Sokoban – Projeto de IIA 2020

António Domingues 89007
Leonardo Freitas 89131
Tiago Dias 88896

- Na resolução do problema proposto, o nosso agente reage consoante o seu estado, e este é descrito pelas posições do Keeper e das respetivas caixas para um determinado mapa.
- O nosso agente divide-se principalmente em dois ficheiros, o search.py e o students.py.
- O ficheiro search.py é adaptado do módulo implementado nas aulas práticas tree_search.py .Contém os métodos abstratos necessários para definir um domínio, a inicialização de um problema dado um estado inicial e um goal, assim como a criação dos nodes para a árvore de pesquisa, e por fim a implementação da nossa árvore de pesquisa que se rege por uma pesquisa do tipo A*.
- O students.py é quem irá fazer a maior parte do trabalho. É lá que são implementados os métodos para definir o domínio, criar um domínio e alternar entre os estados;

SearchDomain()

- Contém os métodos necessários para implementar um domínio. Todos os métodos desta classe são abstratos e serão mais tarde implementados no ficheiro **student.py**.

SearchProblem

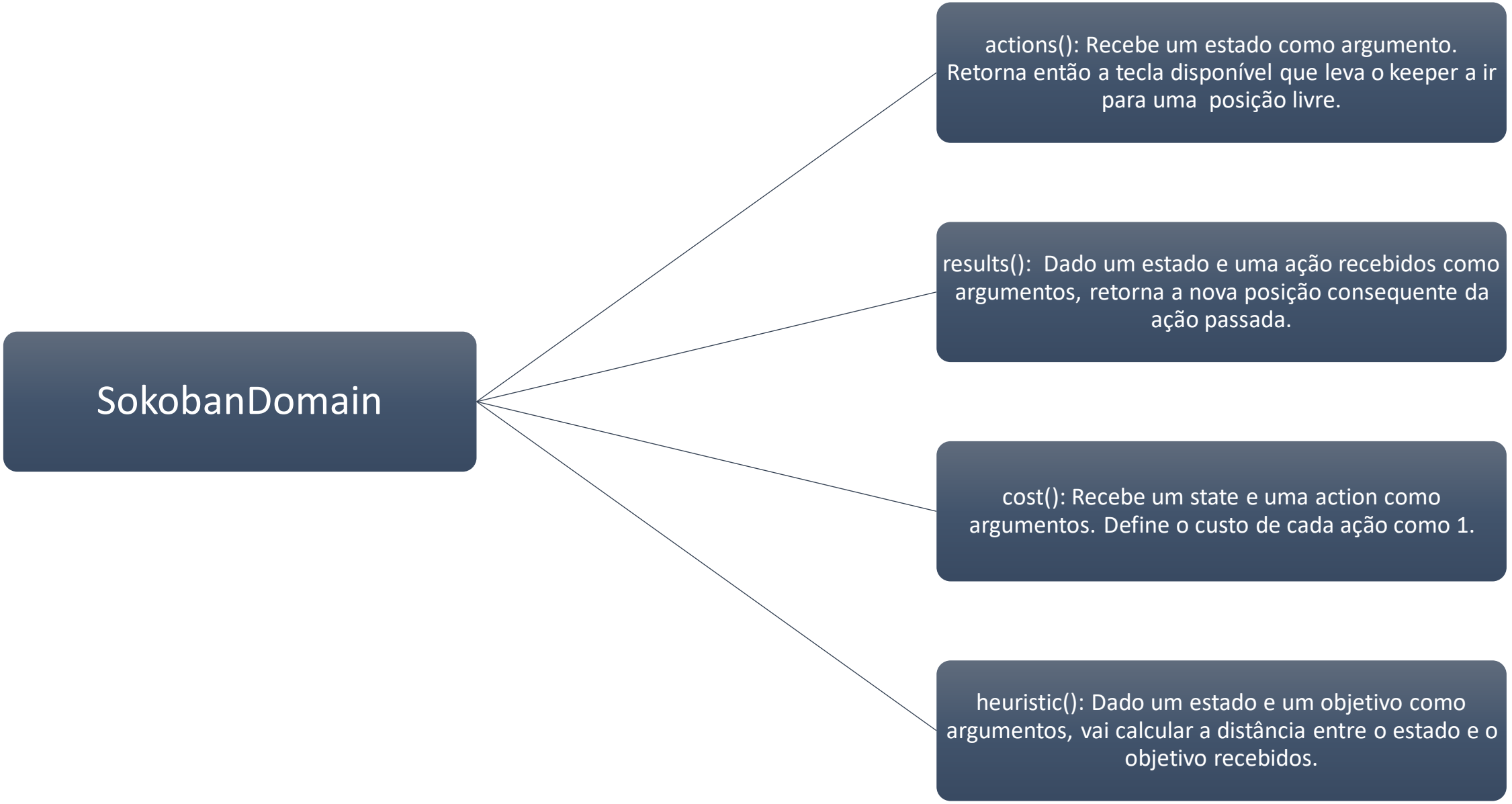
- Esta classe é capaz de criar um problema, recebendo como argumento o domínio em questão, o ponto de partida e um goal

SokobanNode

- Cria e implementa os nós necessários e específicos para a nossa árvore de pesquisa (**SokobanTree**)

SokobanTree

- Cria e implementa a nossa árvore de pesquisa utilizando a estratégia de pesquisa denominada de **A***.


SokobanDomain
actions(): Recebe um estado como argumento. Retorna então a tecla disponível que leva o keeper a ir para uma posição livre.
results(): Dado um estado e uma ação recebidos como argumentos, retorna a nova posição consequente da ação passada.
cost(): Recebe um state e uma action como argumentos. Define o custo de cada ação como 1.
heuristic(): Dado um estado e um objetivo como argumentos, vai calcular a distância entre o estado e o objetivo recebidos.

Objetivos não alcançados

- Subsequentemente ao nosso agente ser capaz de superar os 3 primeiros níveis, foi notória a presença de demasiados estados na tree search, o que levou a um grande aumento de tempo necessário para a resolução do mapa. Após inúmeras tentativas de compor a função para o cálculo das posições deadlocks(para ficar mais rousta), surgiu um resultado, que não o esperado. Devido ao sucedido, não se implementou outra alternativa, tendo a consequência de o agente apenas superar 3 níveis.
- Haveria uma 2ª implementação, sendo que, teriamos 2 tree searchs para um melhor desempenho do agente. Uma para o planeamento dos movimentos do sokoban e outra para os movimentos das caixas. A mesma chegou a ser implementada, contudo não funcional.