Hnefatafl

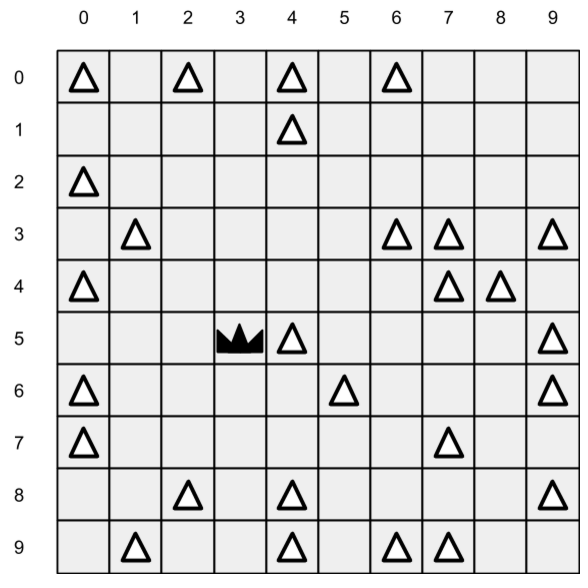
Hnefatafl es un antiguo juego vikingo similar al ajedrez, en el que un jugador tiene un pequeño ejército con un rey que debe escapar, mientras el otro jugador tiene un ejército de más fichas y su objetivo es capturar al rey antes de que escape. En este problema vamos a resolver una versión super simplificada, en la que los soldados se encuentran fijos (no se mueven), y el rey está solo intentando escapar sin ser atacado.

En esta versión el tablero tiene dimensiones de 10x10, y los soldados se encuentran en las posiciones del diagrama “Posiciones iniciales”. El jugador debe lograr que el rey llegue a cualquiera de los bordes del tablero en la menor cantidad posible de movimientos, pero sin ser atacado por ninguno de los soldados en ningún momento (incluso cuando llega al borde).

Los soldados atacan desde casillas adyacentes (no en diagonal), pero como el rey también sabe combatir, los soldados solo pueden matar al rey cuando lo superan en número. Es decir, cuando el rey se encuentra adyacente a un solo soldado, está a salvo, pero cuando se encuentra adyacente a dos o más soldados, es derrotado y muere.

Nunca puede haber dos fichas en la misma posición y los movimientos solo son a casilleros adyacentes (arriba, abajo, izquierda o derecha, no en diagonal, de a 1 casillero).

Posiciones iniciales:



La posición inicial del rey puede variar, vamos a intentar con distintas posiciones que son más o menos difíciles de resolver.

Ejercicios:

1. Implementar la formulación del problema como problema de búsqueda tradicional para ser resuelto con SimpleAI, incluyendo definición de la clase problema y sus métodos: cost, actions, result, is\_goal y heuristic (la heurística puede ser poco precisa, pero **debe** ser admisible).
2. Ejecutar los siguientes métodos de búsqueda, y por cada uno indicar la cantidad de nodos visitados, la profundidad de la solución, el costo de la solución, y el tamaño máximo alcanzado por la frontera:

(las posiciones se expresan como (fila, columna))

* Caso 1: Búsqueda en amplitud, en árbol, partiendo con el rey en la posición (5, 3).
* Caso 2: Búsqueda en amplitud, en grafo, partiendo con el rey en la posición (5, 3).
* Caso 3: Búsqueda en profundidad, en árbol, partiendo con el rey en la posición (5, 3).
* Caso 4: Búsqueda en profundidad, en grafo, partiendo con el rey en la posición (5, 3).
* Caso 5: Búsqueda avara, en árbol, partiendo con el rey en la posición (5, 3).
* Caso 6: Búsqueda avara, en grafo, partiendo con el rey en la posición (5, 3).
* Caso 7: Búsqueda A\*, en árbol, partiendo con el rey en la posición (5, 3).
* Caso 8: Búsqueda A\*, en grafo, partiendo con el rey en la posición (5, 3).

En el formato de entrega se explica qué hacer frente a casos de falta de memoria, demasiado tiempo, o bucles infinitos.

Formato de entrega:

La resolución del ejercicio debe realizarse en un archivo llamado entrega\_1\_tradicional.py, que debe ser subido a la raíz del repositorio git/mercurial del grupo.

El módulo debe tener una función llamada resolver, que reciba los siguientes parámetros:

* metodo\_busqueda: el nombre del método de búsqueda a ejecutar, como string con nombres **exáctamente** iguales a los nombres de las funciones en SimpleAI (ej: 'astar', o 'breadth\_first').
* posicion\_rey: la posición desde la que inicia el juego el rey. Es una tupla, donde el primer elemento es un número representando la fila, y el segundo un número representando la columna.
* controlar\_estados\_repetidos: un booleano indicando si se deben o no controlar estados repetidos en la búsqueda (lo que debe pasarse comograph\_search al método de SimpleAI).

Al llamar a esta función, se debe ejecutar la búsqueda especificada y devolver el **nodo** resultante (lo que devuelve el método de búsqueda de SimpleAI).

Las estadísticas deben subirse también al repositorio en un archivo llamado entrega\_1\_tradicional.txt con una linea por cada caso, con el siguiente formato para cada una:

numero\_de\_caso:cantidad\_nodos\_visitados,profundidad\_solucion,costo\_solucion,largo\_maximo\_frontera

Ejemplo:

1:100,23,10,50

2:60,20,8,40

3:-1,-1,-1,-1

4:...

...

Si alguno de los métodos quedó bloqueado en un bucle infinito, o se quedaron sin memoria, o demoró más de 1 hora en ejecutarse, incluir la linea en el archivo, pero con todos valores en -1, y agregar un segundo archivo llamado explicaciones.txt donde expliquen el motivo.

**Respetar** nombres de archivos, funciones, parámetros y tipos de datos **exáctamente** como se dicen en este enunciado. Cualquier falla por no respetar la interfaz definida, se considera no entregado.

Si quieren probar que lo están haciendo de manera correcta, pueden descargar el script llamado probar\_entrega\_1.py del repositorio de la materia (directorio 2016), y luego de posicionarlo en **el mismo** directorio que su entrega, ejecutarlo de esta forma:

python probar\_entrega\_1.py

Si eso no funciona, pueden estar seguros de que algo no están haciendo bien. En los casos de error más comunes, el script puede explicarles lo que están haciendo mal. En casos más raros, no tanto. **Recuerden que pueden preguntar en el grupo todo lo que necesiten!**

Notas útiles:

* Recuerden que para calcular las estadísticas, pueden usar los visores. El BaseViewer permite calcular estadísticas sin agregar ningún tipo de interacción manual durante la ejecución de los algoritmos, de esta forma:

from simpleai.search.viewers import BaseViewer, astar, SearchProblem

class MiProblema(SearchProblem):

# ...

visor = BaseViewer()

resultado = astar(MiProblema(inicial), viewer=visor)

print resultado # ...

print visor.stats # esto les va a imprimir las estadísticas que necesitan

* El WebViewer es muy útil para probar y visualizar cosas, encontrar problemas, etc. Pero **recuerden** desactivarlo para la versión entregada, de lo contrario cuando la corrección automática trate de llamar a los algoritmos, se va a quedar tildada esperando.
* También **recuerden** que el módulo no debe ejecutar ninguna búsqueda al ser importado. Para ello, utilicen el "truco" del if \_\_name\_\_ == '\_\_main\_\_': que vimos en clases (está en el ejemplo del repo).