Sistemas Inteligentes

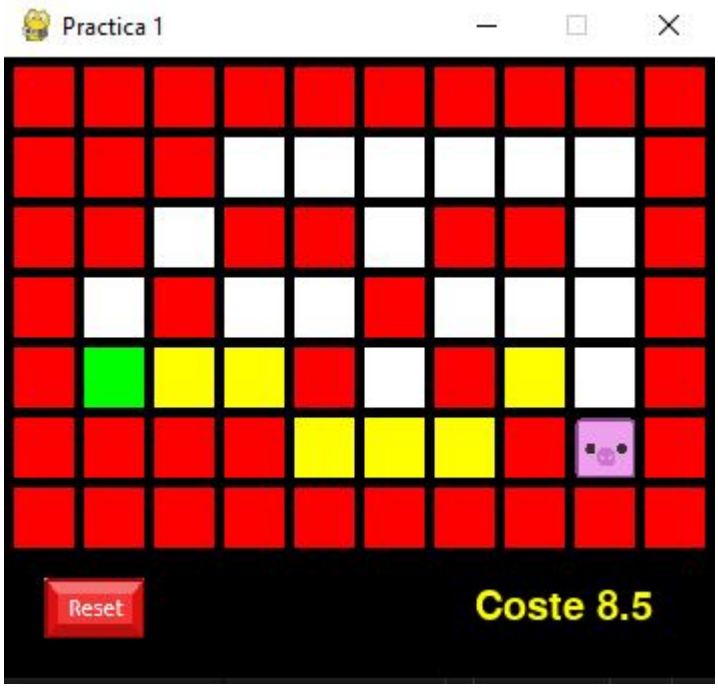
Práctica 1

Nikita Polyanskiy

Y4441167L

1. Traza del algoritmo A*

Podemos como funciona el algoritmo utilizando la distancia manhattan en este mapa de ejemplo (mapa3x3.txt):



A continuacion se detallan las iteraciones y que ocurre en cada iteracion:

--- Nº iteración: 0

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=8,y=5)

Hijo nº0: (x=7,y=4) f=7.5
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=8,y=4) f=8.0
    el hijo no está en listaAbierta, se añade.
```

--- Nº iteración: 1

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=7,y=4)

Hijo nº0: (x=6,y=3) f=9.0
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=7,y=3) f=9.5
    el hijo no está en listaAbierta, se añade.
Hijo nº2: (x=8,y=3) f=11.0
    el hijo no está en listaAbierta, se añade.
Hijo nº3: (x=8,y=4) f=9.5
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
Hijo nº4: (x=6,y=5) f=9.0
    el hijo no está en listaAbierta, se añade.
Hijo nº5: (x=8,y=5) f=11.0
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 2

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=8,y=4)

Hijo nº0: (x=7,y=3) f=9.5
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
Hijo nº1: (x=8,y=3) f=10.0
    el hijo ya está en listaAbierta...
    ...la g nueva (gm) es mejor que la g antigua, se actualiza.
Hijo nº2: (x=7,y=4) f=8.0
    el hijo ya está en listaCerrada.
Hijo nº3: (x=8,y=5) f=10.0
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 3

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=6,y=3)

Hijo nº0: (x=5,y=2) f=10.5
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=7,y=3) f=11.0
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
Hijo nº2: (x=5,y=4) f=8.5
    el hijo no está en listaAbierta, se añade.
Hijo nº3: (x=7,y=4) f=10.5
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 4

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=5,y=4)

Hijo nº0: (x=4,y=3) f=10.0
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=6,y=3) f=12.0
    el hijo ya está en listaCerrada.
Hijo nº2: (x=4,y=5) f=10.0
    el hijo no está en listaAbierta, se añade.
Hijo nº3: (x=5,y=5) f=10.5
    el hijo no está en listaAbierta, se añade.
Hijo nº4: (x=6,y=5) f=12.0
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
```

--- Nº iteración: 5

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=6,y=5)

Hijo nº0: (x=5,y=4) f=8.5
    el hijo ya está en listaCerrada.
Hijo nº1: (x=7,y=4) f=10.5
    el hijo ya está en listaCerrada.
Hijo nº2: (x=5,y=5) f=9.0
    el hijo ya está en listaAbierta...
    ...la g nueva (gm) es mejor que la g antigua, se actualiza.
```

--- Nº iteración: 6

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=5,y=5)

Hijo nº0: (x=5,y=4) f=9.0
    el hijo ya está en listaCerrada.
Hijo nº1: (x=4,y=5) f=9.0
    el hijo ya está en listaAbierta...
    ...la g nueva (gm) es mejor que la g antigua, se actualiza.
Hijo nº2: (x=6,y=5) f=11.0
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 7

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=4,y=5)
```

```
Hijo nº0: (x=3,y=4) f=8.5
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=5,y=4) f=10.5
    el hijo ya está en listaCerrada.
Hijo nº2: (x=5,y=5) f=11.0
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 8

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=3,y=4)

Hijo nº0: (x=3,y=3) f=10.5
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=4,y=3) f=12.0
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
Hijo nº2: (x=2,y=4) f=8.5
    el hijo no está en listaAbierta, se añade.
Hijo nº3: (x=4,y=5) f=12.0
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 9

```
El nodo actual pasa de la listaAbierta a la listaCerrada: (x=2,y=4)

Hijo nº0: (x=1,y=3) f=10.0
    el hijo no está en listaAbierta, se añade.
Hijo nº1: (x=3,y=3) f=12.0
    el hijo ya está en listaAbierta...
    ...la g antigua es mejor, no se hace nada.
Hijo nº2: (x=1,y=4) f=8.5
    el hijo no está en listaAbierta, se añade.
Hijo nº3: (x=3,y=4) f=10.5
    el hijo ya está en listaCerrada.
```

--- Nº iteración: 10

```
Destino encontrado!!! (x=1,y=4)
```

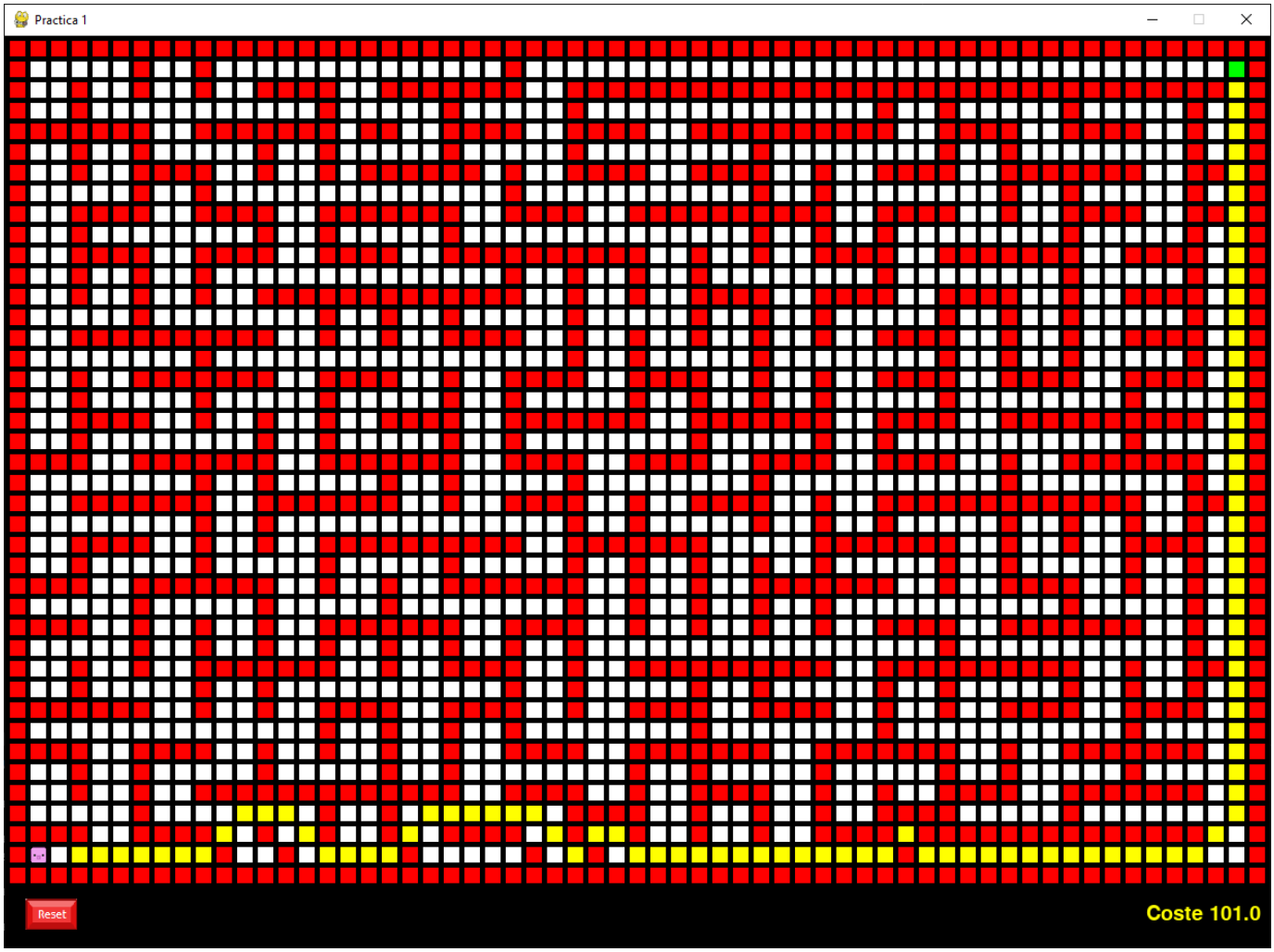
Orden de exploracion:

```
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1  3 -1 -1 -1]
 [-1 10  9  8 -1  4 -1  1  2 -1]
 [-1 -1 -1 -1  7  6  5 -1  0 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

2. Comparación de heurísticas

2.1 Manhattan

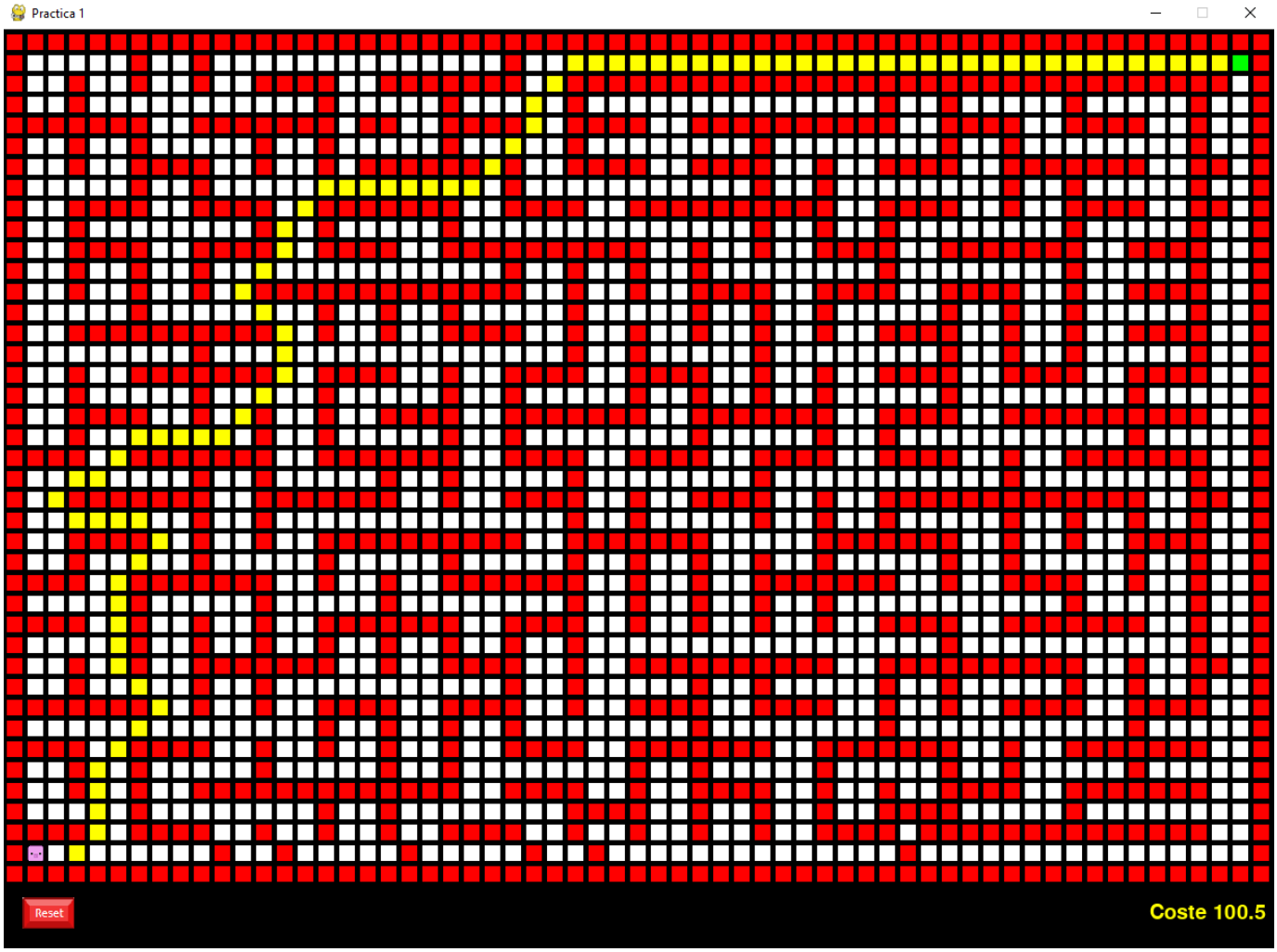
Podemos comprobar que Manhattan no es admisible en el "mapa20x20 - manhattan.txt" (para abrir este mapa primero deberemos cambiar el tamaño de la celda a TAM = 15, (30 por defecto) esta variable se encuentra en la linea 14 de main.py)




```
--- Nº iteración: 721
     Destino encontrado!!! (x=59,y=1)

Orden de exploracion:
[[ -1  -1  -1 ...  -1  -1  -1]
 [ -1  -1  -1 ...  -1 721  -1]
 [ -1  -1  -1 ...  -1 720  -1]
 ...
 [ -1  -1  -1 ... 684  -1  -1]
 [ -1   0   1 ...  -1  -1  -1]
 [ -1  -1  -1 ...  -1  -1  -1]]
```

2.2 h=0

Como podemos ver nos calcula el camino de coste 101 en tan solo 721 iteraciones (721 nodos explorados), pero si lo comparamos con una heuristica admisible (por ejemplo h=0), podemos ver que el verdadero coste optimo es 100.5:

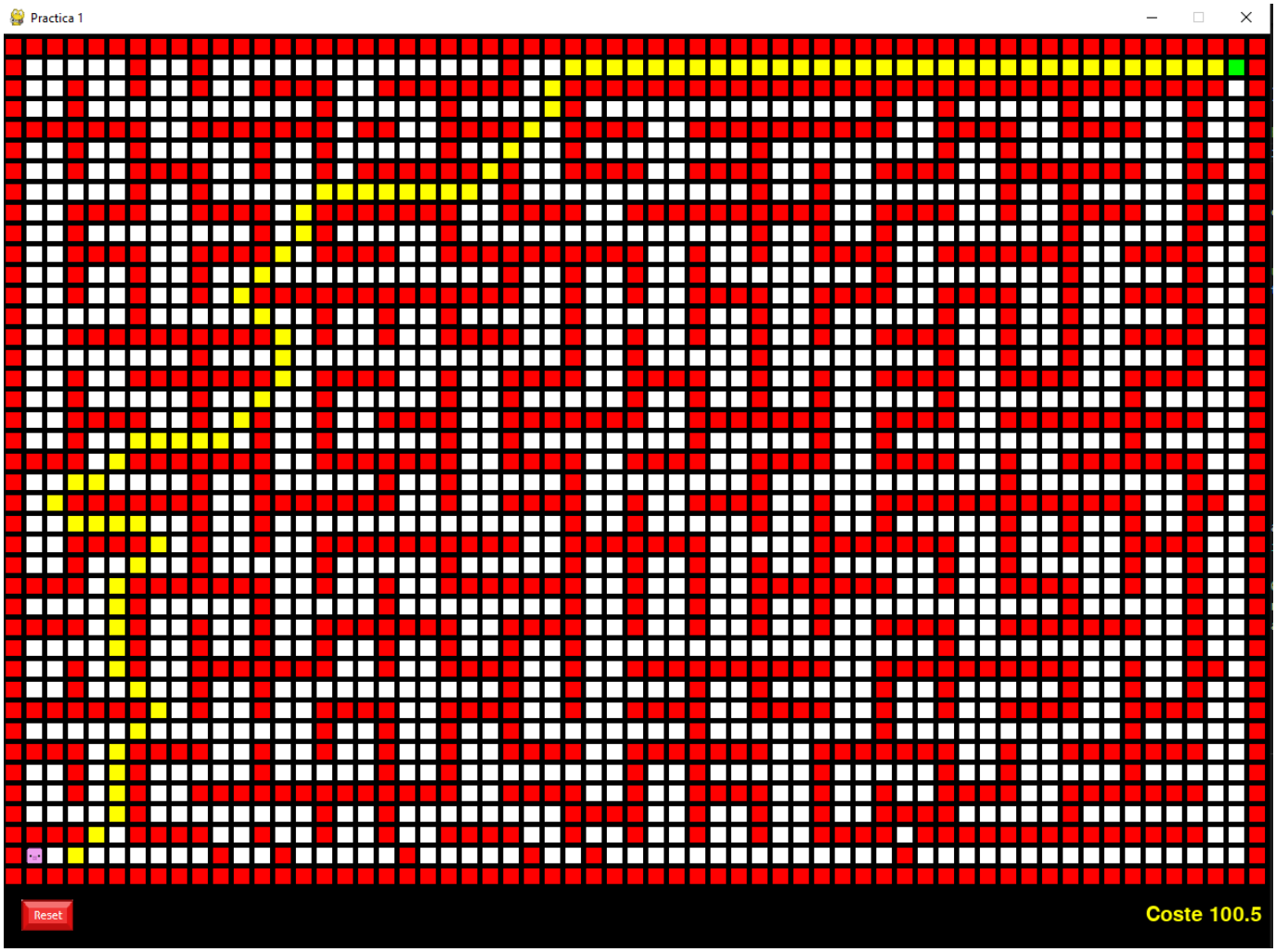



```
--- Nº iteración: 1387
     Destino encontrado!!! (x=59,y=1)

Orden de exploracion:
[[  -1   -1   -1 ...   -1   -1   -1]
 [  -1  951  916 ... 1385 1387   -1]
 [  -1  959  927 ...   -1 1386   -1]
 ...
 [  -1   -1   -1 ...  858  884   -1]
 [  -1    0    1 ...  851  876   -1]
 [  -1   -1   -1 ...   -1   -1   -1]]
```

En este caso en la heuristica h=0 se encontró el camino de coste optimo 100.5 en 1387 iteraciones.

2.3 Euclidea

Y como podemos ver a continuacion, la distancia euclidea a diferencia de la distancia de manhattan, si que es una heuristica admisible, y además es más rápida que la heuristica h=0:



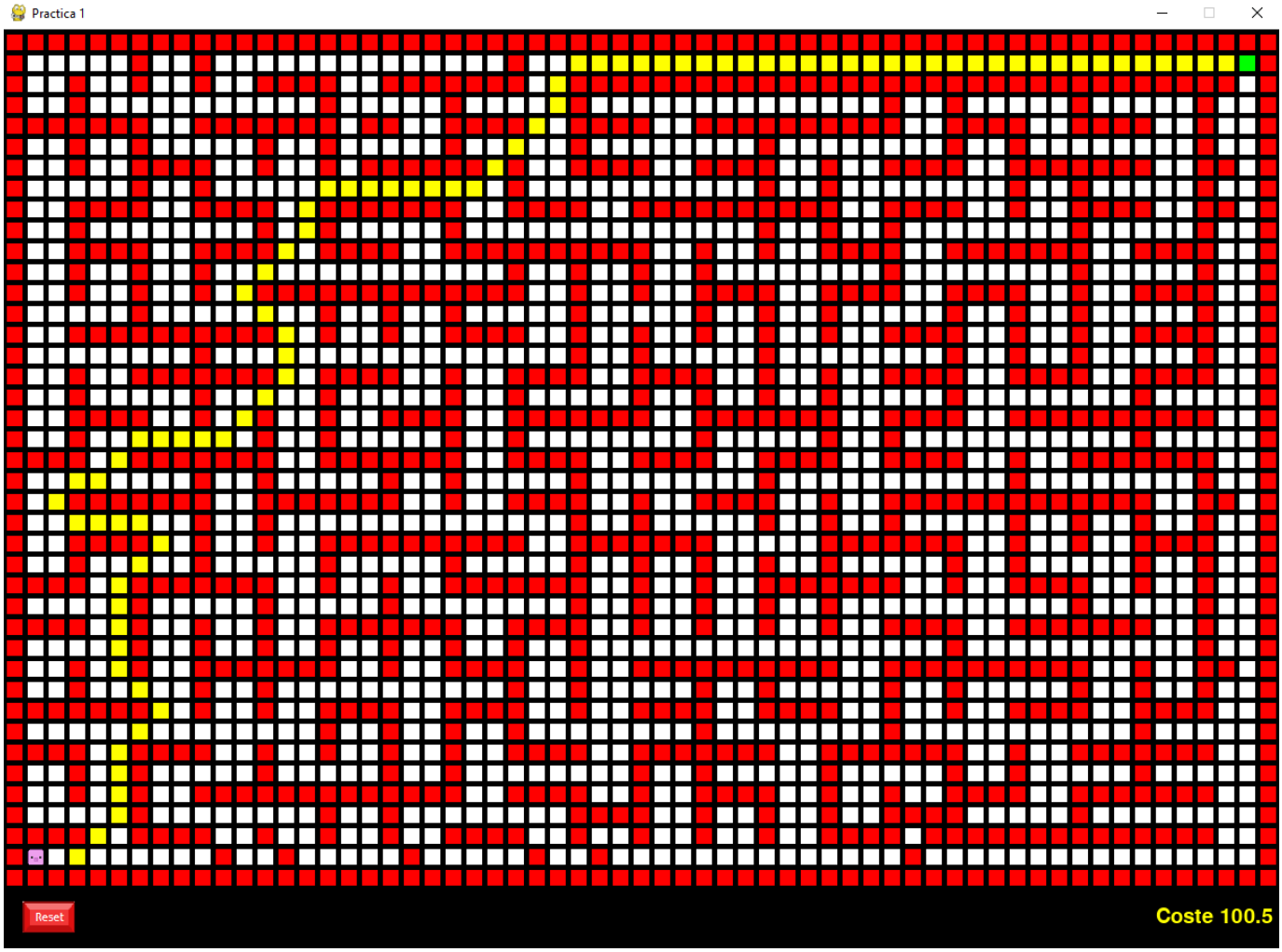

```
--- Nº iteración: 1011
     Destino encontrado!!! (x=59,y=1)

Orden de exploracion:
[[  -1   -1   -1 ...   -1   -1   -1]
 [  -1   -1   -1 ... 1010 1011   -1]
 [  -1   -1   -1 ...   -1   -1   -1]
 ...
 [  -1   -1   -1 ...   -1   -1   -1]
 [  -1    0    1 ...   -1   -1   -1]
 [  -1   -1   -1 ...   -1   -1   -1]]
```

Como podemos ver, ha calculado el camino de coste optimo 100.5, en tan solo 1011 iteraciones.

2.4 Chebyshov

Otra heuristica admisible es la distancia de Chebyshov, que se calcula de la siguiente forma:

Cheb(p1,p2) = max(|p1.x - p2.x|, |p1.y - p2.y|)



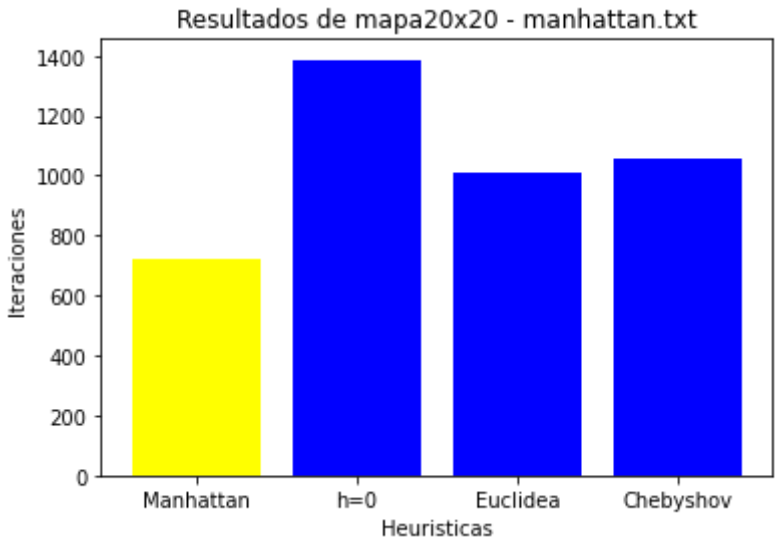

```
--- Nº iteración: 1058
     Destino encontrado!!! (x=59,y=1)

Orden de exploracion:
[[  -1   -1   -1 ...   -1   -1   -1]
 [  -1   -1   -1 ... 1057 1058   -1]
 [  -1   -1   -1 ...   -1   -1   -1]
 ...
 [  -1   -1   -1 ... 1018   -1   -1]
 [  -1    0    1 ...   -1   -1   -1]
 [  -1   -1   -1 ...   -1   -1   -1]]
```

Como podemos ver es una heuristica admisible, pero no es mejor que la heuristica de distancia euclidea, ya que para encontrar el camino de coste optimo 100.5 necesita 1058 iteraciones, 47 iteraciones más que la distancia euclidea.

2.5 Comparación gráfica

A continuación podemos ver una grafica que compara los resultados obtenidos para el mapa utilizado en este apartado.

```
import matplotlib.pyplot as plt

ejeX=["Manhattan","h=0","Euclidea","Chebyshov"]
ejeY=[721,1387,1011,1058]

plt.bar(ejeX, ejeY, color=['yellow','blue','blue','blue'])
plt.xlabel('Heuristicas')
plt.ylabel('Iteraciones')
plt.title('Resultados de mapa20x20 - manhattan.txt')
plt.show()
```



Como podemos ver, aunque Manhattan haya tenido menor número de iteraciones (menor numero de nodos explorados), lo he marcado en amarillo ya que el camino que ha encontrado no es el optimo. Luego podemos ver que la distancia euclidea ha sido la mas rapida para este mapa, y la h=0 la mas lenta.

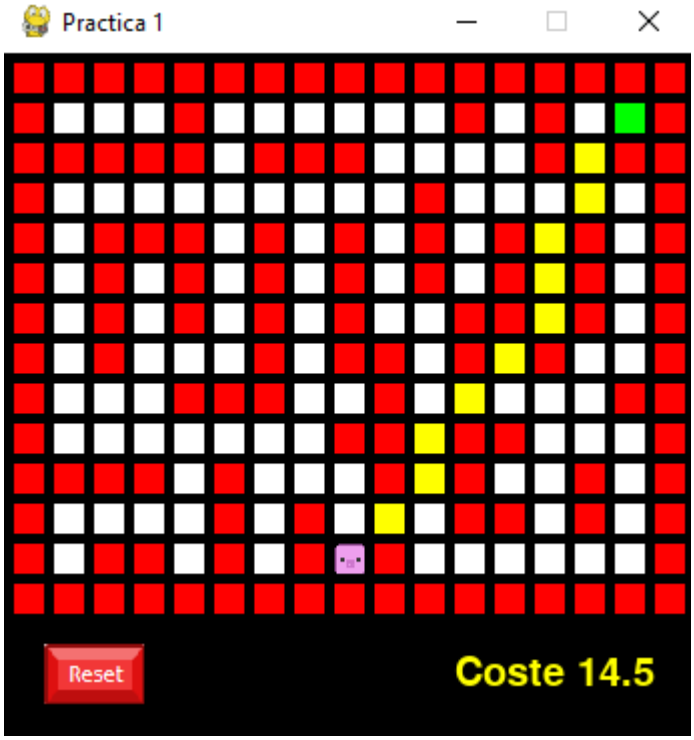
3. Mapas de prueba

Podemos verificar lo comprobado en el apartado anterior con los siguientes mapas de prueba:

3.1 mapa.txt

Mapa proporcionado por el profesorado.

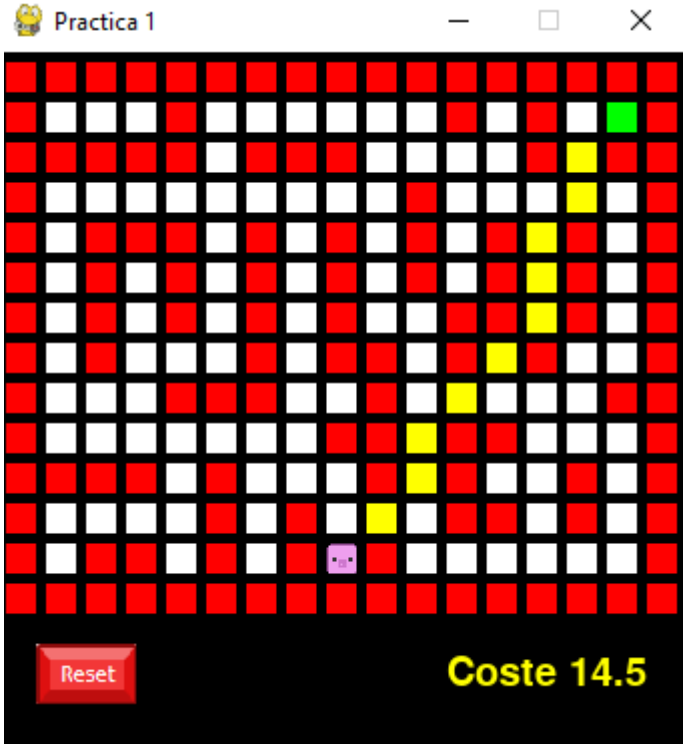
3.1.1 Manhattan




```
--- Nº iteración: 11
     Destino encontrado!!! (x=15,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 11 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 10 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  9 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  8 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  7 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  6 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  5 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  4 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  3 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1  2 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

A diferencia del apartado anterior, en este mapa si que encuentra el camino de coste optimo, y además lo encuentra en tan solo 11 iteraciones.

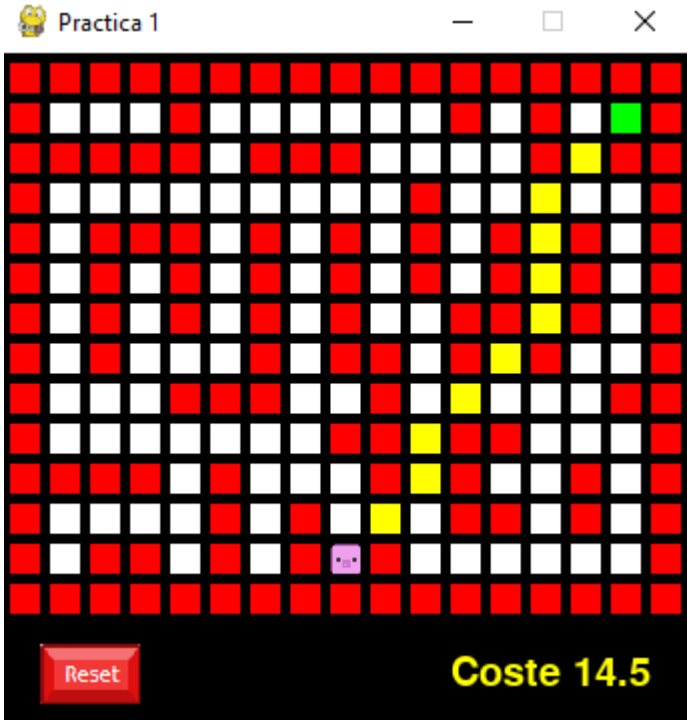
3.1.2 Euclidea




```
--- Nº iteración: 21
     Destino encontrado!!! (x=15,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 19 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 18 17 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 20 -1 14 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 16 -1 12 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 13 -1 -1 11 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 10 -1  9 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  7  6 15 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1  5 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1  4 -1  3 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1  2  1  8 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1  0 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

Para este mapa en concreto la distancia euclidea ya no es la heuristica mas eficaz, ya que esta vez encuentra el camino de coste optimo en 21 iteraciones.

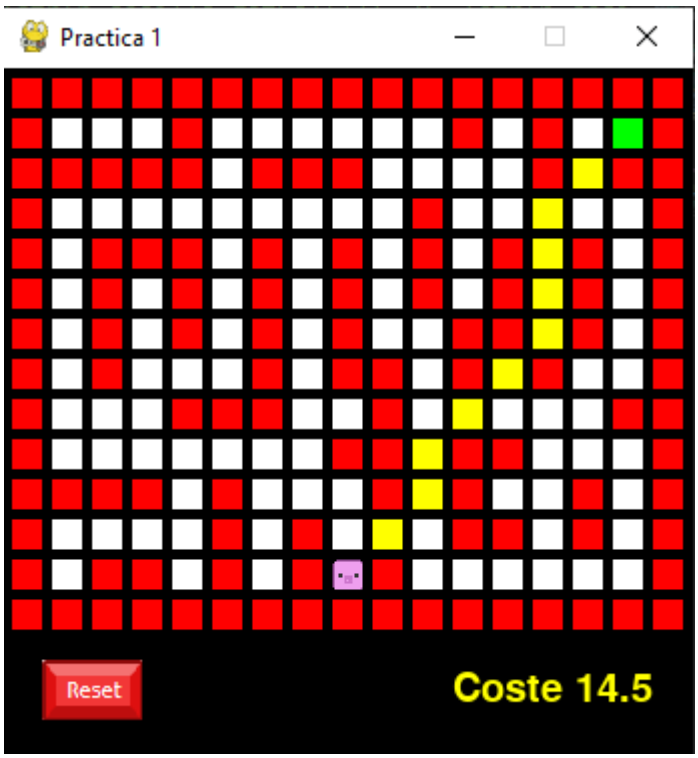
3.1.3 h=0




```
--- Nº iteración: 109
     Destino encontrado!!! (x=15,y=1)

Orden de exploracion:
[[ -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1]
 [ -1  -1  -1  -1  -1  94  97 105  98  95  99  -1 101  -1 106 109  -1]
 [ -1  -1  -1  -1  -1  82  -1  -1  -1  84  90  88  91  -1 100  -1  -1]
 [ -1 107 108 104  93  81  68  64  69  76  -1  77  79  87  89 102  -1]
 [ -1 103  -1  -1  -1  83  -1  52  -1  61  -1  62  -1  75  -1  96  -1]
 [ -1  92  -1  78  -1  85  -1  38  -1  47  -1  48  -1  59  -1  86  -1]
 [ -1  80  -1  63  -1  72  -1  28  -1  34  32  -1  -1  44  -1  73  -1]
 [ -1  65  -1  49  53  70  -1  21  -1  -1  24  -1  29  -1  55  66  -1]
 [ -1  58  45  33  -1  -1  -1  14  15  -1  19  20  27  37  51  -1  -1]
 [ -1  57  42  30  22  16  10   8  -1  -1  13  -1  -1  41  54  67  -1]
 [ -1  -1  -1  -1  25  -1   9   4   3  -1   6  -1  40  36  -1  60  -1]
 [ -1  71  56  39  35  -1  11  -1   1   2   5  -1  -1  26  -1  46  -1]
 [ -1  74  -1  -1  50  -1  17  -1   0  -1   7  12  18  23  31  43  -1]
 [ -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1]]
```

La heuristica h=0 siempre es admisible, pero ineficaz, al explorar una cantidad de nodos muy elevada (en este caso 109).

3.1.4 Chebyshov




```
--- Nº iteración: 36
     Destino encontrado!!! (x=15,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 36 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 35 29 26 30 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 25 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 16 -1 24 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 33 -1 19 11 -1 -1 23 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 20 -1 -1 10 -1 18 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 14  7 -1  9 15 21 34 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 17  5 -1 -1  8 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 13  4  2 -1  6 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 28 -1  1  3 12 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1  0 -1 27 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

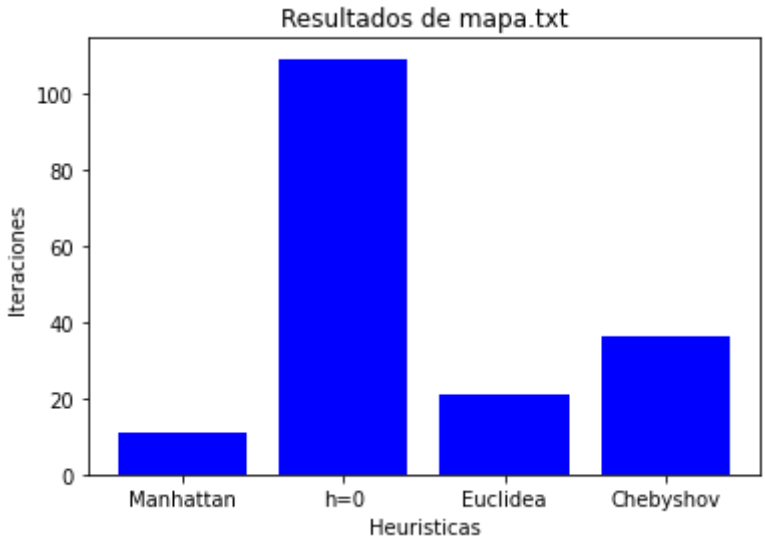
En este caso la heuristica de chebyshov ha encontrado el camino de coste optimo 14.5 en 36 iteraciones.

3.1.5 Comparación gráfica

A continuación podemos ver una grafica que compara los resultados obtenidos para el mapa utilizado en este apartado.

```
import matplotlib.pyplot as plt
```

```
ejeX=["Manhattan","h=0","Euclidea","Chebyshov"]
ejeY=[11,109,21,36]

plt.bar(ejeX, ejeY, color=['blue','blue','blue','blue'])
plt.xlabel('Heuristicas')
plt.ylabel('Iteraciones')
plt.title('Resultados de mapa.txt')
plt.show()
```

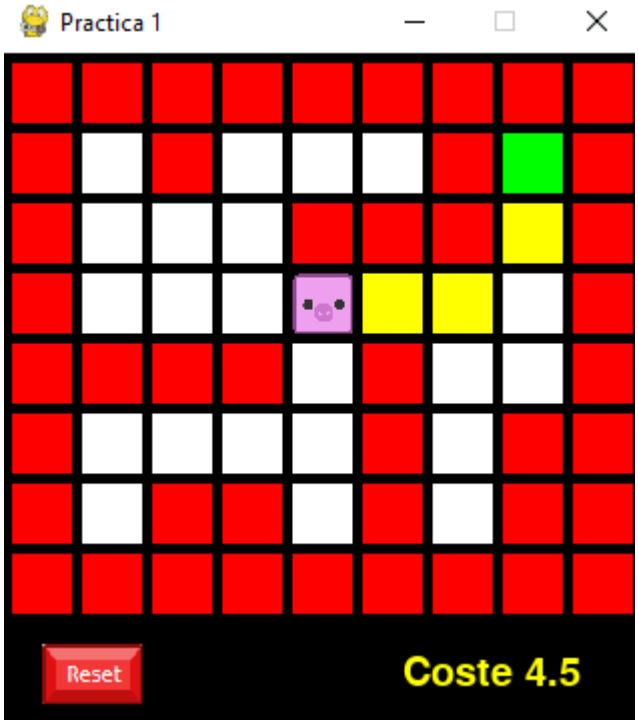


Podemos ver que en este caso la distancia Manhattan si que ha calculado el camino optimo, y además ha sido la mas rápida, con una cantidad de 11 iteraciones (nodos explorados), mientras que la h=0 la mas lenta, con 109 iteraciones.

3.2 mapa2.txt

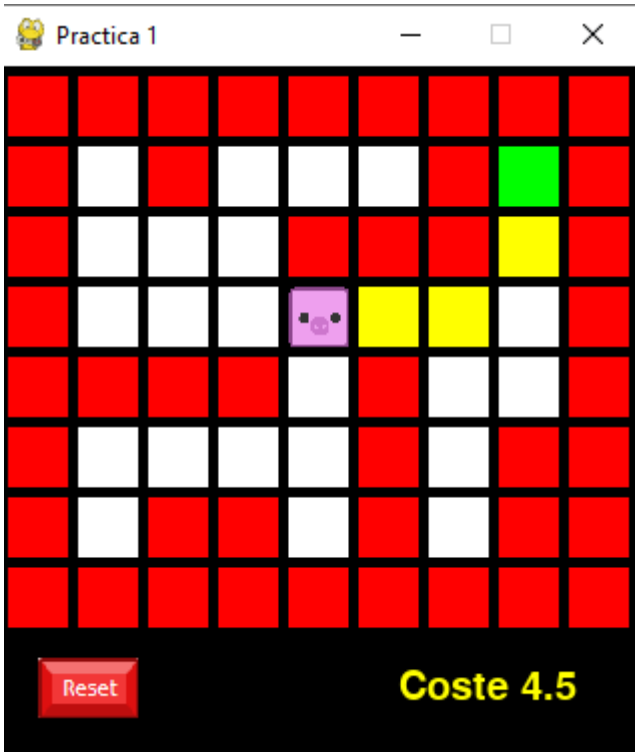
Mapa proporcionado por el profesorado.

3.2.1 Manhattan



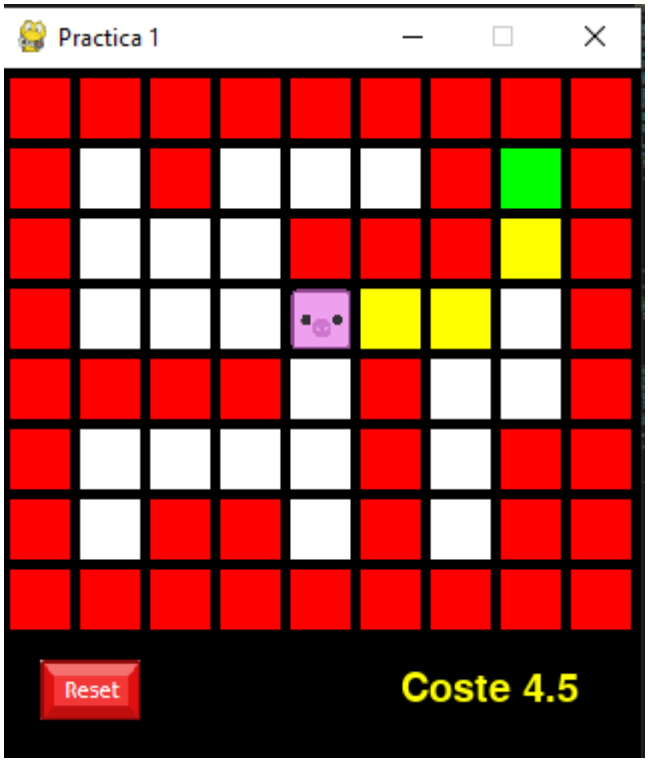

```
--- Nº iteración: 4
     Destino encontrado!!! (x=7,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1  4 -1]
 [-1 -1 -1 -1 -1 -1 -1  3 -1]
 [-1 -1 -1 -1  0  1  2 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.2.2 Euclidea



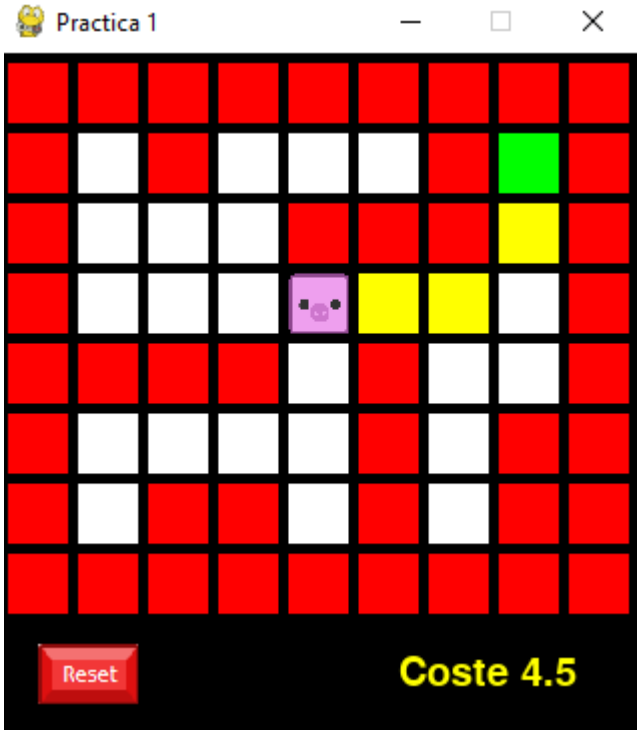

```
--- Nº iteración: 4
     Destino encontrado!!! (x=7,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1  4 -1]
 [-1 -1 -1 -1 -1 -1 -1  3 -1]
 [-1 -1 -1 -1  0  1  2 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.2.3 h=0




```
--- Nº iteración: 23
     Destino encontrado!!! (x=7,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 21 -1 11 12 22 -1 23 -1]
 [-1 16  8  4 -1 -1 -1 17 -1]
 [-1 13  5  1  0  2  6 14 -1]
 [-1 -1 -1 -1  3 -1  9 18 -1]
 [-1 -1 20 10  7 -1 19 -1 -1]
 [-1 -1 -1 -1 15 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.2.4 Chebyshov

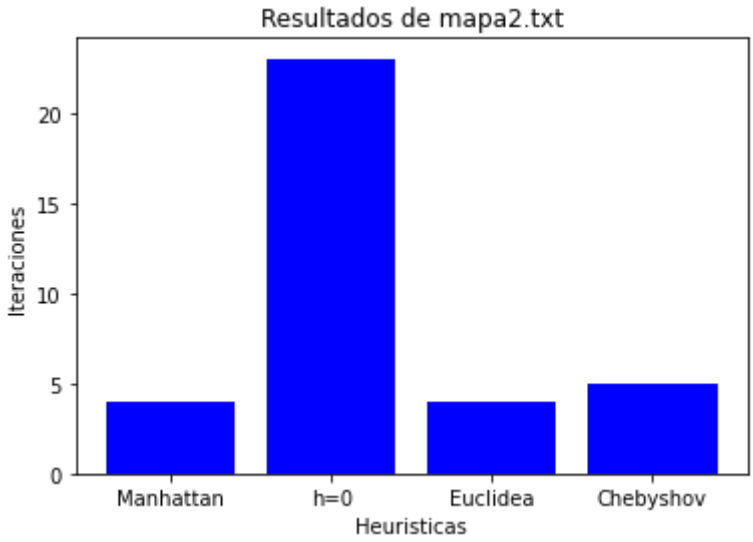



```
--- Nº iteración: 5
    Destino encontrado!!! (x=7,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1  5 -1]
 [-1 -1 -1 -1 -1 -1 -1  4 -1]
 [-1 -1 -1 -1  0  1  3 -1 -1]
 [-1 -1 -1 -1  2 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.2.5 Comparación gráfica

A continuación podemos ver una grafica que compara los resultados obtenidos para el mapa utilizado en este apartado.

```
import matplotlib.pyplot as plt

ejeX=["Manhattan","h=0","Euclidea","Chebyshov"]
ejeY=[4,23,4,5]

plt.bar(ejeX, ejeY, color=['blue','blue','blue','blue'])
plt.xlabel('Heuristicas')
plt.ylabel('Iteraciones')
plt.title('Resultados de mapa2.txt')
plt.show()
```

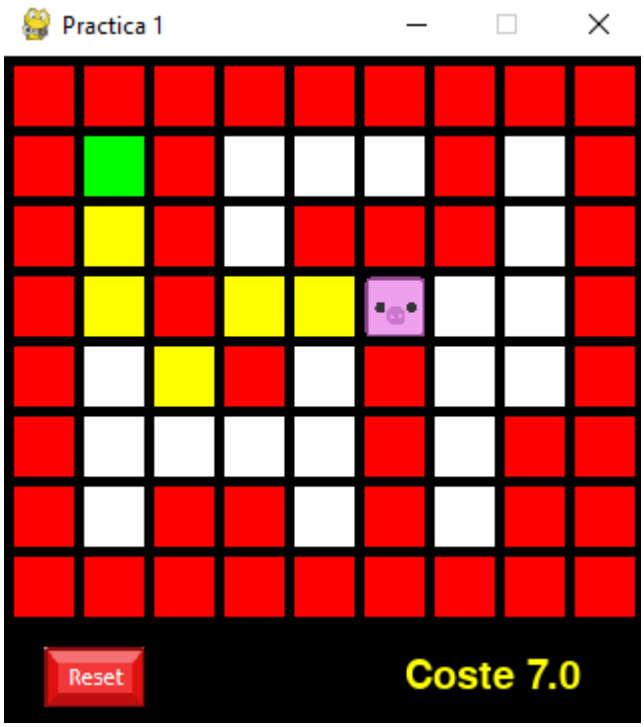


En este caso tanto manhattan como la euclidea han obtenido el mismo resultado de 4 iteraciones, mientras que la h=0 ha obtenido 23.

3.3 mapa3.txt

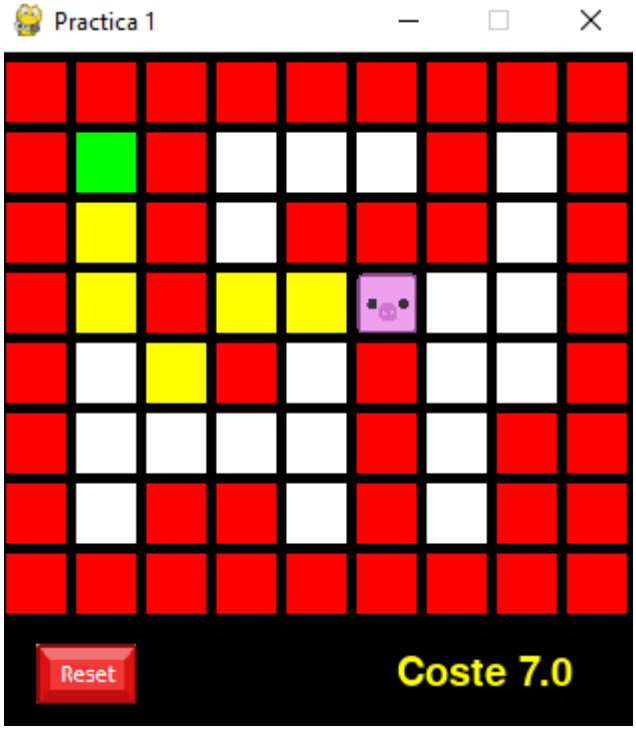
Mapa proporcionado por el profesorado.

3.3.1 Manhattan



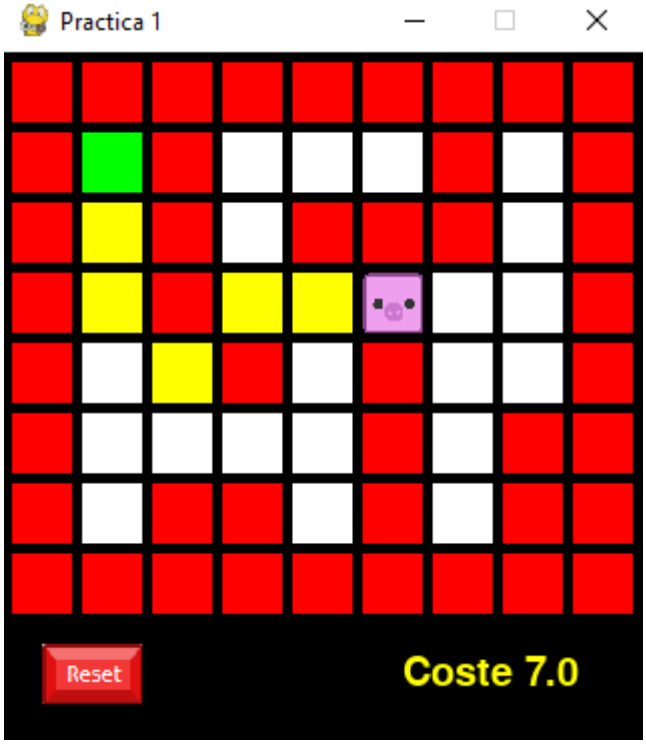
```
--- Nº iteración: 10
     Destino encontrado!!! (x=1,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 10 -1  3  5 -1 -1 -1 -1]
 [-1  9 -1  2 -1 -1 -1 -1 -1]
 [-1  8 -1  4  1  0 -1 -1 -1]
 [-1 -1  7 -1  6 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.3.2 Euclidea



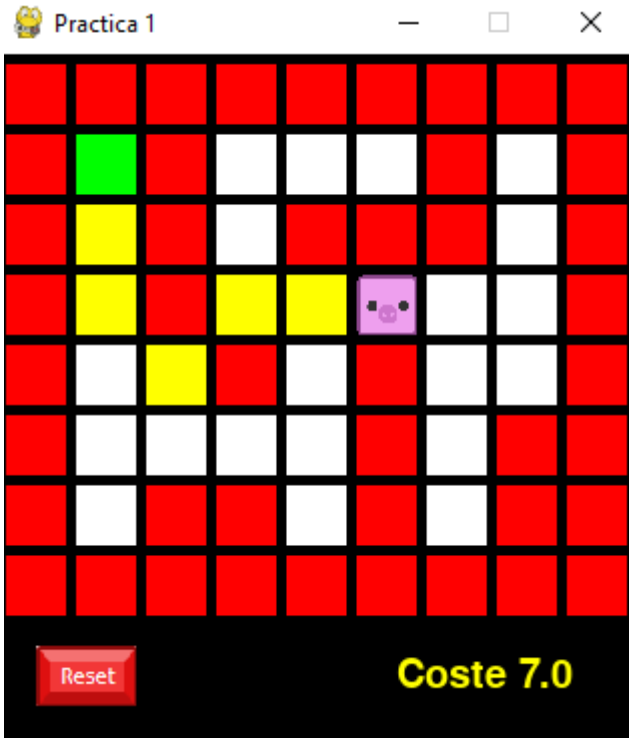

```
--- Nº iteración: 11
     Destino encontrado!!! (x=1,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 11 -1  4  8 -1 -1 -1 -1]
 [-1 10 -1  2 -1 -1 -1 -1 -1]
 [-1  9 -1  3  1  0  6 -1 -1]
 [-1 -1  7 -1  5 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.3.3 h=0




```
--- Nº iteración: 26
     Destino encontrado!!! (x=1,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 26 -1 14 18 23 -1 15 -1]
 [-1 25 -1  7 -1 -1 -1  8 -1]
 [-1 21 -1  5  1  0  2  6 -1]
 [-1 20 13 -1  3 -1  4  9 -1]
 [-1 22 19 12 10 -1 11 -1 -1]
 [-1 24 -1 -1 16 -1 17 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.3.4 Chebyshov

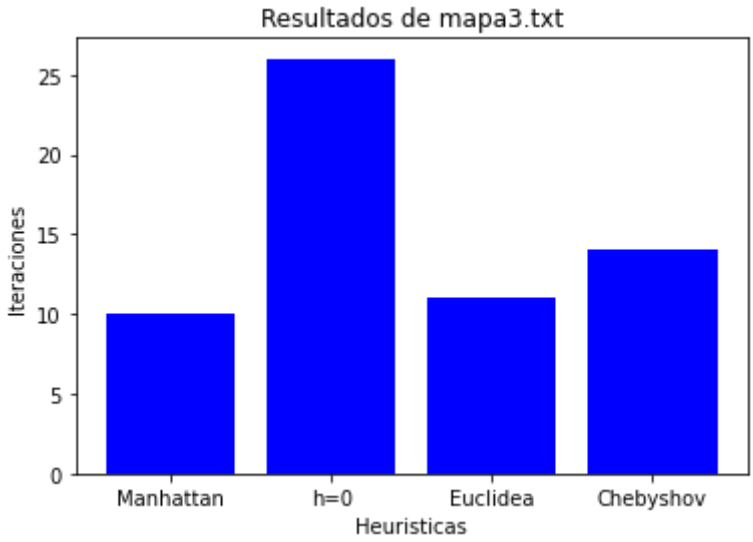



```
--- Nº iteración: 14
     Destino encontrado!!! (x=1,y=1)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 14 -1  5 11 -1 -1 -1 -1]
 [-1 13 -1  4 -1 -1 -1 -1 -1]
 [-1 12 -1  2  1  0  6 -1 -1]
 [-1 -1  8 -1  3 -1  7 -1 -1]
 [-1 -1 -1 10  9 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.3.5 Comparación gráfica

A continuación podemos ver una grafica que compara los resultados obtenidos para el mapa utilizado en este apartado.

```
import matplotlib.pyplot as plt

ejeX=["Manhattan","h=0","Euclidea","Chebyshov"]
ejeY=[10,26,11,14]

plt.bar(ejeX, ejeY, color=['blue','blue','blue','blue'])
plt.xlabel('Heuristicas')
plt.ylabel('Iteraciones')
plt.title('Resultados de mapa3.txt')
plt.show()
```

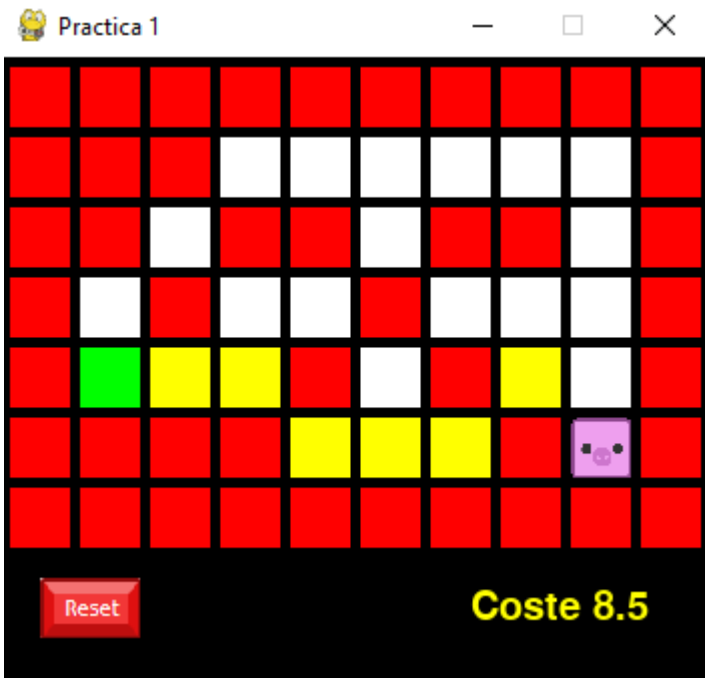


En este caso Manhattan ha sido mas rapido al calcular el camino con 10 iteraciones.

3.4 mapa3x3.txt

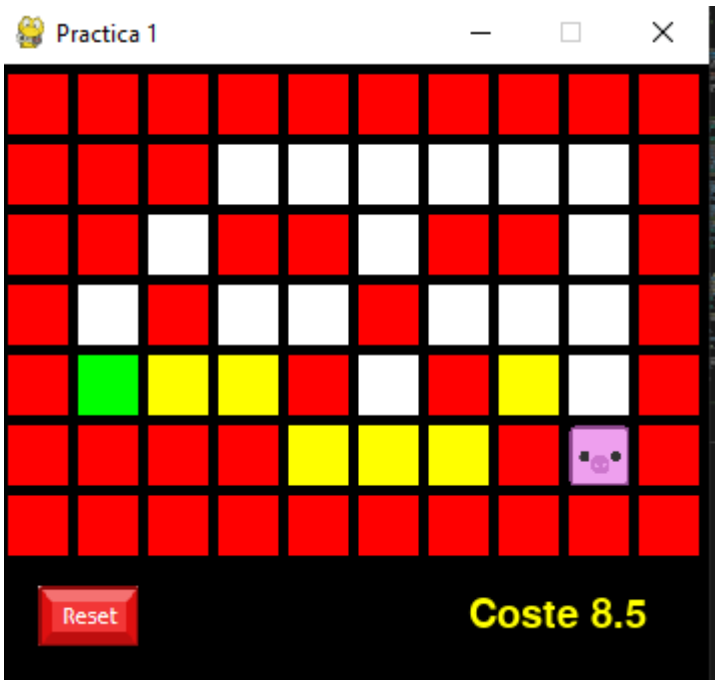
Mapa creado por mi.

3.4.1 Manhattan



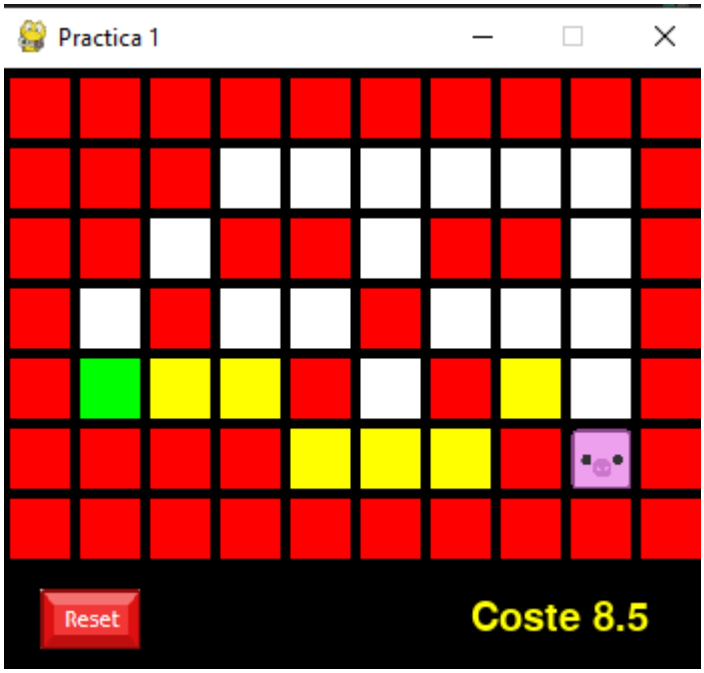
```
--- Nº iteración: 10
     Destino encontrado!!! (x=1,y=4)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1  3 -1 -1 -1]
 [-1 10  9  8 -1  4 -1  1  2 -1]
 [-1 -1 -1 -1  7  6  5 -1  0 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.4.2 Euclidea



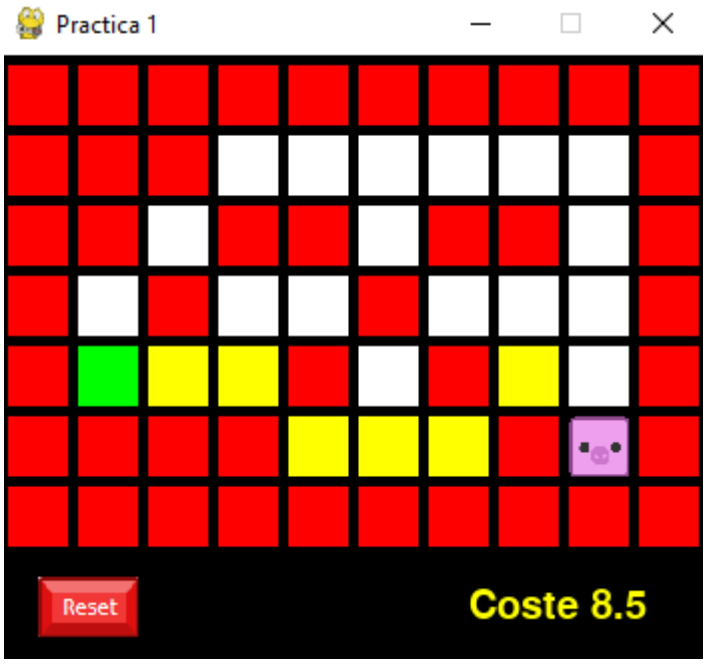

```
--- Nº iteración: 10
     Destino encontrado!!! (x=1,y=4)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1  3 -1 -1 -1]
 [-1 10  9  8 -1  7 -1  1  2 -1]
 [-1 -1 -1 -1  6  5  4 -1  0 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.4.3 h=0




```
--- Nº iteración: 23
     Destino encontrado!!! (x=1,y=4)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 19 16 14 15 12  9 -1]
 [-1 -1 22 -1 -1 10 -1 -1  7 -1]
 [-1 -1 -1 20 17 -1  5  4  3 -1]
 [-1 23 21 18 -1 11 -1  2  1 -1]
 [-1 -1 -1 -1 13  8  6 -1  0 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.4.4 Chebyshov

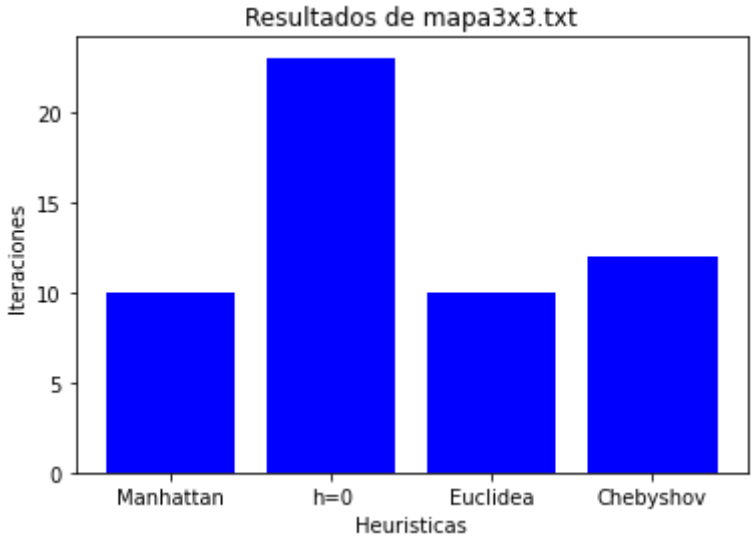


```
--- Nº iteración: 12
    Destino encontrado!!! (x=1,y=4)

Orden de exploracion:
[[-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1  8 -1 -1 -1 -1]
 [-1 -1 -1 -1 -1 -1  3  7 -1 -1]
 [-1 12 11 10 -1  9 -1  1  2 -1]
 [-1 -1 -1 -1  6  5  4 -1  0 -1]
 [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]]
```

3.4.5 Comparación gráfica

A continuación podemos ver una grafica que compara los resultados obtenidos para el mapa utilizado en este apartado.

```
import matplotlib.pyplot as plt

ejeX=["Manhattan","h=0","Euclidea","Chebyshov"]
ejeY=[10,23,10,12]

plt.bar(ejeX, ejeY, color=['blue','blue','blue','blue'])
plt.xlabel('Heuristicas')
plt.ylabel('Iteraciones')
plt.title('Resultados de mapa3x3.txt')
plt.show()
```

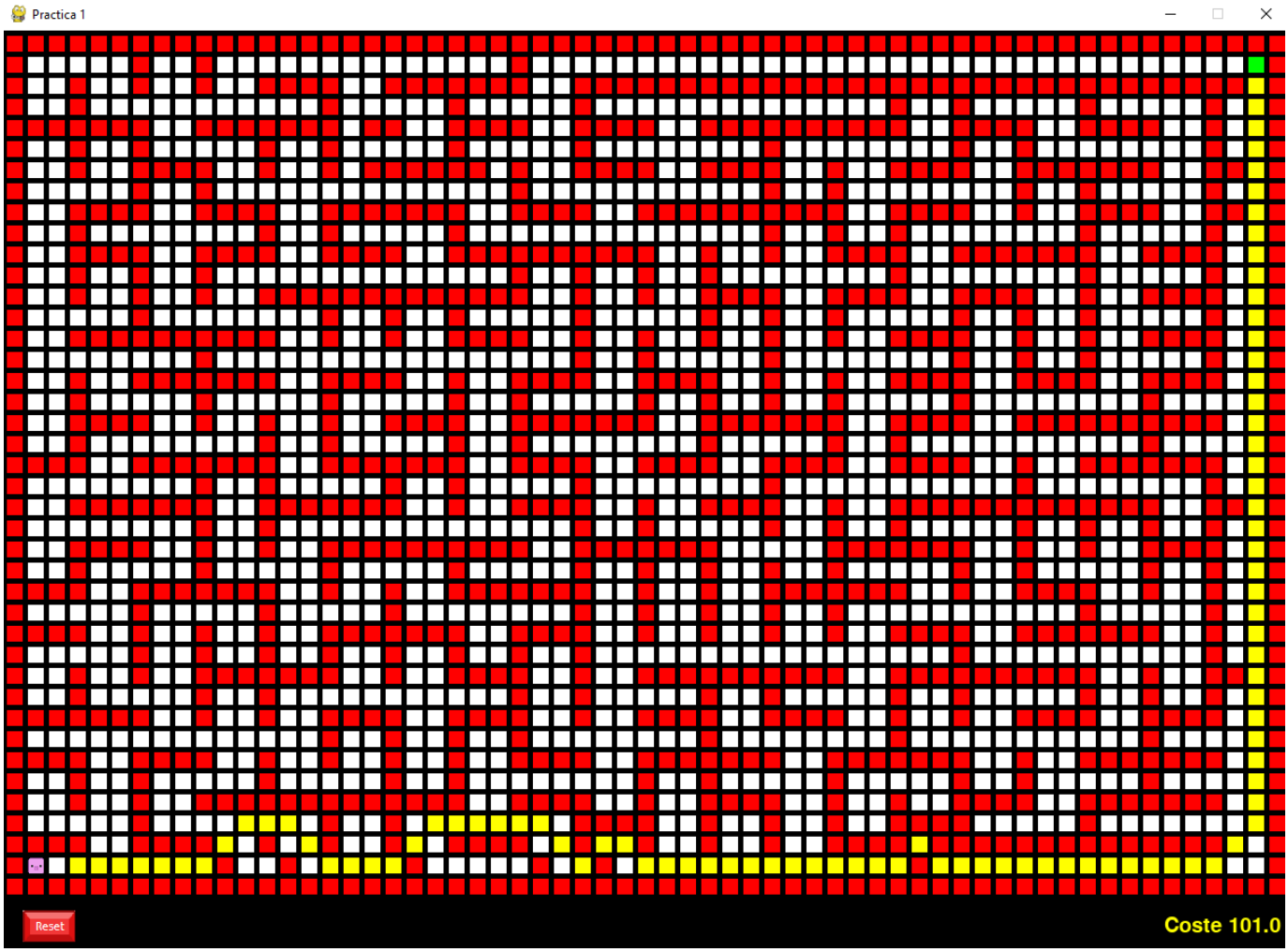


En este caso tanto Manhattan como la euclidea han calculado el camino optimo con 10 iteraciones.

3.5 mapa20x20 - manhattan.txt

Mapa creado por mi.

Este mapa se ha utilizado en el apartado **2. comparacion de heuristicas**, por lo que seria redundante incluir las pruebas otra vez en este apartado. A continuacion se muestra el mapa:



✓ 0s completed at 6:07 PM