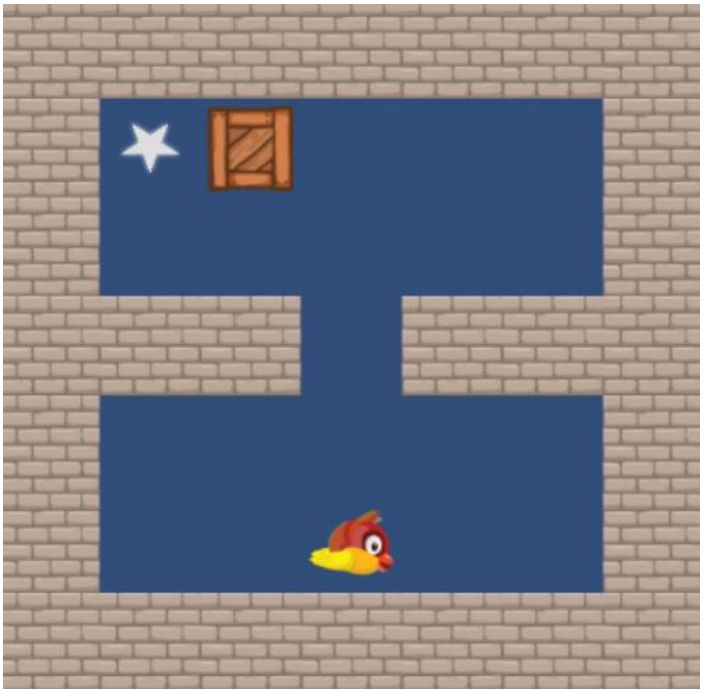
INTELIGÊNCIA ARTIFICIAL

TRABALHO PRÁTICO Nº2

SOKOBAN



Nome:
Pedro Manuel Cerveira Andrade

Nº de Estudante:
2014225147

Email:
uc2014225147@student.uc.pt

Turma Prática:
PL7

Nome:
Luis Alberto Pires Amaro

Nº de Estudante:
2014109216

Email:
lapa@student.dei.uc.pt

Turma Prática:
PL2

Nome:
Joel Filipe Rogão Pires

Nº de Estudante:
2014195242

Email:
jfpires@student.dei.uc.pt

Turma Prática:
PL1

Meta Final

DEI – FCTUC

INTRODUÇÃO

Este trabalho foi desenvolvido no âmbito da unidade curricular de Introdução à Inteligência Artificial do curso de Engenharia Informática da Universidade de Coimbra e regida por Fernando Machado.

Neste trabalho vamos desenvolver e explorar as capacidades de agentes procura. Para isso, vamos tentar resolver o jogo "Sokoban" cujo objetivo é movimentar todas as caixas para determinadas posições no mapa. Isso só é possível usando algoritmos de procura.

Para desenvolver tal projeto usou-se o motor de jogo Unity e a linguagem de programação C#. O Unity foi-nos útil para poder construir cenários de teste e visualizar os comportamentos manifestados pelo nosso agente de procura. A linguagem C# deu vida a esses mesmo agente que manifestavam os comportamentos dos algoritmos que nós próprios havíamos programado previamente.

META I – MODELAÇÃO E IMPLEMENTAÇÃO

Foi nesta meta que tentámos compreender as opções que tomámos na modelação do problema. Na Package fornecida, o algoritmo de pesquisa em largura já havia sido implementado. Coube-nos então implementar métodos de pesquisa cega, de pesquisa em profundidade limitada e aprofundamento progressivo. Todas essas implementações foram testadas e validadas.

A seguir, focámo-nos na implementação dos algoritmos heurísticos, a saber, o A* e a Pesquisa Sôfrega. A heurística utilizada foi:

- O nosso **h(n)** descreve a função que calcula uma estimativa do custo de transitar de um estado s até ao estado final**. h(n)** é então igual ao número de caixas que não estão numa posição destino.

META II – REFINAMENTO E EXPERIMENTAÇÃO

Estando o problema modelado e os algoritmos de pesquisa implementados, coube-nos nesta meta desenvolver heurísticas com melhores propriedades, heurísticas que não estimassem tão largamente o custo. Foi nesta meta também que testamos exaustivamente os algoritmos nos 12 mapas (fornecidos previamente) com as diferentes heurísticas.

Exigiu trabalho de pesquisa e investigação na internet, em artigos e teses para discernir quais as heurísticas que melhor se adequavam ao jogo Sokoban

MODELAÇÃO DO PROBLEMA

O problema de movimentar todas as caixas para determinadas posições no mapa resume-se a um problema de procura no espaço de estados usando um algoritmo de procura apropriado. O nosso **espaço de estados** com as respetivas ligações pode então ser visto como um grafo. Existem ainda **restrições** na movimentação do agente que se prende com o facto de ele não se poder posicionar nas coordenadas (e para além delas) onde se encontram paredes. Abaixo identificamos os vários aspetos deste problema modelado:

Estado do problema? Um estado descreve uma situação possível do problema. Neste caso, o estado será a posição das caixas e do jogador no nível.

Estado Inicial? Descreve a configuração das posições das caixas e do jogador assim que o jogo é inicializado.

Como se define um estado final? O estado final define-se como aquela configuração do problema que constitui a sua solução. Aplicado ao nosso problema, o estado final é aquele onde a posição das caixas se encontram nas posições objetivo.

Quais são os operadores de mudança de estado? Os operadores de mudança de estado são ”Esquerda”,”Direita”, “Cima”, “Baixo”.

Qual a natureza da solução pretendida? Estamos simplesmente focados em encontrar a solução e não a sequência de ações. A solução pretendida deverá não deverá ser uma qualquer, mas aquela que tem o menor custo possível. Querendo nós a solução mais económica e discriminadora, então estamos perante um problema de **otimização** (num ambiente **determinista)**.

Qual o custo associado a cada movimento? A cada movimento o custo é de um e apenas pode realizar um movimento de cada vez. Em cada movimento que fazemos estamos a visitar um novo nó na nossa árvore o que acarreta custos de tempo e de memória (depende se for necessário expandir os nós ou não e/ou se temos memória fixa ou não).

IMPLEMENTAÇÃO DE ALGORITMOS DE PROCURA

ALGORITMOS DE PROCURA CEGA

Pesquisa em Profundidade Limitada

Este algoritmo é uma melhoria do algoritmo de pesquisa em profundidade, uma vez que lida com caminhos infinitos limitando o nível de profundidade da procura. A procura começa na raíz, expandindo um nó, escolhe um dos seus sucessores e expande-o, e o processo repete-se até chegar à profundidade máxima da arvore. Uma vez feito, continua-se esta análise em profundidade a partir de um irmão do último nó analisado (caso exista) ou

então voltar ao nível anterior para continuar o processo. Facilmente se entende que estamos a falar de um algoritmo que não é nem discriminador, nem completo, nem económico.

Estando o algoritmo descrito, prossigamos à sua implementação. O nosso código para este algoritmo foi baseado no código fornecido para a pesquisa em Largura Primeiro. A alteração significativa que fizemos foi guardar os sucessivos nós expandidos numa pilha em vez de numa fila. Também, verificamos sempre se não estamos a ultrapassar o limite do nível de procura sempre que queremos adicionar um nó ao conjunto de nós visitados.

De forma a guardar os nós visitados usamos de um HashMap closedSet.

Aprofundamento Progressivo

Existe ainda um problema que precisa de ser melhorado: não sabemos o valor do nível máximo a que se pode encontrar a solução. O que fazemos é neste algoritmo é então fazer variar o nível de 0 até infinito. Chama-se consecutivamente o algoritmo de profundidade limitada e a cada iteração incrementa-se o nível de procura.

Este algoritmo vai ser completo e ainda é discriminador se o custo associado a todas as transições for igual. Caso contrário não encontra necessariamente a melhor solução. A nível temporal espera-se uma melhoria uma vez que a sua complexidade temporal é r/(r-1) – sendo r o fator de ramificação. Lembramos que é mais rápido dado que a solução é encontrada sem ter de chegar ao nível teoricamente necessário.

ALGORITMOS HEURÍSTICOS

Procura Sôfrega

Neste algoritmo escolhe-se o nó fronteira que aparenta ser o mais promissor de acordo com o valor estimado por h(n). A fronteira da árvore fica assim ordenada pelos seus valores heurísticos e escolhe-se sempre o nó com o valor mais baixo (aquele que está mais perto da solução).

Para implementar tal funcionalidade usamos a classe “PriorityQueue” fornecida previamente pelo professor. Assim, esta fila ordenada recebe os nós da árvore e recebe de acordo com o valor de h(n) que lhe passamos.

Facilmente se percebe que esta solução não é ideal uma vez que nã o é discriminadora nem completa.

A*

O algoritmo A* assume-se como um melhoramento do algoritmo anterior pois, a cada instante, ele tenta escolhe o melhor caminho passando pelo nó n, ultilizando para tal a função **f(n) = g(n) + h(n).** Este algoritmo é sim completo e discriminador.

A implementação deste algoritmo foi idêntica à anterior com apenas uma alteração: o valor com que os nós são adicionados à PriorityQueue é f(n) em vez de h(n).

A eficácia deste algoritmo depende das propriedades da heurística em causa. Assim sendo, nós implementámos mais que uma heurística que passamos a descriminar a seguir no relatório.

HEURÍSTICAS

O que são Heurísticas?

Usar heurísticas no método de pesquisa da solução consiste em utilizar informação e conhecimento sobre o problema para que a pesquisa seja mais eficaz (também chamada de pesquisa informada). A heurística procura então estimar o custo do caminho do nó corrente até ao nó solução. Essa estimativa pode exceder largamente o custo ou não, pode ter boas propriedades ou não mas, independentemente disso, devem ser admissíveis.

HEURÍSTICAS UTILIZADAS

Heurística Inicial

Esta heurística foi a heurística proposta no enunciado. Neste caso, **h(n)** descreve a função que calcula uma estimativa do custo de transitar de um estado s até ao estado final**. h(n)** é então igual ao número de caixas que não estão numa posição destino. Esta heurística é admissível pois o numero de caixas que não estão numa posição destino não pode ultrapassar o número de posições necessárias para pô-las nas posições-objetivo. Implementamos esta heurística fazendo uso da função já disponibilizada "GoalsMissing()".

Primeira Heurística

Depois de algum trabalho de investigação, concluímos que uma boa primeira abordagem a uma heurística serica calcular a distância Euclediana entre cada caixa e a posição objetivo mais próxima. Cria-se então assim uma estimativa mais real do custo para chegar ao estado final do que a heurística anterior (embora ainda não seja a ideal dado que não se tem em conta o movimento do jogador.)

Relativamente à sua implementação, apenas tivemos de iterar cada posição objetivo por cada caixa e calcular a distância mínima através do método "distance" da classe "Vector2" e, por fim, soma-se ao resultado o que é retornado.

Segunda Heurística

Nesta heurística calculamos a distância de Manhattan entre o jogador e a caixa mais próxima dele e entre as caixas e as posições-objetivos. Lembramos que esta distância é a distância entre dois pontos medidao ao longo dos eixos, como numa grelha. Assim, é feita uma estimativa aproximada à distância necessária para resolver o problema, levando em consideração os passos que o jogador precisa fazer para preencher as posições-alvo e a deslocação do jogador para a caixa mais próxima.

É sem dúvida uma heurística com melhores propriedades que as anteriores, no entanto entanto a distância necessária é sobrestimada.

Para implementar apenas tivemos de: para cada caixa no mapa iterar pelas posições objetivo e adicionar ao valor da heurística, a distância a cada caixa; a seguir, calcular a distância do jogador à caixa e verificar se é a distância mínima; finalmente, devolver a soma de todas as distâncias das caixas às posições.

Terceira Heurística

Esta heurística é uma melhoria da anterior. O que se fez desta vez foi, ao invés de somar a distância de todas as caixas até todas as posições objetivo, somamos a distância de todas as caixas à posição objetivo mais próxima. Esta heurística permite agora devolver uma estimativa mais adequado ao custo real (embora a suposição de que a caixa mais próxima é a certa pode ser erróneo).

Para Implementar este algoritmo apenas acrescentámos uma pequena verificação em relação à implementação anterior que permite saber qual a posição-alvo mais próxima da caixa.

Quarta Heurística

Esta heurística revela-se praticamente igual à Segunda Heurística. Trata-se de uma variante do calculo da distância euclidiana que usa exponenciação e a raiz quadrada para calculo do output. Tendo em conta os testes realizados, esta heurística é a que manifesta os piores resultados a seguir àquela sugerida no enunciado (com raras exceções esporádicas como por exemplo no mapa 6).

ADMISSIBILIDADE DAS HEURÍSTICAS

Para que a heurística seja aplicável ao problema tem que ser admissível, isto é, nunca deve sobrestimar o custo real do caminho que leva a que todas as caixas fiquem nas posições-objetivo.: **h(n) <= h_real(n)**. Todas implementadas por nós **à exceção da Terceira Heurística** e incluindo a heurística sugerida no enunciado são admissíveis, pois o seus custos não ultrapassam o número de posições necessárias para pôr as caixas nas posições-objetivo.

DEADLOCKS

Gostaríamos de fazer uma breve menção a este conceito que está implementado no código e que permite com que o nosso jogador não ande ciclicamente à procura da solução indefinidamente. Se existe um estado e que é impossível encontrar uma solução, uma mensagem de alerta surgirá a informar disso.

TESTES E EXPERIMENTAÇÃO

Nesta fase testamos intensivamente os algoritmos implementados sob diferentes heurísticas e tendo em conta diferentes mapas. Comparamos as diferentes performance e registamos (**ver anexos A e B**). Através destes testes podemos fazer uma avaliação das vantagens e desvantagens de certos algoritmos bem como comparar a eficiência das heurísticas utilizadas. Avançamos já que os melhores resultados foram obtidos combinando a Terceira Heurística com o algoritmo de pesquisa informada.

Pesquisa em Largura Primeiro

Teoricamente este algoritmo é muito pouco económico – complexidade espacial e temporal exponenciais. Assim para problemas onde são gerados poucos nós e com um fator r de ramificação baixo o algoritmo até se torna eficiente. Para um grau de complexidade muito maior, este algoritmo torna-se incomportável. Assim, pelos testes realizados dá para perceber que no mapa 01 o algoritmo porta-se bem, mas daí para a frente a sua eficiência é perdida e muito.

Pesquisa em Profundidade Limitada

A complexidade temporal deste algoritmo continua exponencial embora a sua complexidade espacial melhores (linear com limite máximo). Através dos testes facilmente se comprova que os valores obtidos vão de acordo com esta teoria. De todos os algoritmos de procura cega foi este o que maximizou os resultados. Agora existe sempre o problema de não sabermos que limite havemos de aplicar.

Aprofundamento Progressivo

A complexidade espacial continua igual à do algoritmo anterior. Contudo, existe uma sobrecarga temporal pois estamos a analisar os mesmos nós mais que uma vez. Para valores elevados, a sobrecarga teporal é favorável, embora a complexidade temporal se mantenha exponencial.

Este algoritmo, pelos testes que realizamos, foi então aquele que apresentou piores resultados. Sem dúvida útil para problemas de menor complexidade em que não conhecemos o limita mas em que o custo espacial e temporal fica grande à medida que a complexidade aumenta.

Pesquisa Sôfrega

Os nós neste algoritmo vão sendo expandidos hibridamente entre a procura em profundidade e a procura em largura. No limite temos complexidade espacial exponencial pois a fronteira tem de ser mantida toda em memória. Podemos também ter complexidade temporal exponencial no pior caso (pode acontecer que todos os nós tenham de ser expandidos e visitados).

A escolha de uma boa heurística é então fundamental. Com diferentes heurísticas, pela análise dos resultados dos testes, este algoritmo comporta-se favoravelmente quando se usa a Terceira Heurística, verificando-se os melhores resultados de entre todos os algoritmos!

A*

Pelo estudo teórico deste algoritmo sabes que ele é completo, ótimo e discriminador. A sua eficácia depende em muito da função **f**. Como procura sempre o melhor caminho, o custo temporal e espacial é superior ao do Procura Sôfrega. Pela análise dos resultados dos testes verifica-se que este algoritmo não é o que apresente de todo os melhores tempos.

CONCLUSÃO

Com este trabalho conseguimos aprender mais sobre os diferentes algoritmos de pesquisa. Concluímos as suas principais fraquezas e forças. Não existe um algoritmo ideal, perfeito, existem sim algoritmos mais ou menos adequados para determinado contexto/problema. Ficaram bem evidentes para nós as vantagens e desvantagens de cada algoritmo bem como o peso que uma boa heurística pode ter na resolução eficaz do problema.

ANEXO A – RESULTADOS DOS ALGORITMOS DE PROCURA CEGA

Mapa 1	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discriminaç	sim	não	sim
Tempo	22	19	41
Memória	58	50	116
Max		9	
Final Max			5

Mapa 2	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discrimina	sim	sim	sim
Tempo	386	92	767
Memória	1123	288	2301
Max		10	
Final Max			10

Mapa 5	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discriminaç	sim	não	não
Tempo	708161	31527	955331
Memória	1964594	90130	2748378
Max		56	
Final Max			56

Mapa 6	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discrimina	sim	não	não
Tempo	71617	5312	967207
Memória	158845	11328	2070370
Max		215	
Final Max			215

Mapa 9	BFS	LDFS	PLDS
Sucesso	não	sim	não
Discriminaç		não	
Tempo		1350141	
Memória		3650167	
Max		292	
Final Max			

Mapa 10	BFS	LDFS	PLDS
Sucesso	não	não	não
Discrimina			
Tempo			
Memória			
Max			
Final Max			

Mapa 3	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discrimina	sim	não	não
Tempo	134251	2835	105034
Memória	382032	7911	281586
Max		35	
Final Max			35

Mapa 4	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discrimina	sim	não	não
Tempo	13812	1854	55443
Memória	33496	4663	141692
Max		76	
Final Max			60

Mapa 7	BFS	LDFS	PLDS
Sucesso	não	não	não
Discrimina			
Tempo			
Memória			
Max			
Final Max			

Mapa 8	BFS	LDFS	PLDS
Sucesso	sim	sim	sim
Discrimina	sim	não	não
Tempo	1073204	155845	5866057
Memória	2802189	418108	15681530
Max		117	
Final Max			117

Mapa 11	BFS	LDFS	PLDS
Sucesso	não	não	não
Discrimina			
Tempo			
Memória			
Max			
Final Max			

Mapa 12	BFS	LDFS	PLDS
Sucesso	não	não	não
Discrimina			
Tempo			
Memória			
Max			
Final Max			

ANEXO B – RESULTADOS DOS ALGORITMOS HEURÍSTICOS

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 1	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	21	21	9	9	21	19	19	5	5	19
Memória	55	55	27	27	55	50	50	16	16	50

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 2	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	187	128	236	68	559	30	24	379	16	647208
Memória	564	386	695	205	1364	96	77	1112	51	1773987

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 3	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	126745	123701	141805	78481	134534	3506	3450	28252	1507	40861
Memória	362926	354202	406078	219826	384290	9568	9204	81302	4430	109129

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 4	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	13126	10054	4943	7782	10721	9909	4544	3225	4812	4892
Memória	31822	24540	12166	19146	26147	21967	10944	7973	11930	11828

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 5	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	442260	107845	1968	78638	185646	1767	31861	226	81	15089
Memória	1234389	302854	5395	222538	518081	4641	84477	614	225	39488

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 6	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Tempo	71123	54100	8298	26406	68786	9542	4815	3733	4015	3269
Memória	157717	119547	18013	57767	152679	21360	10486	8212	8910	7113

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 7	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	TE	TE	Sim	TE	TE	TE	TE	TE	TE	TE
Discriminação			Sim							
Tempo			769543							
Memória			2137508							

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 8	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminaç	Sim	Sim	Sim	Sim	Sim	Sim	Sim	SIm	SIm	Sim
Tempo	1063578	966662	104085	624471	1523412	28793	6793	4382	5333	1726
Memória	2775789	2523794	270070	1630929	3659452	73864	17740	11506	14208	4429

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 9	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	TE	TE	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Discriminação			SIm	SIm	Sim	Sim	Sim	Sim	SIm	Sim
Tempo			199948	5954088	9107157	18312662	1445877	356883	323896	612099
Memória			524299	15760530	24119981	47360105	3757573	939740	861002	1591136

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 10	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	TE	TE	Sim	Sim	TE	Sim	Sim	Sim	Sim	Sim
Discriminação			Sim	Sim		Sim	Sim	Sim	SIm	SIm
Tempo			299908	8521637		91555	27751	4546	392	37367
Memória			956151	13964782		280561	80972	14343	1251	109410

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 11	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	TE	TE	Sim	Sim	TE	TE	TE	Sim	Sim	Sim
Discriminação			Sim	Sim				SIm	Sim	Sim
Tempo			118352	23156782				299006	3352855	270565
Memória			302599	67521003				765887	8563134	677577

Heuristicas	Inicial	First	Second	Third	Fourth	Inicial	First	Second	Third	Fourth
Mapa 12	A*	A*	A*	A*	A*	Sofrega	Sofrega	Sofrega	Sofrega	Sofrega
Sucesso	TE	TE	TE	TE	TE	TE	TE	TE	TE	TE
Discriminação										
Tempo										
Memória										

*TE Time Exceeded (> 1 hour)