UFU/FACOM Disciplina: IA - Profa. Gina

Primeiro Trabalho de Programação – Grupo 3

Descrição: O objetivo do trabalho consiste em obter uma solução computacional para dois problemas distintos utilizando diferentes algoritmos de busca. As mesmas implementações dos algoritmos devem ser utilizadas para os dois problemas escolhidos.

Os problemas são:

1) Quebra-cabeças de N^2-1 peças deslizantes (N=3 e N=4)
2) Labirinto em uma sala N x N (N entre 6 e 10).

O Quebra-cabeças de N^2-1 peças deslizantes foi visto como o Problema de 8 peças deslizantes (N=3), que além de amplamente conhecido, foi discutido extensivamente em sala. O caso genérico é um quadrado com NxN células onde N^2-1 peças deslizantes. Fazer também para N=4, com 15 peças.

O Problema do Labirinto é relacionado à robótica onde tem-se um ambiente (sala) com obstáculos e caminhos livres. O robô deve encontrar um caminho livre entre um ponto inicial A e um ponto final B dentro da sala. Ao final desse documento é apresentada uma descrição mais detalhada desse problema.

Os algoritmos de busca que devem ser implementados para resolver os 2 problemas são:

1. Busca simples:

- Busca em Largura

- Busca em Profundidade Iterativa;

- Busca de Menor Custo (ou Busca de Custo Uniforme)

2. Busca informada:

- Busca A*

3. Busca Local:

- Subida de Encosta (hill climbing) com reinício aleatório em caso de estagnação (ótimo local);

- Recristalização Simulada (simulated annealing);

Especificações Gerais

A) Para a utilização dos algoritmos de busca informada, será necessário estabelecer uma estimativa de custo final para cada problema.
   - Para o Quebra-cabeças de peças deslizantes o grupo deve permitir o uso das duas estimativas apresentadas para o puzzle de 8 peças (o usuário define qual utilizar): h1: número de peças na posição errada e h2: soma das distâncias até a posição correta.
   - Para o problema do Labirinto, cada grupo deve pensar na solução para estimativa.

B) Para os algoritmos de busca local também é necessário pensar em operações de vizinhança e uma função de avaliação.
   - Para o Quebra-cabeças de peças deslizantes o grupo deve utilizar como avaliação a estimativa similar à apresentada para o puzzle de 8 peças: a1: número de peças na posição correta e a2: 50 - soma das distâncias até a posição correta (trocar 50 por 100, para 15 peças)
   - Para o problema do Labirinto, cada grupo deve pensar na solução para operador e função de avaliação.

C) Nos algoritmos de busca local, o programa também deve apresentar a sequência de passos desde a solução inicial até a obtenção da solução final.

D) Nos algoritmos de busca em árvore, deve-se evitar o processamento redundante na busca: Durante a busca, um mesmo estado pode ser visitado muitas vezes. Para melhorar a eficiência da busca é preciso evitar o processamento de nós repetidos. Se um nó já foi visitado

então ele não deve ser incluído na lista de nós a serem visitados. Você deve manter registro dos estados já visitados.

E) Para cada experimento, o programa deve imprimir algumas informações: (a) solução encontrada (sequência de estados desde o nó inicial até a meta); (b) número total de nós visitados (nós gerados que já foram testados e para os quais, se possível, foram gerados seus sucessores), (c) a profundidade em que a meta foi encontrada; (d) custo da solução

   Por exemplo, produziria a seguinte saída:

   Solução: BBXPP-BBPXP-BXPBP-BPXBP-XPBBP
   número de nós visitados: 20
   profundidade da meta: 4
   custo da solução: 6

Submissão do trabalho.

O envio do trabalho deve conter a listagem de todos os códigos relevantes, listagens dos dados de teste juntamente com a saída produzida por cada um. Além dos códigos um relatório também deverá ser escrito contendo, no mínimo:

- Introdução. Seção que descreve os problemas a serem resolvidos;

- Experimentos. Seção que descreve os experimentos realizados para cada um dos algoritmos em cada um dos problemas investigados. Apresentar Alguns exemplos de execução para cada problema (variando o N e a condição inicial). Esta seção deve incluir um gráfico mostrando o tempo gasto pelo computador em função da quantidade N de cada problema, para cada um dos algoritmos especificados.

- Conclusão. Seção que conclui o relatório mostrando as dificuldades encontradas e observações pessoais a respeito do trabalho desenvolvido.

Observações finais

1. o trabalho poderá ser realizado em grupo constituído por 3 ou no máximo 4 alunos. Observa-se que a sua avaliação será feita de forma individualizada com base em apresentação a ser agendada com professor em que necessariamente todos os componentes do grupo deverão estar presentes.

2. A escolha da linguagem de programação a ser utilizada fica a critério do grupo. Quero apenas lembrar que os algoritmos foram apresentados num formato procedimental, o que permite que o desenvolvimento possa ser feito usando uma linguagem de programação tal como C ou Java, que já foi vista por vocês em outras disciplinas. Caso o grupo opte por uma linguagem diferente das especificadas, a professora deve ser consultada com antecedência para evitar uma escolha inadequada pelo grupo.

Especificação do Problema do Labirinto

No Problema do Labirinto, o robô encontra-se em um ambiente (sala) com obstáculos e caminhos livres. O robô possui um mapa desse ambiente e deve encontrar um caminho livre entre um ponto inicial A e um ponto final B dentro da sala. O mapa é uma matriz quadrada de ordem N (usar ambientes com N entre 6 e 10). A figura abaixo apresenta um labirinto de ordem 10 (10x10), que deve necessariamente ser utilizado pelo grupo. Células com “1” representam posições na sala por onde é possível se mover (ou células livres). As células com “0” representam obstáculos por onde o robô não pode passar (paredes). A célula marcada com “2” representa por onde o robô iniciará o percurso e a célula de marcada com “3” onde ele precisa chegar. O grupo deve elaborar pelo menos 5 ambientes para cada tamanho de N, de forma que sempre exista ao menos quatro soluções possíveis para o problema e a busca deve retornar o melhor caminho. Também devem ser previstos casos onde existem caminhos alternativos que “desaguam” em uma parede. Obs: o código deve ser preparado para aceitar uma nova especificação de labirinto que seja repassada como entrada.