Relatório A*

Projeto de Algoritmos

Renata Sarmet Smiderle Mendes

Playing with wheels

Vamos realizar 2 testes, um com uma entrada pequena, descrita no próprio pdf do problema, e outro com uma entrada um pouco maior, gerada por um dos colegas de classe. Ambas as entradas seguem anexadas junto ao código.

Os resultados dos testes foram:

Algoritmo	Tamanho entrada	Run time (segundos)
Backtracking	*pequena*	0.010
*A**	*pequena*	0.008
Backtracking	*grande*	0.545
*A**	*grande*	0.157

Para a entrada pequena, o código já roda rápido, então não houve tanta mudança, porém para a entrada maior foi possível notar mais a diferença. Além disso, ambos os códigos foram submetidos no UVA, para verificação da validade das respostas e comparação do tempo de execução.

My Submissions

#	Problem	Verdict	Language	Run Time	Submission Date
26406761	10067 Playing with Wheels	Accepted	C++11	0.030	2021-05-16 19:10:25
26406654	10067 Playing with Wheels	Accepted	C++11	0.050	2021-05-16 18:04:12

Figura 1 - Submissão no UVA
#26406761 representa o A e #26406654 representa o Backtracking*

Como podemos ver na Figura 1, ambos foram aceitos e o A* obteve *run time* de 0.030, enquanto que o Backtracking de 0.050.
Percebe-se, portanto, que de fato faz diferença a utilização da heurística para aperfeiçoar o código.

Rat in a Maze

Para o problema do rato no labirinto, foram geradas instâncias de tamanhos 50, 100 e 500. Todas as entradas estão anexadas junto ao código. Para testar, vamos comparar o tempo de execução e a quantidade de quadradinhos que foram de fato visitados, isto é, vieram através do *.top* da fila.

Vamos comparar 5 algoritmos, o *mazeBFS* (dado pelo professor), dois que utilizam a distância manhattan na heurística h, sendo o *maze_a_star_manhattan* (no qual considera para comparação g+h, onde g é o *depth*) e o *maze_a_star_manhattan_no_depth* (no qual considera para comparação apenas h, pois *depth* recebe sempre valor zero), e dois que utilizam a distância euclidiana na heurística h, sendo o *maze_a_star_euclidean* (no qual considera para comparação g+h, onde g é o *depth*) e o *maze_a_star_euclidean_no_depth* (no qual considera para comparação apenas h, pois *depth* recebe sempre valor zero).

Para a instância de tamanho 50:

Algoritmo	**Run time (segundos)**	**Quantidade de quadradinhos visitados**
mazeBFS	0.012	2039
maze_a_star_manhattan	0.012	562
maze_a_star_manhattan_no_depth	0.011	122
maze_a_star_euclidean	0.013	1257
maze_a_star_euclidean_no_depth	0.012	120

Para a instância de tamanho 100:

Algoritmo	**Run time (segundos)**	**Quantidade de quadradinhos visitados**
mazeBFS	0.023	9548
maze_a_star_manhattan	0.022	1930
maze_a_star_manhattan_no_depth	0.021	199
maze_a_star_euclidean	0.044	9228
maze_a_star_euclidean_no_depth	0.023	239

Para a instância de tamanho 500:

Algoritmo	**Run time (segundos)**	**Quantidade de quadradinhos visitados**
mazeBFS	0.303	149203
maze_a_star_manhattan	0.358	82458
maze_a_star_manhattan_no_depth	0.277	1547
maze_a_star_euclidean	0.705	134644
maze_a_star_euclidean_no_depth	0.282	1463

Portanto, não é possível perceber uma mudança tão significativa do tempo em alguns casos, pois o tamanho das instâncias não é tão grande, eu tentei gerar instâncias maiores, porém meu computador não suportou. Porém, é possível perceber que a quantidade de quadradinhos visitados muda bastante entre cada algoritmo. Acredito que a melhor opção seja utilizar o A* porém não considerando a distância do início até o ponto atual (g), apenas a distância restante até o final (h). E nessa configuração, não houve tanta diferença entre a distância manhattan e a euclidiana.