UNIVERSIDAD DE GRANADA

DECSAI

Técnicas de los Si es

Grado en Informática

Cambiar nombre al paquete por src_Carm

Curso 2020-21. Práctica 1

Estrategias Deliberativas

Jesús Giráldez Crú y Pablo Mesejo Santiago

Departamento de Ciencias de la Computación e Inteligencia Artificial

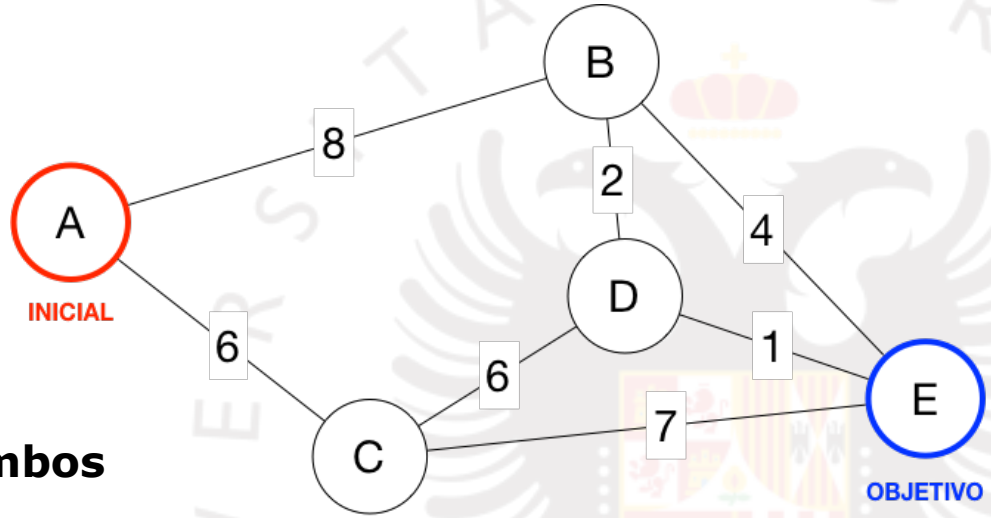
http://decsai.ugr.es

El agente usa el **comportamiento deliberativo simple** para calcular la **ruta óptima entre dos puntos** del mapa (por ejemplo, entre dos gemas)

posible heurística (h(n)) distancia manhatt

Para este comportamiento, se implementará el **algoritmo A*:**

- **Input:**
  - **Un grafo**
  - **Un nodo inicial**
  - **Un nodo objetivo**
- **Output:**
  - **La ruta óptima entre ambos**



	A	B	C	D	E
h(n)	10	4	5	1	0

Algoritmo A*

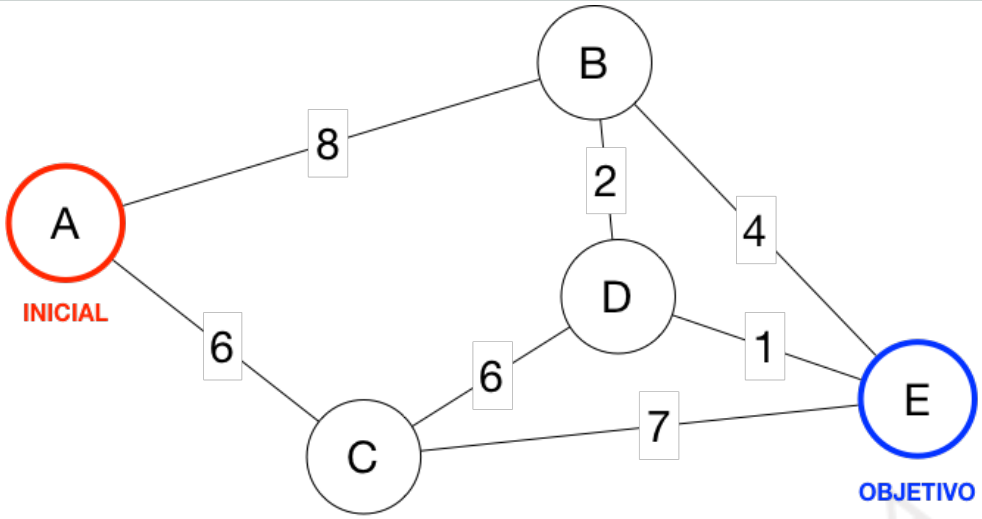
- *abiertos* = [*inicial*] **# nodos pendientes de visitar**
- cerrados = [] **# nodos ya visitados**
- **while True** :
  - *nodo* = ***mejorCandidato***(*abiertos*) **# mejor f(n)**
  - **if** *nodo* == *objetivo*:
    - **break # se expande el nodo objetivo, fin.**
  - *abiertos*.remove(*nodo*)
  - **foreach** *sucesor* **in** ***expandir***(*nodo*) : **# vecinos del *nodo***
    - **if** *sucesor* == *nodo*.parent **:**
      - **Pass *# nodo padre ya visitado***
    - **if** *cerrados*.contains(*sucesor*)
      and ***mejorCaminoA***(*sucesor*): **# menor g(n)**
      - *cerrados*.remove(*sucesor*)
      - *abiertos*.add(*sucesor*) ***# actualizar g(n)***
    - **elif not** *cerrados*.contains(*sucesor*) **and**
      not *abiertos*.contains(*sucesor*) :
      - *abiertos*.add(*sucesor*) ***# nodo no visitado, añadir***
    - **elif** *abiertos*.contains(*sucesor*) **and** **mejorCaminoA**(*sucesor*):
      - *abiertos*.update(*sucesor*) ***# actualizar g(n)***

g + h

en nuestro caso no se puede generar el hijo resultado de rea

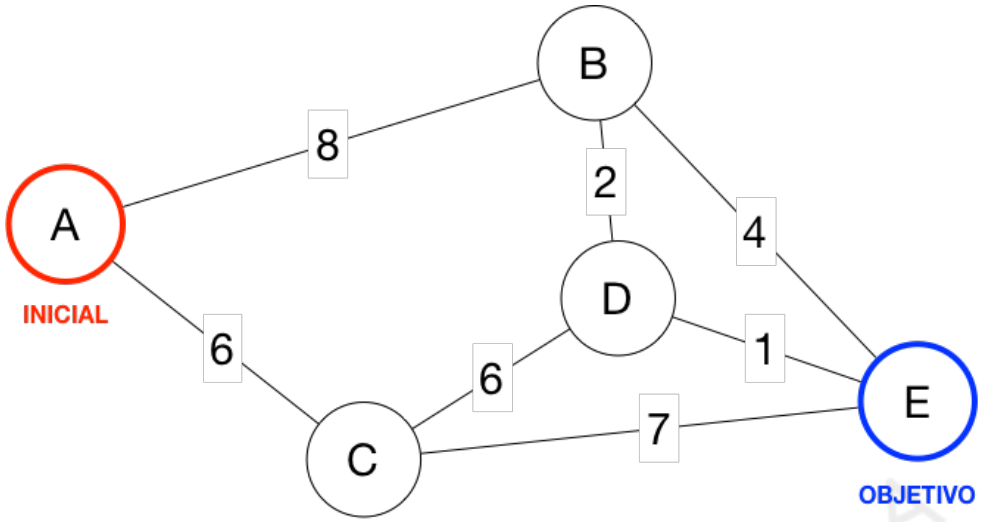
no hacer, nuestra heurística e

Creo que al generar los hijos, e



	A	B	C	D	E
h(n)	10	4	5	1	0

It.	Expandir	Sucesores	Abiertos	Cerrados
0			A(0+10)	-
1	A	B(8+4), C(6+5)	B(8+4), C(6+5)	A
2	C	D(12+1), E(13+0)	B(8+4), D(12+1), E(13+0)	A,C
3	B	D(10+1), E(12+0)	~~D(12+1), E(13+0)~~, D(10+1), E(12+0)	A,C,B
4	D	~~C(16+5)~~, E(11+0)	~~E(12+0)~~, E(11+0)	A,C,B,D
5	E	Fin		A,C,B,D,E



	A	B	C	D	E
h(n)	10	4	5	1	0

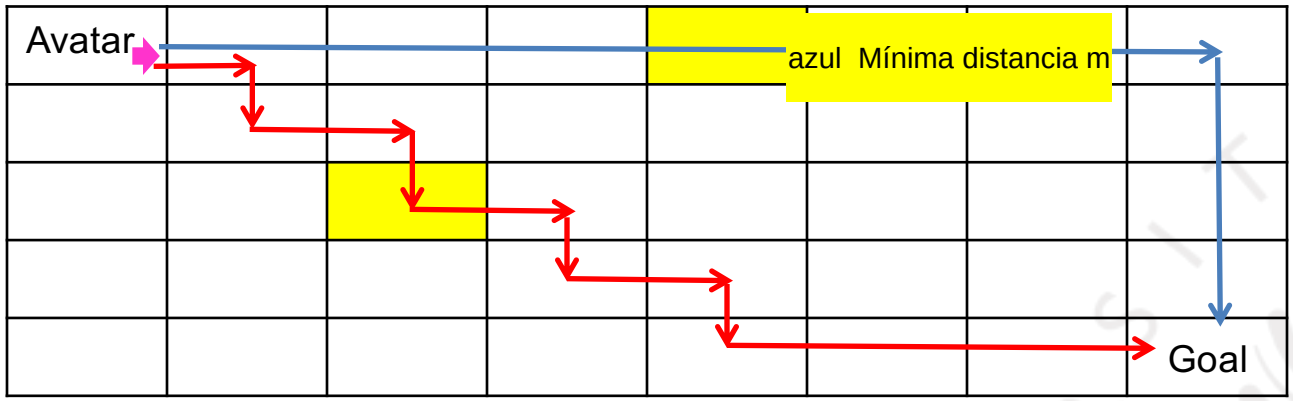
Nodo	Nodo padre	Coste del arco	Ruta	Coste de la ruta
E (Objetivo)	D	1	E - D	1
D	B	2	E – D - B	1+2 = 3
B	A	8	**E – D – B – A**	1+2+8 = **11**
A (Inicial)				

Para más información:

Introduction to the A* algorithm:

https://www.redblobgames.com/pathfinding/a-star/introduction.html

- Elección de mínima distancia ma**heurística**. ¿Cuál?
- **Estru**Si se quiere poner algo de esto en los comentarios del código, indicando por**s** datos necesarios
- Representación del **coste de los arcos**: función g(n)



f: distancia/ticks para llegar de la posición del Avatar al Goal

$f = g + h = 4 + 7$

$f = g + h = 7 + 7$

Porque el cambio de dirección implica un tick extra.

Timeout = 50ms!!!!

A* = Dijkstra + (

buscamos minimizar el número ticks, es decir,

Greedy usar sólo la h(x) Busca el camino más

si se quiere bajar hacia abajo, y el pers

inicializar factor de escala entre tablero y grid

Para saber en que escenario estamos, comprobar si hay gemas o enem

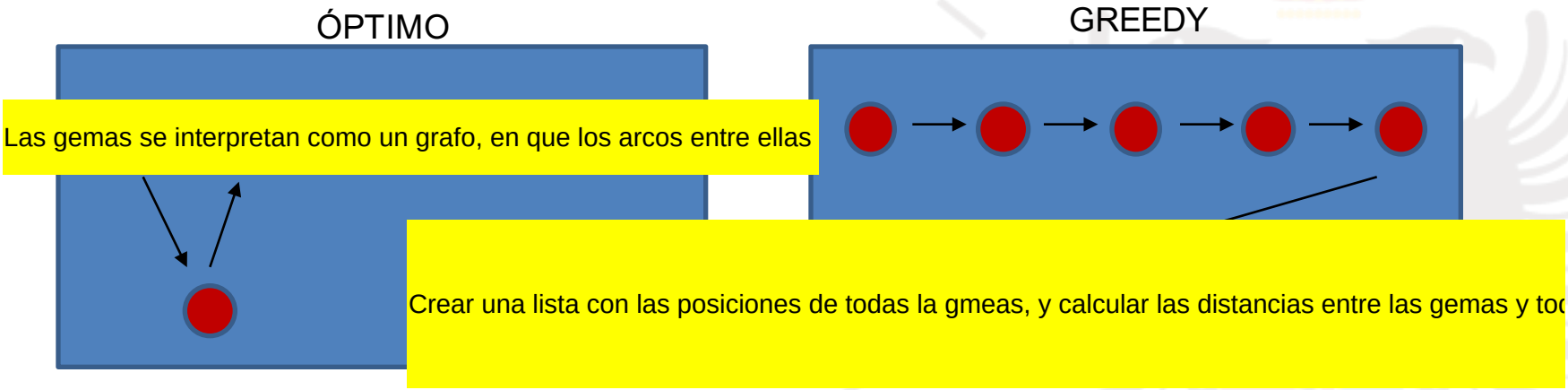
Planteamiento para el problema del deliberativo compuesto:

Este problema se puede **descomponer en dos sub-problemas**:

1. Decidir a qué gema (portal) dirigirse en cada momento
2. Decidir el camino para dirigirse hasta ella

Para el **primer sub-problema**, se empleará una estrategia greedy:

- En cada momento, dirigirse a la gema (portal) más cercana según la **distancia Manhattan**
- **NO** es una estrategia óptima, pero tiene mayor capacidad adaptativa (p.ej, en presencia de enemigos)



Para el **segundo sub-problema**, usaremos el camino óptimo calculado con A* para el problema deliberativo simple

Nota: recordad que hay un límite de tiempo por tick (50ms), por lo que la implementación debe ser suficientemente eficiente para no superar este tiempo.