ACH2024 ALGORITMOS E ESTRUTURAS DE DADOS II

Semestre 2010-2 - Exercício prático – Caminho em grafo– t.02

Contexto: Seja um ambiente virtual representado por um grafo não-dirigido formado por salas (vértices) e ligações entre elas (arestas). O ambiente deve ser percorrido por um agente de software (e.g., um robô) a partir de uma sala marcada como **início** até a sala marcada como **fim**, que é o objetivo final do agente. As salas do ambiente podem estar trancada ou não, e o acesso às salas trancadas só é permitido se o agente tiver obtido previamente a chave de acesso. A chave dá acesso automático a todas as salas trancadas do ambiente, e ela é adquirida pelo agente quando este passa pela sala marcada como **chave**.

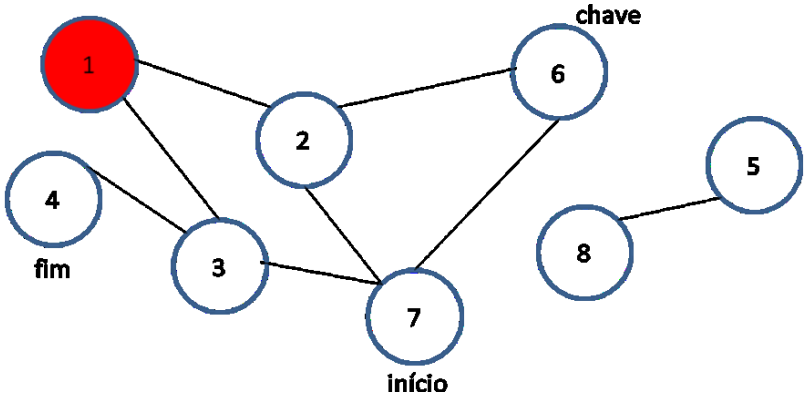
Descrição do EP: A partir do projeto exemplo disponibilizado no COL (*projetoEP.dev*), editar o módulo *trabalho.cpp* implementando a seguinte função **caminho** sobre uma estrutura de dados do tipo grafo não dirigido representado por listas de adjacências:

```
NO *caminho(VERTICE *g, int inicio, int fim, int chave)
```

Entrada: um grafo de listas de adjacências **g** representando as salas possíveis (vértices) e os caminhos entre elas (arestas), a identidade das salas **início**, **fim** e **chave**. Dentro dos vértices de **g** há também um campo booleano **aberto** indicando se a sala encontra-se já aberta (i.e., sem necessidade de encontrar a chave) - veja o módulo *grafos.h* do projeto disponibilizado no COL.

Saída: uma lista ligada de inteiros contendo um percurso válido de **início** até **fim**, i.e., um percurso de salas adjacentes que não inclua acesso inválido a salas trancadas sem que a chave tenha sido obtida.

Exemplos: Considerando-se que no grafo abaixo o vértice vermelho (1) está trancado, são exemplos de soluções válidas: {7,3,4}, {7,6,2,1,3,4} e {7,3,7,2,6,7,3,4}. Exemplos de soluções incorretas incluem {7,2,1,3,4}, que atravessa a sala trancada (1) sem ter obtido a chave em (6), {7,6,3,4} que salta do vértice (6) para uma posição não-adjacente (3), e {7,2,1} que não termina no estado final esperado.



O que implementar e como:

1. A estrutura a ser manipulada é um grafo não-dirigido contendo no mínimo 2 e no máximo 20 vértices. Os vértices são numerados de 1 a 20. Vértices que não tenham adjacências devem ser desconsiderados. O grafo pode ser conexo ou não.
2. O projeto exemplo fornecido contém uma função *main.cpp* que pode ser usada para execução e teste do seu programa (já contendo os tipos de dados manipulados e outras especificações globais) e o módulo *trabalho.cpp* onde deve ser realizado o trabalho propriamente dito.
3. Para ser considerada válida, a lista ligada fornecida como resposta deve obrigatoriamente atender aos seguintes requisitos.
   - A lista de resposta deve seguir o formato especificado no projeto, com um inteiro em cada nó.
   - O campo *prox* do último nó da lista deve obrigatoriamente apontar para NULL.
   - Não use nó cabeça, sentinela, circularidade ou encadeamento duplo. Siga o modelo do projeto.
   - Se um caminho for encontrado, o primeiro nó da resposta contém obrigatoriamente o nro. do vértice **início**, e o último nó contém obrigatoriamente o nro. do vértice **fim**.
   - Se não existe caminho possível, a lista simplesmente fica vazia (NULL).

- Todos os elementos da lista devem representar uma sequencia ordenada de vértices adjacentes no grafo, ou seja, o agente não pode "voar" de uma sala para outra que não seja adjacente.
- Uma sala *i* tal que **g[*i*].aberto** seja *false* só pode ser incluída no percurso se a sala **chave** tiver sido alcançada antes de *i*, ou seja, se o vértice **chave** aparecer antes de *i* na sequência.
- Uma vez obtida a chave, assume-se que todas as salas são acessíveis.

4. Não há restrições quanto ao nro. de vezes que cada sala é visitada, ou se o agente anda em círculos, refaz o caminho etc. mas deve ser observada a ordem de adjacência das salas, começando por **início**.
5. Um dado problema **(g,início,fim,chave)** pode ter múltiplas soluções possíveis, mas seu objetivo é achar apenas uma solução válida *qualquer*, ou então retornar uma lista vazia (NULL).
6. Para alguns percursos mais simples pode ser desnecessário encontrar **chave**, ou seja, o fim pode ser alcançável de forma direta se todas as salas de **início** até **fim** já estiverem abertas.
7. O valor da sala **chave** pode ser definido como sendo -1. Isso significa que não existe nenhuma **chave** naquele ambiente (mas ainda assim pode haver um caminho possível).
8. O grafo pode ser desconexo, ou seja, pode haver vértices inacessíveis a partir do **início**.
9. Tenha em mente que tudo que for necessário para executar a função solicitada deve obrigatoriamente estar em *trabalho.cpp*, ou seu EP estará incompleto.
10. Além da função acima, solicita-se que o aluno complete as funções *aluno1*, *aluno2*, *nrousp1* e *nrousp2* e *turma,* também presentes no módulo *trabalho.cpp* para fins de identificação dos autores. Para trabalhos individuais, deixe as informações do segundo autor em branco.
11. IMPORTANTE: o seu programa será corrigido de forma *automática*, e por isso você não pode alterar as assinaturas da função solicitada, nem os tipos de dados ou especificações em *grafos.h*.

Modalidade de desenvolvimento: Este trabalho pode ser desenvolvido individualmente ou em duplas desde que previamente informadas até a data da P1. Após esta data, novas duplas não serão aceitas. Por favor não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.

Ferramentas de desenvolvimento: 0 programa deve ser compilável no Dev-C++ versão 4.9.9.2. sob Windows XP ou Vista. Por favor teste seu programa no DevC++ antes de entregá-lo.

O que entregar: Apenas o arquivo *trabalho.cpp* extraído do projeto implementado, observando que ele deve conter todas as rotinas necessárias para a execução das funções solicitadas.

Como entregar: A entrega será realizada via sistema COL até a data e hora de inicio da P2. EPs entregues após este horário serão desconsiderados.

Critérios de avaliação: O programa será testado com 10 chamadas fornecendo configurações variadas de entrada. Cada resposta correta vale 1,0 ponto. Como o número de testes realizados na avaliação é necessariamente reduzido, um programa que esteja funcionando apenas de forma parcial pode facilmente ficar com uma nota muito abaixo do esperado. Por este motivo, sugere-se que o programa seja testado de forma *exaustiva*, com vários tipos de entradas e com especial atenção a casos excepcionais, para assim garantir um resultado satisfatório.

Penalidade prevista: Semelhança com outros trabalhos: nota zero a todos os envolvidos.

Lembrete: Este EP deve ser desenvolvido obrigatoriamente por todos os alunos de AED2. Sua nota faz parte da 2ª. avaliação da disciplina, que é calculada como o *mínimo* entre P2 e EP. A nota do EP não é passível de substituição e, caso este não seja entregue, implica nota zero na 2ª. avaliação independentemente do resultado na P2 ou SUB.

Bom trabalho a todos!