

**PRÁCTICA 2: Búsquedas Informadas y no Informadas**

**Grado en Ingeniería Informática**

# Curso 2014/2015

**Inteligencia Artificial:** Grupo 3 de prácticas

**Identificador y nombre del 1º ratón:** m15C04Practica2a MickeyDFS

**Identificador y nombre del 2º ratón:** m15C04Practica2b CakeAE

**Profesor:** José Manuel Fuertes García

**Alumnos:** David Fernández Puentes y Mª De Guía Martos Valenzuela

En esta práctica debemos implementar dos ratones que funcionen correctamente en el entorno del juego MOUSERUN. Cada uno de estos ratones implementa un algoritmo de búsqueda distinto. El primero realiza una búsqueda no informada ya que implementa la *búsqueda en profundidad iterativa.*  El segundo en cambio utiliza el algoritmo de búsqueda informada conocido como *A\*.* Los ratones tendrán que explorar el laberinto de manera inteligente para encontrar el queso

**Ratones**

# Mickey:

Mickey es nuestro primer ratón, está compuesto por 4 variables globales, *contadorMovimientos* y *bombasQuedan* son 2 contadores que determinan cuando debemos poner una bomba. *Laberinto* es una tabla hash en la que vamos almacenando cada una de las celdas por las que pasa Mickey y *camino* es la pila en la que almacenamos los movimientos óptimos para llegar al queso.

El funcionamiento de este ratón viene determinado por las siguientes funciones:

* 1. **DFS**: es la función que implementa el algoritmo pedido en la práctica. Lo primero que hace es almacenar la casilla en el laberinto. Después, comprueba los movimientos posibles que nos acerquen al queso y si lo hay será lo que devolvamos, sino significa que ya tenemos la mejor posición posible, por lo que sacaremos de la pila el camino a seguir.
  2. **HeurísticaManhattan:** para calcular la distancia entre donde nos encontramos y donde está el queso usamos esta función, que comprueba la distancia actual con la del movimiento que queremos realizar, para saber si es un buen movimiento.
  3. **PosiblesMovimientos**: devuelve true si puede realizar el movimiento solicitado y false si no.
  4. **Move**: esta es la función principal, ya que es la que le indica al ratón que es lo que debe hacer en cada turno. Como variables de entrada tiene la casilla en la que nos encontramos y la casilla en la que está el queso. Si estas variables tienen el mismo valor no tenemos por qué devolver nada ya que hemos encontrado el objetivo, sino es así entonces llamaremos al algoritmo primero en profundidad.

# Cake:

El segundo ratón se llama Cake, está compuesto por 6 variables globales, *contadorMovimientos* y *bombasQuedan* son 2 contadores que determinan cuando debemos poner una bomba. *Laberinto* es una tabla hash en la que vamos almacenando cada una de las celdas por las que pasa Cake, para ello hemos creado una variable nodo que almacena que movimientos son posibles desde ella (subir, bajar, derecha e izquierda) y *camino* es la pila en la que almacenamos los movimientos óptimos para llegar al queso. Además, hemos añadido una variable booleana para saber si una casilla es accesible y otra tabla hash para almacenar la información que contiene cada uno de los nodos por los que pasamos.

El funcionamiento de este ratón viene determinado por las siguientes funciones:

* 1. **AlgoritmoAEstrella**: es la función que implementa el algoritmo pedido en la práctica. Si tenemos la casilla que contiene al queso localizada y accesible la añadimos a la variable camino para que el ratón lo siga, si no calculamos que celda conocemos que es la más cercana al mismo. Una vez tenemos la posición de la celda más cercana calcularemos los movimientos almacenados hasta encontrar la casilla objetivo.
  2. **GetPadre:** comprueba si en el mapa hash de nodos visitados está la celda que contiene al queso. Si no es así entonces almacenamos para cada casilla en la que se encuentra el ratón que posibles movimientos podrá hacer para así comunicárselo a la función *AlgoritmoAEstrella* pasándolo con la variable *anteriores*.
  3. **getMenorDistancia**: es la función que usamos cuando no conocemos donde se encuentra el queso, mide la distancia de la casilla *queso*  y la casilla más cercana al mismo que si nos es conocida.
  4. **getDireccion:** sirve para saber qué dirección debe seguir el ratón según las coordenadas al queso. En caso de estar a una diferencia mayor a 1, siempre iremos a la izquierda
  5. **Move**: esta es la función principal, ya que es la que le indica al ratón que es lo que debe hacer en cada turno. Como variables de entrada tiene la casilla en la que nos encontramos y la casilla en la que está el queso. Si el laberinto no tiene almacenado la posición en que estamos, creamos un nodo nuevo con la información necesaria y lo almacenamos.

Después tenemos que llamar al algoritmo AEstrella para que almacene el camino a seguir y así asegurarnos que Cake se mueve correctamente

**COMPETICIÓN**

La fase de molestia a los adversarios la hemos definido igual para nuestros dos ratones. Tenemos un contador de movimientos y un contador de bombas que determinarán, en caso de ser posible, cuando nos toca poner la bomba. En cada búsqueda, sin que el queso sea comido o explotemos, dispondremos de un máximo de 5 bombas que iremos soltando cada 20 movimientos.

**Pruebas de los ratones**

Se han realizado distintas pruebas de los ratones, en el primer caso hemos probado con un tablero mediano y un tiempo reducido con 5 ejecuciones del problema, en el segundo hemos reducido el tablero obteniendo los siguientes resultados.

Tamaño del tablero: 10x10 Número de Quesos: 10

Tiempo de Ejecución: 30 Segundos

|  |  |  |
| --- | --- | --- |
| EJECUCIONES: 10/10/10/30 | | |
| Ejecución | mickeyDFS | cakeAE |
| 1 | 0 | 4 |
| 2 | 0 | 0 |
| 3 | 2 | 1 |
| 4 | 1 | 0 |
| 5 | 2 | 2 |

Tamaño del tablero: 5x5 Número de Quesos: 10

Tiempo de Ejecución: 30 Segundos

|  |  |  |
| --- | --- | --- |
| EJECUCIONES: 5/5/10/30 | | |
| Ejecución | mickeyDFS | cakeAE |
| 1 | 4 | 6 |
| 2 | 6 | 4 |
| 3 | 4 | 5 |
| 4 | 4 | 3 |
| 5 | 4 | 4 |

Tras las 10 ejecuciones del programa con distintos tamaños de tablero podemos observar que el tanto Mickey como Cake ganan y alcanzan un número parecido de quesos, por lo que ambos algoritmos resultan útiles en este tipo de problemas.