Agente autónomo para o jogo **Sokoban**

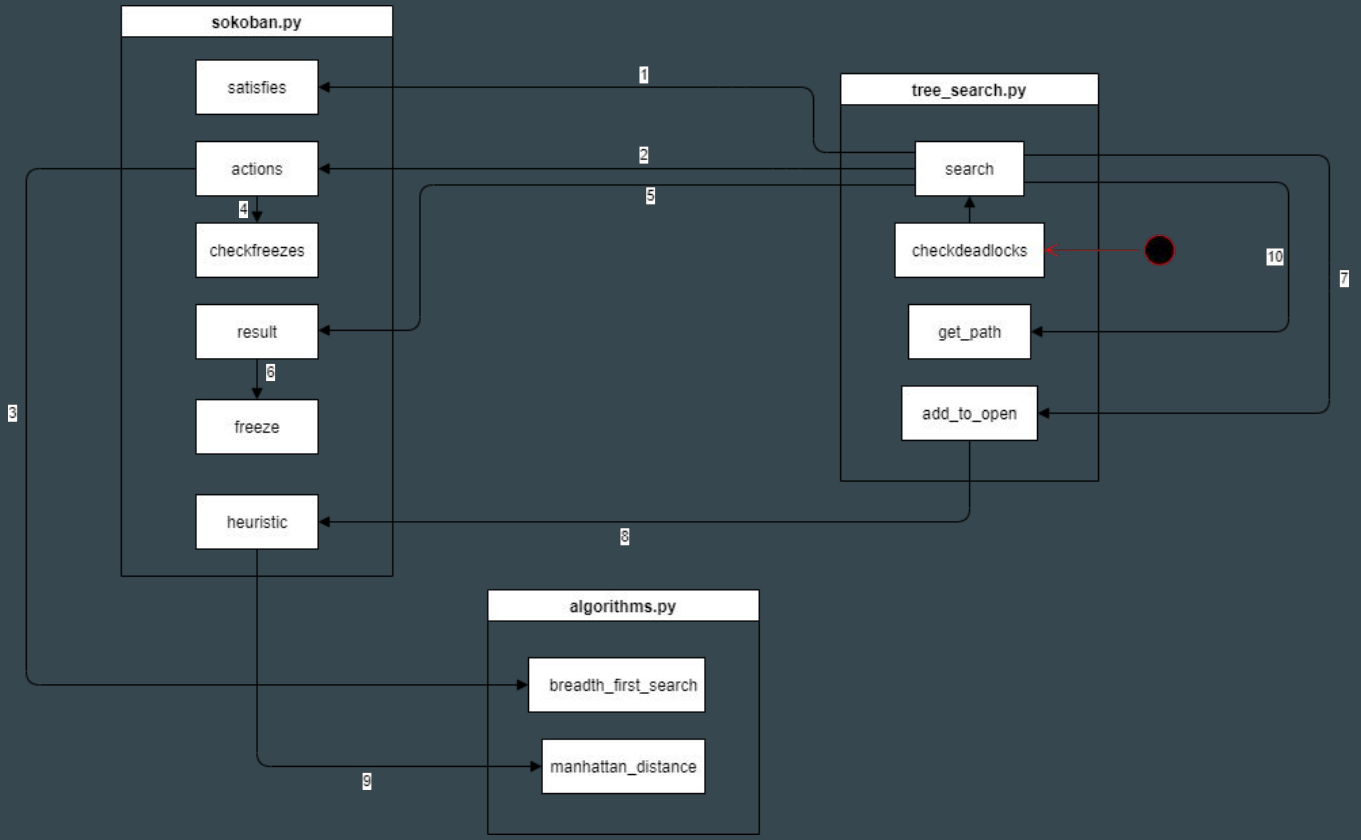
•••

Inteligência Artificial / Introdução à Inteligência Artificial
Ano Letivo de 2020/2021

Diogo Gomes
Luís Seabra Lopes

Pedro Tavares 93103
Gonçalo Pereira 93310

Ciclo de vida da aplicação



Os números representam a ordem de execução das funções no ciclo:

- search()
- satisfies()
- actions()
- breadth_first_search()
- checkfreezes()
- result()
- freeze()
- add_to_open()
- heuristic()
- manhattan_distance()
- get_path()

sokoban.py

- ACTIONS - Decide se o novo node é criado. Nesta função, verificam-se todos os movimentos possíveis. Verifica se o movimento está dentro do mapa e se obedece às restrições impostas: se o jogador se consegue mover até à posição para executar a ação ou se a caixa faria parte de uma das listas de deadlocks, simple ou freeze. Isto diminui a quantidade de nodes criados e deepcopies, que é um processo dispendioso.
- CHECKFREEZES -Verifica se da action resultaria um freeze deadlock. Caso resulte, este não é adicionado aos novos nodes. Para isso é utilizado o set de tuplos definido inicializado no construtor SearchTree, que vai conter todos os freeze deadlocks até ao momento.
- RESULT - Altera o mapa após o atualizar com a action, caso esta não gere um freeze deadlock que ainda não foi encontrado.
- FREEZE - Função chamada no RESULT de forma a verificar se o mapa apresenta algum freeze que ainda não tenha acontecido em iterações anteriores.
- HEURISTIC - Função que através da distância de manhattan verifica a “distância” até à solução.

tree_search.py

- SEARCH - Função onde se procura a solução. É nesta função onde as funções *actions(), result(), add_to_open() e get_path()* são chamadas, sendo o centro do programa. Ao receber as ações possíveis, guardamos as que não correspondem a nenhum tipo de deadlock encontrado anteriormente. Caso o novo estado seja criado, vai ser argumento de um novo node que é adicionado à lista open_nodes, que corresponde aos nodes por abrir.
- CHECKDEADLOCKS - Um deadlock é uma tile para onde, em qualquer situação, não podemos empurrar uma caixa. Para os calcular, simulamos substituir cada goal por uma caixa e tentamos puxá-la para todos as tiles possíveis. Se alguma tile não é alcançável em todos os goals, quer dizer que não conseguiríamos empurrar uma caixa dessa tile para um goal. Guardamos essas tiles no set *deadlocks*, e o agente nunca vai tentar colocar uma caixa nessas tiles, poupando a formação de bastantes nós.
- GET_PATH - Função que devolve o caminho que o keeper percorre até ao resultado final.
- ADD_TO_OPEN - Função que adiciona os novos nós à fila existente, dependendo da estratégia pretendida.

algorithms.py

- BREADTH_FIRST_SEARCH - Pesquisa em largura. Recebe o mapa, a tile inicial e a tile final. Cria nós com todas as opções de caminho a percorrer, evitando tiles bloqueadas (paredes ou caixas) ou já visitadas na pesquisa. Após encontrar uma solução, devolve o caminho entre a tile inicial e final.
- MANHATTAN_DISTANCE - Devolve a Distância de Manhattan entre duas posições.

Conclusão

Estratégias de otimização do programa:

- Simple Deadlocks
- Freeze Deadlocks

Estratégia usada para pesquisar nodes:

- Greedy

Implementações:

- Recursividade