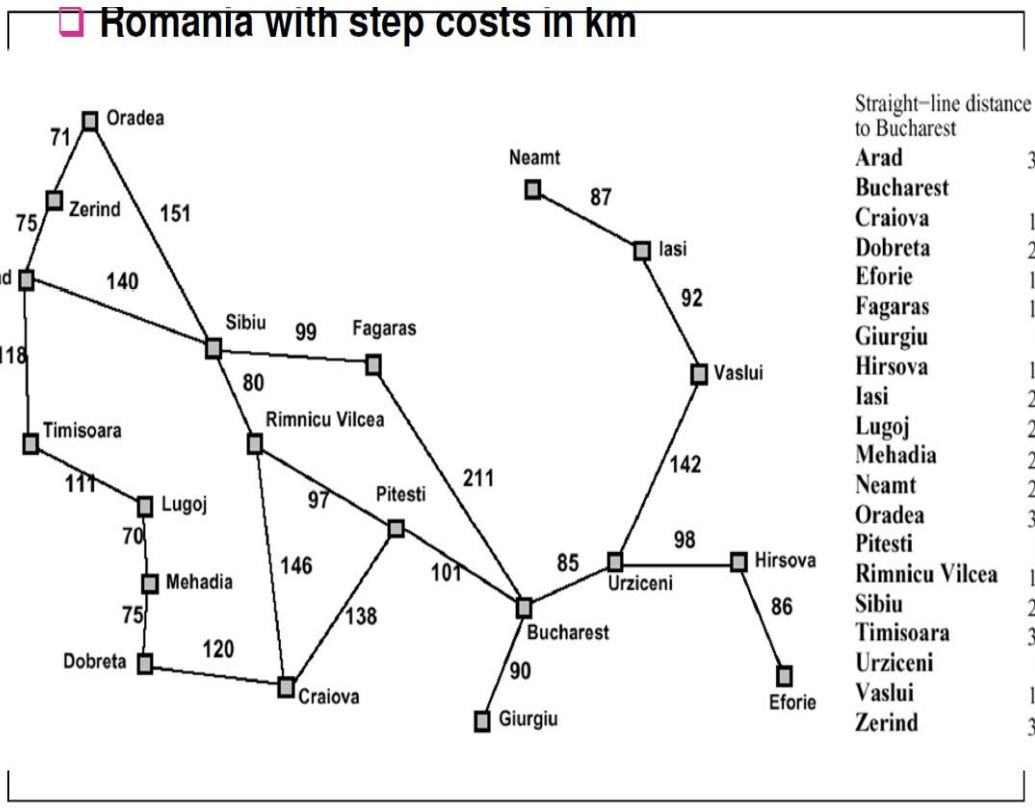
Informe Inteligencia Artificial: Búsqueda A*.



Ingeniería Informática ULL.
Manuel Andrés Carrera Galafate

Introducción al algoritmo A*

El algoritmo de búsqueda A*se clasifica dentro de los algoritmos de búsqueda en grafos de tipo heurístico o informado.

El funcionamiento del algoritmo se basa en una función (f(n)) que es la suma de otras dos (h(n) y g(n)). h(n) es la función heurística estimada para un nodo dado n, que podría ser por ejemplo la distancia en línea recta desde el nodo n hasta el nodo destino (para el nodo final su h(n) = 0). g(n) es la suma real de las distancias desde el nodo inicial hasta el nodo actual (Si estuvieramos en el tercer nodo desde el inicial, su g(n) es la suma del coste del nodo inicial al segundo y el coste del segundo al tercero).
f(n) pues sería la suma de la distancia que hemos recorrido para llegar hasta el nodo n y la distancia que estimamos para llegar al nodo final.

Para resolver el problema del camino mínimo, en primer lugar consideraremos al nodo inicial como un nodo prometedor, veremos si este nodo es el nodo final, y si no lo es, generaremos sus sucesores (Nodos a los que podemos llegar desde el nodo directamente). A continuación, calcularemos la función f(n) de cada uno de los sucesores, y a continuación en cada iteración escogeremos al nodo con menor coste asociado de la función para explorar. Este algoritmo lo repetiremos hasta llegar al nodo final, teniendo en cuenta que un nodo que esté en la lista de predecesores del nodo n no lo añadiremos a los sucesores de este. Esto significa que si para llegar al nodo 3 hemos recorrido los nodos 1, 4 y 6, y entre los sucesores del 3 está el 6, no consideraremos a este último como sucesor.

En esta práctica recrearemos en el lenguaje de programación C++ este algoritmo, teniendo 2 ficheros, 1 que contenga el grafo con los costes asociados, y un segundo fichero que contenga la heurística de todos los nodos a un nodo X concreto.

Estructuras de datos

Para la realización de esta práctica hemos usado la siguiente estructura:

Una clase nodo, que contendrá el identificador del nodo, un puntero a su padre, una heurística al nodo objetivo asociada y un coste del camino que lleva hasta este nodo.

Una clase grafo, que contendrá el número de nodos y aristas del grafo, y además la información del mismo en un vector de vectores de pares. El vector de parejas representa todos los sucesores que tiene un nodo iésimo, donde el primer elemento de la pareja es el índice del nodo sucesor y el segundo el coste. El vector externo representa el nodo iésimo. Es decir, la posición v[0][0] nos retornaría el primer sucesor (con su coste asociado) del nodo 0, y v[7][8] nos retornaría el noveno sucesor del nodo 7. La estructura que utilizaremos realmente es la de una matriz dispersa, ya que si un nodo, el 0 por ejemplo, solo tiene 3 sucesores y existen 200 nodos en el grafo no tiene sentido almacenar 197 posiciones extra, de ahí que necesitemos el índice del sucesor además del coste, para poder recuperar el grafo real. También tenemos que considerar que la manera de almacenarla es tal que del nodo 1 al 2 tenemos un camino con coste 20, del nodo 2 al 1 tendremos un camino de coste 20. Para los caminos inexistentes, a la hora de introducirlos en el vector de vectores comprobaremos que el coste sea mayor a 0, en otro caso consideraremos que no existe camino entre los 2 puntos o que es de coste infinito.

Finalmente, tendremos una clase árbol, que será la que genere el camino. Para el correcto funcionamiento del árbol necesitaremos que contenga un grafo de donde se extraerá la información de los sucesores y costes de los nodos en general, un vector que contiene la heurística de los distintos nodos al nodo destino (Si el nodo destino es el 14, la heurística sería el camino en línea recta de los distintos nodos al 14), un contador para los nodos generados e inspeccionados, un puntero raíz que estará a nulo, y será a donde apunte el primer nodo de la búsqueda, y una lista de nodos prometedores de donde escogeremos en cada iteración el nodo con mejor función de evaluación asociada. La decisión de implementarlo en forma de lista se debe a que cada vez que se escoge un elemento como "el más prometedor" se debe eliminar de la lista de nodos prometedores, y una

extracción es menos costosa computacionalmente en una lista que en un vector.

Explicación de la implementación.

Cuando ejecutemos el programa tendremos que pasar por parámetro 2 ficheros de donde leer, uno para extraer el grafo y otro para obtener el vector de heurística. A continuación, se nos pedirá que establezcamos un nodo inicial y final para la búsqueda (El nodo final de la búsqueda debería establecerse teniendo en cuenta que sea el adecuado para el fichero de heurística). El fichero del grafo se utilizará para construir el árbol, el de heurística se introducirá en la búsqueda para permitir mayor flexibilidad (Un grafo puede tener un vector de heurística por cada nodo del grafo).

Una vez hayamos introducido todos los datos necesarios para la búsqueda, el algoritmo comenzará por crear un nodo inicial, que tendrá como identificador el nodo inicial que hemos establecido antes, coste 0, padre el puntero nulo y como heurística la que ponga en el fichero en la posición v_heurística[id_del_nodo]. Este nodo se introducirá entre los nodos prometedores, se escogerá como el nodo prometedor y comenzará la parte iterativa:

-Se observará si el nodo N escogido es el nodo final.
-Si no es el nodo final, se eliminará el nodo de los prometedores y se generarán sus sucesores. Para generar los sucesores se observará que estos no estén en el camino desde el nodo N hasta el inicio. Los nodos generados tendrán como padre al nodo N y como coste el coste de N + el coste de N al nodo sucesor i. Estos sucesores se añadirán en la lista de nodos prometedores.
El nuevo nodo N será el que tenga menor f(n) asociado de la lista de nodos prometedores.
-Si el nodo escogido (el que tiene menor f(n) asociado) es el nodo final, ese nodo y su recorrido hasta el nodo origen mediante los punteros a los padres representan el camino mínimo entre los 2 nodos. Por ello, podemos almacenar el recorrido de nodos en una pila y luego vaciar la pila sumando los costes asociados para tener el camino en el sentido correcto y con la suma de los costes el coste del camino.

Resultados

Nodos	Aristas	N_Ini	N_Fin	Camino	Distancia	N_Generados	N_Inspeccionados
15	19	0	7	1 -> 2 -> 4 -> 12 -> 5 -> 13 -> 8 ->	43	90	51
15	19	0	7	1 -> 2 -> 4 -> 12 -> 5 -> 13 -> 8 ->	43	90	52
15	19	1	13	2 -> 4 -> 12 -> 5 -> 13 -> 8 -> 14 ->	41	79	49
15	19	0	7	1 -> 2 -> 4 -> 12 -> 5 -> 13 -> 8 ->	43	123	80

Algunas soluciones de los grafos propuestos en clase (Grafo 1 y 2 con heurísticas 1 y 2)