Inteligência Artificial

Thiago Henrique Leite da Silva, RA: 139920

Aula 05 - Exercício prático Busca informada (Teoria)

1) (0,5) Descreva o algoritmo de busca Dijkstra. Em seguida identifique qual a diferença/semelhança entre Dijkstra e a busca de custo uniforme e a busca A*?

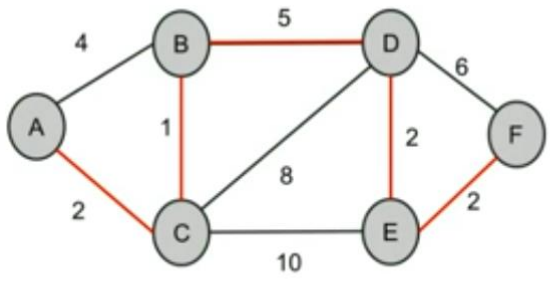
O algoritmo de Dijkstra é um algoritmo de menor caminho para percorrer grafos.

Ele tem complexidade O(E+V\log(V)), onde V é o número de vértices e E é o número de arestas.

A forma com que o algoritmo realiza a busca para encontrar o menor caminho é bem simples de entender, basicamente, o que ele faz é, partindo de um vértice inicial, busca entre seus adjascentes o vértice com a menor distância, tendo encontrado este vértice, a partir dele, busca novamente entre seus adjacentes aquele com menor distância relativa, sendo que agora, também somamos a distância já percorrida, ou seja, estamos analisando a distância total.

No exemplo abaixo, conseguimos ver sua atuação, onde a coluna de vértices são as possibilidades que temos para percorrer a partir de um dado vértice, já a representação e parêntes é da seguinte forma:

(custo ou distância, vértice ao qual estamos vindo)



Vértice	Peso01	Peso02	Peso03	Peso04	Peso05	Peso06
A	(0,A)					
B	(4,A)	(3,C)	(3,C)			
C	(2,A)	(2,A)				
D		(10,C)	(8,B)	(8,B)		
E		(12,C)		(10,D)	(10,D)	
F				(14,D)	(12,E)	(12,E)

A semelhança deste algoritmo com o de busca uniforme, é que em ambos estamos levando em conta a distância total, não olhando somente para as margens, o que faz total diferença no momento de encontrarmos uma solução mais otimizada, já a diferença, é que na busca de custo uniforme, a cada iteração, olhamos para a distância total de todos os nós folhas, e expandimos aquele com o menor valor, já no algoritmo de Dijkstra, após termos tomado uma decisão, não olhamos novamente para os vértices que ficaram para trás.

Falando agora da busca A*, o fator determinante que diferencia ela das duas anteriores, é a heurística, que nada mais é do que uma classificação de alternativas na etapa de ramificação do grafo, ou seja, vamos definir uma informação a mais, além do custo, que irá nos ajudar na hora de tomar uma decisão otimizada. Portanto, este algoritmo é uma adaptação do algoritmo de Dijkstra com a Busca em

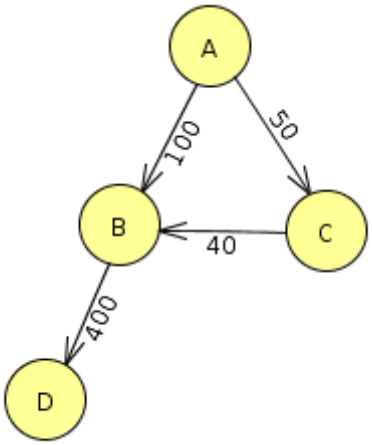
Largura, dessa forma, conseguimos extrair os pontos positivos de ambos, gerando um melhor resultado.

Fonte: Algoritmo de Dijkstra

2) (0,5) Quais das seguintes alternativas são falsas e quais são verdadeiras? Explique suas respostas.

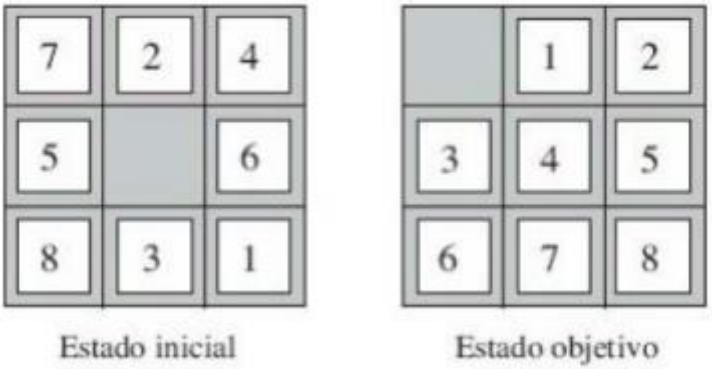
a. A busca em profundidade sempre expande pelo menos tantos nós quanto a busca A* com uma heurística admissível.

Falso. A busca em profundidade, dependendo do grafo, pode buscar menos nós do que a busca A*, ainda que com isso esteja perdendo em otimização. Isso acontece, pois, a busca A* está procurando o caminho mais otimizado, ou seja, em uma situação em que é melhor passar pelos vértices C e B para chegar em D, por exemplo, a busca em profundidade poderia optar por ir de B para D diretamente, sem analisar os custos como forma de otimizar a busca.



Porém este é um caso específico, em linhas gerais a busca A* percorrerá menos vértices até obter uma solução.

b. h(n) = 0 é uma heurística admissível para o quebra-cabeças de 8 peças.



Verdadeira. A definição exige apenas que, para uma heurística h(n) ser admissível, ela precisa ter valor menor ou igual que o custo c(n). No nosso problema, c(n) seria da seguinte forma:

c(n) = a distância do estado atual do nó até seu estado final

Desta forma, como a distância será no mínimo zero, a heurística é admissível, pois para todo n,

h(n) <= c(n).

c. Em robótica, A* não é útil porque as percepções, estados e ações são contínuas.

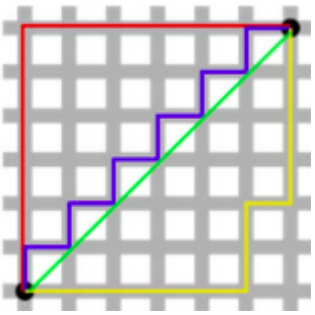
Verdadeiro. A complexidade deste algoritmo depende muito da heurística, e em um espaço discreto, teríamos dois grandes problemas, encontrar uma heurística admissível, algo que seria bastante utópico, já que estamos falando de um espaço que está em constante mudança. Além disso, no pior caso, a quantidade de vértices que é expandida é exponencial, de acordo com a profundidade da solução, sendo O(b^d), onde b é a profundidade e do fator de ramificação, ou seja, o número médio de nós adjacentes a um outro qualquer. Ou seja, em um ambiente contínuo, provavelmente o algoritmo não terá fim, pois a quantidade de nós que sucedem um outro é vasta, e a profundidade é muito maior do que em um ambiente contínuo. Em resumo, o algoritmo A* só funciona em ambientes discretos.

d. A busca em largura é completa mesmo se os custos de passos iguais a zero forem permitidos.

Verdadeiro. Precisaríamos apenas que tratar a ordem de inserção na fila, para que os nós mais profundos continuassem mais ao final, permitindo assim o desenvolvimento do algoritmo de busca.

e. Assuma que a torre pode se mover em um tabuleiro de xadrez qualquer quantidade de quadrados em linha reta, verticalmente ou horizontalmente, mas não pode pular sobre as peças. A distância de Manhattan é uma heurística admissível para o problema de movimentar a torre do quadrado A para o B no menor número de movimentos.

Falsa. Por conta de permitirmos agora o movimento na diagonal, a distância de Manhattan não seria mais admissível; sabemos que uma heurística admissível não pode superestimar um resultado, e isso aconteceria para os movimentos na diagonal, vamos observar a imagem abaixo:



Imagine que queremos chegar do ponto no canto inferior esquerdo até o ponto superior direito, cada movimento para cima e para baixo tem custo 1, se andássemos na diagonal da maneira com que mostra a figura para calcular seu custo, obteríamos custo 12. Já se pegássemos o caminho em vermelho, teríamos custo 12 também, ou seja, segundo o algoritmo, irmos pela diagonal, ou irmos tudo para cima, depois para direita, teria o mesmo custo, algo que sabemos que é falso, pois o movimento na diagonal neste caso seria muito melhor. Logo, podemos concluir que a distância de Manhattan funciona apenas para movimentos diagonais e verticais.