Trabalho Prático 1: Klotski

FCUP

Elementos de

Inteligência Artificial e

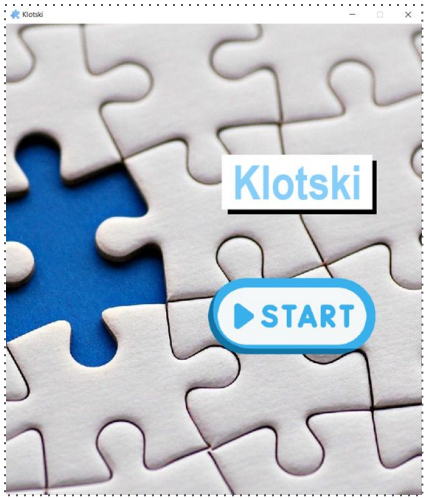
Ciência de Dados

2º. Semestre 2022/23

Gonçalo Esteves (up202203947)

Mariana Gomes (up202206615)

Especificação do Trabalho



O trabalho encontra-se dividido por **4 ficheiros e 1 pasta**:

→ **Constants.py**: contém as cores usadas nas peças e tabuleiro. Este também apresenta as dimensões da janela do PyGame bem como as do tabuleiro.

→ **Button.py**: contém a classe que permite uma melhor implementação dos menus e operações (exit, main menu, reset, solve) de modo a tornar a interface "user friendly".

→ **Level.py**: contém a classe responsável pela implementação dos **níveis** presentes no jogo (utilizada no próximo ficheiro).

→ **Game.py**: contém a classe 'game', que irá dar vida ao jogo, contendo múltiplos níveis de diferentes dificuldades.

→ **Images**: contém as **imagens** usadas nos botões do jogo (exit, main menu, reset, solve) e no fundo dos menus.

Formulação do Problema

REPRESENTAÇÃO DO ESTADO:

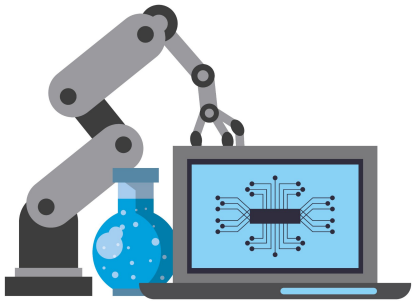
→ O estado é representado através das posições atuais das peças utilizando para tal **listas de tuplos**.

MOVIMENTOS POSSÍVEIS:

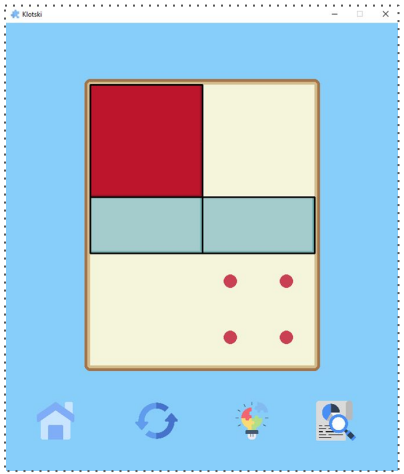
→ As peças são capazes de transitar dentro do tabuleiro tendo em conta os movimentos: **Cima, Baixo, Esquerda e Direita**. Note-se que as coordenadas das peças não podem ultrapassar o número de linhas nem o número de colunas do tabuleiro.

PRÉ-CONDIÇÕES:

→ A peça pode ser movida numa certa direção, se houver, nesta, um **espaço vazio** para o qual possa transitar.



Formulação do Problema



ESTADO INICIAL:

→ Qualquer estado correspondente à **configuração inicial** do jogo (com a peça vermelha fora do local de destino).

ESTADO OBJETIVO:

→ Estado em que a peça vermelha se encontra no local de **destino**.

PEÇAS:

→ As peças deslocam-se quadrado a quadrado, podendo estas ser a combinação de múltiplos quadrados.

HEURÍSTICA:

→ A heurística utilizada avalia a **distância Manhattan** da posição atual do quadrado vermelho à posição final.

Algoritmos de Pesquisa

Tendo em conta este problema de Pesquisa, foram utilizados vários algoritmos:

Algoritmos de Pesquisa Desinformada:

→ **Breadth-First-Search** (pesquisa em largura realizada de forma sequencial).

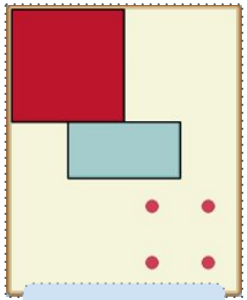
→ **Iterative Deepening** (pesquisa em profundidade limitada feita de forma sequencial).

Algoritmos de Pesquisa Informada:

→ **Greedy Search** (pesquisa tendo em conta o valor da heurística).

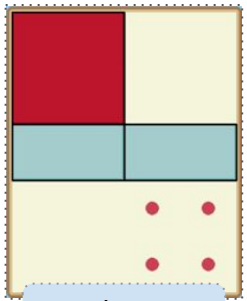
→ **A* Algorithm** (pesquisa que tem por base tanto o valor da heurística como os caminhos de menor custo).

Resultados Experimentais | Easy Mode



Nível 3

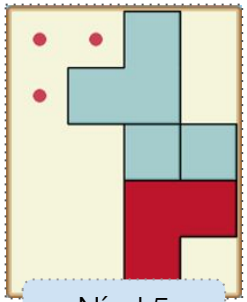
Algorithm	Steps to Solution	Nodes Explored	Time
Breadth-First-Search	6	70	0.159s
Iterative Deepening	13	13	0.015s
Greedy Search (H1)	6	6	0.006s
Greedy Search (H2)	7	7	0.009s
A* Search (H1)	6	6	0.006s
A* Search (H2)	7	7	0.009s



Nível 4

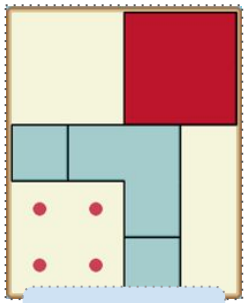
Algorithm	Steps to Solution	Nodes Explored	Time
Breadth-First-Search	8	283	2.858s
Iterative Deepening	10	10	0.009s
Greedy Search (H1)	11	22	0.031s
Greedy Search (H2)	10	16	0.027s
A* Search (H1)	9	20	0.024s
A* Search (H2)	9	16	0.027s

Resultados Experimentais | Easy Mode



Nível 5

Algorithm	Steps to Solution	Nodes Explored	Time
Breadth-First-Search	6	436	8.04s
Iterative Deepening	175	175	2.227s
Greedy Search (H1)	6	7	0.432s
Greedy Search (H2)	6	7	0.01s
A* Search (H1)	6	7	0.011s
A* Search (H2)	6	7	0.011s

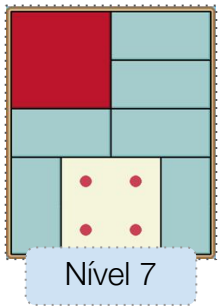


Nível 8

Algorithm	Steps to Solution	Nodes Explored	Time
Breadth-First-Search	9	2019	194.751s
Iterative Deepening	62	62	0.315s
Greedy Search (H1)	17	46	0.57s
Greedy Search (H2)	12	38	0.1s
A* Search (H1)	11	45	0.08s
A* Search (H2)	11	28	0.074s

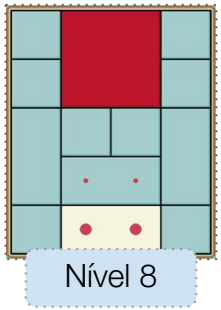
Resultados Experimentais | Hard Mode

Devido à **grande** quantidade de recursos e tempo utilizada pelos Algoritmos de Pesquisa Desinformada, optou-se por apenas avaliar os Algoritmos de Pesquisa Informada nos Níveis mais difíceis.



Nível 7

Algorithm	Steps to Solution	Nodes Explored	Time
Greedy Search (H1)	34	11	0.312s
Greedy Search (H2)	34	108	0.319s
A* Search (H1)	23	445	0.851s
A* Search (H2)	26	115	0.242s



Nível 8

Algorithm	Steps to Solution	Nodes Explored	Time
Greedy Search (H1)	83	728	3.55s
Greedy Search (H2)	59	576	1.979s
A* Search (H1)	39	18687	58.621s
A* Search (H2)	41	3152	10.408s

Conclusões

Partindo dos Resultados Experimentais obtidos, podemos aferir que:

→ Os **Algoritmos de Pesquisa Desinformada** requerem mais tempo e recursos para encontrar a solução, podendo esta não ser ótima. Neste tipo de algoritmos, destacam-se a Pesquisa em Largura (Breadth-First-Search) e a Pesquisa em Profundidade Iterativa. Apesar da Pesquisa em Largura encontrar uma melhor solução comparativamente à Pesquisa em Profundidade iterativa, esta explorada uma **maior quantidade de nós**, pelo que apresenta um **maior tempo de pesquisa**. Assim, apesar dos algoritmos proporcionarem boas soluções para problemas simples, estes acabam por não ser ideais na resolução de problemas mais complexos, optando-se por **Algoritmos de Pesquisa Informada** (Uso de heurísticas).

Conclusões

→ Os **Algoritmos de Pesquisa Informada** são capazes de resolver tanto problemas mais simples, como problemas mais complexos. Porém, a **qualidade das heurísticas** usadas nos algoritmos (Greedy Search e A* Search) é imprescindível na obtenção de uma boa solução. A primeira heurística calcula a distância Manhattan da posição atual da peça vermelha até à sua posição final, ao passo que a segunda não só tem em atenção esta distância como adiciona uma **penalização** caso exista algum bloco que impeça o movimento da peça vermelha em direção ao objetivo. Deste modo, a primeira heurística gera uma **melhor solução** relativamente à segunda, apesar de pesquisar uma maior quantidade de nós (maior uso de memória) e consumir uma maior quantidade de tempo. Por outro lado, a segunda heurística proporciona uma solução de uma forma **mais rápida e usando menos memória**.

→ Por fim, conclui-se que **não existe uma escolha exata** do algoritmo a usar visto nenhum ser capaz de obter uma solução ótima (com 100% de certeza) sem analisar todos os movimentos possíveis. Assim, a escolha recai sobre o algoritmo que melhor resolve o problema em questão, tendo em conta a **complexidade** (simples ou complexo) e os requisitos na obtenção da solução (**qualidade da solução, velocidade e memória utilizada**).

Referências Bibliográficas

→ “PyGame Tutorial”. Disponível em: https://www.geeksforgeeks.org/pygame-tutorial/

→ Diapositivos disponibilizados pelos docentes da Unidade Curricular e utilizados nas Aulas Teóricas e Práticas.