Departamento de Ingeniería
Informática y de Sistemas
Universidad de La Laguna

Informe Practica 2: A*

Airam Prieto González alu0101546377

C/ Padre Herrera s/n
38207 La Laguna
Santa Cruz de Tenerife. España

T: 900 43 25 26

ull.es

Índice

Funcionamiento.

La función `RecorridoAEstrella` implementa el algoritmo de búsqueda A* (A-star) para encontrar el camino más corto desde un nodo inicial hasta un nodo final en un laberinto. A continuación se describe brevemente el diseño del algoritmo y las estructuras de datos utilizadas:

1. Estructuras de datos utilizadas:

- **Nodo (`Nodo *`):** Los nodos representan las celdas del laberinto. Cada nodo tiene un identificador que corresponde a sus coordenadas `(x, y)` dentro del laberinto, un coste acumulativo para llegar a ese nodo (`g`), y una referencia al nodo padre para poder reconstruir el camino final.
- **vector<Nodo *> open_nodes**: Lista de nodos abiertos. Contiene nodos candidatos a ser explorados en las siguientes iteraciones. Se inicia con el nodo de partida.
- **vector<Nodo *> closed_nodes**: Lista de nodos cerrados. Contiene los nodos ya explorados, evitando que se visiten nuevamente.
- **Nodo *current_node**: Apunta al nodo que se está procesando en la iteración actual del algoritmo.
- **Nodo nodo_final_**: Nodo que representa la posición de destino en el laberinto.

2. Inicialización:

- La función comienza identificando el nodo de inicio (`current_node`) y el nodo de destino (`nodo_final_`).
- Se añaden los nodos hijos inmediatos del nodo inicial a la lista de nodos abiertos (`open_nodes`), y se seleccionan como candidatos válidos usando la función `SelectDefChildren`.

3. Iteración principal:

- El algoritmo entra en un bucle `while` que continúa hasta que se encuentre el destino o no haya más nodos abiertos por explorar:
    1. **Verificación de solución:** Si la lista de nodos abiertos (`open_nodes`) está vacía, se determina que no hay solución posible y se imprime un mensaje indicando que no se encontró el camino.
    2. **Selección del nodo a explorar:** Se elige el nodo con el menor valor de `f` (coste total estimado) de la lista de nodos abiertos usando la función `SelectMinorF`.

3. **Comprobación de destino:** Si el nodo seleccionado es el nodo destino, se ha encontrado el camino. Se reconstruye el camino trazado y se imprime el coste total y el camino mínimo.
4. **Actualización de listas:** El nodo seleccionado se elimina de la lista de nodos abiertos y se añade a la lista de nodos cerrados para marcarlo como explorado.
5. **Generación de hijos:** Se calculan los nodos hijos del nodo actual y se añaden a la lista de nodos abiertos, siempre que no estén ya en la lista de cerrados y se optimiza su coste si corresponde.

4. Funciones auxiliares:

- `CalculateChildren`: Calcula los nodos hijos adyacentes que se pueden explorar desde el nodo actual.
- `SelectDefChildren`: Selecciona los hijos válidos y los añade a la lista de nodos abiertos, evitando nodos repetidos.
- `SelectMinorF`: Selecciona el nodo de menor coste total `f` de la lista de nodos abiertos.
- `PrintIteration`: Imprime la información de la iteración actual en el archivo de salida (`file_out`).

5. Coste total `f` de cada nodo:

- El valor `f` se calcula como la suma de dos componentes:
  - `g`: Coste acumulativo para llegar al nodo desde el inicio.
  - `h`: Estimación heurística de la distancia restante hasta el nodo destino.

6. Salida final:

- Si se encuentra el camino, la función imprime la ruta trazada desde el nodo de destino hasta el de partida y marca el camino en el laberinto.
- Si no se encuentra, se imprime un mensaje indicando que no hay solución.

7. Heurística alternativa:

- La distancia euclidiana, $\sqrt{(x_i - x_{final})^2 + (y_i - y_{final})^2}$

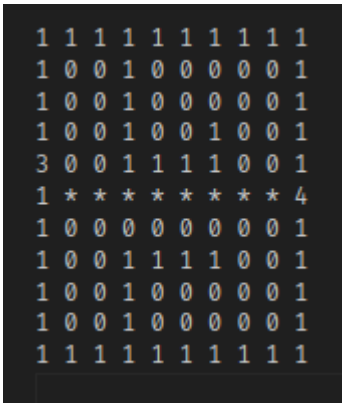
8. Tabla

Maze	n	m	S	E	Camino	Coste	Generados	Inspeccionados

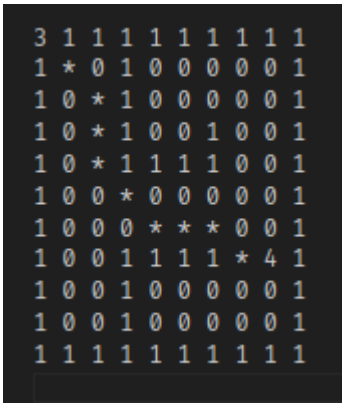
M1-A	11	10	4-0	5-9	5-9 <- 5-8 <- 5-7 <- 5-6 <- 5-5 <- 5-4 <- 5-3 <- 5-2 <- 5-1 <- 4-0	47	6-2 6-3 6-4 2-1 3-2 6-5 6-6 4-7 6-7 4-8 6-8 1-2 1-1 5-9 7-1 7-2	4-0 5-1 5-2 5-3 3-1 5-4 4-2 5-5 5-6 5-7 2-2 5-8 4-1 6-1
M1-B	11	10	0-0	7-8	7-8 <- 7-7 <- 6-6 <- 6-5 <- 6-4 <- 5-3 <- 4-2 <- 3-2 <- 2-2 <- 1-1 <- 0-0	60	2-1 3-1 4-1 5-2 6-3 5-4 5-5 5-6 6-7 5-7 6-1 8-7 7-8 6-8 8-6 8-8 7-2 7-1	0-0 1-1 2-2 3-2 4-2 5-3 6-4 1-2 6-5 6-6 5-1 7-7 6-2
M2-A	17	17	4-0	12-16	12-16 <- 12-15 <- 12-14 <- 12-13 <- 13-12 <- 12-11 <- 12-10 <- 12-9 <- 11-8 <- 11-7 <- 11-6 <- 10-5 <- 9-4 <- 8-3 <- 7-3 <- 6-2 <- 5-1 <- 4-0	103	10-9 1-1 9-8 13-11 10-1 11-2 8-9 9-10 14-12 13-13 14-11 12-1 9-11 11-13 11-14 13-14 14-8 13-7 14-7 14-9 13-2 13-1 13-3 11-15 13-15 14-10 12-16 7-10 8-11 7-9 7-11 15-13 14-14 15-12 15-14	4-0 5-1 6-2 7-3 8-3 9-4 3-1 10-5 4-2 11-6 5-3 7-1 8-2 11-7 6-3 9-3 10-4 11-8 11-5 12-9 2-2 9-6 3-3 10-7 12-10 4-1 6-1 4-3 9-1 10-8 12-11 7-2 10-2 11-9 11-3 11-10 11-4 1-3 9-5 2-1 11-11 3-2 10-6 5-2 9-9 13-12 8-1 11-1 10-10 12-13 9-2 13-8 12-2 10-3 10-11 12-14 13-9 12-3 1-2 12-15 13-10 2-3 9-7 8-10 14-13
M2-B	17	17	5-5	10-10	10-10 <- 9-10 <- 8-10 <- 7-10 <- 6-10 <- 5-10 <- 4-9 <- 3-8 <- 3-7 <- 4-6 <- 5-5	58	3-5 2-7 2-6 2-8 3-9 5-9 5-11 4-11 6-9 6-11 7-9 7-11 2-5 8-9 8-11 1-9 2-10 1-8 1-10 3-11 2-11 9-9 9-11 10-10 10-9 10-11	5-5 6-6 7-7 4-6 5-7 7-5 7-6 3-7 4-5 5-6 3-8 6-7 4-9 5-10 6-10 3-6 7-10 6-5 4-7 2-9 3-10 8-10 4-10 9-10
M3-A	5	5	4-4	1-0	1-0 <- 1-1 <- 0-2 <- 1-3 <- 2-4 <- 3-3 <- 4-4	40	4-3 0-3 0-4 0-1 1-0 2-0	4-4 3-3 3-2 2-4 1-3 3-4 0-2 1-4 1-1
M3-B	5	5	4-2	0-2	0-2 <- 1-3 <- 2-4 <- 3-3 <- 4-2	31	3-4 1-4 0-3 0-2 0-4	4-2 3-2 3-3 2-4 4-3 1-3

Fotos.

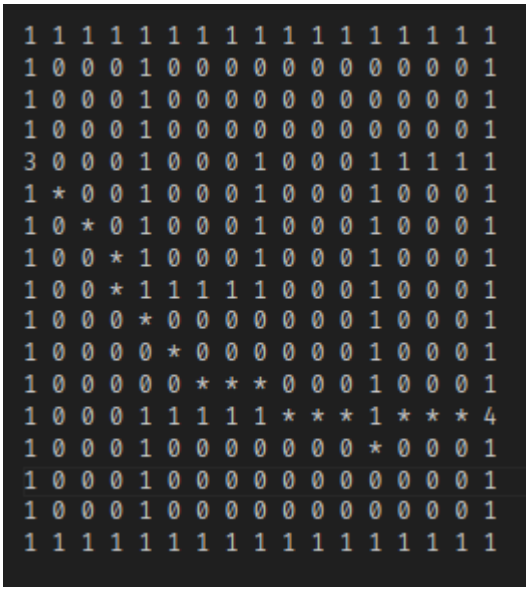
M1-A



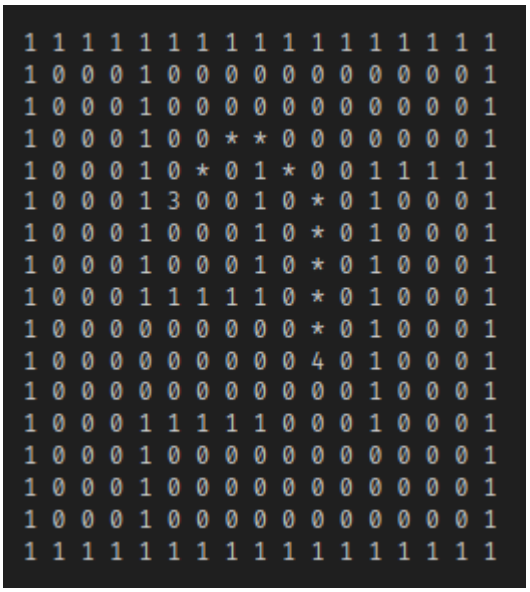
M1-B



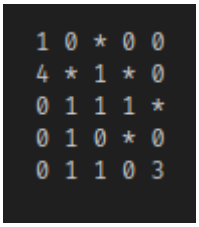
M2-A


1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
3 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1
1 * 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
1 0 * 0 1 0 0 0 1 0 0 0 1 0 0 0 1
1 0 0 * 1 0 0 0 1 0 0 0 1 0 0 0 1
1 0 0 * 1 1 1 1 1 0 0 0 1 0 0 0 1
1 0 0 0 * 0 0 0 0 0 0 0 1 0 0 0 1
1 0 0 0 0 * 0 0 0 0 0 0 1 0 0 0 1
1 0 0 0 0 0 * * * 0 0 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 * * * 1 * * * 4
1 0 0 0 1 0 0 0 0 0 0 0 * 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

M2-B


1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 * * 0 0 0 0 0 0 0 1
1 0 0 0 1 0 * 0 1 * 0 0 1 1 1 1 1
1 0 0 0 1 3 0 0 1 0 * 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1 0 * 0 1 0 0 0 1
1 0 0 0 1 0 0 0 1 0 * 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 0 * 0 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 * 0 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 4 0 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

M3-A



M3-B