Universidad Rey Juan Carlos

Algoritmos de búsqueda I

Práctica 1.Parte 1 – Enunciados de los problemas

P1. n-Puzzle

El problema del n-puzzle es un clásico rompecabezas deslizante que se utiliza como un problema de búsqueda y planificación en inteligencia artificial.

En su versión tradicional el rompecabezas involucra un tablero 3x3 con ocho fichas numeradas del 1 al 8 y un espacio en blanco.

El objetivo es alcanzar un estado específico partiendo desde una disposición inicial de las fichas mediante la realización de movimientos permitidos.

Requisitos del programa:

- *El usuario podrá definir el tamaño del n-puzzle, donde n es el tamaño de piezas a acomodar.*
- *El usuario podrá definir un estado inicial y un estado final.*

Ejemplo de estado inicial para un 15-puzzle:

2	1	9	4
11	5	10	3
6	8		7
12	15	14	13

Ejemplo de estado final:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Algoritmos de búsqueda I

Práctica 1.Parte 1 – Enunciados de los problemas

P2- Torres de Hanoi

En el problema de las Torres de Hanoi hay tres agujas (*A*, *B* y *C*) y *n* discos de distintos tamaños que reposan sobre la aguja *A* (*D1*, *D2*, ..., *Dn*).

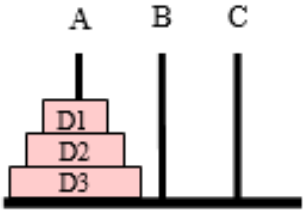
Es un requisito que en cualquier momento los discos estén apilados de menor a mayor.

El proceso es pasar la torre de discos de la aguja *A* bien a la aguja *B o* bien a la aguja *C*. En cada operación es posible mover un disco de una aguja a otra, siempre y cuando dicho disco no se coloque sobre un disco de menor tamaño.

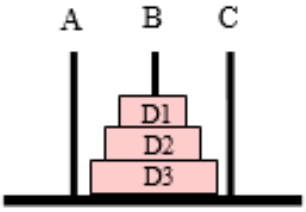
A continuación, se muestra el estado inicial y un estado meta para el caso de *n*=3 discos.

Requisitos del programa:

- *El usuario podrá definir el número n de discos a mover entre las torres.*
- *El usuario podrá definir el estado inicial, colocando en diferentes posiciones los discos, siempre y cuando respete la regla de colocar discos más pequeños sobre los grandes.*



Ejemplo de estado inicial



Ejemplo de estado final

Algoritmos de búsqueda I

Práctica 1.Parte 1 – Enunciados de los problemas

P3. Mundo de los bloques apilados

El problema del mundo de los bloques aborda la planificación y la resolución de problemas en un entorno de bloques apilables.

Se tiene un conjunto de bloques y un espacio de trabajo donde se pueden apilar los bloques.

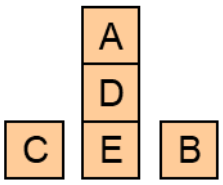
Cada bloque puede estar en una de las ubicaciones del espacio de trabajo o encima de otro bloque.

A continuación, se muestra el estado inicial y un estado meta para el caso de *n*=5 bloques.

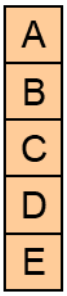
Requisitos del programa:

- *El usuario podrá definir el número n de bloques a apilar.*
- *El usuario podrá definir el estado inicial y el estado final, colocando los bloques en diferentes posiciones.*

Ejemplo de estado inicial:



Ejemplo de estado final:



Algoritmos de búsqueda I

Práctica 1.Parte 1 – Enunciados de los problemas

P4. Cuadrados latinos

En el juego de los "cuadrados latinos" se parte de un tablero n x n vacío o con alguna otra configuración.

En su versión clásica el tablero es de 3 x 3 y en cada posición vamos colocando números del 1 al 9, ninguno de los cuales se puede repetir.

El objetivo es tener el tablero completo, es decir un número en cada posición de este.

En el estado final se busca que el valor de la suma de las filas, columnas y diagonales sea siempre el mismo valor: 15 (para el caso de números del 1 al 9).

Requisitos del programa:

- *El usuario podrá definir el estado inicial, colocando los números en diferentes posiciones, pudiendo omitir algunos de ellos.*
- *Debe respetarse que en ese estado inicial no se repitan números y que estén limitados del 1 al 9.*

Ejemplo de estado inicial:

2		4
	5	3
6	1	

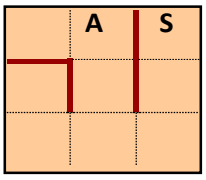
Ejemplo de estado final:

2	9	4
7	5	3
6	1	8

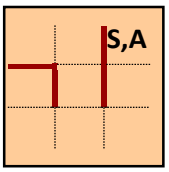
Algoritmos de búsqueda I
Práctica 1.Parte 1 – Enunciados de los problemas

P5. Recorrido de un laberinto

Considere el problema el laberinto que se presenta en la siguiente figura (estado inicial):



El agente A tiene el objetivo de encontrar la salida S (estado final).



Las únicas acciones de las que el agente dispone son los movimientos (derecha, arriba, abajo, e izquierda) desde el cuadrado en el que se encuentra en un momento dado a un cuadrado adyacente.

Cada una de estas acciones sólo es posible si en la dirección correspondiente no existe una barrera ni se saldría del tablero. Cada acción tiene un coste de una unidad.

De antemano, el agente conoce el mapa del laberinto y la posición de la salida, pero no las posiciones de las barreras. Además, el agente es capaz de identificar en cada momento su propia posición en el mapa.

Requisitos del programa:

- *El usuario podrá definir el estado inicial, colocando la salida S y el agente A en el laberinto.*
- *El usuario podrá colocar las "barreras" del laberinto.*