|  |
| --- |
| UB |
| Intel·ligència Artificial |
| Pràctica 2 |
| **Pedro HerruzoSanchez, DNI: 46415512V, NIUB: 14582341.** |
| **Carlos Omar Cortés Hinojosa, DNI: 38882037P, NIUB: 14896523.** |
|  |
|  |

Índex

[INTRODUCCIÓ 2](#_Toc369696452)

[DISSENY 3](#_Toc369696453)

[AUTOEVALUACIÓ 4](#_Toc369696454)

[PROVES REALITZADES 4](#_Toc369696455)

[ERRORS DETECTATS 4](#_Toc369696456)

|  |
| --- |
|  |

# INTRODUCCIÓ

Aquesta memòria completa el codi de la practica en la que es demanava la implementació de les estratègies de cerca bàsiques al món d’en *pacman*. Concretament es demanava la implementació de les tècniques de cerca: DFS (*Depth First Search)*, BFS (*Breath First Search)*  i l’algoritme A\* amb diferents heurístiques. Per això s’ha utilitzat el codi base subministrat que s’encarrega de manejar tota l’aplicació excepte les estratègies de cerca demanades i el problema de les cantonades.

A més d’aquesta memòria tot el codi de la pràctica està comentat explicant més en detall com s’han implementat els diferents algoritmes , heurístiques i el problema de les cantonades.

# DISSENY

Com s’ha comentat a la introducció al llarg de la implementació s’ha continuat el disseny del codi proporcionat.

A la implementació dels algoritmes la única diferencia important entre els demanats era la gestió de la frontera:

* DFS la frontera es una pila,.
* BFS la frontera es una cua.
* A\* es una cua de prioritats ordenada per el cost fins el node més el valor de la heurística.

D’aquesta manera els tres algoritmes han estat implementats en una única funció de cerca que rep la frontera i l’heuristica per paràmetre. En el cas del DFS i el BFS que no utilitzen heurístiques s’utilitza una nul·la que sempre val 0.

A la pràctica també es demanava la realització del problema de les cantonades (*Corners Problem* al codi), per a la implementació s’ha basat en la del *SearchProblem*  que ja venia implementat, afegint una llista on es guarden les cantonades que queden per visitar. A més, s’ha implementat la funció dels successors que bàsicament s’assegurava de actualitzar la llista de cantonades pendents a visitar si el successor en qüestió es una cantonada. Amb aquest disseny per a saber si s’ha arribat a l’estat objectiu (*GoalState)* és suficient amb que la longitud de la llista on guardem les cantonades pendents sigui zero.

Per últim es demana la implementació de dues heurístiques, una pel problema de les cantonades i una pel problema del menjar:

* Heurística de les cantonades: basada en el fet de que el cost del camí que visita quatre corners al món del *pacman* sempre serà més gran que el camí que visita els quatre corners seguint una línia recta (seguint el rectangle). També és clar que el camí òptim visitarà les quatre cantonades en un ordre concret. Per tant, sabem que si es calcula el cost de visitar els quatre corners en línia recta per a tots els ordres possibles i prenent el mínim de tots els calculats aquest serà un valor més petit o igual al cost real. Per tant la heurística es admissible.

Com que només hi ha quatre cantonades el cost de calcular els 24 posibles ordres en que es visiten les cantonades es totalment assumible

* Heurística del problema del menjar: després de provar diferents heurístiques s’ha escollit la millor en nombre de nodes tenint en compte el cost computacional. Per a cada node el valor de la seva heurística es el nombre de menjars que queden al mapa. Clarament, la heurística és admissible ja que per a menjar n menjars, com a mínim, el camí te cost n. Amb aquesta heurística s’aconsegueix que es prioritzi avançar cap a punts on hi ha menjars. A més, és consistent ja que si ens movem n posicions com a molt podrem menjar n flocs i per tant la heurística no podrà

# Autoevalució

A l’hora de valorar el treball realitzat l’entorn de treball subministrat proporcionava dues eines: el temps trigat i el nombre de nodes expandits.

En el cas del DFS i el problema de cerca d’un únic menjar al *mediumMaze*es trobael de cost 130 tal com diu l’enunciat.

A continuació una tabla on es mostren els resultats obtinguts:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Implementació | | Dades de l’enunciat | |
| Nodes | Temps | Nodes | Temps |
| BFS-*CornersProblem* al *medimCorners* | 1921 | 0.0 s | Menys de 2000 | - |
| A\*-*CornersHeuristic* al *mediumCorners* | 787 | 0.0 s | 800-Molt bé | - |
| A\*-FoodHeuristic | 8679 | ~2 s | ~16000 | ~13 s |

Es veu que els resultats obtinguts compleixen amb els demanats. Tant a BFS com a DFS els són els espertats. En el cas de A\* amb *CornersHeuristic*  s’obtenen resultats molt bons segons l’enunciat,

# PROVES REALITZADES

El procediment que s’ha seguit per a testejar el codi ha consistit en utilitzar totes les comandes de que es disposava i veure que es solucionen els diferents problemes correctament i amb uns nombres de nodes i temps acceptables. Els resultats més destacables de les proves realitzades es troben a la taula d’aquesta pàgina

# ERRORS DETECTATS

En un principi

#TODO

· guardar tot camí per a cada node🡪 guardar nomes el pare

· la primera heurística que es va emprar no era admissible, bla ,bla ,bla

A la funció de cerca genèrica (*genericSearch)*  s’utilitza un diccionari per a emmagatzemar tota la informació que necessitem de cada node, és a dir, el cost, el node des de el qual arribem i la direcció per la que arribem. En un principi es feia que la clau dels diccionaris fos la tupla que representa la posició en forma d’*string* això però no era gens eficient i de cara a la implementació final s’utilitza la tupla com a clau del diccionari.