

**SISTEMAS INTELIGENTES.**

**PRÁCTICA DE PLANIFICACIÓN**

**1.** El problema del puzzle 8 también puede ser resuelto como un problema de planificación.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |  |
| 1 | 2 | 3 |
| 4 | 5 |  |
| 7 | 8 | 6 |

**Figura 1. Estado Inicial Figura 2. Estado Final**

1. **Podemos definir el problema con una única acción que sería:**

operator slide(T,S1,S2)

pre: on(T,S1), tile(T), blank(S2), adjacent(S1,S2)

post: on(T,S2), blank(S1),~on(T,S1), ~blank(S2)

Para tal caso debes hacer lo siguiente:

1) Incluir la acción *slide* en el fichero ***domainPuzzle8.txt***

2) Escribir la descripción del estado inicial y final según las figuras 1 y 2 en el fichero ***problemPuzzle8Facil.txt***.

start(

tile(t1), tile(t2), tile(t3),

tile(t4), tile(t5), tile(t6),

tile(t7), tile(t8), blank(s9),

on(t1,s1), on(t2,s2), on(t3,s3),

on(t4,s4), on(t5,s5), on(t6,s6),

on(t7,s7), on(t8,s8),

adjacent(s1,s2), adjacent(s1,s4),

adjacent(s2,s3), adjacent(s2,s5),

adjacent(s3,s6), adjacent(s4,s7),

adjacent(s4,s5), adjacent(s5,s8),

adjacent(s5,s6), adjacent(s7,s8),

adjacent(s6,s9), adjacent(s8,s9),

adjacent(s2,s1), adjacent(s4,s1),

adjacent(s3,s2), adjacent(s5,s2),

adjacent(s6,s3), adjacent(s7,s4),

adjacent(s5,s4), adjacent(s8,s5),

adjacent(s6,s5), adjacent(s8,s7),

adjacent(s9,s6), adjacent(s9,s8))

goal(

blank(s6)

)

3) Ejecuta el planificador

***java –jar javagp.jar –nopddl –d domainPuzzle8.txt –p problemPuzzle8Facil.txt***

y escribe el resultado:

INFORMACION:Runnig planner, maximun memory:455,5MB

INFORMACION:Expanding graph

INFORMACION:Extracting solution

INFORMACION:Plannig took 489ms (0s)

INFORMACION:Total memory used: 254,5MB

INFORMACION:Plan found:

slide(t6,s6,s9)

INFORMACION: Plan length: 1

4) Modifique la configuración inicial y final según las figuras 3, 4. y almacena en el fichero ***problemPuzzle8.txt***.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 |  |
|  | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

**Figura 3. Estado Inicial Propuesto Figura 4. Estado Final Propuesto**

Inicial

start(

tile(t1), tile(t2), tile(t3),

tile(t4), tile(t5), tile(t6),

tile(t7), tile(t8), blank(s9),

on(t1,s1), on(t2,s2), on(t3,s3),

on(t4,s4), on(t5,s5), on(t6,s6),

on(t7,s7), on(t8,s8),

adjacent(s1,s2), adjacent(s1,s4),

adjacent(s2,s3), adjacent(s2,s5),

adjacent(s3,s6), adjacent(s4,s7),

adjacent(s4,s5), adjacent(s5,s8),

adjacent(s5,s6), adjacent(s7,s8),

adjacent(s6,s9), adjacent(s8,s9),

adjacent(s2,s1), adjacent(s4,s1),

adjacent(s3,s2), adjacent(s5,s2),

adjacent(s6,s3), adjacent(s7,s4),

adjacent(s5,s4), adjacent(s8,s5),

adjacent(s6,s5), adjacent(s8,s7),

adjacent(s9,s6), adjacent(s9,s8))

goal(

blank(s1)

)

4.1. Establezca el objetivo en base al espacio en blanco (blank(p1)), almacena en el fichero ***problemPuzzle8Sec4.1.txt***. y vuelve a ejecutar el planificador

***java –jar javagp.jar –nopddl –d domainPuzzle8.txt –p problemPuzzle8Sec4.1.txt***

Objetivo

goal(

blank(s1))

y escribe el resultado:

INFORMACION: Running planner, maximum memory:455,5MB

INFORMACION: Expanding graph

INFORMACION: Goals not possible with 1 steps

INFORMACION: Expanding graph

INFORMACION: Goals not possible with 2 steps

INFORMACION: Expanding graph

INFORMACION: Goals not possible with 3 steps

INFORMACION: Expanding graph

INFORMACION: Extracting solution

INFORMACION: Plan found with 4 steps

INFORMACION: Planning took 984ms (0s)

INFORMACION: Total memory used: 245,5MB

INFORMACION: Plan found:

slide(t8,s8,s9)

slide(t7,s7,s8)

slide(t4,s4,s7)

slide(t1,s1,s4)

INFORMACION: Plan length: 4

4.2. Establezca el objetivo en base a las fichas (on(f1,p2)…), almacena en el fichero ***problemPuzzle8Sec4.2.txt***. y vuelve a ejecutar el planificador

***java –jar javagp.jar –nopddl –d domainPuzzle8.txt –p problemPuzzle8Sec4.2.txt***

Objetivo

goal(

on(t1,s2), on(t2,s3), on(t3,s4),

on(t4,s5), on(t5,s6), on(t6,s7),

on(t7,s8), on(t8,s9))

y escribe el resultado:

slide(t6,s6,s9)

slide(t5,s5,s6)

slide(t4,s4,s5)

slide(t7,s7,s4)

slide(t8,s8,s7)

slide(t6,s9,s8)

slide(t5,s6,s9)

slide(t3,s3,s6)

slide(t2,s2,s3)

slide(t4,s5,s2)

slide(t6,s8,s5)

slide(t8,s7,s8)

slide(t7,s4,s7)

slide(t6,s5,s4)

slide(t3,s6,s5)

slide(t5,s9,s6)

slide(t8,s8,s9)

slide(t7,s7,s8)

slide(t6,s4,s7)

slide(t3,s5,s4)

slide(t4,s2,s5)

slide(t1,s1,s2)

Plan length: 22

1. **El problema puede ser definido mediante cuatro acciones (OPCIONAL)**

operator moveUp (T,PX,PY,BY)

pre: tile(T), position(PX), position(PY), position(BY), dec (BY,PY), blank(PX,BY), at(T,PX,PY)

post: ~blank(PX,BY),~at(T,PX,PY),blank(PX,PY), at(T,PX,BY)

operator moveDown (T,PX,PY,BY)

pre: tile(T), position(PX), position(PY), position(BY), inc (BY,PY), blank(PX,BY), at(T,PX,PY)

post: ~blank(PX,BY),~at(T,PX,PY),blank(PX,PY), at(T,PX,BY)

operator moveLeft (T,PX,PY,BX)

pre: tile(T), position(PX), position(PY), position(BX), dec (BX,PX), blank(BX,PY), at(T,PX,PY)

post: ~blank(BX,PY),~at(T,PX,PY),blank(PX,PY), at(T,BX,PY)

operator moveRight (T,PX,PY,BX)

pre: tile(T), position(PX), position(PY), position(BX), inc (BX,PX), blank(BX,PY), at(T,PX,PY)

post: ~blank(BX,PY),~at(T,PX,PY),blank(PX,PY), at(T,BX,PY)

Para tal caso debemos repetir los puntos anteriores, observando el tiempo que toma al planificador la resolución del problema.