Ejercicio 1. Rush hour (22/30 puntos)
Tenemos un tablero con N filas y M columnas. El juego Rush Hour contiene un tablero donde varios coches (ocupan 2 casillas) y camiones (3 casillas) se encuentran atrapados en un atasco.

El juego consiste en encontrar una serie de movimientos de los coches y camiones de manera que el coche rojo llegue a la salida del atasco.

Por ejemplo, el siguiente tablero tiene la siguiente solución:

Imagen tomada de: http://heuristieken.nl/wiki/index.php?title=File:Rushhour6x6_1.jpg#file

```
-------------------------
| . | . | a | b | b | c |
-------------------------
| . | . | a | . | . | c |
-------------------------
| . | . | a | x | x | c |   ← Salida
-------------------------
| . | . | . | d | e | e |
-------------------------
| f | g | g | d | . | . |
-------------------------
| f | . | . | d | h | h |
-------------------------
```

[('a', 'down'), ('f', 'up'), ('f', 'up'), ('f', 'up'), ('f', 'up'), ('g', 'left'), ('a', 'down'), ('a', 'down'), ('x', 'left'), ('x', 'left'), ('x', 'left'), ('a', 'up'), ('a', 'up'), ('g', 'right'), ('d', 'up'), ('d', 'up'), ('g', 'right'), ('g', 'right'), ('a', 'down'), ('a', 'down'), ('x',

'right'), ('b', 'left'), ('b', 'left'), ('d', 'up'), ('e', 'left'), ('c', 'down'), ('f', 'down'), ('b', 'left'), ('f', 'down'), ('f', 'down'), ('f', 'down'), ('h', 'left'), ('x', 'left'), ('a', 'up'), ('a', 'up'), ('a', 'up'), ('e', 'left'), ('e', 'left'), ('d', 'down'), ('e', 'left'), ('a', 'down'), ('b', 'right'), ('b', 'right'), ('b', 'right'), ('a', 'up'), ('b', 'right'), ('d', 'up'), ('e', 'right'), ('e', 'right'), ('e', 'right'), ('a', 'down'), ('a', 'down'), ('f', 'up'), ('h', 'left'), ('h', 'left'), ('h', 'left'), ('a', 'down'), ('x', 'right'), ('f', 'up'), ('f', 'up'), ('f', 'up'), ('x', 'left'), ('a', 'up'), ('a', 'up'), ('a', 'up'), ('e', 'left'), ('e', 'left'), ('e', 'left'), ('a', 'down'), ('g', 'left'), ('g', 'left'), ('g', 'left'), ('a', 'down'), ('a', 'down'), ('x', 'right'), ('d', 'down'), ('d', 'down'), ('d', 'down'), ('x', 'right'), ('x', 'right'), ('c', 'down'), ('c', 'down'), ('x', 'right')]

Se pide:

(a) Determinar el número de estados posibles para un tablero de dimensión MxN con K coches. Por simplificar, supondremos que todos los coches son de longitud 2.
1/20: Suponiendo que solo hay 3 coches.
1/20: Suponiendo que solo hay k coches.

(b) (2/20) Indica cómo representarías un estado de ese juego y no olvides justificarlo.

(c) (2/20) Indica cuál es el factor de ramificación y no olvides justificarlo.

(d) (2/20) Indica cuál(es) es/son el estado(s) final(es).

(e) (2/20) Determina un heurístico para este problema. Justifica si sería admisible o no.

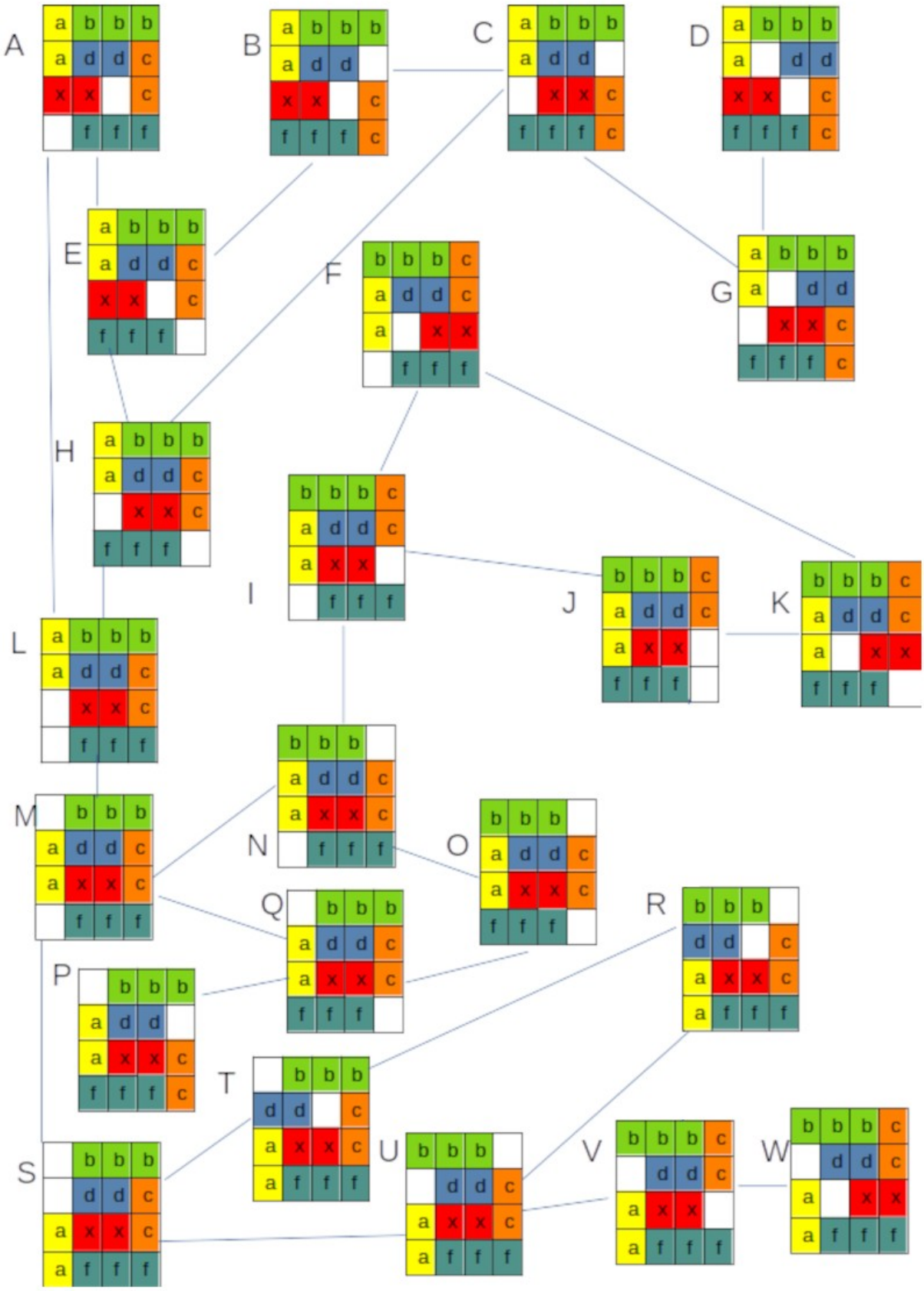
Dado el grafo de estados, se pide calcular el árbol de búsqueda, el conjunto de nodos expandidos (en el orden en que se expanden) y la solución (camino) obtenido por los siguientes algoritmos. En caso de empate, se elegirá por orden alfabético. Para el camino, la respuesta se debe dar en la forma ‘S − A − D − G.’.

(f) (2 puntos) ¿Qué estados expandirá y qué camino devolverá DFS para este problema de búsqueda?

(g) (2 puntos) ¿Qué estados expandirá y qué camino devolverá BFS para este problema de búsqueda?

(h) (3 puntos) Supongamos que los movimientos tienen un costo asociado dependiendo del tipo de vehículo. Algunos vehículos consumen menos gasolina, y su movimiento tiene un coste de 1, mientras que en otros vehículos el movimiento tiene un coste de 2. En el ejemplo anterior, sabemos que los vehículos a, b y f no son eficientes energéticamente, y los demás vehículos sí.
¿Qué estados expandirá y qué camino devolverá UCS para este problema de búsqueda?

(i) (3 puntos) ¿Qué estados expandirá y qué camino devolverá A* para este problema de búsqueda usando el heurístico previamente definido? (este ejercicio se hará sin tener en cuenta la consideración de l apregunta (I)).


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W

(j) (2/20) Implementa la función getSuccessors() que, dado un estado, obtenga todos los estados válidos accesibles desde ese estado.

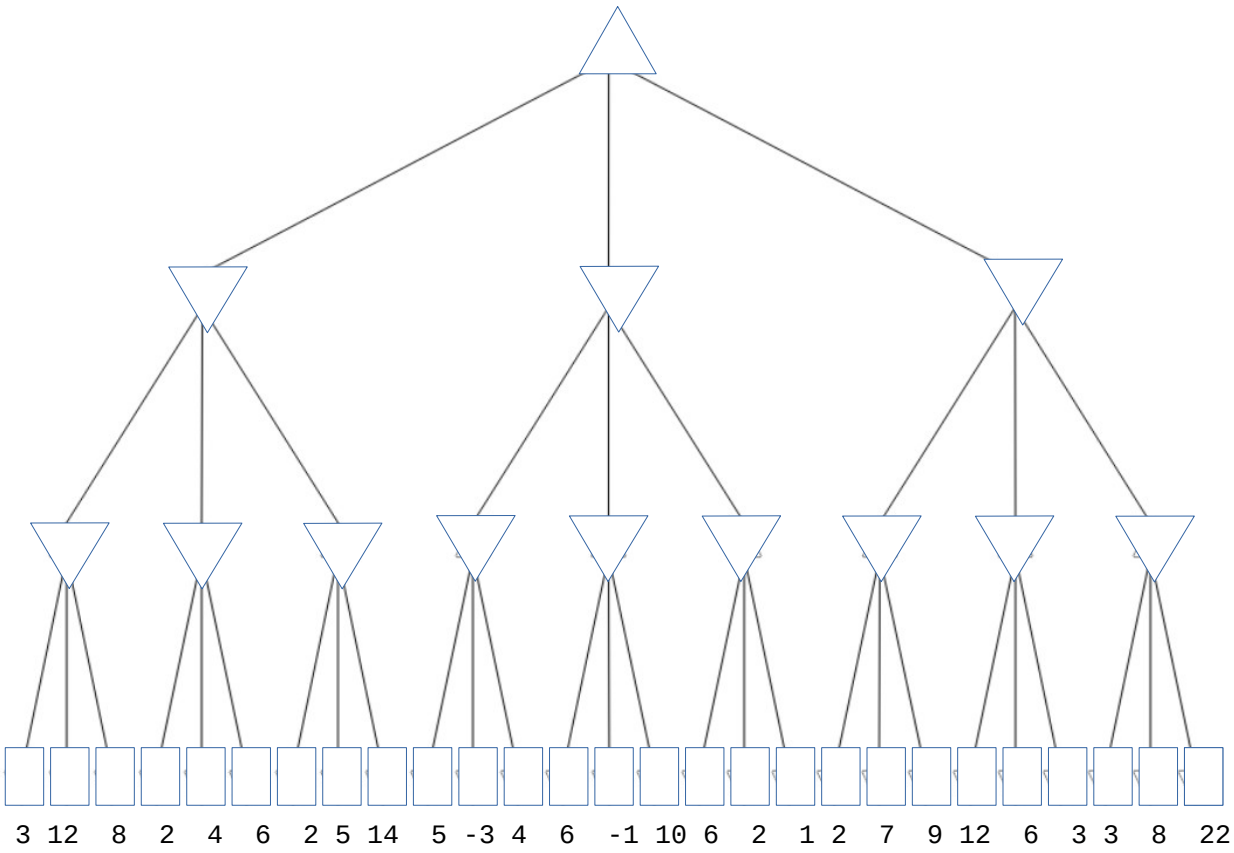
Primeramente, define la representación que usarás para codificar un estado:

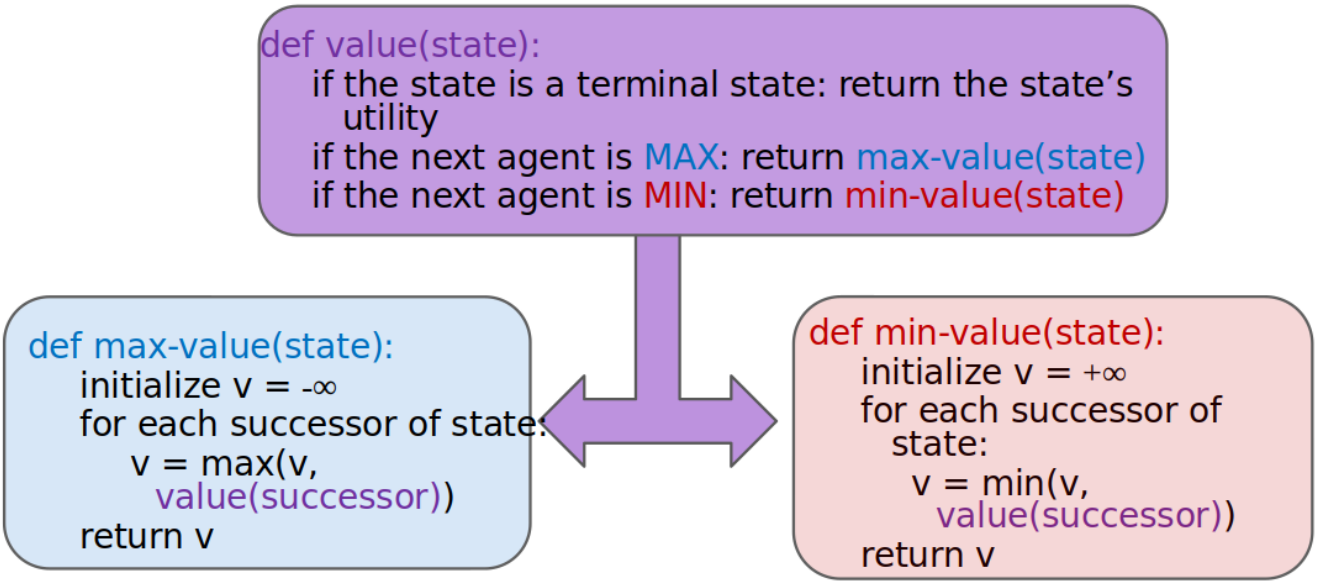
```
def getSuccessors(self, state):
   succs = []

    return succs
```

Ejercicio 2. (8/30 puntos)
En una nueva versión del juego, se tiene un tipo de coche especial cuyo objetivo es chocar contra el coche rojo. Este tipo de coche solo ocupa una casilla y puede moverse una posición en las 4 direcciones.
Por ejemplo, en el siguiente tablero, los coches adversarios se representan por medio de los caracteres * y #:

```
-------------------------
| . | . | a | b | b | c |
-------------------------
| . | . | a | . | . | c |
-------------------------
| . | . | a | x | x | c |
-------------------------
| . | . | . | d | e | e |
-------------------------
| f | g | g | d | . | * |
-------------------------
| f | # | . | d | h | h |
-------------------------
```

a) (2 puntos) Calcular el valor minimax del nodo raíz (sin poda alfa beta)





En la versión Alfa-Beta el Dispacher es el mismo con una mínima adaptación que consiste en añadir Alfa y Beta como parámetros para informar a los hijos de los valores que han ido tomando

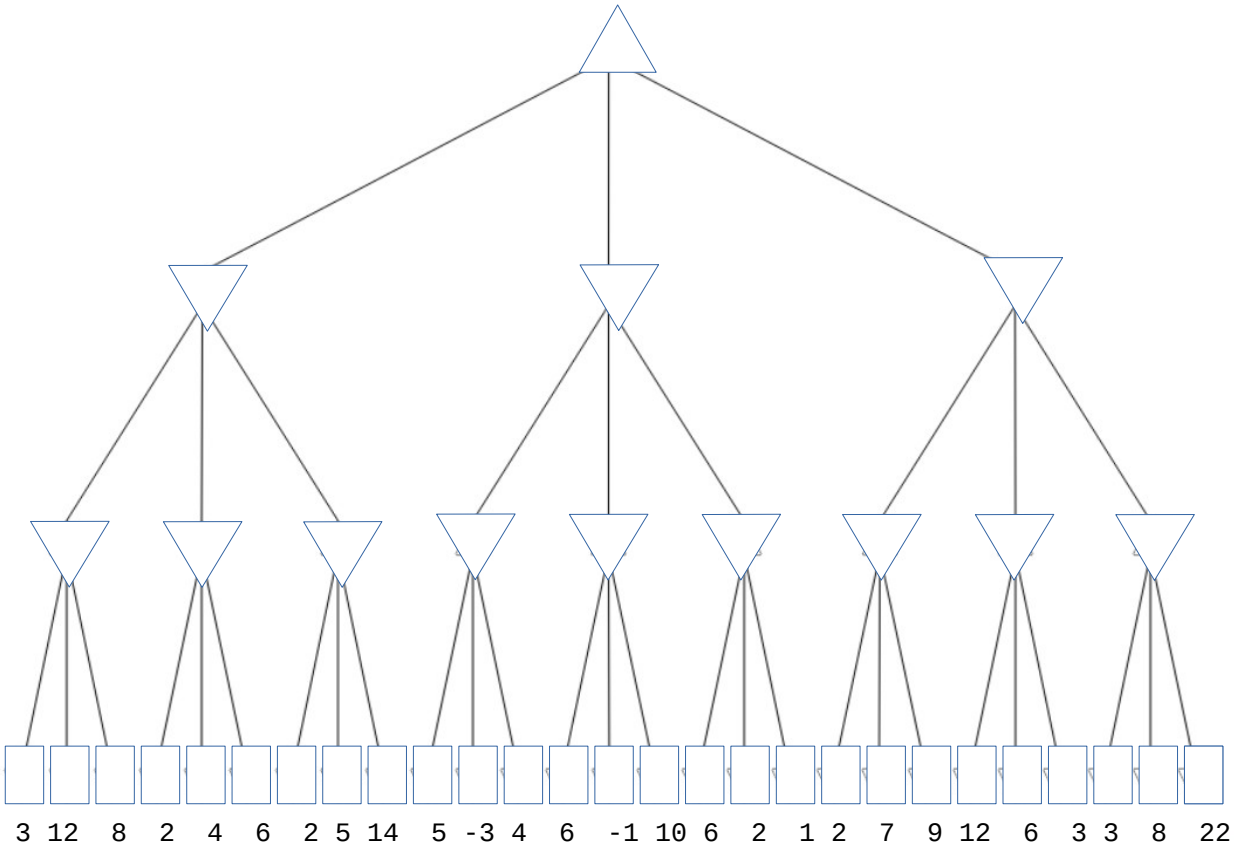
α: la mejor opción de MAX's en el camino a la raíz
β: la mejor opción de MIN's en el camino a la raíz

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

b) (3 puntos) Calcular el valor minimax del nodo raíz usando poda alfa beta.

¿Cuántos nodos elimina el algoritmo alfa beta? Indica ese valor como el porcentaje sobre el total de nodos del árbol de búsqueda.



c) (3 puntos) En una nueva variante del juego, una vez que nuesro coche realiza un movimiento, dependiendo del estado del terreno (mojado, resbaladizo, …) nuestro coche se vuelve a mover de manera aleatoria. Posteriormente, le toca mover al coche adversario. Calcula el valor del nodo en la raíz del árbol.