Prácticas

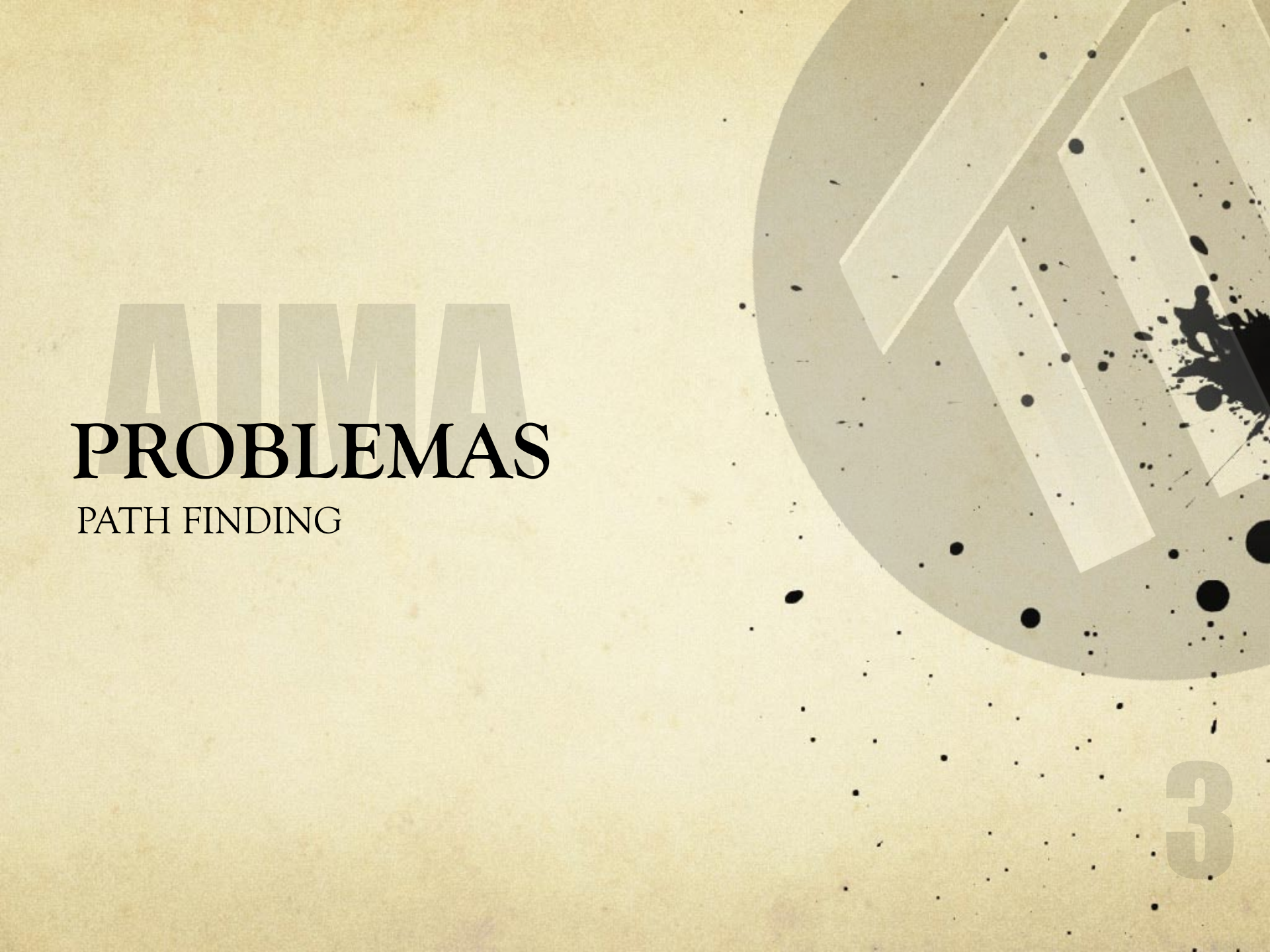
Sistemas Inteligentes I

Sesión 2. Búsqueda

José A. Montenegro Montes
monte@lcc.uma.es

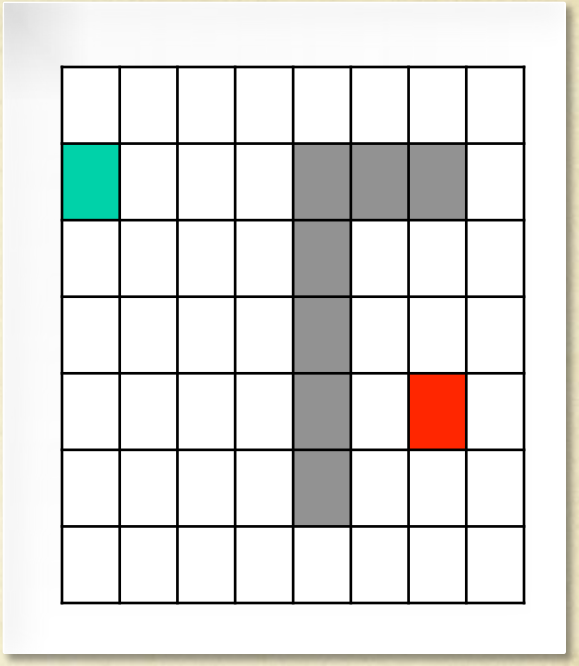
Resumen

- PATH FINDING
- Práctica EntregarRouteApp
- Algoritmo A*

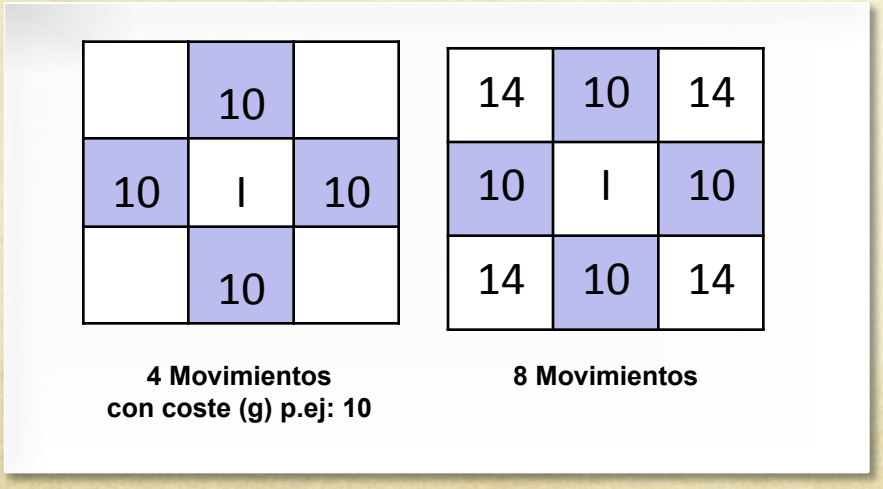

AIMA
PROBLEMAS
PATH FINDING
3

Path Finding

• Camino desde un nodo origen (verde) a un nodo destino (rojo), evitando obstáculos (gris), sobre un mapa que está dividido en cuadrículas.

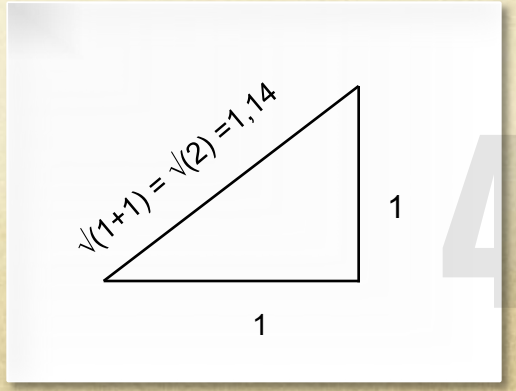


• Los movimientos permitidos en cada paso desde estado actual (nodo I), dependen del problema:



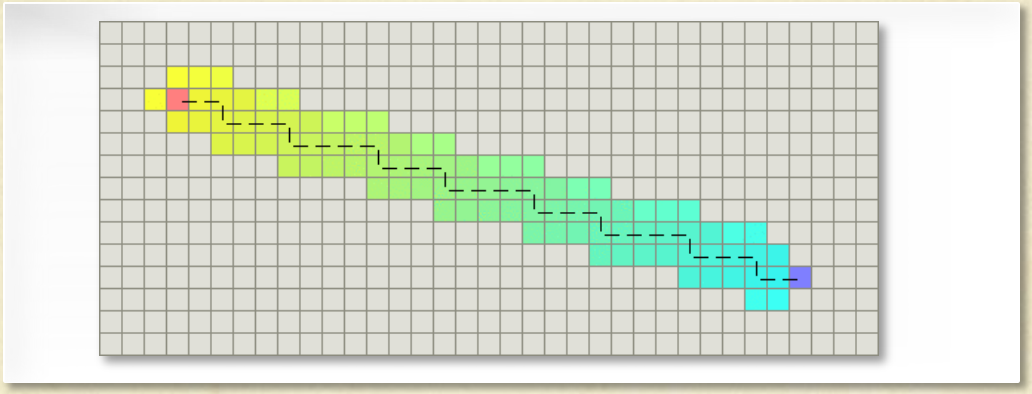
4 Movimientos con coste (g) p.ej: 10

8 Movimientos

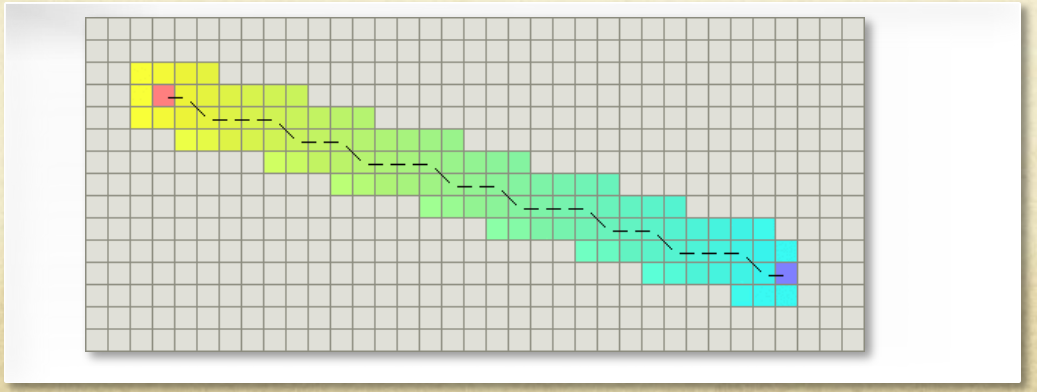


Path Finding Heurísticas

- **Distancia Manhattan**: h(n) = (abs(n.x-goal.x) + abs(n.y-goal.y)).

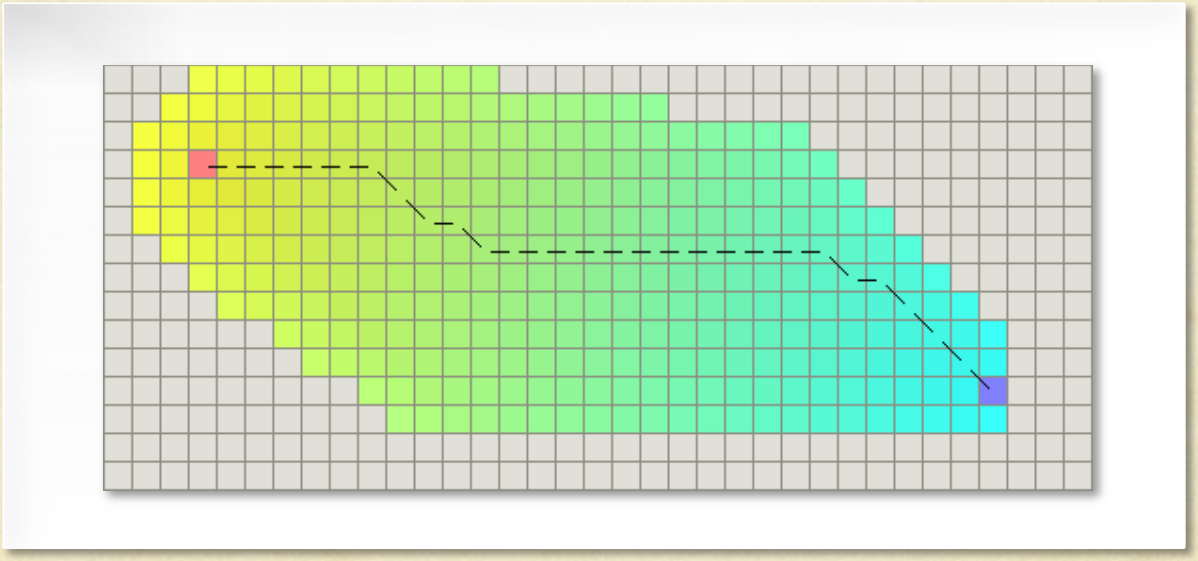


- **Distancia Diagonal**: h(n) = max(abs(n.x-goal.x), abs(n.y-goal.y))



Path Finding Heurísticas

- **Distancia Euclídea:** h(n) = sqrt((n.x-goal.x)^2 + (n.y-goal.y)^2)



http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

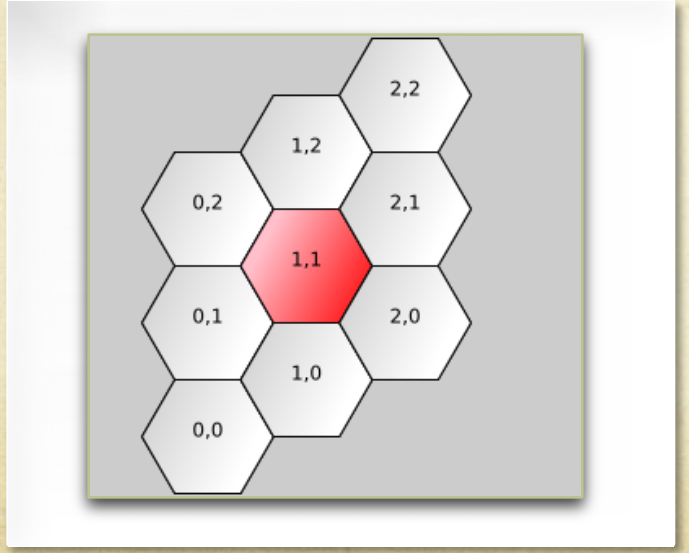
Otros Casos

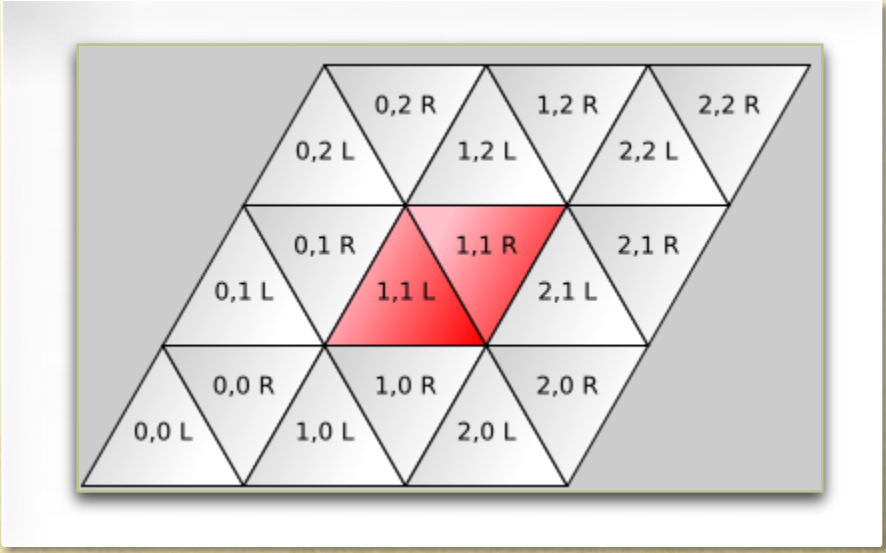
- Animación flash casillas hexagonales:

  http://www-cs-students.stanford.edu/~amitp/game-programming/a-star-flash/main-hexagon.swf

- Animación flash casillas triangulares:

  http://www-cs-students.stanford.edu/~amitp/game-programming/a-star-flash/main-triangle.swf

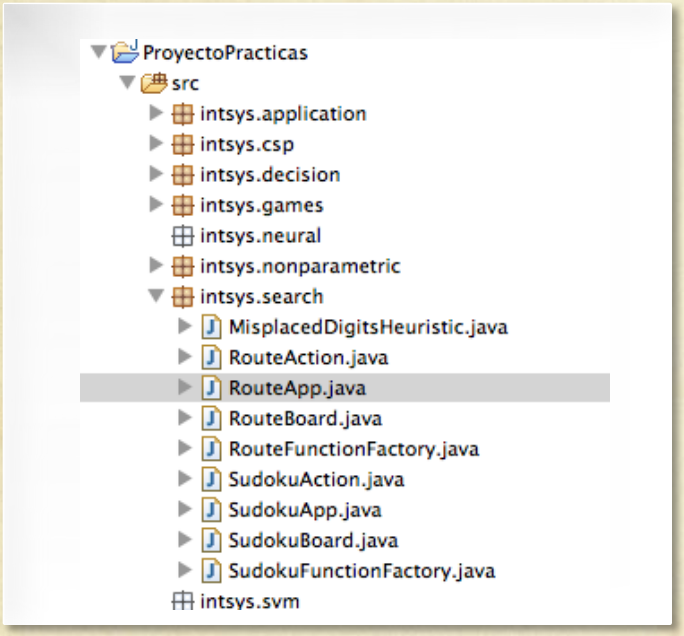


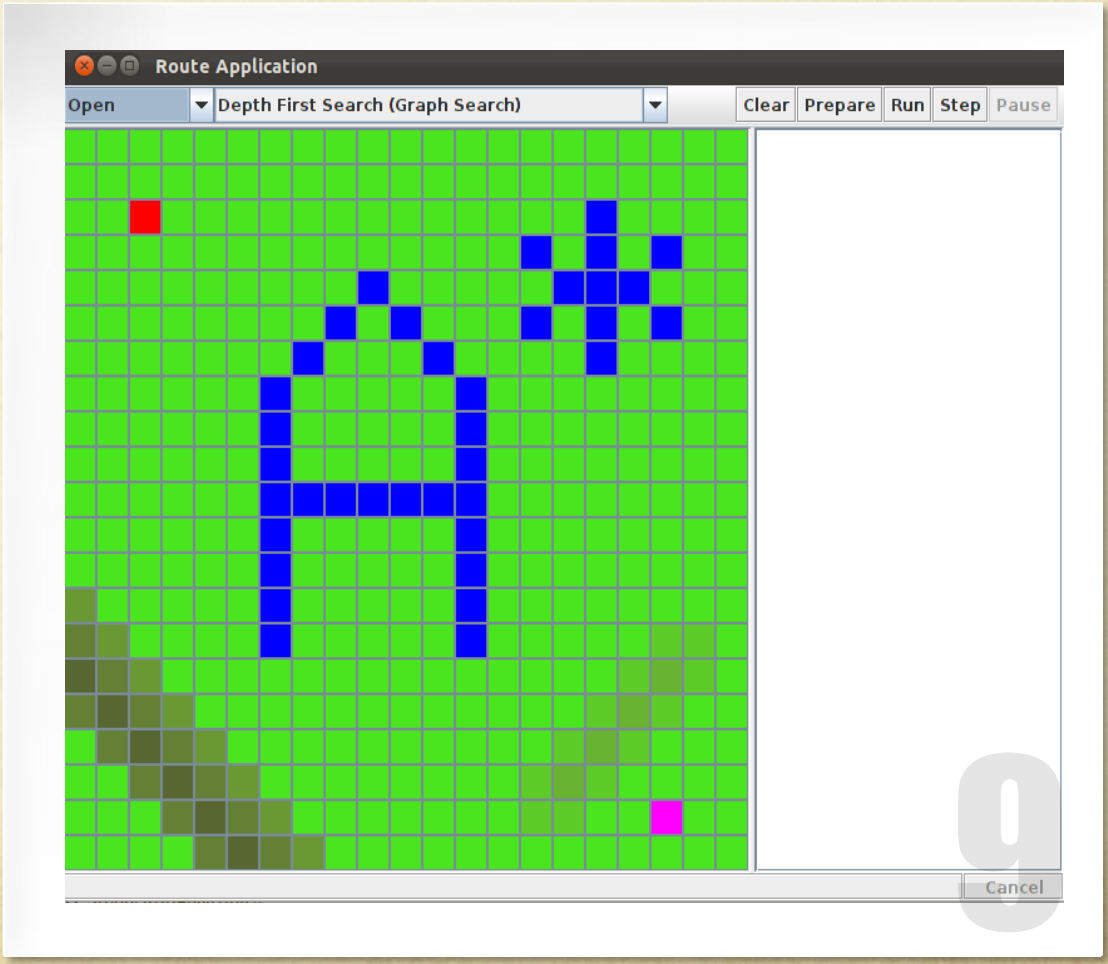


Práctica Entregar

RouteApp

Intsys.search.RouteApp





Observar coste de las casillas, además de los obstáculos.

Ejercicio 1. RouteApp

Implemente heurística Distancia Euclídea

La clase RouteHeuristicFunc en *RouteFunctionFactory implemente el método*

public double h(Object state)

h(n) = sqrt((n.x-goal.x)^2 + (n.y-goal.y)^2)

Ejercicio 2. RouteApp

Añada heurística Distancia Manhattan y Distancia Diagonal al programa.

1. En la clase RouteApp añada dos entradas

addSearchAlgorithm("A search (Manhattan heuristic)",*
new AStarSearch(new GraphSearch(),
RouteFunctionFactory.*getMHeuristicFunction()));*

addSearchAlgorithm("A search (Distancia Diagonal heuristic)",*
new AStarSearch(new GraphSearch(),
RouteFunctionFactory.*getDHeuristicFunction()));*

2. Implemente los métodos:
RouteFunctionFactory.*getMHeuristicFunction()));*
RouteFunctionFactory.*getDHeuristicFunction()));*

Apéndice

Algoritmo A*

Algoritmo A*

```
cerrado:= {},          abiertos := {inicio}
g [inicio] := 0,       h [inicio] := heuristica(inicio, objetivo)
f [inicio] := g [inicio] + h [inicio].

MIENTRAS abiertos no es vacio
    actual:= nodo menor valor f
    SI actual = objetivo devolver ACIERTO
    Eliminar actual de abiertos y  Añadir actual a cerrados

        PARA CADA vecido (actual)
            SI  vecino esta en cerrado CONTINUAR
            g := g [actual] + distancia(actual,vecino)

            SI vecino no esta en abierto
                añadir vecino abierto
                h [vecino] := heuristica(vecino, objetivo)
                mejorado := verdad
            SI NO SI g < g  [vecino]  mejorado := VERDAD
                SINO                  mejorado := VERDAD

        SI mejorado == VERDAD
            padre[vecino] := actual
            g [vecino] := g
            score[vecino] := g[vecino] + h [vecino]

devolver FALLO
```

Inicialización

Expansión nodos

Algoritmo A*

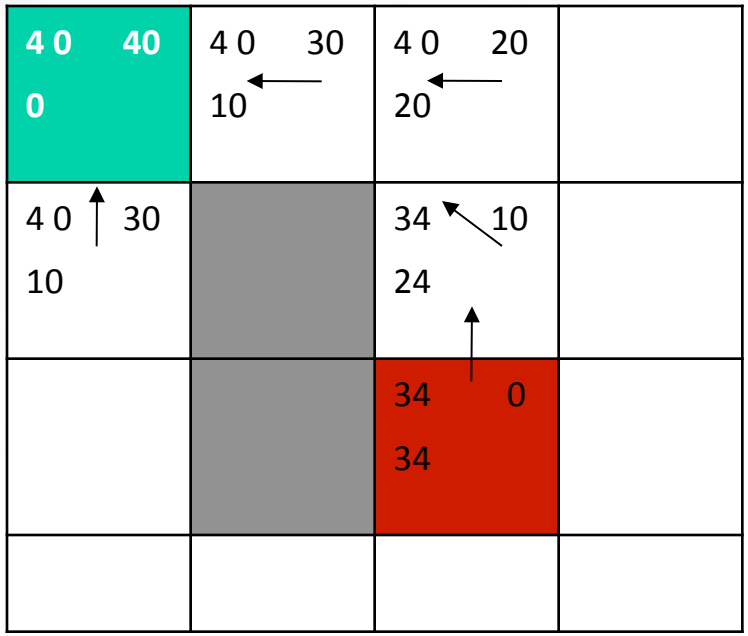
```
OPEN = priority queue containing START
CLOSED = empty set
while lowest rank in OPEN is not the GOAL:
  current = remove lowest rank item from OPEN
  add current to CLOSED
  for neighbors of current:
    cost = g(current) + movementcost(current, neighbor)
    if neighbor in OPEN and cost less than g(neighbor):
      remove neighbor from OPEN, because new path is better
    if neighbor in CLOSED and cost less than g(neighbor): **
      remove neighbor from CLOSED
    if neighbor not in OPEN and neighbor not in CLOSED:
      set g(neighbor) to cost
      add neighbor to OPEN
      set priority queue rank to g(neighbor) + h(neighbor)
      set neighbor's parent to current

reconstruct reverse path from goal to start
by following parent pointers
```

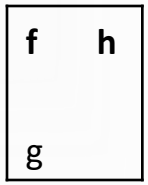
Otra versión del Algoritmo

Cambia condiciones

Ejemplo Algoritmo A*



Cada Casilla



Nodo Padre

Sistemas Inteligentes

José A. Montenegro Montes

monte@lcc.uma.es