|  |
| --- |
| Asignatura: |
| Ingeniería del Conocimiento |

|  |
| --- |
| Título del documento: |
| **Laboratorio 1:**  **Práctica de búsqueda** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fecha |
| Grupo: |  | 21 |  | 12/02/18 |
|  |  |  |  |  |
| Miembros: | 1- | Gonzalo de las Heras |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre de fichero: |  | Fecha: |  |  |  |  |
| LAB01-INCO-GRUPO21-MEMORIA.docx **(.pdf)** |  | **30/10/18** |  |  |  |  |

Índice

[1 Solución lógica 4](#_Toc528608151)

[1.1 Espacio de Estados 4](#_Toc528608152)

[1.1.1 Estado 4](#_Toc528608153)

[1.1.2 Operadores 4](#_Toc528608154)

[1.1.3 Coste g(n) 4](#_Toc528608155)

[1.2 Mapa 4](#_Toc528608156)

[1.3 Algoritmos de búsqueda 4](#_Toc528608157)

[1.4 Implementación en Python 7](#_Toc528608158)

[1.4.1 Estado 7](#_Toc528608159)

[1.4.2 Operadores 7](#_Toc528608160)

[1.4.3 Algoritmos 7](#_Toc528608161)

[1.4.4 Heurística 7](#_Toc528608162)

[1.5 Parámetros de ejecución de la búsqueda 8](#_Toc528608163)

[1.6 Bancos de prueba y aclaraciones 8](#_Toc528608164)

[1.6.1 Ficheros de prueba 8](#_Toc528608165)

[1.6.2 Accesibilidades 8](#_Toc528608166)

[2 Resolución de los ejercicios y discusión 9](#_Toc528608167)

[2.1 Ejercicio 1 9](#_Toc528608168)

[2.1.1 Distancia Manhattan 9](#_Toc528608169)

[2.1.2 Distancia Chebyshov 10](#_Toc528608170)

[2.1.3 Distancia Octil 11](#_Toc528608171)

[2.1.4 Comparativa 11](#_Toc528608172)

[2.1.4.1 Admisibilidad 12](#_Toc528608173)

[2.1.4.2 Consistencia 13](#_Toc528608174)

[2.1.4.3 Dominancia 14](#_Toc528608175)

[2.1.4.4 Comparativa 15](#_Toc528608176)

[2.2 Ejercicio 2 15](#_Toc528608177)

[2.2.1 Apartado A 15](#_Toc528608178)

[2.2.2 Apartado B 16](#_Toc528608179)

[2.3 Ejercicio 3 17](#_Toc528608180)

[2.3.1 Apartado A 17](#_Toc528608181)

[2.3.2 Apartado B 18](#_Toc528608182)

[2.4 Ejercicio 4 19](#_Toc528608183)

[2.4.1 Apartado A 19](#_Toc528608184)

[2.4.2 Apartado B 20](#_Toc528608185)

[2.4.3 Apartado C 20](#_Toc528608186)

[2.5 Reflexiones finales 20](#_Toc528608187)

[3 Bibliografía 24](#_Toc528608188)

Índice de ilustraciones

[Ilustración 1. Mapa BootyBay. 6](#_Toc528608189)

[Ilustración 2. Heurística del enunciado. 9](#_Toc528608190)

[Ilustración 3. Cuadro ejemplo distancia Manhattan. 9](#_Toc528608191)

[Ilustración 4. Heurística distancia Manhattan. 10](#_Toc528608192)

[Ilustración 5. Cuadro ejemplo distancia Chebyshov. 10](#_Toc528608193)

[Ilustración 6. Heurística distancia Chebyshov. 10](#_Toc528608194)

[Ilustración 7. Cuadro ejemplo distancia Octil. 11](#_Toc528608195)

[Ilustración 8. Heurística distancia Octil. 11](#_Toc528608196)

[Ilustración 9. Cuadro ejemplo admisibilidad de la distancia Manhattan. 12](#_Toc528608197)

[Ilustración 10. Cuadro ejemplo admisibilidad de la distancia Chebyshov. 12](#_Toc528608198)

[Ilustración 11. Cuadro ejemplo admisibilidad de la distancia Octil. 13](#_Toc528608199)

[Ilustración 12. Cuadro ejemplo consistencia de la distancia Manhattan. 13](#_Toc528608200)

[Ilustración 13. Cuadro ejemplo consistencia de la distancia Chebyshov. 14](#_Toc528608201)

[Ilustración 14. Cuadro ejemplo consistencia de la distancia Octil. 14](#_Toc528608202)

[Ilustración 15. Gráfica costes ejecución distancia Octil. 16](#_Toc528608203)

[Ilustración 16. Gráfica nodos expandidos ejecución distancia Octil. 17](#_Toc528608204)

[Ilustración 17. Gráfica costes ejecución distancia Chebyshov. 18](#_Toc528608205)

[Ilustración 18. Gráfica nodos expandidos ejecución distancia Chebyshov. 19](#_Toc528608206)

[Ilustración 19. Resultado ejecución Ejercicio 4. 20](#_Toc528608207)

[Ilustración 20. Ejemplo del coste total sin penalizar caminos en diagonal. 21](#_Toc528608208)

[Ilustración 21. Ejemplo del coste total sin penalizar caminos en diagonal (camino alternativo). 21](#_Toc528608209)

[Ilustración 22. Ejemplo del coste total penalizando caminos en diagonal. 21](#_Toc528608210)

[Ilustración 23. Ejemplo del coste total penalizando caminos en diagonal (camino alternativo). 22](#_Toc528608211)

[Ilustración 24. Resultado A\* sin penalizar el camino en diagonal. 22](#_Toc528608212)

[Ilustración 25. Resultado A\* penalizando el camino en diagonal. 22](#_Toc528608213)

[Ilustración 26. Comparativa penalizando y sin penalizar el camino diagonal. 23](#_Toc528608214)

Índice de tablas

[Tabla 1. Operadores 5](#_Toc528608215)

[Tabla 2. Comparativa de las distintas heurísticas. 15](#_Toc528608216)

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

# Solución lógica

## Espacio de Estados

### Estado

Un estado será representado por un par de coordenadas (X, Y) que indican la posición dentro de la matriz del mapa.

* El estado inicial será la posición desde la cual se parte.
* El estado final será la posición a la cual se tiene que llegar.

### Operadores

Existen 8 operadores, uno para moverse en cada dirección horizontal, vertical y diagonal.

Una celda es transitable si cumple los criterios de accesibilidad. Por ejemplo, los muros no son celdas transitables y no se puede acceder al agua desde el terreno. No obstante, estas condiciones dependen del enunciado.

En la tabla 1 se detallan los distintos operadores.

### Coste g(n)

El coste de transitar una celda será 1, salvo en el último ejercicio que dependerá del tipo de celda que sea (terreno, agua, pantano, árbol).

## Mapa

El mapa viene representado por una matriz en el que cada celda indica el coste de tránsito y si es transitable o no (si no lo es se indica con un -1). El cumplimiento de los criterios de accesibilidad se evalúa aparte.

## Algoritmos de búsqueda

Los algoritmos de búsqueda a implementar son A\* y Avara.

Para poder utilizar el mismo código para todos los enunciados, cada algoritmo de búsqueda recibe por parámetro: el mapa, puntos de inicio y fin, costes de las celdas, si los árboles son transitables, D2 y si se penaliza en el coste el moverse diagonalmente.

|  |  |  |
| --- | --- | --- |
| **Operador** | **Precondición** | **Acción** |
| Ir Arriba | Existe una celda superior transitable, dentro de los límites del mapa. | Transición y coste += Coste de la celda. |
| Ir Abajo | Hay una celda inferior transitable (y dentro de los límites del mapa). | Transición y coste += Coste de la celda. |
| Ir Izquierda | Hay una celda a la izquierda transitable (y dentro de los límites del mapa). | Transición y coste += Coste de la celda. |
| Ir Derecha | Hay una celda a la derecha transitable (y dentro de los límites del mapa). | Transición y coste += Coste de la celda. |
| Ir Diagonal Superior Izquierda | Hay una celda transitable en la diagonal superior izquierda. | Transición y coste += Coste de la celda. |
| Ir Diagonal Superior Derecha | Hay una celda transitable en la diagonal superior derecha. | Transición y coste += Coste de la celda. |
| Ir Diagonal Inferior Izquierda | Hay una celda transitable en la diagonal inferior izquierda. | Transición y coste += Coste de la celda. |
| Ir Diagonal Inferior Derecha | Hay una celda transitable en la diagonal inferior derecha. | Transición y coste += Coste de la celda |

Tabla 1. Operadores

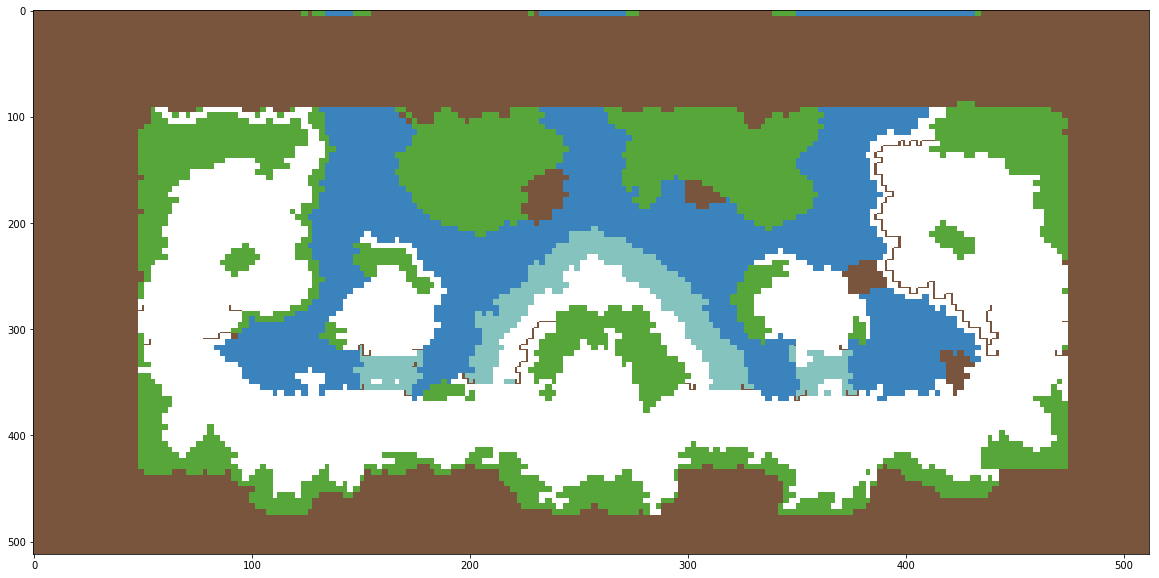


Ilustración 1. Mapa BootyBay.

## Implementación en Python

Para mejorar la organización del código, se ha optado por un diseño modular basado en objetos, los cuales son:

* Nodo: contiene la abstracción de un estado del Estado de Estados.
* Mapa: contiene la funcionalidad del programa principal.

Estos objetos se complementan con las funciones para la búsqueda A\* y Avara, además del cálculo de la heurística.

### Estado

Un estado consistirá en un objeto de la clase Nodo con los siguientes atributos:

* Posición: una tupla (X, Y) que indica la posición dentro de la matriz que representa el mapa. Este atributo identifica únicamente el estado dentro del Espacio de Estados. No puede haber dentro de las listas de ABIERTO o CERRADO dos objetos Nodo que tengan la misma posición pues habría 2 estados iguales.
* F: Suma de H y G.
* G: Coste acumulado para llegar a ese estado desde el inicial.
* H: valor de la heurística.
* Padre: Referencia al nodo padre. Esto sirve para cuando le halle la solución, poder trazar el camino recorrido.

### Operadores

Los operadores serán almacenados en una lista en el que cada operador es una tupla que indica la coordenada en la que se moverá (arriba, abajo, izquierda, derecha y diagonales):

Movimientos = [(-1, 0), (1, 0), (0, -1), (0, 1), (-1, -1), (1, 1), (1, -1), (-1, 1)]

### Algoritmos

Para una mejor organización y evitar repetir el mismo código, se ha separado del flujo principal del programa la implementación de A\* y Avara. Las listas de ABIERTO y CERRADO contienen objetos de la clase Nodo.

### Heurística

Por el mismo motivo que los algoritmos de búsqueda, se ha separado la implementación de la función para el cálculo de la heurística del flujo principal del programa. También, para poder emplear la heurística con los distintos valores de D y D2, además de los distintos costes de las celdas, se ha refactorizado la función para recibir estos valores como parámetros de la función.

## Parámetros de ejecución de la búsqueda

Las búsquedas se pueden realizar con ciertas modificaciones, las cuales se indican mediante el constructor del programa principal:

Los parámetros de la ejecución son:

* El valor de D2 en la función heurística.
* Los costes de los distintos tipos de celdas.
* Si la celda de tipo árbol es transitable.
* Si la celda de tipo agua es accesible desde el terreno.
* Si se penaliza el coste tras realizar un movimiento en diagonal.

## Bancos de prueba y aclaraciones

### Ficheros de prueba

Debido a las restricciones de accesibilidad indicadas en el enunciado y a algunas de las búsquedas propuestas en el fichero proporcionado, el cálculo del recorrido óptimo es muy costoso en tiempo o el recorrido es demasiado simple como para poder comparar correctamente los algoritmos.

Por ello, se han creado nuevos bancos de pruebas:

* BancoDePruebas.map.scen: Para los ejercicios 2,3 y 4.
* ReflexionesFinales.map.scen: Para las reflexiones finales.

### Accesibilidades

Según el enunciado:

* Pantano: transitable y accesible desde terreno.
* Agua: transitable si estás dentro, pero no accesible desde terreno.

Suposiciones:

* Pantano: accesible desde terreno y desde el agua.
* Terreno: accesible desde pantano.
* Agua: accesible desde solo el pantano.

De esta forma aclaramos que para ir del agua al terreno, o viceversa, hay que pasar por el pantano.

# Resolución de los ejercicios y discusión

## Ejercicio 1

Partimos de:

|  |
| --- |
| def heuristica(celda):  dx = abs(celda.x - goal.x)  dy = abs(celda.y - goal.y)  return D\*(dx+dy) + (D2-2\*D) \* min(dx,dy) |

Ilustración 2. Heurística del enunciado.

La cual es una generalización de varias fórmulas heurísticas, las cuales analizamos a continuación.

Para los ejemplos, asumiremos que el coste de tránsito de todas las celdas es siempre 1.

### Distancia Manhattan

Derivando la fórmula anterior podemos obtener la distancia Manhattan, que es la suma de la distancia al objetivo en cuanto al número filas y columnas (dx + dy).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 6 | 7-Fin |
| 2 |  |  |  |  |
| 1 |  |  |  |  |
| Inicio |  |  |  |  |

Ilustración 3. Cuadro ejemplo distancia Manhattan.

En este caso la distancia Manhattan es 3 + 4 = 7.

Para obtenerla se tiene que cumplir que:

* D = 1.
* (D2-2\*D) \* min(dx,dy) = 0, para lo cual:
  + (D2-2\*D) = 0 -> D2 = 2\*D -> Si D = 1 -> D2 = 2.

De este modo:

-> ~~1 \*~~ (dx+dy) + (2-2\*1) \* min(dx,dy)

-> dx+dy + ~~0 \* min(dx,dy)~~

|  |
| --- |
| def heuristica(celda):  dx = abs(celda.x - goal.x)  dy = abs(celda.y - goal.y)  return dx + dy |

Ilustración 4. Heurística distancia Manhattan.

### Distancia Chebyshov

Derivando la fórmula anterior podemos obtener la distancia Chebyshov que representa el número de movimientos que necesitaría la pieza del rey en el ajedrez para llegar a la casilla destino.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | 3 | 3 | 4-Fin |
| 2 | 2 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 4 |
| Inicio | 1 | 2 | 3 | 4 |

Ilustración 5. Cuadro ejemplo distancia Chebyshov.

Operando con:

* D = 1.
* D2 = 1.

-> ~~1 \*~~ (dx+dy) + (1-2~~\*1~~) \* min(dx,dy)

-> (dx + dy) – min(dx, dy)

|  |
| --- |
| def heuristica(celda):  dx = abs(celda.x – goal.x)  dy = abs(celda.y – goal.y)  return (dx + dy) – min(dx, dy) |

Ilustración 6. Heurística distancia Chebyshov.

### Distancia Octil

Derivando la fórmula anterior podemos obtener la distancia Octil, que una extensión de la distancia Manhattan para admitir movimientos en diagonal. Para ello, el coste de tránsito en diagonal es el valor de la diagonal del cuadrado con lado = al coste indicado. Como en este caso el coste es 1, la diagonal es √2.

Siguiendo el ejemplo anterior donde la distancia Manhattan es: 3 + 4 = 7.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | 3\*√2 | 3\*√2+1 |
|  |  | 2\*√2 |  |  |
|  | √2 |  |  |  |
| Inicio |  |  |  |  |

Ilustración 7. Cuadro ejemplo distancia Octil.

La distancia Octil es 7 + (√2 – 2) \* 3 = 5,2426, que es igual a √2 \* 3 + 1 (la diagonal de cada casilla recorrida + la distancia de una casilla)

Operando:

* D = 1.
* D2 = √2.

-> ~~1 \*~~ (dx+dy) + (√2 - 2~~\*1~~) \* min(dx,dy)

-> (dx+dy) + (√2 - 2) \* min(dx,dy)

|  |
| --- |
| def heuristica(celda):  dx = abs(celda.x - goal.x)  dy = abs(celda.y - goal.y)  return (dx+dy) + (√2 - 2) \* min(dx,dy) |

Ilustración 8. Heurística distancia Octil.

### Comparativa

A continuación, comparamos las anteriores heurísticas en cuanto a sus propiedades de admisibilidad, consistencia y dominancia.

Tendremos en cuenta que todas las celdas transitables tienen un coste de 1, que se pueden hacer movimientos en horizontal, vertical y diagonal (los 8 operadores) y seguiremos el ejemplo de las ilustraciones anteriores.

#### Admisibilidad

Para cumplir el criterio de admisibilidad: h\*(n) ≥ h(n) ∀n.

* Distancia Manhattan:

No es admisible pues la heurística calcula 7 y el coste real es 4, ya no se cumple que el coste real sea mayor o igual a las heurísticas de todos los nodos. Por lo tanto, A\* no encontraría el óptimo, el algoritmo A\* sería solo el algoritmo A.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 5 | 6 / 3 | H=7 / G=4 -Fin |
| 2 |  | 2 |  |  |
| 1 | 1 |  |  |  |
| Inicio |  |  |  |  |

Ilustración 9. Cuadro ejemplo admisibilidad de la distancia Manhattan.

* Distancia Chebyshov:

Es admisible para todo n, la heurística es igual que el coste real. La distancia Chebyshov es la mínima que necesita la pieza del rey para llegar a una casilla, es decir, la distancia óptima.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 3 | 3 | 3 | 4-Fin |
| 2 | 2 | 2 | 3 | 4 |
| 1 | 1 | 2 | 3 | 4 |
| Inicio | 1 | 2 | 3 | 4 |

Ilustración 10. Cuadro ejemplo admisibilidad de la distancia Chebyshov.

* Distancia Octil:

No es admisible porque la heurística calcula 5,2426 y el coste real es 4. En este caso, A\* también sería A. Si el coste de moverse en diagonal fuera √2 y en horizontal o vertical 1, sí sería admisible.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | 3\*√2 | 3\*√2+1 |
|  |  | 2\*√2 |  |  |
|  | √2 |  |  |  |
| Inicio |  |  |  |  |

Ilustración 11. Cuadro ejemplo admisibilidad de la distancia Octil.

#### Consistencia

Para cumplir el principio de consistencia: h(padre) ≤ h(hijo) + k (padre, hijo) ∀n.

* Distancia Manhattan:

No es consistente, la heurística en el inicio es 7, si nos movemos en diagonal la heurística es 5 por lo que 7 ≤ 5 + 1 no es cierto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fin |
|  |  |  |  |  |
|  | H = 5 |  |  |  |
| H = 7 |  |  |  |  |

Ilustración 12. Cuadro ejemplo consistencia de la distancia Manhattan.

* Distancia Chebyshov:

Sí es consistente, se cumple que h(padre) ≤ h(hijo) + k (padre, hijo) ∀n.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fin |
|  |  |  |  |  |
| H = 4 | H = 3 |  |  |  |
| H = 4 | H = 3 |  |  |  |

Ilustración 13. Cuadro ejemplo consistencia de la distancia Chebyshov.

* Distancia Octil:

No es consistente, la heurística en el inicio es 5,2426, si nos movemos en diagonal la heurística es 3,8284 por lo que 5,2426 ≤ 3,8284 + 1 no es cierto.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | Fin |
|  |  |  |  |  |
| H = 4,2426 | H = 3,8284 |  |  |  |
| H = 5,2426 | H = 4,2426 |  |  |  |

Ilustración 14. Cuadro ejemplo consistencia de la distancia Octil.

#### Dominancia

Una heurística es dominante si: hi(n) ≥ h(n) ∀h ∀n.

Si una heurística es dominante sobre otra, significa que expande menos nodos que sobre la que domina.

En nuestro caso el orden de dominancia es:

1. Manhattan.
2. Octil.
3. Chebyshov.

#### Comparativa

|  |  |  |  |
| --- | --- | --- | --- |
| **Distancia** | **Admisible** | **Consistente** | **Dominante** |
| Manhattan | No | No | 1º |
| Chebyshov | Sí | Sí | 3º |
| Octil | No | No | 2º |

Tabla 2. Comparativa de las distintas heurísticas.

## Ejercicio 2

La heurística empleada es la distancia Octil.

### Apartado A

Para las condiciones de nuestro problema (coste siempre 1 y 8 operadores) no es admisible, ni consistente y en orden de las 3 heurísticas estudiadas (Manhattan, Chebyshov y Octil) ocupa la 2º posición. Eso significa que expandirá menos nodos que Chebyshov pero más que Manhattan.

La demostración se encuentra en el Ejercicio 1.

### Apartado B

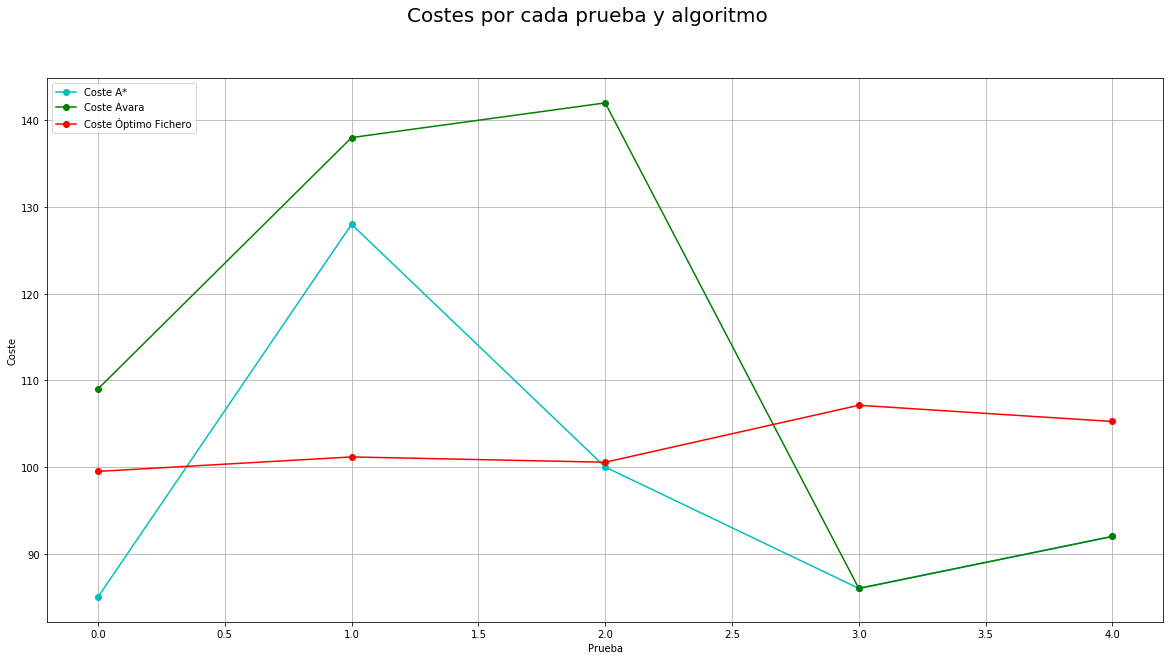


Ilustración 15. Gráfica costes ejecución distancia Octil.

* Costes A\*:
  + [85, 128, 100, 86, 92]
* Costes Avara:
  + [109, 138, 142, 86, 92]
* Costes óptimos fichero:
  + [99.49747467, 101.16652222, 100.55634918, 107.12489166, 105.25483398]

Tras la ejecución del programa encontramos que los costes calculados no siempre mejoran el coste del fichero. No podemos emitir un juicio acerca del motivo porque no sabemos bajo que condiciones exactas se calcularon los valores “óptimos” que figuran en el fichero.

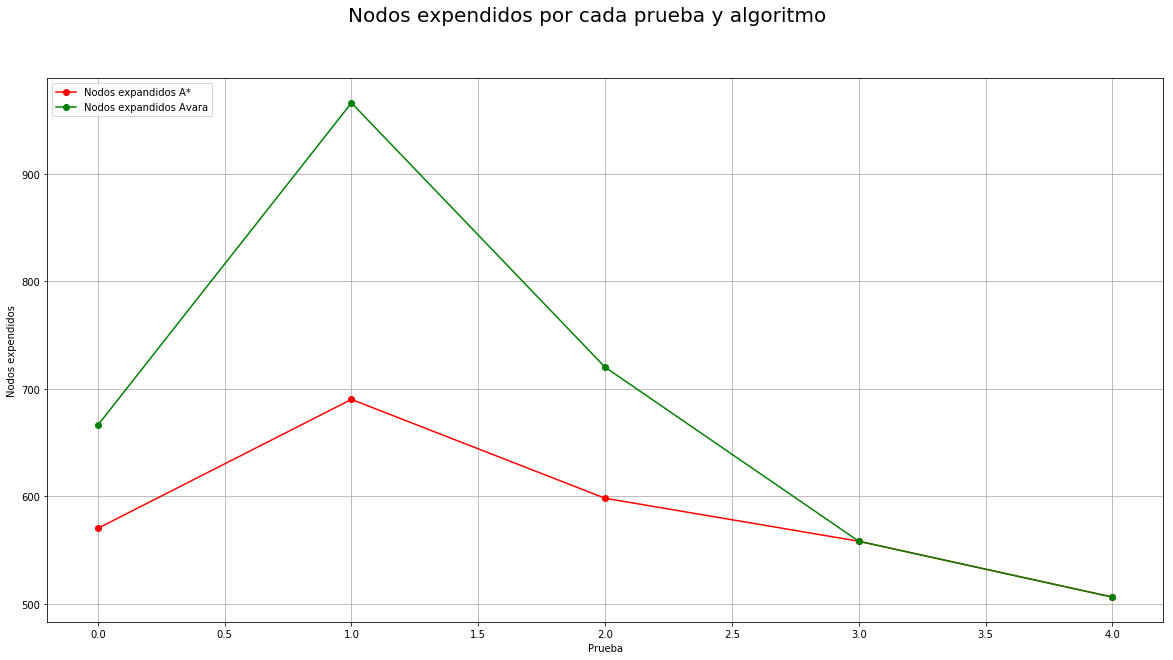


Ilustración 16. Gráfica nodos expandidos ejecución distancia Octil.

* Nodos expandidos A\*:
  + [570, 690, 598, 558, 506]
* Nodos expandidos Avara:
  + [666, 966, 720, 558, 506]

Observamos que A\* encuentra mejores rutas que Avara y, además, expandiendo igual o menos nodos.

## Ejercicio 3

### Apartado A

La función heurística utilizada es la distancia Chebyshov la cual es admisible y consistente.

La demostración se encuentra en el Ejercicio 1.

### Apartado B

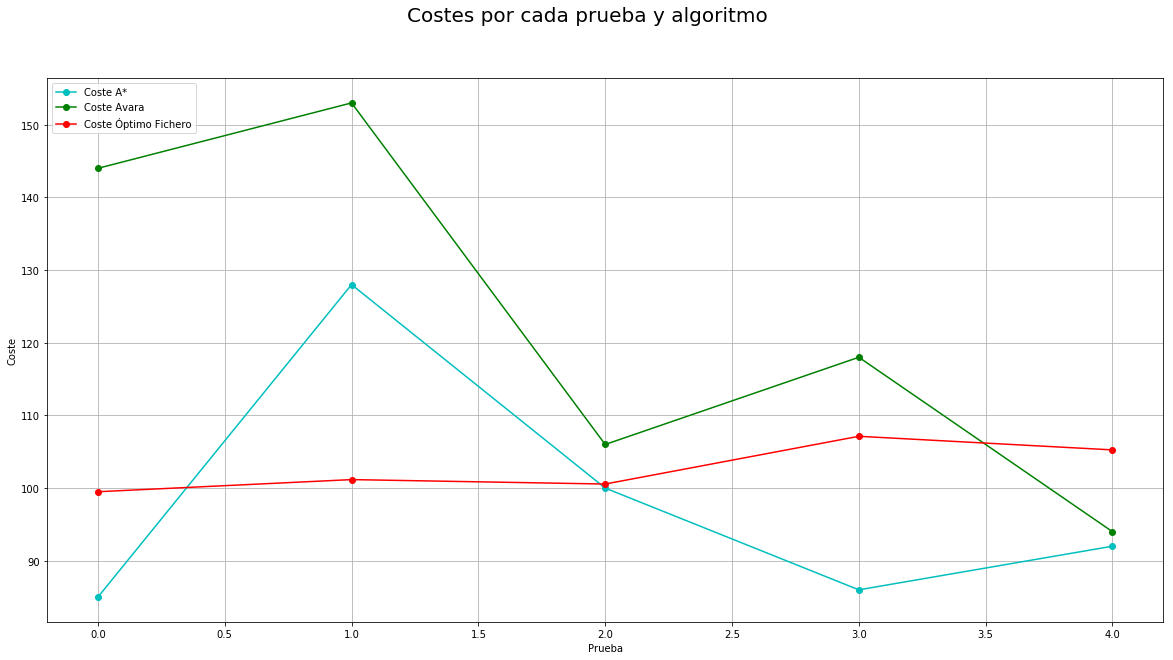


Ilustración 17. Gráfica costes ejecución distancia Chebyshov.

* Costes A\*:
  + [85, 128, 100, 86, 92]
* Costes Avara:
  + [144, 153, 106, 118, 94]
* Costes óptimos fichero:
  + [99.49747467, 101.16652222, 100.55634918, 107.12489166, 105.25483398]

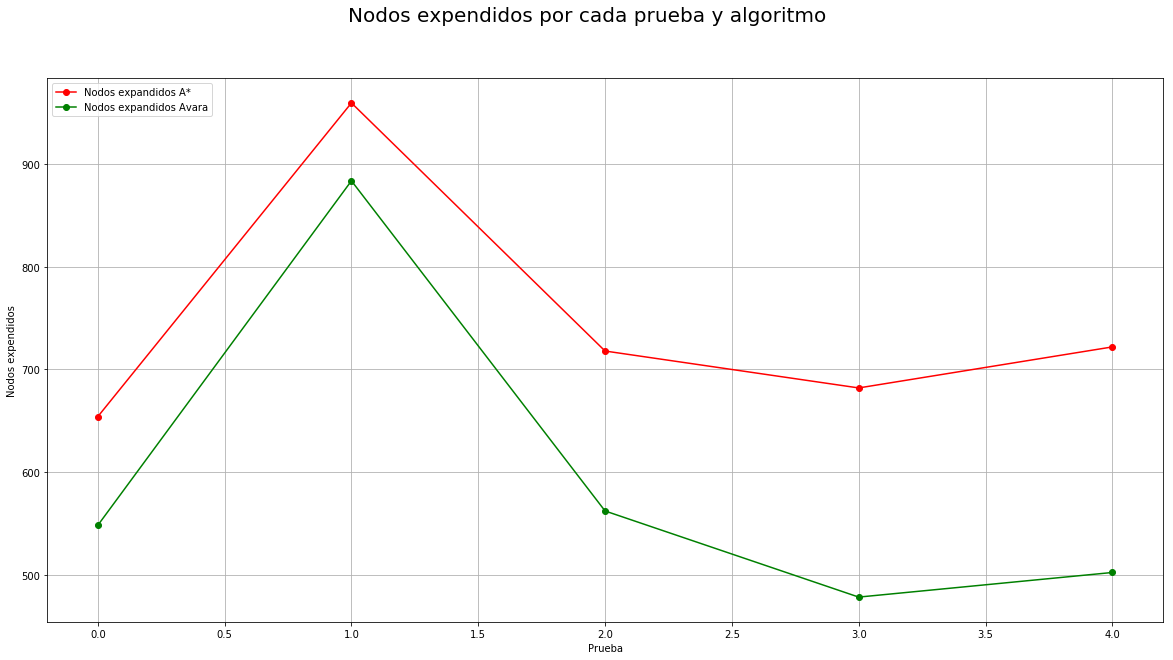


Ilustración 18. Gráfica nodos expandidos ejecución distancia Chebyshov.

* Nodos expandidos A\*:
  + [654, 960, 718, 682, 722]
* Nodos expandidos Avara:
  + [548, 884, 562, 478, 502]

Observamos que:

* El número de nodos expandidos con la heurística distancia Chebyshov es mayor que con distancia Octil debido a que Octil es dominante sobre Chebyshov.
* Los caminos calculados por A\* son del mismo coste.
* Con la distancia Octil A\* expande menos nodos que Avara, sin embargo, con la distancia Chebyshov Avara expande menos nodos que A\*.

## Ejercicio 4

### Apartado A

La función heurística utilizada sigue siendo la distancia Chebyshov, pero ponderando D y D2 según el coste de transitar por el tipo de celda.

### Apartado B

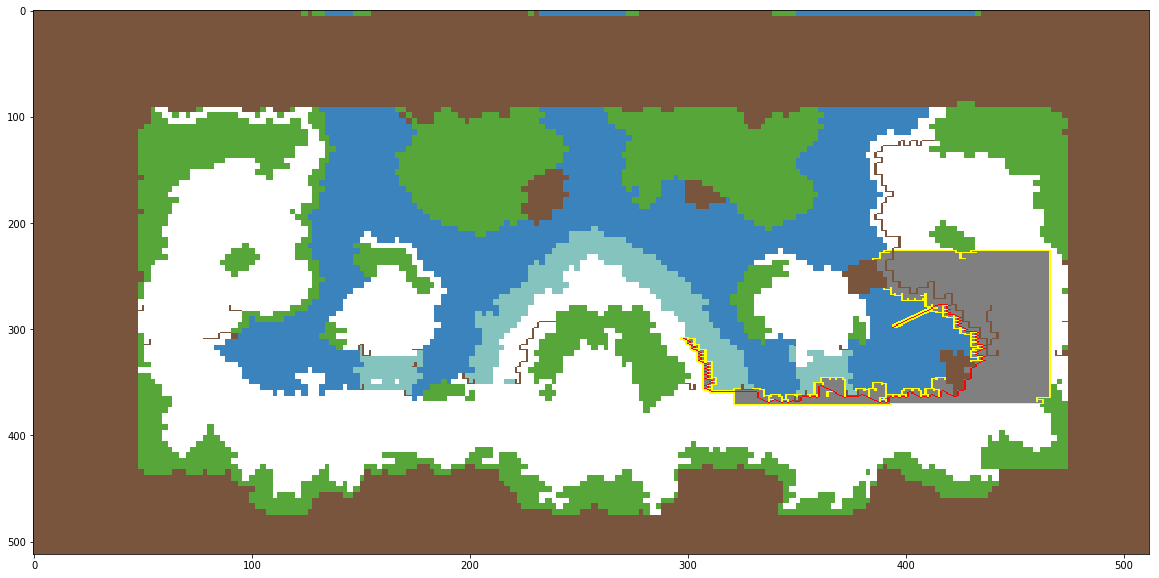


Ilustración 19. Resultado ejecución Ejercicio 4.

En el resultado observamos como el algoritmo premia los nodos con menor coste, en este caso el terreno cuyo coste es 1 y solo se adentra en celdas con costes superiores cuando se han expandido varios nodos de coste menor.

### Apartado C

A\* irá expandiendo los nodos de menor f (expandirá varios nodos de menos f antes que uno de f más grande). Esto significa que, si el terreno vale 1 y el agua 4, A\* explora primero caminos por el terreno, porque su f es menor. Solo expandirá nodos con costes g mayores cuando haya expandido varios nodos de coste menor, de este modo premia en la exploración los caminos más rápidos.

## Reflexiones finales

Al examinar los caminos óptimos calculados por A\*, intuitivamente parece que realmente no es el óptimo. Esto es debido a que, a nuestro ojo, penalizamos un camino en diagonal porque la diagonal de un cuadrado es más grande que su lado. Sin embargo, el coste de nuestro problema es igual ya sea en diagonal o en horizontal/vertical (coste = 1) por lo que existen caminos a priori “más costosos” pero que en realidad no lo son.

Veamos un ejemplo:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
|  |  |  |  |  |
|  |  |  |  |  |

Ilustración 20. Ejemplo del coste total sin penalizar caminos en diagonal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 |  |  |  | 4 |
|  | 1 |  | 3 |  |
|  |  | 2 |  |  |

Ilustración 21. Ejemplo del coste total sin penalizar caminos en diagonal (camino alternativo).

Aunque en la ilustración 21 el camino se mueve en diagonal y podamos pensar que el camino es más largo, los dos caminos son de coste igual y por tanto da igual cual de ellos escoger.

Sin embargo, si en el algoritmo indicamos que el coste de tomar un camino en diagonal es la longitud de la diagonal del cuadrado con lado igual al coste de transición de la celda, observamos como el camino calculado es, en apariencia, más natural.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 |
|  |  |  |  |  |
|  |  |  |  |  |

Ilustración 22. Ejemplo del coste total penalizando caminos en diagonal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 |  |  |  | 4\*√2 |
|  | √2 |  | 3\*√2 |  |
|  |  | 2\*√2 |  |  |

Ilustración 23. Ejemplo del coste total penalizando caminos en diagonal (camino alternativo).

Un ejemplo en nuestro mapa sería:

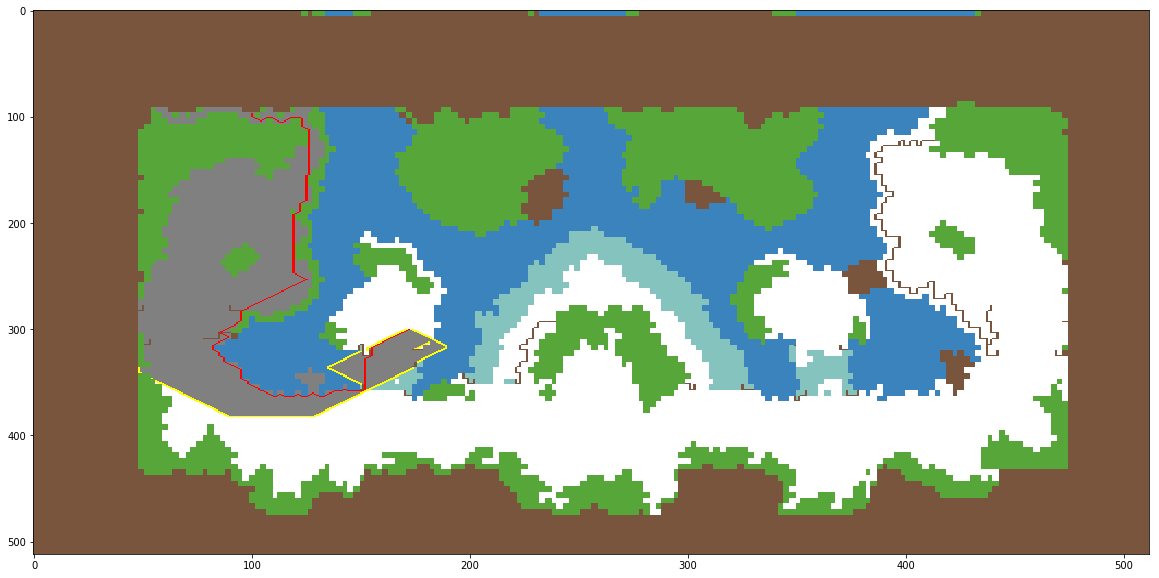


Ilustración 24. Resultado A\* sin penalizar el camino en diagonal.

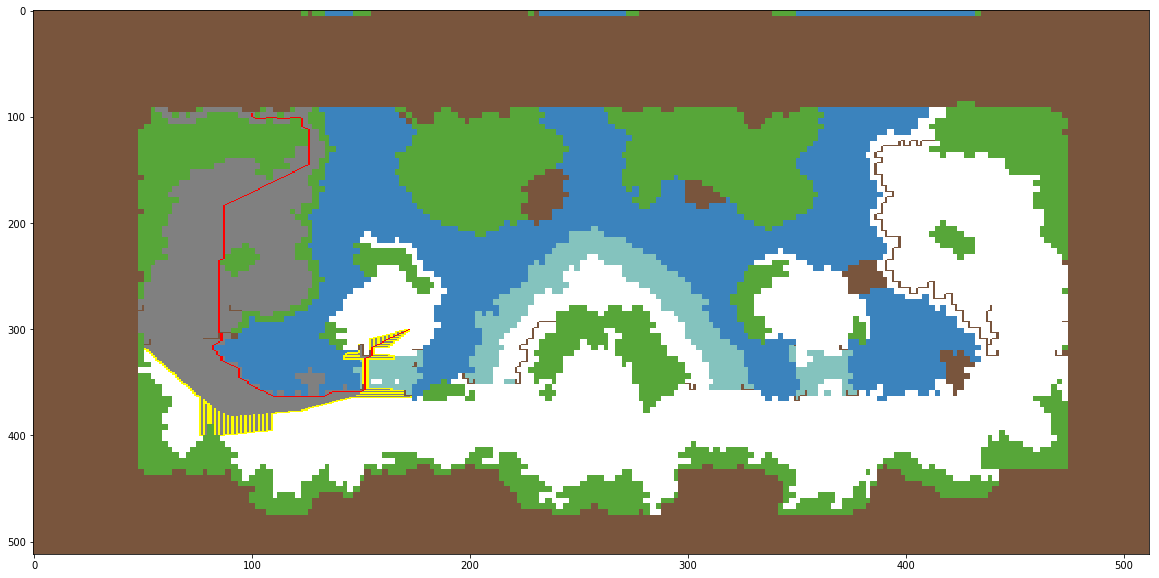


Ilustración 25. Resultado A\* penalizando el camino en diagonal.

Ampliado:

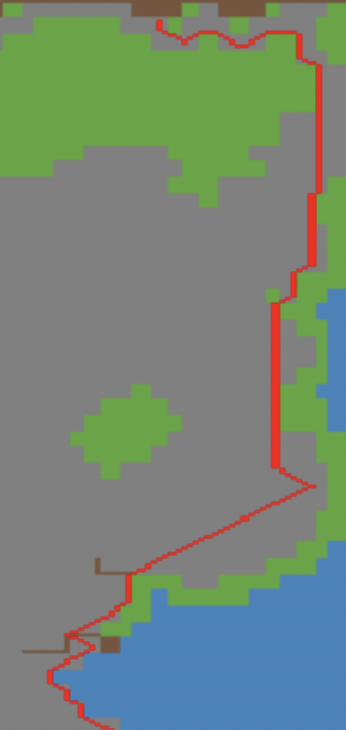


Ilustración 26. Comparativa penalizando y sin penalizar el camino diagonal.

La imagen de la izquierda, correspondiente al camino que penaliza los tránsitos en diagonal, acorta lo máximo posible las curvas y parece, al ojo humano, más corto que el otro.

# Bibliografía

**García-Tejedor, Álvaro.** Moodle UFV. Recursos de la asignatura. [En línea] <http://moodleufv.ufv.es/moodle/>.

**Patel, Amit.** Heuristics. *Red Blob Games.* [En línea] https://www.redblobgames.com/.