1. Encontrar una representación adecuada del espacio para los siguientes problemas:

   a) El juego de los 8 números (Taken):
   Dada una configuración de ocho números en un tablero de 3×3, llevarlo mediante el desplazamiento de números, a la siguiente situación final

1	2	3
8		4
7	6	5

   b) La torre de Hanói:
   Hay 64 discos de diámetro decreciente en un poste y hay que pasarlos a otro poste, utilizando un tercero para los pasos intermedios. Sólo puede moverse un disco a la vez, siempre deben estar en algún poste y no se puede colocar un disco sobre otro de menor tamaño.

   c) Cuadrado Latino:
   Consiste en llenar un cuadrado de 3×3 con un elemento del conjunto $\{1, 2, 3\}$ de forma tal que en cada fila y columna no haya elementos repetidos.

Soluciones

a) Notaremos con $0\,(M)$ al par ordenado formado por la fila y columna donde se encuentra el elemento 0 en la matriz M.

- $S = \{M \in \mathcal{M}_{3\times 3}\,([\![0, 8]\!]) : |\{m_{ij}/i, j \in [\![1, 3]\!]\}| = 9\}$.
- $G = \begin{bmatrix} 1 & 2 & 3 \\ 8 & 0 & 4 \\ 7 & 6 & 5 \end{bmatrix}$.
- Sea $(f, c) = 0\,(M)$, definimos los siguientes operadores:
  - Mover arriba:

$$\uparrow (M) = \begin{cases} M & \text{si } f = 1 \\ M' & \text{si } f \neq 1 \end{cases}$$

$$\text{donde } m'_{ij} = \begin{cases} m_{f-1,c} & \text{si } (i, j) = (f, c) \\ 0 & \text{si } (i, j) = (f-1, c) \\ m_{ij} & \text{en otro caso} \end{cases}$$

- Mover abajo:

$$\downarrow (M) = \begin{cases} M & \text{si } f = 3 \\ M' & \text{si } f \neq 3 \end{cases}$$

$$\text{donde } m'_{ij} = \begin{cases} m_{f+1,c} & \text{si } (i,j) = (f,c) \\ 0 & \text{si } (i,j) = (f+1,c) \\ m_{ij} & \text{en otro caso} \end{cases}$$

- Mover a la derecha:

$$\rightarrow (M) = \begin{cases} M & \text{si } c = 3 \\ M' & \text{si } c \neq 3 \end{cases}$$

$$\text{donde } m'_{ij} = \begin{cases} m_{f,c+1} & \text{si } (i,j) = (f,c) \\ 0 & \text{si } (i,j) = (f,c+1) \\ m_{ij} & \text{en otro caso} \end{cases}$$

- Mover a la izquierda:

$$\leftarrow (M) = \begin{cases} M & \text{si } c = 1 \\ M' & \text{si } c \neq 1 \end{cases}$$

$$\text{donde } m'_{ij} = \begin{cases} m_{f,c-1} & \text{si } (i,j) = (f,c) \\ 0 & \text{si } (i,j) = (f,c-1) \\ m_{ij} & \text{en otro caso} \end{cases}$$

b) Sea $\mathbb{L}$ el conjunto de todas las listas con coeficientes en $[\![1, 64]\!]$ cuya longitud no es mayor a 64.

- $S = \{(A, B, C) : A, B, C \in \mathbb{L} \wedge P(A) \wedge P(B) \wedge P(C) \wedge Q(A) \wedge Q(B) \wedge Q(C)\}$, donde:
  - $P(X) = \forall x_i \in X : x_i \leq x_{i+1}$
  - $Q(X) = |X| = |\{x/x \in X\}|$
- $I = ([1, 2, 3, 4, 5, \cdots, 60, 61, 62, 63, 64], [], [])$.
- $G = ([], [], [1, 2, 3, 4, 5, \cdots, 60, 61, 62, 63, 64])$.

Definimos los siguientes operadores:

- $\overrightarrow{ab}\,(A,B,C) = \begin{cases} (A,B,C) & \text{si } a_1 > b_1 \\ (tail\,(A)\,,[a_1]+B,C) & \text{en otro caso} \end{cases}$
- $\overrightarrow{ac}\,(A,B,C) = \begin{cases} (A,B,C) & \text{si } a_1 > b_c \\ (tail\,(A)\,,B,[a_1]+C) & \text{en otro caso} \end{cases}$
- $\overleftarrow{ba}\,(A,B,C) = \begin{cases} (A,B,C) & \text{si } b_1 > a_1 \\ ([b_1]+A,tail\,(B)\,,C) & \text{en otro caso} \end{cases}$
- $\overrightarrow{bc}$: Análogo.
- $\overleftarrow{ca}$: Análogo.
- $\overleftarrow{cb}$: Análogo.

$c)$

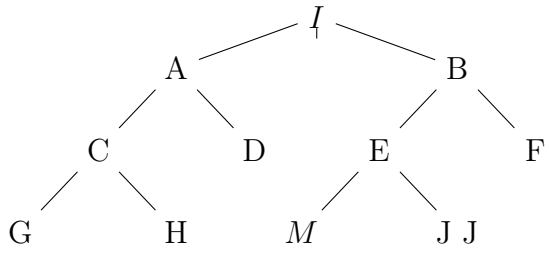
- $S = \{M \in \mathcal{M}_{3\times 3}\,([\![0,3]\!]) : P\,(M) \wedge P\,(M^t)\}$, donde $P\,(M) = \forall i \in [\![1,3]\!]\,(\forall j \in [\![1,3]\!] : m_{ij} = 0 \vee m_{ij} \neq m_{ij+1})$.
- $I = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$.
- $G = \{M \in S / \forall m_{ij} \in M : m_{ij} \neq 0\}$.
- Sea $f\,(M,n,f,c) = M'$ donde $m_{ij} = \begin{cases} n & \text{si } (i,j) = (f,c) \\ m_{ij} & \text{en otro caso} \end{cases}$

  definimos el siguiente operador:

  - $g\,(M,n,f,c) = \begin{cases} M & \text{si } f\,(M,n,f,c) \notin S \\ f\,(M,n,f,c) & \text{en otro caso} \end{cases}$

2. Escribir en pseudocódigo el algoritmo para la búsqueda a lo ancho y en profundidad, a partir del algoritmo de búsqueda general y realizando las especificaciones necesarias para cada caso.

   Dar la evolución de la Lista de Espera de Nodos, al aplicar las estrategias de búsqueda a lo ancho y en profundidad, al problema representado en el siguiente árbol, donde I es el estado inicial y M es el estado meta.



Solución

```
def dfs(problem):
    l = [(problem.startState, [])]

    while l:
        node, actions = l.pop()

        if problem.isGoalState(node):
            return node, actions

        for succ, action, _ in problem.getSuccessors(node):
            newActions = actions + [action]
            l += [(succ, newActions)]
```

- `l ~ ["I"]`.
- `l ~ ["A", "B"]`.
- `l ~ ["C", "D", "B"]`.
- `l ~ ["G", "H", "D", "B"]`.
- `l ~ ["H", "D", "B"]`.
- `l ~ ["D", "B"]`.
- `l ~ ["B"]`.
- `l ~ ["E", "F"]`.
- `l ~ ["M", "J J", "F"]`.

```
def bfs(problem):
    l = [(problem.startState, [])]

    while l:
        node, actions = l.pop()

        if problem.isGoalState(node):
            return node, actions

        for succ, action, _ in problem.getSuccessors(node):
            newActions = actions + [action]
            l = [(succ, newActions)] + l
```

- `l ~ ["I"]`.
- `l ~ ["A", "B"]`.
- `l ~ ["B", "C", "D"]`.
- `l ~ ["C", "D", "E", "F"]`.
- `l ~ ["D", "E", "F", "G", "H"]`.
- `l ~ ["E", "F", "G", "H"]`.
- `l ~ ["F", "G", "H", "M", "J J"]`.
- `l ~ ["G", "H", "M", "J J"]`.
- `l ~ ["H", "M", "J J"]`.
- `l ~ ["M", "J J"]`.

3. Representar el problema de los Misioneros y Caníbales descripto a continuación, mediante un espacio de estados y aplicar búsqueda primero a lo ancho para resolver el problema. Encontrar la solución de menos pasos.

   Tres misioneros y tres caníbales se encuentran en una orilla. Junto a una canoa en la que pueden cruzar 1 o 2 personas. Hay que encontrar la forma de pasarlos a todos a la otra orilla pero teniendo en cuenta que en ningún momento el número de misioneros sea menor que el de los caníbales.

Solución

- $S = \{(m, c, b, M, C) : m, c, M, C \in [\![0, 3]\!] \wedge b \in \mathbb{B} \in \wedge P\,(m, c, M, C)\}$ donde $P\,(m, c, M, C) = (m = 0 \vee m \geq c) \wedge (M = 0 \vee M \geq C) \wedge m + M = 3 \wedge c + C = 3$.
- $I = (3, 3, \top, 0, 0)$.
- $G = (0, 0, \bot, 3, 3)$.
- $f_i\,(m, c, \top, M, C) = (m - i, c, \bot, M + i, C)$ para $1 \leq i \leq 2$ siempre que el resultado pertenezca al espacio de estados.
- $f_i\,(m, c, \bot, M, C) = (m + i, c, \top, M - i, C)$ para $1 \leq i \leq 2$ siempre que el resultado pertenezca al espacio de estados.
- $g_i\,(m, c, \top, M, C) = (m, c - i, \bot, M, C + i)$ para $1 \leq i \leq 2$ siempre que el resultado pertenezca al espacio de estados.
- $g_i\,(m, c, \bot, M, C) = (m, c + i, \top, M, C - i)$ para $1 \leq i \leq 2$ siempre que el resultado pertenezca al espacio de estados.
- $h_i\,(m, c, \top, M, C) = (m - i, c - i, \bot, M + i, C + i)$ para $i = 1$ siempre que el resultado pertenezca al espacio de estados.
- $h_i\,(m, c, \bot, M, C) = (m + i, c + i, \top, M - i, C - i)$ para $i = 1$ siempre que el resultado pertenezca al espacio de estados.

4. En el juego de los 8 números (Taken) con la siguiente configuración inicial:

2	8	3
1	6	4
7		5

   a) Resolverlo mediante la estrategia de búsqueda en profundidad estableciendo algún límite apropiado de profundidad y controlando los nodos de estados repetidos.

   b) Desarrollar las etapas de búsqueda considerando el método A* considerando:

   - h_1 = Cantidad de números fuera de lugar.
   - h_2 = Distancia en cuadras (de Manhattan).

Comparar los resultados obtenidos con ambas heurísticas (solución óptima, cantidad de nodos expandidos) en este caso de resolución y analizar si este comportamiento es un caso particular o si se puede generalizar a otros casos del juego.

Soluciones

a)

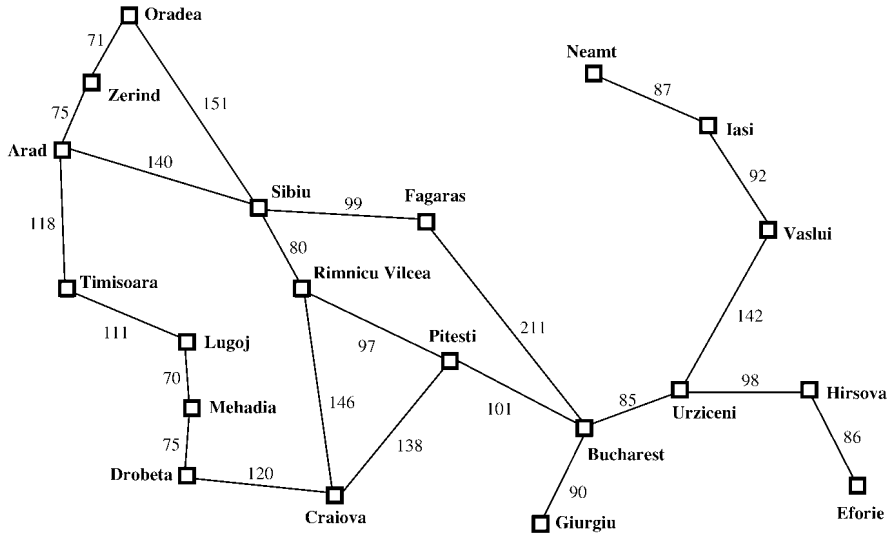
```
def dfsWithLimit(problem, limit):
  l = [(problem.startState, 0, [])]

  while l:
    node, level, actions = l.pop()

    if problem.isGoalState(node):
      return node, actions

    for succ, action, _ in problem.getSuccessors(node):
      if level < limit:
        l += [(succ, level + 1, actions + [action])]

def dfsWithAncestors(problem):
  l = [(problem.startState, [], [])]

  while l:
    node, ancestors, actions = l.pop()

    if problem.isGoalState(node):
      return node, actions

    for succ, action, _ in problem.getSuccessors(node):
      if succ not in ancestors:
        newAncestors = ancestors + [node]
        l += [(succ, newAncestors, actions + [action])]
```

b) COMPLETAR.

5. Considerando el siguiente mapa de Rumania con las rutas existentes y las distancias en línea recta entre las distintas ciudades y Bucharest (Ejemplo extraído de Russell & Norvig).

   a) Aplique el algoritmo de búsqueda de costo uniforme para encontrar una solución al problema de ir desde Arad a Bucharest.

   b) Explorar el árbol de búsqueda con el método A* usando como heurística la distancia en línea recta, indicar en cada nodo el número de expansión y el valor de g y h.



Distancia en linea recta a Bucharest

- Bucharest: 0.
- Giurgiu: 77.
- Urziceni: 80.
- Pitesti: 100.
- Hirsova: 151.
- Craiova: 160.
- Eforie: 161.
- Fagaras: 176.
- Rimnicu Vilcea: 193.
- Vaslui: 199.
- Iasi: 226.
- Neamt: 234.
- Mehadia: 241.
- Drobeta: 242.
- Lugoj: 244.
- Sibiu: 253.
- Timisora: 329.
- Arad: 366.
- Zerind: 374.
- Oradea: 380.

Soluciones

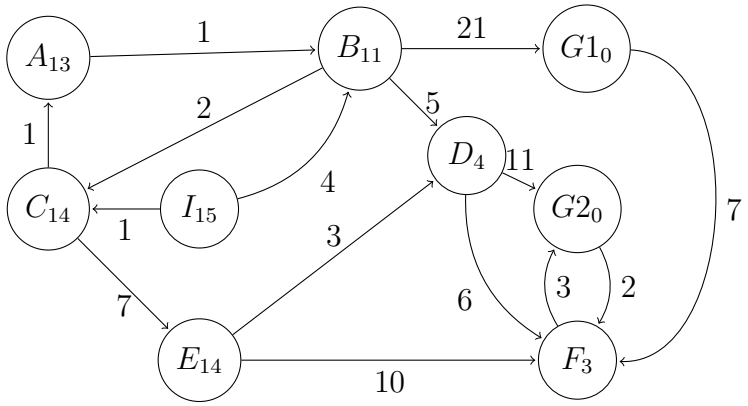
a) Puede observarse en negrita la expansión del nodo actual y debajo de cada nodo, el costo total de ruta.

a	$\underset{\mathbf{75}}{\boldsymbol{A\to Z}}$, $\underset{\mathbf{118}}{\boldsymbol{A\to T}}$, $\underset{\mathbf{140}}{\boldsymbol{A\to S}}$
az	$\underset{118}{A\to T}$, $\underset{140}{A\to S}$, $\boldsymbol{A\to Z\to O}$ **75+71=146**
azt	$\underset{140}{A\to S}$, $\underset{146}{A\to Z\to O}$, $\boldsymbol{A\to T\to L}$ **118+111=229**
azts	$\underset{146}{A\to Z\to O}$, $\boldsymbol{A\to S\to R}$ **140+99=220**, $\underset{229}{A\to T\to L}$, $\boldsymbol{A\to S\to F}$ **140+99=239**, $\boldsymbol{A\to S\to O}$ **140+151=291**
aztso	$\underset{220}{A\to S\to R}$, $\underset{229}{A\to T\to L}$, $\underset{239}{A\to S\to F}$, $\underset{291}{A\to S\to O}$, $\boldsymbol{A\to Z\to O\to S}$ **146+151=297**
aztsor	$\underset{229}{A\to T\to L}$, $\underset{239}{A\to S\to F}$, $\underset{291}{A\to S\to O}$, $\underset{297}{A\to Z\to O\to S}$, $\boldsymbol{A\to S\to R\to P}$ **220+97=317** , $\boldsymbol{A\to S\to R\to C}$ **220+146=366**
aztsorl	$\underset{239}{A\to S\to F}$, $\underset{291}{A\to S\to O}$, $\underset{297}{A\to Z\to O\to S}$, $\boldsymbol{A\to T\to L\to M}$ **229+70=299** , $\underset{317}{A\to S\to R\to P}$, $\underset{366}{A\to S\to R\to C}$
aztsorlf	$\underset{291}{A\to S\to O}$, $\underset{297}{A\to Z\to O\to S}$, $\underset{299}{A\to T\to L\to M}$, $\underset{317}{A\to S\to R\to P}$, $\underset{366}{A\to S\to R\to C}$, $\boldsymbol{A\to S\to F\to B}$ **239+211=450**
aztsorlf	$\underset{297}{A\to Z\to O\to S}$, $\underset{299}{A\to T\to L\to M}$, $\underset{317}{A\to S\to R\to P}$, $\underset{366}{A\to S\to R\to C}$, $\underset{450}{A\to S\to F\to B}$
aztsorlf	$\underset{299}{A\to T\to L\to M}$, $\underset{317}{A\to S\to R\to P}$, $\underset{366}{A\to S\to R\to C}$, $\underset{450}{A\to S\to F\to B}$
aztsorlfm	$\underset{317}{A\to S\to R\to P}$, $\underset{366}{A\to S\to R\to C}$, $\boldsymbol{A\to T\to L\to M\to D}$ **299+75=374** , $\underset{450}{A\to S\to F\to B}$
aztsorlfmp	$\underset{366}{A\to S\to R\to C}$, $\underset{374}{A\to T\to L\to M\to D}$, $\boldsymbol{A\to S\to R\to P\to B}$ **317+101=418** , $\underset{450}{A\to S\to F\to B}$, $\boldsymbol{A\to S\to R\to P\to C}$ **317+138=455**
aztsorlfmpc	$\underset{374}{A\to T\to L\to M\to D}$, $\underset{418}{A\to S\to R\to P\to B}$, $\underset{450}{A\to S\to F\to B}$, $\underset{455}{A\to S\to R\to P\to C}$, $\boldsymbol{A\to S\to R\to C\to D}$ **366+120=486** , $\boldsymbol{A\to S\to R\to C\to P}$ **366+138=504**
aztsorlfmpcd	$\boxed{\underset{418}{A\to S\to R\to P\to B}}$, $\underset{450}{A\to S\to F\to B}$, $\underset{455}{A\to S\to R\to P\to C}$, $\underset{486}{A\to S\to R\to C\to D}$, $\boldsymbol{A\to T\to L\to M\to D\to C}$ **374+120=494** , $\underset{504}{A\to S\to R\to C\to P}$

b) Puede observarse debajo de cada nodo $g(n) + h(n) = f(n)$.

a	$\boldsymbol{A\to S}$ **140+253=393**, $\boldsymbol{A\to T}$ **118+329=447**, $\boldsymbol{A\to Z}$ **75+374=449**
as	$\boldsymbol{A\to S\to R}$ **220+80=300**, $\boldsymbol{A\to S\to F}$ **239+178=417**, $A\to T$ 118+329=447, $A\to Z$ 75+374=449, $\boldsymbol{A\to S\to O}$ **291+380=671**
asr	$\boldsymbol{A\to S\to R\to P}$ **317+98=415** , $A\to S\to F$ 239+178=417, $A\to T$ 118+329=447, $A\to Z$ 75+374=449, $\boldsymbol{A\to S\to R\to C}$ **366+160=526** , $A\to S\to O$ 291+380=671
asrp	$A\to S\to F$ 239+178=417, $\boldsymbol{A\to S\to R\to P\to B}$ **418+0=418** , $A\to T$ 118+329=447, $A\to Z$ 75+374=449, $A\to S\to R\to C$ 366+160=526 , $\boldsymbol{A\to S\to R\to P\to C}$ **455+160=615** , $A\to S\to O$ 291+380=671
asrpf	$\boxed{A\to S\to R\to P\to B}$ 418+0=418, $A\to T$ 118+329=447, $A\to Z$ 75+374=449, $\boldsymbol{A\to S\to F\to B}$ **450+0=450** , $A\to S\to R\to C$ 366+160=526 , $A\to S\to R\to P\to C$ 455+160=615 , $A\to S\to O$ 291+380=671

6. Considérese el grafo dirigido de la figura, que representa un espacio de estados, siendo I el estado inicial y $G1$ y $G2$ los dos estados objetivos. El número que figura en cada estado corresponde al valor de una función heurística h' que estima el coste mínimo necesario para pasar de ese estado al objetivo más cercano. Cada arista está etiquetada con un número que representa el coste real de atravesar dicha arista.



a) ¿Es h' una heurística admisible? Justifique.

b) Indicar qué estado objetivo se alcanzará (si es que se alcanza alguno), qué estados se expandirán y en qué orden para cada uno de los siguientes algoritmos de búsqueda:

1) Primero en profundidad.
2) A*.
3) Escalada simple (hill climbing).

Cuando dos nodos tengan las mismas características por el criterio de selección que se esté usando, se seleccionará el primero por orden alfabético. Además, se evitarán las repeticiones de estados.

Soluciones

a) No es admisible pues $h'(E) = 14 > h^*(E) = 12$.

b)

1) Primero en profundidad con chequeo de ancestros:

i	$\boldsymbol{I \to B}, \boldsymbol{I \to C}$
ib	$\boldsymbol{I \to B \to C}, \boldsymbol{I \to B \to D}, \boldsymbol{I \to B \to G1}, I \to C$
ibc	$\boldsymbol{I \to B \to C \to A}, \boldsymbol{I \to B \to C \to E}, I \to B \to D, I \to B \to G1, I \to C$
ibca	$I \to B \to C \to E, I \to B \to D, I \to B \to G1, I \to C$
ibcae	$\boldsymbol{I \to B \to C \to E \to D}, \boldsymbol{I \to B \to C \to E \to F}, I \to B \to D, I \to B \to G1, I \to C$
ibcaed	$\boldsymbol{I \to B \to C \to E \to D \to F}, \boldsymbol{I \to B \to C \to E \to D \to G2}, I \to B \to C \to E \to F, I \to B \to D, I \to B \to G1, I \to C$
ibcaedf	$\boxed{I \to B \to C \to E \to D \to G2}, I \to B \to C \to E \to F, I \to B \to D, I \to B \to G1, I \to C$

2) A*:

i	$\underset{\mathbf{4+11=15}}{\boldsymbol{I\to B}}$, $\underset{\mathbf{1+14=15}}{\boldsymbol{I\to C}}$
ib	$\underset{\mathbf{9+4=13}}{\boldsymbol{I\to B\to D}}$, $\underset{1+14=15}{I\to C}$, $\underset{\mathbf{6+14=20}}{\boldsymbol{I\to B\to C}}$, $\underset{\mathbf{25+0=25}}{\boldsymbol{I\to B\to G1}}$
ibd	$\underset{1+14=15}{I\to C}$, $\underset{\mathbf{15+3=18}}{\boldsymbol{I\to B\to D\to F}}$, $\underset{6+14=20}{I\to B\to C}$, $\underset{\mathbf{20+0=20}}{\boldsymbol{I\to B\to D\to G2}}$, $\underset{25+0=25}{I\to B\to G1}$
ibdc	$\underset{\mathbf{2+14=16}}{\boldsymbol{I\to C\to A}}$, $\underset{15+3=18}{I\to B\to D\to F}$, $\underset{6+14=20}{I\to B\to C}$, $\underset{20+0=20}{I\to B\to D\to G2}$, $\underset{\mathbf{8+14=22}}{\boldsymbol{I\to C\to E}}$, $\underset{25+0=25}{I\to B\to G1}$
ibdca	$\underset{\mathbf{3+11=14}}{\boldsymbol{I\to C\to A\to B}}$, $\underset{15+3=18}{I\to B\to D\to F}$, $\underset{6+14=20}{I\to B\to C}$, $\underset{20+0=20}{I\to B\to D\to G2}$, $\underset{8+14=22}{I\to C\to E}$, $\underset{25+0=25}{I\to B\to G1}$
ibdca	$\underset{15+3=18}{I\to B\to D\to F}$, $\underset{6+14=20}{I\to B\to C}$, $\underset{20+0=20}{I\to B\to D\to G2}$, $\underset{8+14=22}{I\to C\to E}$, $\underset{25+0=25}{I\to B\to G1}$
ibdcaf	$\boxed{\underset{\mathbf{18+0=18}}{\boldsymbol{I\to B\to D\to F\to G2}}}$, $\underset{6+14=20}{I\to B\to C}$, $\underset{20+0=20}{I\to B\to D\to G2}$, $\underset{8+14=22}{I\to C\to E}$, $\underset{25+0=25}{I\to B\to G1}$

3) Escalada simple: COMPLETAR.

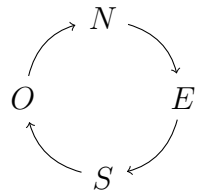
7. Un robot se mueve en una grilla de $n \times n$. En un determinado momento, el robot se encontrará en un celda (determinada por su posición x indicando la fila y la posición y indicando la columna), con una orientación la cual puede tener los siguientes valores: N (Norte), S (Sur), E (Este) y O (Oeste).

   Los movimientos que puede realizar el robot en esta grilla son:

   - Avanza: el robot se dirige a la celda contigua (costo 2). Esta celda depende de la orientación del robot. La tabla siguiente resume la ubicación del robot después de ejecutar esta acción:

Orientación	Próxima ubicación
N	$x-1, y$
S	$x+1, y$
E	$x, y+1$
O	$x, y-1$

   - Gira: el robot gira 90^{o} en el sentido de las agujas del reloj. El siguiente gráfico muestra la rotación del robot (costo 1)



El robot tendrá una posición inicial en la grilla y se le solicitará que llegue a alguna celda destino. Además se especificarán un grupo de celdas prohibidas. Por lo tanto, el robot tendrá que resolver como problema determinar el camino a seguir en la grilla para alcanzar el destino final, sin pasar por las celdas prohibidas.

Se pide:

a) Representar el problema de desplazamiento del robot mediante un espacio de estados.

b) Definir una función heurística y aplicar búsqueda A* para hallar el camino de menor costo entre (1,1) si el robot comienza con orientación N y (3,4) cualquier orientación, en una grilla de (5×5), teniendo como celdas prohibidas: (1,2), (1,4) y (2,2).

Soluciones

a) COMPLETAR.

b) COMPLETAR.

8. Dado el siguiente problema, consistente en partir de una configuración $[0X0X0X0X]$ pasar a otra $[0000XXXX]$, moviendo de a dos letras adyacentes a la vez (cualquier par) desplazándolas juntas y eliminando el hueco dejado.

   a) Representar el problema mediante espacio de estados.

   b) Proponer una heurística e implementar un método de búsqueda de modo de obtener la configuración deseada con el menor número de movimientos.

   Soluciones

   a) COMPLETAR.

   b) COMPLETAR.

9. Supóngase que usted está diseñando un compilador para una máquina con un solo registro y las siguientes instrucciones:

UNO	Poner en el registro el valor 1
DOBLE	Duplicar el contenido del registro
SUMAR_UNO	Sumar uno al contenido del registro
RESTAR_UNO	Restar uno al contenido del registro

Estas instrucciones sólo pueden ejecutarse cuando el contenido del registro no es divisible por 3. Cuando esto ocurre, la única instrucción disponible es:

DIVIDIR	Dividir el valor del registro por 3

El problema es cómo poner el número 7 en el registro. Puede suponerse que el registro está inicializado en 1.

a) Formular este problema en términos de espacios de estado.

b) Supóngase que los costos de los operadores sobre un registro que contiene el número n son:

UNO	1
DOBLE	N
SUMAR_UNO	1
RESTAR_UNO	1
DIVIDIR	2N/3

Esto es, cada operador tiene un costo equivalente a la diferencia que produce respecto del valor inicial del registro, y la función heurística $h(n) = |7 - n|$. Dibuje el árbol de búsqueda utilizando A* y determine la solución encontrada.

Soluciones

a) COMPLETAR.

b) COMPLETAR.

10. Considere un problema de búsqueda en un universo cuyos estados sean: A, B, C, D, E. Las acciones posibles, en este universo, son dadas por la siguiente tabla:

Inicial	Final	Costo
A	B	3
A	C	2
B	C	5
B	D	3
C	D	5
C	B	1
C	E	11
D	E	5
E	B	8

$nodo$	$h_1(nodo)$	$nodo$	$h_2(nodo)$
A	8	A	7
B	6	B	8
C	6	C	5
D	4	D	5
E	0	E	0

a) ¿Es A* optimo con la heurística h_1? ¿Es A* óptima con h_2?

b) ¿Son las siguientes heurísticas admisibles?

1) $h_1(n)$
2) $h_2(n)$
3) mín $\{h_1(n), h_2(n)\}$
4) $h_1(n) + h_2(n)$
5) $[h_1(n) + h_2(n)]/2$
6) máx $\{h_1(n), h_2(n)\}$
7) $h_1(n) \cdot h_2(n)$
8) $[h_1(n) \cdot h_2(n)]/2$

Soluciones

a) Puesto que ambas heurísticas son admisibles, podemos garantizar entonces que tambien son completas y optimas.

b)

1) Verdadero.
2) Verdadero.
3) Verdadero: puesto que h_1 y h_2 son admisibles, es facil ver que también lo será el mínimo entre ellas.
4) Falso: para el nodo D resulta $h_1(D) + h_2(D) = 4 + 5 = 9 > h^*(D) = 5$.
5) Verdadero.
6) Verdadero: puesto que h_1 y h_2 son admisibles, es facil ver que también lo será el máximo entre ellas.
7) Falso: para el nodo D resulta $h_1(D) \cdot h_2(D) = 4 \cdot 5 = 20 > h^*(D) = 5$.
8) Falso: para el nodo D resulta $h_1(D) \cdot h_2(D)/2 = 4 \cdot 5/2 = 10 > h^*(D) = 5$.

11. Consideremos un puzzle consistente en una fila con 4 fichas con la siguiente configuración inicial:

N	N	B	B	V

Hay dos fichas negras (N), dos blancas (B) y una casilla vacía (V). El puzzle permite los siguientes movimientos:

- Una ficha puede moverse a una casilla vacía adjunta con costo unidad.
- Una ficha puede saltar sobre otras dos como máximo hasta la celda vacía, con costo igual al número de fichas saltadas.
- El objetivo es llegar a tener todas las fichas blancas a la izquierda de todas las negras (sin importar la posición de la casilla vacía).

a) Especificar el problema como búsqueda en un espacio de estados.

b) Especificar una función heurística h para este problema, analizar si es admisible.

c) Encontrar una solución utilizando el algoritmo A* con esa función heurística. Mostrar el árbol de búsqueda desarrollado.

Soluciones

a)

- $S = \left\{ M \in \mathcal{M}_{1\times 5}\left(\{6, 1, -1\}\right) : \sum_{i=1}^{5} m_i = 6 \right\}$.
- $I = (1, 1, -1, -1, 6)$ (puede observarse que el 6 representa al vacío, 1 a las negras y -1 a las blancas).
- $G = \{M \in S / \forall m_i \in X : m_i = 6 \vee m_i \leq m_{i+1}\}$.
- Sea $V(M)$ el indice del casillero vacío, definimos los siguientes operadores:
  - $\leftarrow (M) = \begin{cases} M & \text{si } V(M) = 1 \\ M' & \text{en otro caso} \end{cases}$ donde $m'_i = \begin{cases} 6 & \text{si } i = V(M) - 1 \\ m_{i-1} & \text{si } i = V(M) \\ m_i & \text{en otro caso} \end{cases}$
  - $\leftarrow^2 (M) = \begin{cases} M & \text{si } V(M) \leq 2 \\ M' & \text{en otro caso} \end{cases}$ donde $m'_i = \begin{cases} 6 & \text{si } i = V(M) - 2 \\ m_{i-2} & \text{si } i = V(M) \\ m_i & \text{en otro caso} \end{cases}$
  - $\leftarrow^3 (M) = \begin{cases} M & \text{si } V(M) \leq 3 \\ M' & \text{en otro caso} \end{cases}$ donde $m'_i = \begin{cases} 6 & \text{si } i = V(M) - 3 \\ m_{i-3} & \text{si } i = V(M) \\ m_i & \text{en otro caso} \end{cases}$
  - Pueden definirse análogamente los operadores $\rightarrow$, $\rightarrow^2$ y $\rightarrow^3$.

b) Sea M^* la matriz resultante de eliminar la columna que contiene al vacío de la matriz M, definimos:

$$h(M) = m_1^* \cdot p(m_1^*) + m_2^* \cdot p(m_2^*) + m_3^* \cdot q(m_3^*) + m_4^* \cdot q(m_4^*)$$

donde $p(-1) = 0, p(1) = 1, q(-1) = 1, q(1) = 0$. Observemos que esta heurística cuenta la cantidad de piezas fuera de lugar, por lo cual siempre será menor o igual al costo de moverla al casillero vacío; luego la heurística es admisible.

c) Arriba de cada tupla pueden observarse los operadores utilizados a partir del estado inicial, y debajo $g(M) + h(M) = f(M)$. En negrita se observan los nodos agregados a la lista. Vale la pena notar que no se agregan nodos repetidos.

$$\overset{\mathbf{(1,1,-1,-1,6)}}{\underset{\mathbf{0+4=4}}{}}$$

$$\overset{\leftarrow^{\mathbf{3}}}{\underset{\mathbf{2+2=4}}{\mathbf{(1,6,-1,-1,1)}}};\ \overset{\leftarrow}{\underset{\mathbf{1+4=5}}{\mathbf{(1,1,-1,6,-1)}}};\ \overset{\leftarrow^{\mathbf{2}}}{\underset{\mathbf{1+4=5}}{\mathbf{(1,1,6,-1,-1)}}}$$

$$\overset{\leftarrow^{\mathbf{3}},\rightarrow^{\mathbf{2}}}{\underset{\mathbf{3+2=5}}{\mathbf{(1,-1,-1,6,1)}}};\ \overset{\leftarrow^{\mathbf{3}},\leftarrow}{\underset{\mathbf{3+2=5}}{\mathbf{(6,1,-1,-1,1)}}};\ \overset{\leftarrow^{\mathbf{3}},\rightarrow}{\underset{\mathbf{3+2=5}}{\mathbf{(1,-1,6,-1,1)}}};\ \overset{\leftarrow}{\underset{1+4=5}{(1,1,-1,6,-1)}};\ \overset{\leftarrow^{2}}{\underset{1+4=5}{(1,1,6,-1,-1)}}$$

$$\boxed{\overset{\leftarrow^{\mathbf{3}},\rightarrow^{\mathbf{2}},\leftarrow^{\mathbf{3}}}{\underset{\mathbf{5+0=5}}{\mathbf{(6,-1,-1,1,1)}}}};\ \overset{\leftarrow^{3},\leftarrow}{\underset{3+2=5}{(6,1,-1,-1,1)}};\ \overset{\leftarrow^{3},\rightarrow}{\underset{3+2=5}{(1,-1,6,-1,1)}};\ \overset{\leftarrow}{\underset{1+4=5}{(1,1,-1,6,-1)}};\ \overset{\leftarrow^{2}}{\underset{1+4=5}{(1,1,6,-1,-1)}};\ \overset{\leftarrow^{\mathbf{3}},\rightarrow^{\mathbf{2}},\rightarrow}{\underset{\mathbf{4+2=6}}{\mathbf{(1,-1,-1,1,6)}}}$$

12. Hay una sola pila de 7 monedas. Se debe dividir la pila original en sucesivas pilas tal que en cada paso, una pila se descompone en 2 de distinto tamaño. Esta tarea continúa hasta que todas las pilas tienen solo una o dos monedas. Se plantea encontrar una solución que maximice la cantidad de pilas con una moneda. Se pide entonces:

a) Representar el problema en espacio de estados definiendo los operadores para resolver el problema de descomposición de forma general.

b) Definir una heurística y aplicar el algoritmo A.*

Soluciones

a) COMPLETAR.

b) COMPLETAR.

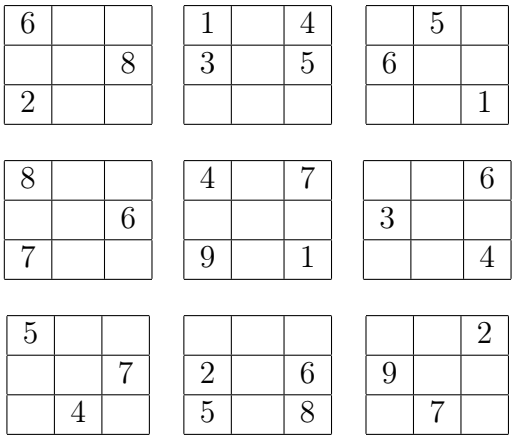
13. Representar como un problema de Satisfacción de Restricciones (CSP) al problema de las 5 reinas descripto a continuación. La meta del problema de las 5 reinas es poner 5 reinas en un tablero de ajedrez de 5×5, de manera que ninguna de ellas esté en posibilidad de atacarse mutuamente (no esté en la misma fila, o columna o diagonal). Generalice la representación al problema de N reinas.

    Solución COMPLETAR.

14. Represente como un CSP el conocido problema criptoaritmético «send+more=money» donde debe asignarse un dígito distinto a cada letra.

    Solución COMPLETAR.

15. Considere el siguiente Sudoku y represéntelo como un CSP:



Solución COMPLETAR.