

# Ubicame Univalle

## Proyecto Práctico de Inteligencia Artificial

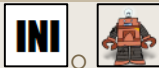







Caicedo Hidalgo Geraldine  
 Universidad del Valle, Escuela de la Ingeniería de Sistemas y Computación  
 Mayo del 2015 Cali, Colombia  
 1527691 - 3743  
 {geraldine.caicedo@correounivalle.edu.co}

**Abstracto.** Este documento presenta el análisis y diseño del proyecto práctico del curso de Introducción a la Inteligencia Artificial, contiene la definición del problema, abordaje del mismo, donde se presenta como se implementaron los algoritmos de búsqueda no informada y búsqueda informada, se analizan las heurísticas implementadas, se muestran las pruebas hechas a la implementación y el análisis de los resultados, finalizando con las conclusiones del proyecto.

**Palabras Calve.** Búsqueda Informada, Búsqueda no Informada, Heurística, Inteligencia Artificial.

## I. Introducción

La Universidad del Valle ha construido un prototipo de Robot llamado Ubicame en UV, el robot debe buscar algún profesor o sitio de la Universidad que alguna persona requiera, el Robot tiene información del ambiente que contiene algunos obstáculos, su ubicación inicial y el objetivo. El ambiente se representa con una matriz de enteros de tamaño  $n \times m$ , se presenta un cuadro con el elemento, el valor en la matriz, el costo de estar en cada elemento y la imagen que lo representara en el mapa:

Elemento	Representación	Costo	Imagen
Inicio	0	0	
Pared	1	/-/	
Espacio Disponible	2	1	
Piso Resbaloso	3	3	
Lugar con Alto Flujo de Personas	4	4	
Lugar Restringido	5	6	
Recarga del Robot	6	5	
Meta	7	1	

El robot es un prototipo y solo puede avanzar 6 casillas antes de necesitar recargar, si se queda sin energía, entonces se detiene sin remedio, el robot puede moverse arriba, abajo, izquierda y derecha.

La aplicación debe leer un archivo de entrada con el siguiente formato que representa, en la primera línea, el tamaño del ambiente, y en la segunda línea la matriz del ambiente.

Ejemplo:



Datos en el Archivo:

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

La salida debe ser un archivo de texto o en consola con la siguiente información:

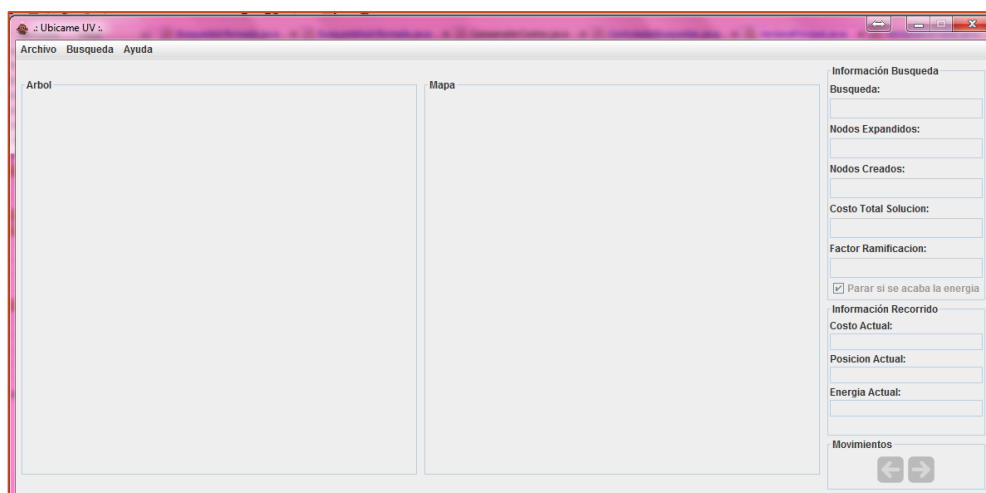
```
Nodos expandidos: X
Nodos Creados: Y
Costo total solución: Z
Factor ramificación: W
```

## II. Abordaje del Problema

Se abordara el problema utilizando dos tipos de búsquedas, Informada y No Informada, para la no informada se eligió el algoritmo de búsqueda de Costo Uniforme. En la busque Informada se implementó la búsqueda A\*, con dos heurísticas diferentes.

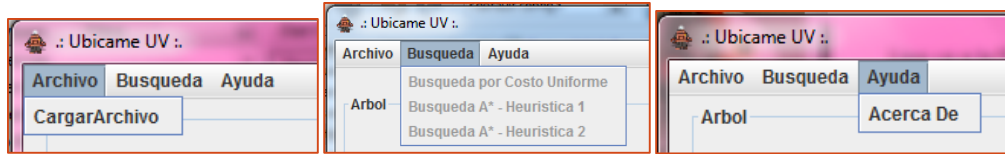
### A. Descripción de la Interfaz

La implementación se inicia con una ventana inicial:





Esta tiene una barra de herramientas en donde se puede elegir el archivo que generara el mapa, el tipo de búsqueda que se quiere hacer y una ayuda que muestra información sobre el proyecto.



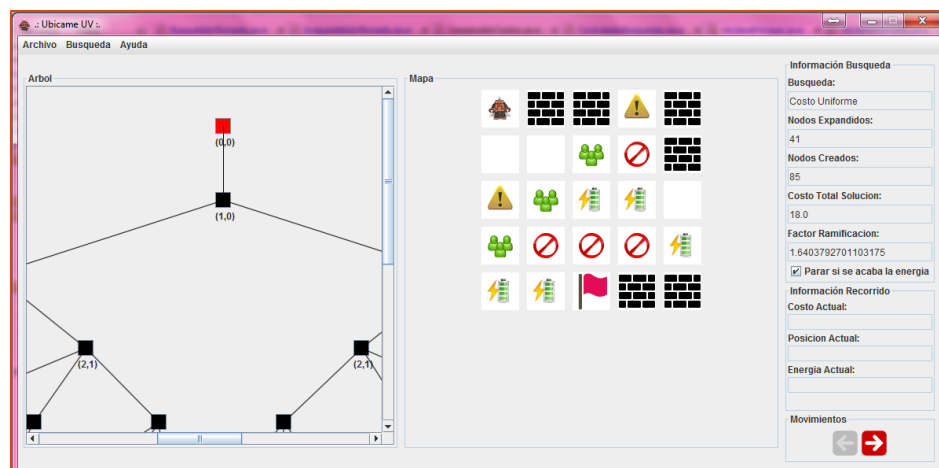
La selección de búsquedas se habilita cuando se haya cargado un archivo.

La ventana principal está dividida en 3 partes:



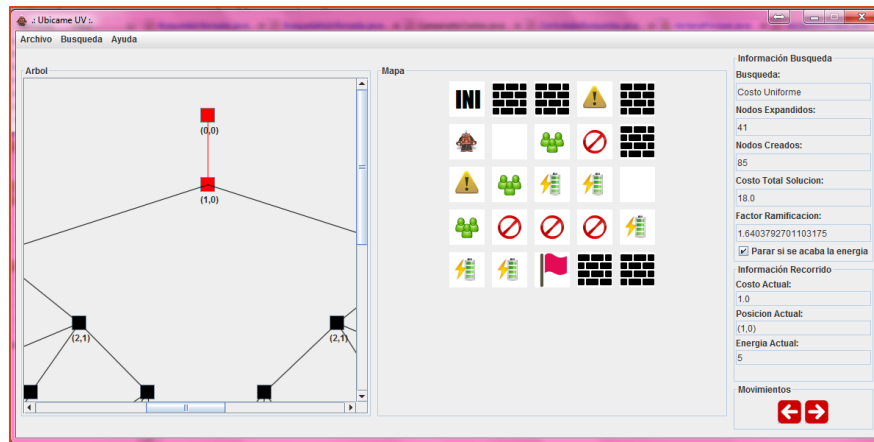
- **1** Zona Árbol: en esta parte se muestra el árbol expandido que se genera en las búsquedas.
- **2** Zona Mapa : en esta zona se mostrara el mapa que se genera según el archivo el
- **3** Zona Información: en esta zona se muestra la información de la búsqueda, (nombre, nodos expandidos, nodos creados, costo total, factor ramificación), se puede elegir si se detiene la búsqueda cuando se acaba la energía o no, se muestra la información del recorrido que se esté haciendo (costo actual, posición actual, energía actual) y los botones de movimiento en el mapa.

Así se verá la interfaz cuando hayamos elegido un archivo y elegido una búsqueda:





Utilizando los botones de movimiento se puede seguir la ruta elegida según la búsqueda, mostrando el movimiento paso a paso en el mapa y en el árbol.



## B. Estructura de la Implementación

EL proyecto está dividido en 3 paquetes principales, Control, Lógica e Interfaz, a continuación se presenta que contiene cada paquete.

- **Control:** Este paquete contiene las clases que se encargan del movimiento de los datos desde el archivo de texto, y desde la lógica a la interfaz.
  - *ControladorBusquedas:* Esta clase se encarga de pasar la información desde la lógica hacia la interfaz.
  - *ControladorArchivo;* Esta clase se encarga de la lectura y escritura de archivos.
- **Lógica:** Este paquete contiene las clases que se encargan de hacer las búsquedas, generar las heurísticas y general el árbol expandido además de toda su información.
  - *Nodo:* Esta clase representa el objeto nodo, contiene la información que debe tener el nodo (costo, coordenada, energía del robot, ruta del nodo, índice, índice del padre, profundidad del nodo)
  - *PuntoRuta:* Esta clase representa el objeto Punto Ruta, contiene la información de un punto en la ruta de un nodo. (costo, coordenada, energía del robot)
  - *ComparadorCostos:* Esta clase se encarga de hacer la comparación que se necesita en la cola de prioridad utilizada para sacar el nodo de menor costo.
  - *BusquedaNoInformada:* Esta clase se encarga de desarrollar la búsqueda no informada por costo uniforme, utilizando una cola de prioridad de objetos nodos y generando así el nodo final con la ruta elegida por el algoritmo y el árbol generado por esta misma, además de la información de su información.



- *BusquedaInformada*: Esta clase se encarga de desarrollar la búsqueda informada A\*, utilizando dos heurísticas diferentes, la primera heurística es la distancia manhattan del punto a la meta, la segunda depende del valor manhattan a la meta, valor manhattan a la zona de carga más cercana y la energía del robot, al igual que la búsqueda no informada, utiliza una cola de prioridad de objetos nodos, y genera el nodo final y el árbol expandido con su información.
- **Interfaz**: Este paquete contiene las clases que se encargan de mostrar en pantalla al usuario los resultados de las búsquedas.
  - *DibujaArbol*: Esta clase se encarga de calcular las posiciones y las aristas de los nodos del árbol a dibujar según un arreglo de nodos generado por la búsqueda.
  - *NuevoCanvas*: Esta clase dibuja el árbol en un `JPanel`, utilizando los datos de la clase *DibujaArbol*, crea los botones con la información correspondiente y los dibuja en un nuevo canvas.
  - *VentanaAcercaDe*: Esta clase representa la ventana de información de ayuda en la interfaz principal del programa, muestra los datos del desarrollador.
  - *VentanaPrincipal*: Esta clase representa la ventana principal del programa, en esta se muestra el árbol, el mapa y la información de la búsqueda que se espera de salida, también incluye unos botones de movimiento que ayudaran a moverse en la ruta elegida por el algoritmo paso a paso.

## C. Priority Queue

En la implementación de las dos búsquedas se utiliza el Priority Queue, una clase de java que implementa una cola de prioridad, al crear una Priority Queue se debe establecer el tamaño inicial de la cola y un valor de comparación.

En el proyecto se creó una clase que implementa `Comparator`, teniendo en cuenta que debemos comparar los costos de los nodos en las dos búsquedas, utilizamos un `Comparator` de Nodos `Comparator<Nodo>`, la lógica del comparador es la siguiente, compara los costos del nodo a remover con el resto que se encuentran en la cola, si el costo es el menor de todos, se remueve de la cola, si hay otro menor, se compara este con el resto de la cola, en caso de tener nodos con costos iguales, se elige el que primero se haya creado, es decir, el que entro primero a la cola.

## D. Búsqueda General

Debido a los requerimientos del proyecto, no se puede simplemente sacar los nodos de la cola de prioridad hasta que esta esté vacía, se deben sacar los nodos hasta que se llegue a la meta, o, hasta que la energía del robot se acabe, para esto, en las dos búsquedas, se implementó con un `while` que se



ejecuta mientras la cola no este vacía,; dentro del `while` se crea un nodo nuevo donde se guarda el nodo que se remueve de la cola, si este nodo cumple la condición de que el robot tiene energía aun y que no es el nodo meta, se procede a crear sus hijos y se suma uno al número de nodos expandidos, si el nodo a expandir es la meta el `while` se detiene. Si la energía del robot llega a 0, se guarda el nodo donde se llegó a cero, que es el nodo de menor costo hasta ahora y se sigue buscando hasta encontrar la meta, si no se encuentra, retorna el nodo guardado. Al final tendremos el árbol con los nodos creados y el nodo final que contiene la ruta tomada para llegar hasta él.

Para elegir los hijos del nodo se busca en todas las direcciones del nodo, se suma o resta uno a las coordenadas del nodo dependiendo la dirección, si se va a arriba, se resta 1 a  $y$ , si se va abajo se suma uno a  $y$ , si se va a la izquierda se resta uno a  $x$ , si se va a la derecha se suma uno a  $x$ .

Cuando ya se tiene la coordenada de los posibles hijos del nodo, se procede a crear y agregar los hijos del nodo si cumple con estas condiciones:

- si al sumar o restar los  $x$  y  $y$ , las coordenadas son negativas o se pasan del tamaño inicial del mapa, entonces no se crea el hijo nodo.
- si en la coordenada a buscar se encuentra una pared, es decir, el tipo del nodo es 1, no se crea el hijo nodo.
- si la coordenada a buscar ya se encuentra en la ruta del nodo anterior, no se crea el hijo nodo.

Cuando ya se puede crear el nodo, se aumenta en uno el número de nodos creados, se copia la ruta del nodo padre y se añada la coordenada actual, se calcula la energía del robot y se agregan datos que se necesitaran más adelante como un índice para establecer el orden de creación, el índice del nodo padre y la profundidad del árbol donde se encontrara el nodo, se crea el nodo y se agrega a la cola y al árbol para después dibujarlo. Antes de crear el nodo se calcula el costo que tendrá ir a ese nodo, en esta parte es donde varía según el algoritmo de búsqueda.

## E. Algoritmo de Búsqueda no Informada

El algoritmo de búsqueda no informada que se implemento fue la búsqueda por costo uniforme, esta consiste en expandir el nodo de menor costo sin importar a que profundidad este, si se llega al nodo meta, o en este problema, si se le acaba la energía al robot, el árbol se dejara de expandir.

En este caso cuando se expande un nodo y se van a crear los hijos, el algoritmo suma el costo del nodo y el costo del nodo a crear según el tipo de nodo que es. Así se agrega el valor del costo a cada nodo y con la cola de prioridad se saca el nodo que tenga menos costo, se expande y así sucesivamente hasta que cumpla con las condiciones de parada.

Se eligió la búsqueda por costo uniforme ya que el problema presenta una variación de costos según el camino que se elija, y como siempre se quiere llegar a la solución óptima y que cueste menos, es mejor tener en cuenta los costos de cada movimiento.

## F. Algoritmo de Búsqueda Informada

El algoritmo de búsqueda informada que se implementó fue la búsqueda A\*, la cual consiste en principio de expandir el nodo con menor costo como la

búsqueda por costo uniforme, pero en este caso se tiene en cuenta una heurística, la cual es una función que asigna a cada nodo un valor que corresponde al costo estimado de llegar a la meta estando en dicho nodo, en caso de la búsqueda A\* no se usa solo el costo o la heurística, para elegir que nodo expandir se suma la heurística y el costo del nodo, dando como resultado el costo total  $f(x) = g(x) + h(x)$ .

Para elegir la heurística se tiene una función que cambia el valor a calcular según un índice elegido en la ventana principal.

### 1. Primera Heurística

La primera heurística consiste simplemente en el valor manhattan del nodo a la meta, decimos que esta heurística es admisible ya que en cualquier punto del mapa la heurística toma un valor menor al costo real, ya que el valor manhattan no toma en cuenta los costos de cada casilla, para esta heurística el mapa solo tiene casillas vacías y la meta, entonces toma el valor del costo en cada casilla como 1, calculando así el costo mínimo.

### 2. Segunda Heurística

La segunda heurística consiste en la multiplicación del valor manhattan del nodo a la meta ( $mh(x,y)$ ), por la suma del valor manhattan del nodo a un nodo de carga ( $mh(x,z)$ ) y el valor de la energía restante para que el robot este cargado completamente, es decir, la resta de la energía actual y la energía máxima del robot ( $E_{ma} - E_a$ ). Se quiere tener en cuenta al calcular la ruta la energía del robot para llegar a la meta antes de que el robot se detenga por la falta de energía.

Al operar estos 3 datos nos da una heurística que no es admisible, ya que el valor de la heurística puede tomar un valor mayor al costo real, para esto se debe encontrar un numero para multiplicar o dividir que nos dé una heurística admisible, para este tomaremos un caso hipotético donde tengamos un mapa  $n \times n$  vacío (solo con espacios vacíos, sin obstáculos) con la meta en el extremo del mapa, y una zona de carga cerca a la meta, diríamos que la meta está en la posición  $(n,n)$  y la energía está en la posición  $(n,n-1)$  o  $(n-1,n)$ , la posición inicial del robot seria  $(0,0)$ , y estaría sin energía, tenemos entonces que la heurística sería:

$$\begin{aligned} (mh(x,y) * (mh(x,z) + (E_{ma} - E_a))) &= (2n * (n + (n - 1)) + 6) \\ &= 2n * (2n + 5) = 4n^2 + 10n \end{aligned}$$

Para que la heurística sea admisible, esta heurística debe ser menor al costo total, ya que el mapa esta vacío y el costo de moverse a un espacio vacío es 1, tendremos que el costo total será igual al valor manhattan del punto a la meta, entonces:

$$4n^2 + 10n \leq 2n$$

Para que la heurística este cerca del valor real intentaremos que  $4n^2 - 10n$  sea cercano a  $2n$ , como el valor de la heurística es muy grande, se encontrara un valor que al dividir la heurística la vuelva admisible:

$$\frac{4n^2 + 10n}{x} = 2n \Rightarrow \frac{1}{x} = \frac{2n}{4n^2 + 10n} \Rightarrow x = \frac{4n^2 + 10n}{2n}$$

$$x = 2n + \frac{10n}{2n} \Rightarrow x = 2n + \frac{5}{n}$$

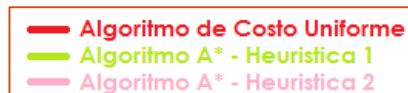
Tenemos entonces el valor por el que debemos dividir el valor de la heurística para que sea admisible.

$$\frac{4n^2 + 10n}{2n + \frac{5}{n}} \leq 2n$$

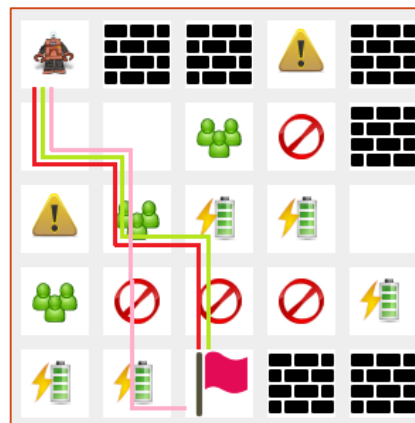
Esta desigualdad es cierta, no importa qué valor tome  $n$ , la heurística siempre será menor que el costo real; en este caso el mapa solo tiene espacios vacíos, así que el costo mínimo a la meta será el valor manhattan  $2n$ , pero si tenemos obstáculos en el camino, el costo real aumentara pero la heurística seguirá siendo menor que el costo mínimo, así que podemos decir que la heurística diseñada es admisible.

### III. Pruebas y Análisis de Resultados

Para hacer las pruebas se crearon 7 mapas adicionales al dado en el documento del proyecto, se analizaran los resultados dados en cada mapa con cada algoritmo, la ruta de cada algoritmo se muestra en el mapa y está identificada con la siguiente notación de colores:



#### A. Mapa 1



- Algoritmo por costo uniforme**  
Ruta elegida:  $[(0,0), (1,0), (1,1), (2,1), (2,2), (3,2), (4,2)]$   
Nodos Expandidos: 41  
Nodos Creados: 85  
Costo Total: 18  
Factor de Ramificación: 1.64037
- Algoritmo A\* - Heurística 1**  
Ruta elegida:  $[(0,0), (1,0), (1,1), (2,1), (2,2), (3,2), (4,2)]$   
Nodos Expandidos: 25  
Nodos Creados: 55



Costo Total: 18

Factor de Ramificación: 1.65395

- **Algoritmo A\* - Heurística 2**

Ruta elegida:  $[(0,0),(1,0),(1,1),(2,1),(3,1),(4,1),(4,2)]$

Nodos Expandidos: 36

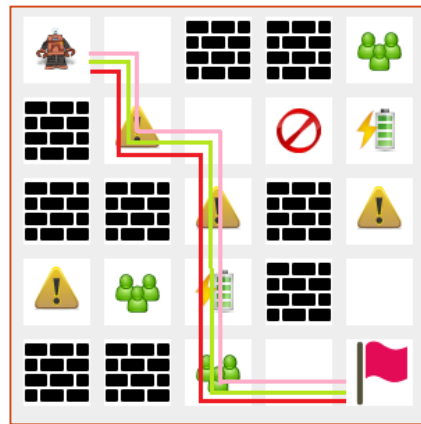
Nodos Creados: 78

Costo Total: 18

Factor de Ramificación: 1.62497

El algoritmo de costo uniforme y la heurística manhattan toman la misma ruta, la heurística 2 toma una ruta diferente pero que tiene el mismo costo, También vemos que la heurística manhattan fue la que menos nodos creo y el algoritmo de costo uniforme el que más creo nodos.

## B. Mapa 2



- **Algoritmo por costo uniforme**

Ruta elegida:  $[(0,0), (0,1), (1,1), (1,2), (2,2), (3,2), (4,2), (4,3), (4,4)]$

Nodos Expandidos: 12

Nodos Creados: 15

Costo Total: 19

Factor de Ramificación: 1.31950

- **Algoritmo A\* - Heurística 1**

Ruta elegida:  $[(0,0), (0,1), (1,1), (1,2), (2,2), (3,2), (4,2), (4,3), (4,4)]$

Nodos Expandidos: 10

Nodos Creaos: 13

Costo Total: 19

Factor de Ramificación: 1.30200

- **Algoritmo A\* - Heurística 2**

Ruta elegida:  $[(0,0), (0,1), (1,1), (1,2), (2,2), (3,2), (4,2), (4,3), (4,4)]$

Nodos Expandidos: 10

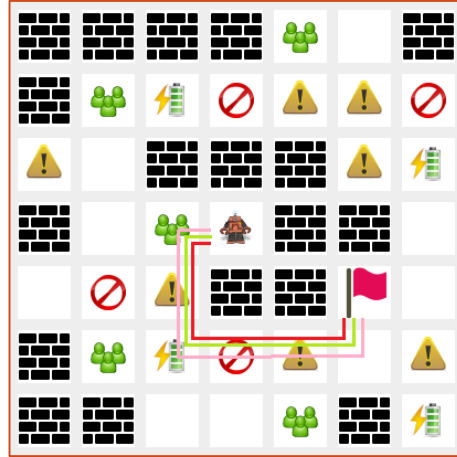
Nodos Creados: 13

Costo Total: 19

Factor de Ramificación: 1.30200

En este mapa el algoritmo de costo uniforme, y las dos heurísticas toman la misma ruta, con la diferencia que en el algoritmo A\* las dos heurísticas, se crean y expanden menos nodos, llegando al mismo costo.

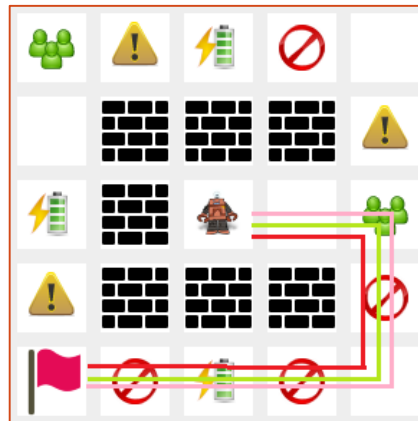
### C. Mapa 3



- **Algoritmo por costo uniforme**  
Ruta elegida: [(3,3),(3,2),(4,2),(5,2),(5,3),(5,4),(5,5), (4,5)]  
Nodos Expandidos: 46  
Nodos Creados: 66  
Costo Total: 23  
Factor de Ramificación: 1.46557
- **Algoritmo A\* - Heurística 1**  
Ruta elegida: [(3,3),(3,2),(4,2),(5,2),(5,3),(5,4),(5,5), (4,5)]  
Nodos Expandidos: 30  
Nodos Creados: 47  
Costo Total: 23  
Factor de Ramificación: 1.42180
- **Algoritmo A\* - Heurística 2**  
Ruta elegida: [(3,3),(3,2),(4,2),(5,2),(5,3),(5,4),(5,5), (4,5)]  
Nodos Expandidos: 37  
Nodos Creados: 57  
Costo Total: 23  
Factor de Ramificación: 1.44647

Los tres algoritmos toman la misma ruta y obtienen el mismo costo, pero el algoritmo de costo uniforme crea y expande más nodos que el resto de las búsquedas, y la heurística manhattan expande y crea menos nodos que los demás.

## D. Mapa 4



- **Algoritmo por costo uniforme**  
Ruta elegida: [(2,2),(2,3),(2,4),(3,4),(4,4),(4,3),(4,2),(4,1),(4,0)]  
Nodos Expandidos: 15  
Nodos Creados: 16  
Costo Total: 30  
Factor de Ramificación: 1.26630
- **Algoritmo A\* - Heurística 1**  
Ruta elegida: [(2,2),(2,3),(2,4),(3,4),(4,4),(4,3),(4,2),(4,1),(4,0)]  
Nodos Expandidos: 13  
Nodos Creados: 14  
Costo Total: 30  
Factor de Ramificación: 1.31101
- **Algoritmo A\* - Heurística 2**  
Ruta elegida: [(2,2),(2,3),(2,4),(3,4),(4,4),(4,3),(4,2),(4,1),(4,0)]  
Nodos Expandidos: 15  
Nodos Creados: 16  
Costo Total: 30  
Factor de Ramificación: 1.26630

En este mapa el algoritmo de costo uniforme, y las dos heurísticas toman la misma ruta, con la diferencia que en el algoritmo A\* la primera heurística crea y expande menos nodos que los otros dos algoritmos.

## E. Mapa 5

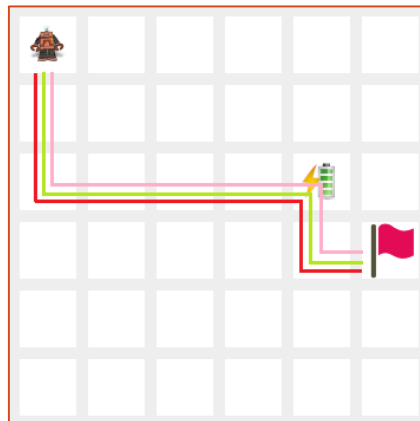




- **Algoritmo por costo uniforme**  
Ruta elegida:  $[(0,0),(1,0)]$   
Nodos Expandidos: 1  
Nodos Creados: 2  
Costo Total: 1  
Factor de Ramificación: 1.44224
- **Algoritmo A\* - Heurística 1**  
Ruta elegida:  $[(0,0),(1,0)]$   
Nodos Expandidos: 1  
Nodos Creados: 2  
Costo Total: 1  
Factor de Ramificación: 1.44224
- **Algoritmo A\* - Heurística 2**  
Ruta elegida:  $[(0,0),(1,0)]$   
Nodos Expandidos: 1  
Nodos Creados: 2  
Costo Total: 1  
Factor de Ramificación: 1.44224

En este caso la meta está cerca al inicio, todos los algoritmos implementados encuentran la misma ruta de solución y expanden y crean la misma cantidad de nodos y tienen al final el mismo costo.

## F. Mapa 6



- **Algoritmo por costo uniforme**  
Ruta elegida:  $[(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(2,4),(3,4),(3,5)]$   
Nodos Expandidos: 165  
Nodos Creados: 410  
Costo Total: 12  
Factor de Ramificación: 1.72831
- **Algoritmo A\* - Heurística 1**  
Ruta elegida:  $[(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(2,4),(3,4),(3,5)]$   
Nodos Expandidos: 146  
Nodos Creados: 365  
Costo Total: 12

Factor de Ramificación: 1.80396

- **Algoritmo A\* - Heurística 2**

Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(2,4),(3,4),(3,5)]

Nodos Expandidos: 161

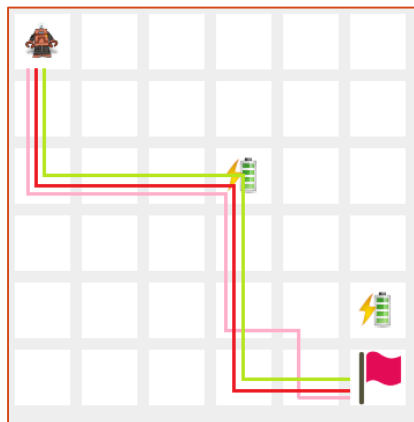
Nodos Creados: 401

Costo Total: 12

Factor de Ramificación: 1.82147

En este mapa Los algoritmos tienen la misma ruta y el mismo costo, pero los nodos expandidos varían, el costo uniforme crea y expande más nodos que los demás, y la primera heurística crea y expande menos nodos que el resto.

## G. Mapa 7



- **Algoritmo por costo uniforme**

Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(3,3),(4,3),(5,3),(5,4),(5,5)]

Nodos Expandidos: 876

Nodos Creados: 1881

Costo Total: 14

Factor de Ramificación: 1.78605

- **Algoritmo A\* - Heurística 1**

Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(3,3),(4,3),(5,3),(5,4),(5,5)]

Nodos Expandidos: 271

Nodos Creados: 664

Costo Total: 14

Factor de Ramificación: 1.71883

- **Algoritmo A\* - Heurística 2**

Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3),(3,3),(4,3),(4,4),(5,4),(5,5)]

Nodos Expandidos: 371

Nodos Creados: 908

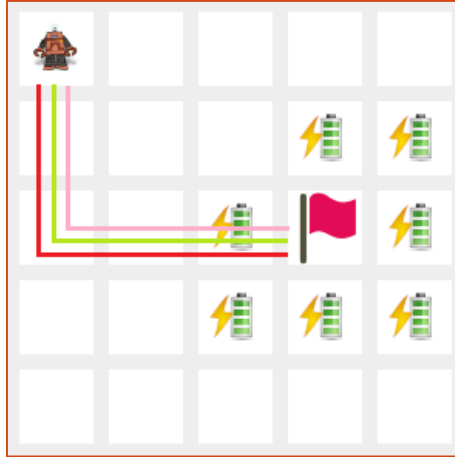
Costo Total: 14

Factor de Ramificación: 1.76419

En este mapa todos los algoritmos van primero por la zona de carga antes que abalanzarse sobre la meta, la segunda heurística hace un ligero cambio en la ruta que llevaba idéntica a las otras dos debido a

la cercanía de la carga, pero al final decide ir por la meta directamente, en este caso el algoritmo de costo uniforme crea y expande más nodos que el resto de las búsquedas, y la primera heurística menos que los demás.

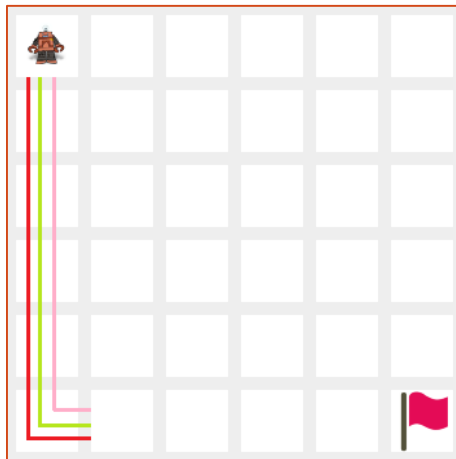
## H. Mapa 8



- **Algoritmo por costo uniforme**  
Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3)]  
Nodos Expandidos: 70  
Nodos Creados: 156  
Costo Total: 9  
Factor de Ramificación: 1.88142
- **Algoritmo A\* - Heurística 1**  
Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3)]  
Nodos Expandidos: 63  
Nodos Creados: 137  
Costo Total: 9  
Factor de Ramificación: 1.85133
- **Algoritmo A\* - Heurística 2**  
Ruta elegida: [(0,0),(1,0),(2,0),(2,1),(2,2),(2,3)]  
Nodos Expandidos: 69  
Nodos Creados: 153  
Costo Total: 9  
Factor de Ramificación: 1.87689

En este mapa los tres algoritmos siguen la misma ruta, su costo final es el mismo, el costo uniforme expande y crea más nodos que el resto, y la heurística uno menos que los demás.

## I. Mapa 9



- **Algoritmo por costo uniforme**  
Ruta elegida:  $[(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (5,1)]$   
Nodos Expandidos: 101  
Nodos Creados: 244  
Costo Total: 6  
Factor de Ramificación: 1.98905
- **Algoritmo A\* - Heurística 1**  
Ruta elegida:  $[(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (5,1)]$   
Nodos Expandidos: 101  
Nodos Creados: 244  
Costo Total: 10  
Factor de Ramificación: 1.98905
- **Algoritmo A\* - Heurística 2**  
Ruta elegida:  $[(0,0), (1,0), (2,0), (3,0), (4,0), (5,0), (5,1)]$   
Nodos Expandidos: 101  
Nodos Creados: 244  
Costo Total: 8  
Factor de Ramificación: 1.98905

En este mapa ninguno de los algoritmos llega a la solución, dado que antes de llegar se le acaba la energía al robot, y no hay ninguna zona de carga en el mapa. El costo es igual en todos, al igual que el número de nodos expandidos y creados.

## IV. Conclusiones

- A pesar de encontrar la respuesta, el algoritmo de búsqueda por costo uniforme expande y crea más nodos de los necesarios haciendo la búsqueda.
- El algoritmo de búsqueda A\* con la distancia manhattan como heurística alcanza la respuesta optima del problema ya que es admisible, además expande y crea menos nodos que las demás búsquedas, pero no tiene en cuenta los obstáculos del mapa, como el costo de cada casilla y la energía agotante del robot.



- El algoritmo de búsqueda A\* con la segunda heurística implementada alcanza una solución óptima del problema, a pesar de expandir y crear más nodos que la primera opción de heurística (Distancia Manhattan), esta opción tiene en cuenta la energía del robot, y las zonas de recarga, pone en prioridad la llegada a la meta, pero teniendo en cuenta la energía del robot.
- La primera heurística domina a la segunda heurística, ya que esta toma valores en los nodos, más grandes que los valores de la segunda heurística.
- En algunos casos de las pruebas, la segunda heurística tienen un factor de ramificación menor que la primera heurística.

## V. Bibliografía

1. [Free Icons] [Iconos para Software] [consultado:16/04/2015] [2010-2015] [Link: <http://www.softicons.com/>]
2. [A Visual Guide to Layout Managers] [Guía de Layout] [consultado:20/04/2015] [ Link: <https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html>]
3. [How I use a PriorityQueue? – Stack OverFlow] [consultado: 15/04/2015] [2019-2015] [Link: <http://stackoverflow.com/questions/683041/java-how-do-i-use-a-priorityqueue>]
4. [Diapositivas del Curso Inteligencia Artificial] [Algoritmos de Búsqueda] [Febrero-Junio 2015]