Relatório EP1

Yara Grassi Gouffon - 4172560

Estatísticas

Busca	tinyMaze		mediumMaze		bigMaze	
	Nós	*Custo*	*Nós*	*Custo*	*Nós*	*Custo*
DFS	15	10	146	130	390	210
BFS	15	8	269	68	620	210
IDS	64	8	10236	68	60211	210
UCS	15	8	269	68	620	210
A*	14	8	222	68	549	210

Discussão

Dados os resultados, é fácil identificar o A* como algoritmo de melhor desempenho em modo geral. Em 2 das 3 fases, encontrou o caminho ótimo com menos expansões de nós--ou seja, de maneira mais efetiva. O único caso em que ficou em segundo lugar foi no bigMaze, em que o DFS precisou de pouco mais da metade de expansões para achar um caminho que por acaso é ótimo. No entanto, visto que o DFS de maneira geral não é completo nem ótimo, o A* continua como escolha mais segura.

O BFS se saiu relativamente bem, fazendo por volta de 70 expansões a mais que o A*. Em problemas mais complexos, isso poderia ser um problema. A principal vantagem é oferecer um caminho ótimo, assim como a UCS, que teve o mesmo número de expansões (provavelmente porque as ações tinham todas o mesmo custo). Já o IDS teve um número quase 100 vezes maior de expansões comparado ao BFS no bigMaze. Talvez a minha implementação esteja errada--caso contrário, visto que a vantagem do UCS é oferecer um caminho ótimo utilizando menos memória que o BFS, me pareceria bom utilizá-lo apenas em um problema que ocuparia memória demais com o BFS ou cujo tamanho não é conhecido.

Questões

Questão 1 - Busca em Profundidade

Sim, é possível ver na visualização da fase que o algoritmo tende a seguir uma "trilha" até o fim em vez de explorar todos os lados. Isso corresponde a seguir um caminho no grafo até atingir o estado meta ou encontrar uma folha, como esperado de uma busca em profundidade.

O Pac-Man não se move para todos os estados explorados, pois é retornado apenas o caminho que leva ao estado meta. Os estados explorados que não levam ao objetivo são descartados.

Questão 2 - Busca em Largura

Sim, a implementação feita encontra uma solução de custo mínimo, pois explora todos os nós atingíveis a uma certa distância do estado inicial antes de explorar mais longe. Assim, obrigatoriamente encontra a primeira ocorrência possível do estado meta. É possível verificar o resultado comparando com os custos encontrados pela UCS e pelo A*, que também são ótimos.

Questão 3 - Busca de Custo Uniforme

A função custo do StayEastSearchAgent aumenta para posições mais à esquerda do mapa e diminui para posições à direita. Assim, o agente prioriza a exploração do lado leste da sala o tanto quanto pode, e apenas vai para o lado oeste quando não há outra opção para atingir o objetivo. De maneira similar, a função custo do StayWestSearchAgent aumenta para posições à direita e diminui para a esquerda, de maneira que o agente imediatamente busca o lado oeste do mapa e vai ao leste apenas quando não tem escolha, retornando à esquerda em direção ao objetivo assim que pode.

Questão 4 - Busca de Aprofundamento Iterativo

Uma DFS segue um caminho até o fim antes de tentar outro. Assim, sem a modificação, a IDS poderia seguir um caminho longo até um estado X, atingir a profundidade máxima, e depois voltar e encontrar outro caminho que chega até X mas é mais curto. Sem a modificação, esse caminho seria descartado, tornando a solução subótima. Ao permitir que um nó seja descartado apenas se seu custo for maior, garantimos que a solução será ótima. Não seria aconselhável utilizar busca em árvore neste EP porque é fácil retornar a uma posição já visitada e o algoritmo perderia muito em eficiência explorando o mesmo estado várias vezes--possivelmente ficando até preso em um loop.

Questão 5 - Busca Heurística

O algoritmo A* tende a ser mais eficiente por possuir conhecimento específico do problema, representado pela função heurística. Por meio dela, o algoritmo é capaz de estimar quais nós têm mais potencial e devem ser explorados primeiro. A razão para se implementar uma heurística consistente é para garantir a otimalidade da solução, na versão em grafo. Caso fosse uma busca em árvore, seria necessário apenas que a heurística fosse admissível.

Questão 6 - Agentes adversariais

O agente reativo, como o nome indica, leva em consideração apenas suas próprias percepções para decidir a ação a ser tomada--ele reage ao que acontece em cada rodada. Já o agente minimax tenta prever as ações dos inimigos de maneira a maximizar o desempenho no pior caso, ou seja, em um jogo como este onde a 'morte' é duramente penalizada, prioriza a sobrevivência e tem mais chances de ganhar--ou ao menos, de não perder tão rápido.

Algumas mudanças possíveis seriam ajustes no tratamento de fantasmas próximos, talvez rever os valores relativos a quão próximos os fantasmas estão ou à quantidade de fantasmas por perto. Caso fosse possível identificar o tipo de cada fantasma, talvez fosse possível retornar um valor menor ou maior dependendo da posição do Pac-Man e do estilo da movimentação do fantasma (aleatória, ou perseguindo ou fugindo do Pac-Man, por exemplo).

Melhorias

Seria interessante poder rodar os autograders por partes. No caso do Minimax, quando eu estava testando a q3 precisava esperar a q1 e q2 concluírem, o que pode demorar mais ou menos dependendo da função de avaliação.