ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

PCS3848 - Inteligência Artificial - Primeiro Exercício Programa

Implementação do jogo Pacman utilizando algoritmo de busca A*

Daniel Lavedonio de Lima - 8992882
daniel.lavedonio.lima@usp.br

Gabriel da Cunha Rodrigues - 8992930
gabriel.cunha.rodrigues@usp.br

25 de setembro de 2019

1. **Link para o código: https://github.com/Lavedonio/PCS3438-2019-EP1**

2. **Descrição do código:**

O algoritmo desenvolvido no arquivo *search.py,* na função *aStarSearch(problem, heuristic=nullHeuristic),* foi amplamente comentado no código para um bom entendimento de seu funcionamento. A estrutura de dados escolhida não se baseou nas estruturas prontas do arquivo *util.py*, por conveniência do grupo. Ela consiste em uma lista que contém as informações de estado, caminhos possíveis e custo em cada elemento.

O loop principal do código é um *while* que se repete enquanto a borda não estiver vazia. Inicialmente, determina-se em *lowest_f* o primeiro elemento da borda como sendo o de menor custo total, ou seja, tendo a menor soma do custo de caminho com o custo da heurística determinada para seu estado, armazenando seu índice em *lowest_f_position*. A partir desse índice expande-se a borda para os sucessores daquela posição. Em seguida, percorre-se um *for* entre cada novo elemento de borda, somando o *custo_acumulado* com o da posição, adicionando a nova direção na lista de caminhos e verificando se a nova posição já foi percorrida. Depois, é avaliado se o elemento em questão é a posição final. Se for, ele retorna o caminho total acumulado até aquela posição; senão esse novo elemento é adicionado na *borda* e nos já *explorados* e o processo total é repetido.

3. **Descrição das rotinas de heurística testadas**

Utilizando a *manhattanHeuristic,* que utiliza a distância Manhattan como base, o algoritmo desenvolvido gerou um Score de 300 pontos, expandindo 549 nós e achando um caminho com um custo total de 210. Já na *euclideanHeuristic,* baseada na distância euclidiana, o algoritmo também gerou um Score de 300, mas expandindo 557 nós. O caminho achado também teve o custo total de 210. A *foodHeuristic,* da mesma forma, obteve um Score de 300 e achou um caminho com custo total de 210, mas expandindo 617 nós. A

cornersHeuristic necessita de outras funções para poder ser executada, portanto não foi possível ser testada.

Nota-se que a *manhattanHeuristic* é um pouco mais eficiente que a *euclideanHeuristic,* pois precisou expandir menos nós para achar o mesmo caminho, obtendo a mesma pontuação no jogo. A *foodHeuristic* demonstrou ser a menos eficiente entre as heurísticas testadas, expandindo muito mais nós do que as outras.