Documentación Tercera Tarea Programada

Danny Chaves Chaves, dxnnx@me.com, Minor Sancho Valverde,
tivin.minor10@gmail.com, Oscar Chavarria Campos, ocach14@hotmail.com

October 25, 2016

1 Introducción

Los algoritmos de optimizacion son un área activa de investigación en la inteligencia articial. Tales algoritmos buscan una solución óptima de un problema, muchas veces bajo un conjunto de restricciones y usualmente buscan minimizar el costo computacional de realizar tal búsqueda. Ejemplos de enfoques de optimizacion son los algoritmos basados en el gradiente, algoritmos genéticos, algoritmos simplex, etc. Uno de los enfoques más usados desde la Inteligencia artificial es el de árboles de búsqueda, y entre los algoritmos populares en este enfoque se puede mencionar el algoritmo A*, tambien llamado A estrella. Los algoritmos de búsqueda definen un problema como un conjunto de variables, cuyos valores óptimos o resueltos es necesario encontrar, y muchas veces también, la secuencia de pasos para llegar a tal estado.

2 Análisis del Problema

El problema del rompecabezas deslizante es una variante de problemas particulares, usados comúnmente para estudiar distintos algoritmos de resolución de problemas en la inteligencia artificial. Consiste en un tablero de M = N x N fichas o entradas, con cada ficha etiquetada con un número n = 0, 1, 2, . . . , M. En todo el tablero la etiqueta de la ficha no se puede repetir. Este se puede representar como una matriz de con M columnas y N filas, donde cada espacio es un numero del 0 hasta N x M - 1 ,y para lograr ordenar la matriz solo se pueden hacer 4 tipos de movimientos, hacia arriba,abajo,izquierda,derecha, y la única ficha que se podrán intercambiar es el cero con cualquier otro numero, ya que el cero representa un hoyo donde la ficha se podrá mover y dejar el lugar vació. Su solución se da cuando se logran ordenar todos estos numero dejando el cero como ultimo elemento en la matriz.

3 Diseño de la solución

Para solucionar dicha matriz se utilizo el algoritmo A estrella, el cual se base en desarrollar un árbol de búsqueda, donde a diferencia de otros algotimos de búsqueda este utiliza un heuristica para decidir que ramificación desarrollar. Para la implementacion de este algoritmo se usaron 3 heuristicas, las cuales son: Heuristica de Manhattan, Jaro, Euclides. Donde calcula la huristica de cada posible movimiento para verificar el apto.

3.1 Heuristica de Manhattan

Es utiliza la distancia de Manhattan la cual, conceptualiza la distancia entre dos puntos, y se encarga de calcular el recorrido mas corto para conectar estos dos puntos. La sumatoria de la distancia de Manhattan para todas las fichas en el tablero, donde si se toma un punto como posicion incial y otro como el estado meta puede llegar a formular la heurística HM o aproximación de cuántos movimientos son necesarios para llegar al estado meta.

3.2 Heuristica de Jaro

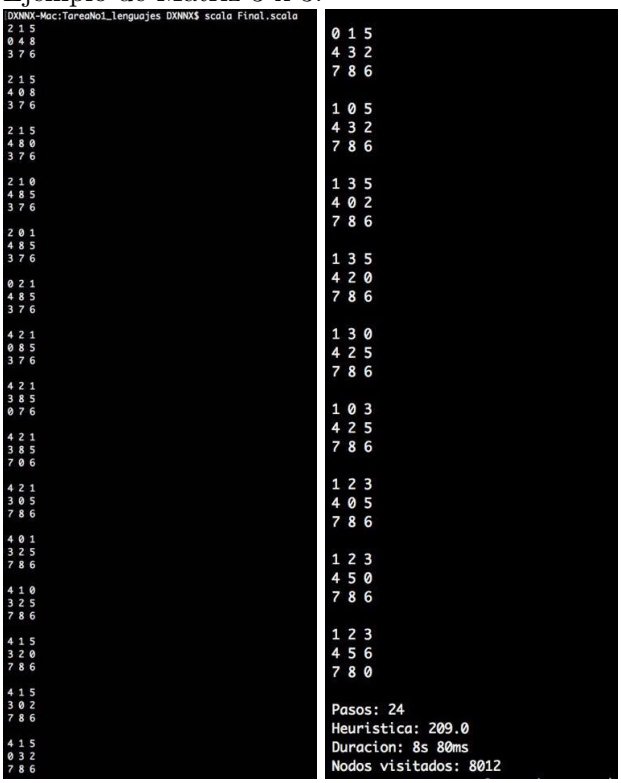
La distancia Jaro-Winkler mide la similitud entre dos cadenas,derivada de la distancia Jaro que se utiliza principalmente en la detección de duplicados. Más distancia Jaro-Winkler entre dos cadenas, la más alta que son similares. Esto es particularmente adecuada para el tratamiento de cadenas cortas tales como nombres o contraseñas. El resultado es normalizado a fin de tener una medida de entre 0 y 1, con el cero representa la ausencia de similitud.

3.3 Heuristica de Euclides

La distancia euclidiana o euclídea es la distancia "ordinaria" entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.

4 Pruebas

Ejemplo de Matriz 3 x 3:



4.1 Parametros

Para poder solucionar el problema, el algorimo siempre recibira 4 parametros los cuales son:

- Dimension de la matriz,ya que el algoritmo implementa la solucion para matrices de mayor a 3 x 3.
- Heuristica, donde se puede elegir entre 3 tipos de heuristicas, 0 para Manhattan, 1 para Jaro y 2 para Euclides.
- Matriz inicial, puede ser cualquier matriz. En caso que se desee una matriz meta, se deja una matriz vacia
- Matriz Meta Personalizada, ya que el usuario puede personalizar a la matriz meta que se quiere llegar.

En la siguiente figura se muestra un ejemplo de los parametros.

```
Pasos: 25
Heuristica: 232.0
Duracion: 1s 455ms
Nodos visitados: 3245
DXNNX-Mac:TareaNo1_lenguajes DXNNX$ scala Final.scala 3 1 "1 6 8 5 0 7 3 2 4"
```

4.2 Heuristica de Manhattan

Heuristica de Manhattan:

```
DXNNX-Mac:TareaNo1_lenguajes DXNNX$ scala Final.scala 3 2 "" "1 2 3 7 8 0 4 5 6"
4 0 6
5 2 7
3 1 8

4 2 6
5 0 7
3 1 8

4 2 6
5 7 0
3 1 8

4 2 0
5 7 6
3 1 8

4 0 2
5 7 6
3 1 8

4 7 2
5 0 6
3 1 8

4 7 2
5 1 6
3 0 8

4 7 2
5 1 6
0 3 8

4 7 2
0 1 6
5 3 8

0 7 2
4 1 6
5 3 8

7 0 2
4 1 6
5 3 8

7 1 2
4 0 6
5 3 8

7 1 2
4 3 6
5 0 8

7 1 2
4 3 6
5 8 0

7 1 2
4 3 0
5 8 6
```

4.3 Heuristica de Jaro

Heuristica de Jaro:

```
Pasos: 25
Heuristica: 232.0
Duracion: 1s 455ms
Nodos visitados: 3245
DXNNX-Mac:TareaNo1_lenguajes DXNNX$ scala Final.scala 3 1 "1 6 8 5 0 7 3 2 4"
```

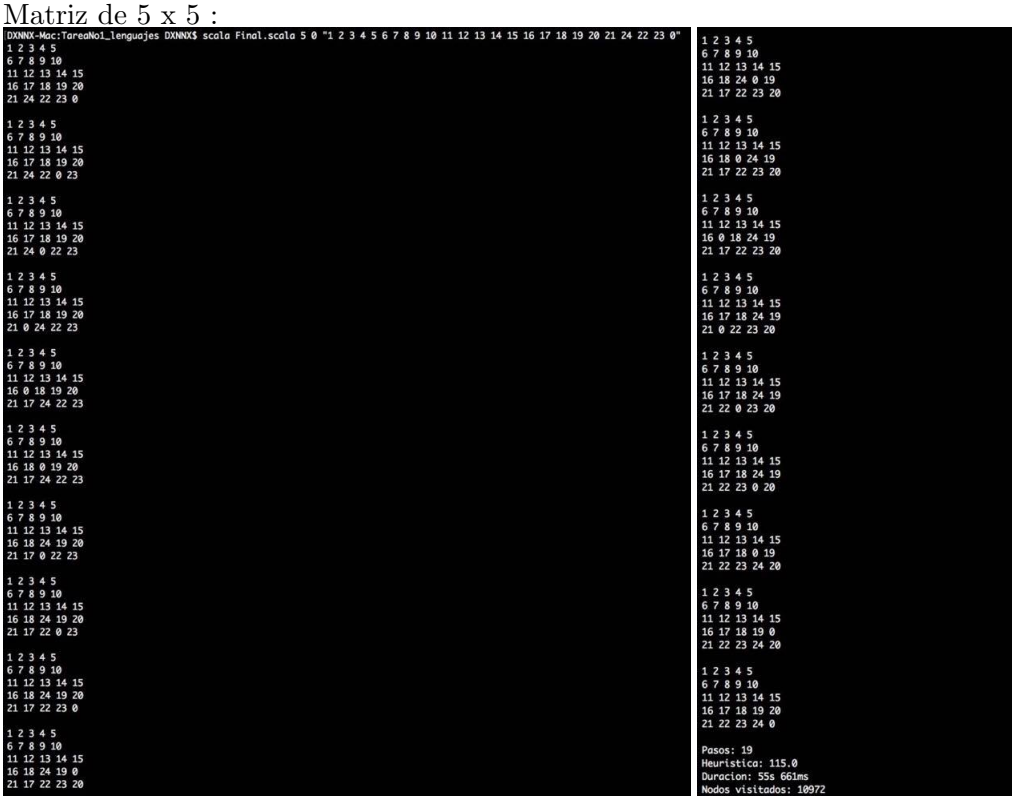
4.4 Heuristica de Euclides

Heuristica de Euclides:

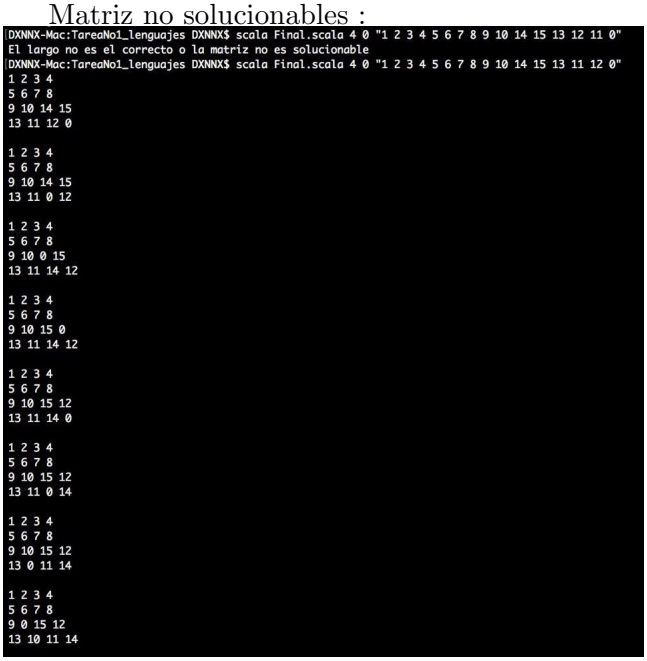
```
Pasos: 25
Heuristica: 196.34563013492053
Duracion: 2s 968ms
Nodos visitados: 4540
DXNNX-Mac:TareaNo1_lenguajes DXNNX$ scala Final.scala 3 0 "1 6 8 5 0 7 3 2 4"
```

4.5 Funcionalidades Extras

Matriz de 5 x 5 :



Matriz no solucionables :



5 Conclusión

Durante la realización de este proyecto logramos apreciar el gran potencial de Scala, el cual brinda muchas ventajas con herramientas funcionales como lo es su paradigma, que otros lenguajes de programación no brindan. Entre las mayores dificultades que encontramos, el desarrollo del árbol de búsqueda, ya que manejamos muy poco este lenguaje y tiene muy poca información en internet para su desarrollo, se puede encontrar algunos sitios que dan algunos ejemplo, pero no son claros. En cuanto sus características especificas, se tuvo que aprender como implementar tanto el algoritmo estrella y sus heuristicas, ya que el programa tenia que contar con partes funcionales. Con esta tarea logramos ampliar nuestros conocimientos en un lenguaje tan poderoso como scala pero tan poco conocido y utilizable a comparación con otros lenguajes.