Trabajo Práctico N° 7

UNIDAD 4: Algoritmos y Aplicaciones avanzadas

Objetivo
Ejercitar el uso de técnicas de recursión y backtracking, listas enlazadas y algoritmos de búsqueda

Problema
Se tiene una matriz en memoria, con el siguiente formato:

```
1000011111
1100010000
0100010000
0100010000
0100010000
0100011000
0100001111
0100000001
0100000001
0111111110
```

Los valores en cero indican un espacio libre, los valores en 1 indican un obstáculo.

Se solicita por teclado una posición A=(x, y) inicial, y una posición B=(x, y) final, y se requiere encontrar un camino de A hasta B, sin salirse de la matriz y esquivando los obstáculos. Se debe imprimir por pantalla el camino, en forma de pares (x, y), o bien se debe indicar que no existe tal camino si no es posible llegar.

Para encontrar el camino de A a B, recorra recursivamente el árbol de búsqueda que se genera a partir del mapa mencionado. Cada posición del mapa representa un nodo, y los hijos de dicho nodo son todas las posiciones del mapa alrededor del mismo. Por ejemplo, si la posición inicial es (5,6), a partir de ese nodo es posible pasar a uno de sus 8 posibles nodos hijos, a saber: {(4,5), (4,6), (4,7), (5,5), (5,7), (6,5), (6,6), (6,7)}. Estos nodos deben recorrerse de a uno, recursivamente, hasta que se llega a la posición final deseada, o se determina que es un camino sin salida, en cuyo caso se retorna al punto de decisión anterior y se continúa con el siguiente nodo.

En cada punto de decisión, se deben verificar las siguientes condiciones, a fin de validar que es un nodo hijo válido:

1. Que el nodo en cuestión no está fuera de la matriz (subíndices i y j deben ser >= 0 y < que la cantidad de filas y columnas respectivamente)
2. Que el nodo en cuestión esté marcado con un cero, indicando que no es un obstáculo
3. Que el nodo en cuestión no haya sido explorado anteriormente, dado que en ese caso podría caerse en una recursión de profundidad infinita. Esto puede realizarse cambiando el valor almacenado en la matriz para los nodos ya explorados (ej: utilizar un valor distinto de 0 y 1).

Sugerencia: es posible recorrer todos los nodos hijos de un determinado nodo (i,j) mediante un doble bucle anidado, donde el bucle más externo itera desde i-1 a i+1 y el bucle más interno itera desde j-1 hasta j+1. En este caso se debe verificar, además de los puntos anteriores, que no se explore el propio nodo.

Sólo debe imprimirse el camino en caso de llegar al destino. Caso contrario se debe imprimir un mensaje indicando que no existe camino entre A y B.

Ejercicios complementarios (opcionales)

Ordenamiento y búsqueda

Implemente los algoritmos de ordenamiento bubblesort y quicksort, y el algoritmo de búsqueda binaria. Implemente cada uno de ellos en una función. Utilice recursión para implementar quicksort y búsqueda binaria. Escriba además un programa (main) que:

1. genere un arreglo aleatorio de 1000 elementos, cada uno tomando un valor entre 0 y 1000 (utilice la función rand() y el operador módulo %);
2. guarde el valor 500 en una posición aleatoria dentro del arreglo;
3. ordene el arreglo mediante bubblesort o quicksort (a elección del usuario);
4. y finalmente que busque el elemento 500 dentro del arreglo especificado por el usuario y muestre por pantalla el subíndice donde se encontró.

Listas enlazadas

Implemente una biblioteca con funciones para manipulación de listas doblemente enlazadas. Cree un nodo para la lista mediante struct (el contenido de cada nodo es a libre elección del alumno), e implemente funciones para

5. agregar un elemento a la lista,
6. eliminar un elemento de la lista,
7. modificar un elemento de la lista
8. mostrar todos los elementos de la lista.