

Ricardo Iván Valdés Rodríguez | C411

Informe de IA

BERKLEY

# Problema 3 y 4

Usando un método auxiliar que es el que recorre los adyacentes y los va añadiendo en la cola de prioridad, al UCS le pasamos a la cola con prioridad la función de prioridad dependiendo del menor costo que tiene hasta el momento así iríamos expandiendo el árbol. Mientras que en el aStartSearch expandimos usando como función de prioridad el costo calculado hasta el momento, más el valor de la heurística en la posición del state.

# Problema 5

En este problema se deben recoger las frutas que están situadas en las esquinas, defino mi state de la forma:

State = ((x, y), food)

Donde la tupla x, y es la posición actual mientras que food es una matriz donde tiene marcada en True las posiciones de las frutas que quedan.

Posición inicial: Devuelvo el state con la posición inicial dado startingGameState junto al food dado por getFood () dado por startingGameState.

Objetivo: Devuelvo True cuando la matriz este vacía (no tenga ninguna casilla con True).

Sucesores:

Recorro las 4 posiciones a las que me puedo trasladar solamente las cuento como validas cuando no exista un muro, entonces coloco esa posición a la que me voy a mover como False y devuelve esa posición valida. Devolviendo un (state, action, cost).

Problema 6

La heurística busca usando el camino manhattan desde posición hasta cada uno de los alimentos, y nos quedamos con la mitad, se puede observar que si no existe comida en ninguna de las esquinas da 0 y es consistente y admisible ya que siempre h(a) < g(a) ya que nos quedamos con la menor forma de caminar desde la posición hasta cada comida y con la mitad de todo en caso de que una comida este a la mitad de otro camino hacia otra comida.

Problema 7

La heurística busca el máximo camino entre la posición actual y una comida, con esto nos daría para cada momento la comida mas lejana por lo tanto recorriendo con aStarSearch nos iríamos quedando con el camino más pequeño en cada iteración.

Demostración:

h(a) >= 0, maze siempre devuelve valores mayores que 0 y solo daría será cuando no exista camino hacia una comida dando el goalState por lo tanto devolvemos 0 en este caso.

h(a) <= C(a) siempre se cumple ya que maze busca usando bfs el menor camino posible

entonces si h(a) – h(b) <- C(a,b)

Sabemos que existe un camino desde la posición a hacia b, sino nunca seria entraría a este caso

Si las comidas más lejanas entre h(a) y h(b) se encuentran dentro del camino, entonces h(a) > C(a,b) – h(b) porque si esto no se cumpliera se podría formar una distancia de una posición a una fruta mas lejana y eso sería imposible ya que h(a) < C(a,b), lo mismo se cumple para h(b) por eso su diferencia se encuentra acotada entre 0 y el C(a,b).

Si las comidas se encontraran fuera del camino, tendríamos dos caso

1. Cuando ambos se dirigen a la misma comida se cumple pue h(a) – h(b) es C(a,b) o un numero negativo.
2. Van hacia comidas distintas, ambas comidas no podrían quedar fuera del camino ya que sino se podría formar un camino mas grande hacia una comida y estos son los caminos máximos.

En ese caso tomemos h(A) como la distancia de la posición de A con la comida que se encuentra afuera, h(A) – h(B) <= C(A,B), porque en caso contrario h(B) no estaría tomando la comida más lejana (la comida que busca A estaría más lejana que la comida que buscó) y eso es imposible. Es decir, la distancia de A con la comida más lejana tiene totalmente contenido a C(A, B), donde lo que sobra es la distancia de B a dicha comida.

Problema 8

El estado final solo devuelve True cuando el punto donde estoy parado este situado sobre una comida.

En el método simplemente llamo al método ya creado aStarSearch pasándole el problema dado y una null.heuristic