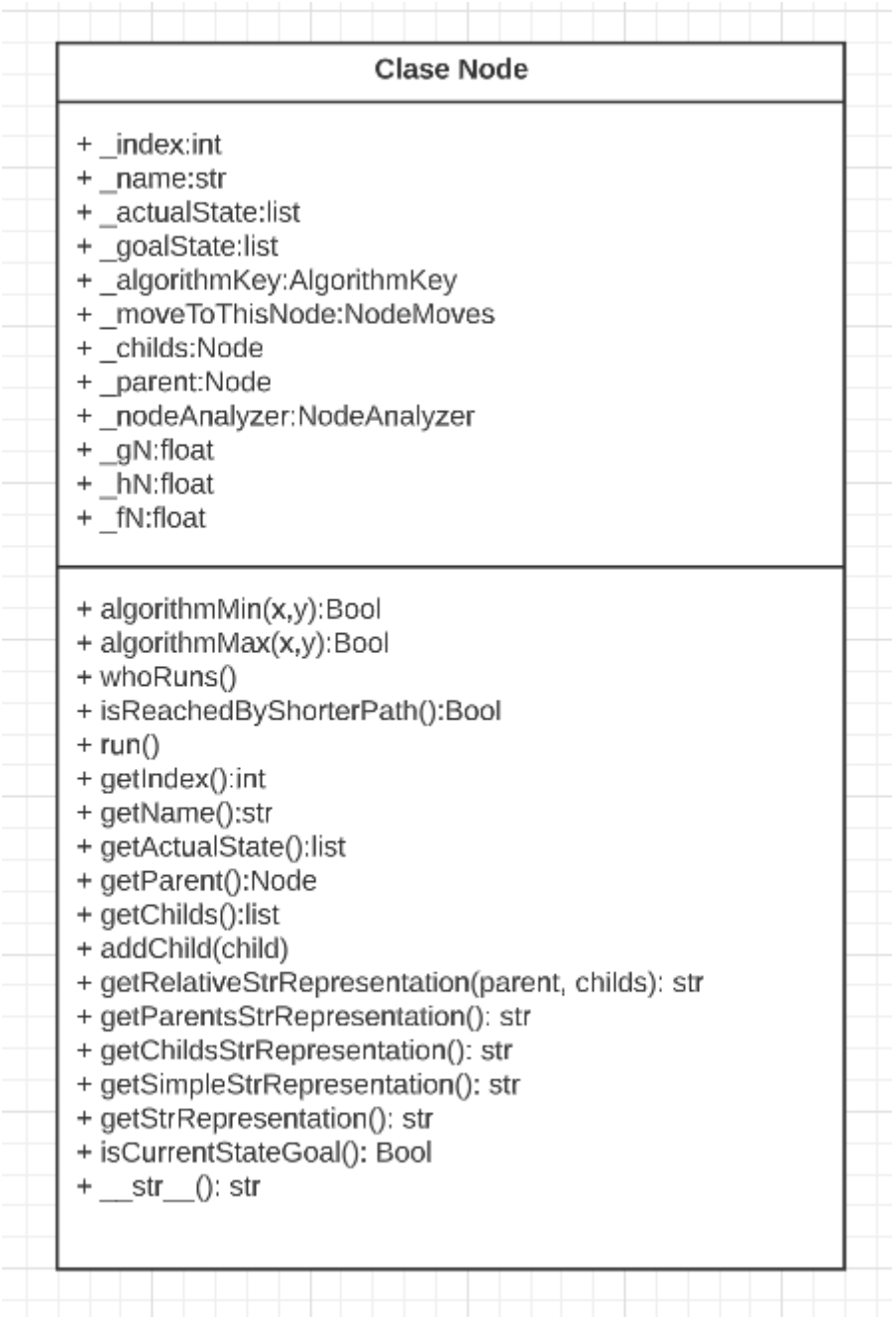

Figura 5. Diagrama UML de la clase Node utilizada para implementar el algoritmo A*

El resultado de este algoritmo nos permite encontrar la mejor serie de pasos (mejor f(n)) que son necesarios para llegar de un estado inicial a un estado final. Con eso en mente, se diseñó un programa compuesto por cinco clases que proveen la funcionalidad necesaria para ejecutar el algoritmo A*.

Análisis de tiempos de ejecución

Iteración	Tiempo de ejecución
1	3.58s
2	0.10s
3	1.99s
4	0.25s
5	0.67s

Estadísticas	
Promedio [s]	1.318
Media [s]	0.67
Moda [s]	0.1, 0.25, 0.67, 1.99, 3.58
Rango [s]	3.48
Media geométrica	0.6536574010998233
Raíz de Mínimo cuadrado	1.8599946236481437
Valor mínimo [s]	0.1
Valor máximo [s]	3.58
Total de valores	5
Suma de valores	6.59
Varianza de la población	1.722456
Varianza de la muestra	2.15307

Desviación Estándar de la población	1.3124237121
Desviación Estándar de la muestra	1.4673343177

Análisis de la función propuesta

Pensamos en utilizar como g(n) la misma función que la original dado que teníamos la idea de darle un costo al movimiento si es que la casilla que se movía llegaba más cerca a su dirección pero reflexionando encontramos que el costo para moverse debería ser igual para ayudarle al algoritmo a converger fácilmente.

Sin embargo, en cuanto a h(n) tenemos la propuesta de hacer un cálculo de diferencia entre las posiciones actuales de los números y las posiciones meta.

El considerar este nuevo h(n) debería hacer converger al algoritmo en menos pasos aunque con una mayor cantidad de comparaciones de por medio.