ACH2024 ALGORITMOS E ESTRUTURAS DE DADOS II

Semestre 2023-1 - Exercício prático – Caminho em grafo– t.04

Contexto: Seja um ambiente virtual representado por um grafo não-dirigido ponderado formado por salas (vértices) e ligações entre elas (arestas). Cada ligação é associada a um custo de deslocamento representado pelo peso da aresta. 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 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 *modelo.txt* disponibilizado com esta atividade, implementar a seguinte função *caminho* sobre uma estrutura de dados do tipo grafo não dirigido ponderado:

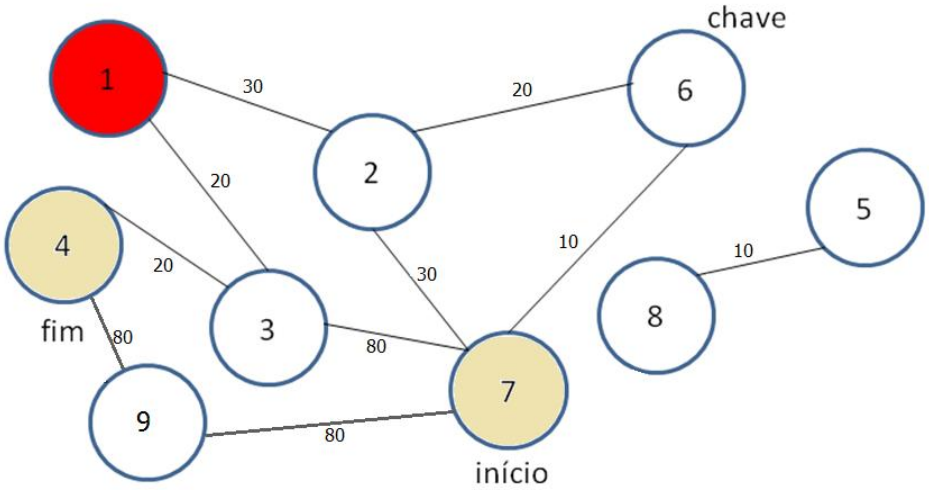
```
NO *caminho(int N, int A, int *ijpeso, int *aberto, int início, int fim, int chave);
```

Grafo de entrada: Os parâmetros *N*, *A* e o vetor *ijpeso* definem um grafo de *N* vértices e A arestas. Os vértices são numerados de 1 a *N* e as arestas são representadas pelo vetor *ijpeso* de tamanho 3 * *A*, contendo triplas na forma vértice origem *i*, vértice destino *j*, peso. Por exemplo, dado *A*=2 e um vetor *ijpeso*={1,2,3,4,5,6}, o grafo em questão possui uma aresta entre os vértices 1 e 2 com peso 3, e uma segunda aresta entre 4 e 5 com peso 6. Além disso, cada uma das *N* salas (vértices numerados 1..*N*) pode estar aberta (1) ou fechada (0), conforme indicado pelo vetor de *N* elementos *aberto* (que contém apenas valores 0 e 1). Para manipulação desta estrutura pelo EP, sugere-se assim que seja primeiramente criado um grafo (em listas de adjacências ou em matriz) a partir deste conjunto de parâmetros.

Parâmetros do problema: Além da especificação do grafo em si, os inteiros *início*, *fim* e *chave* representam as salas com estas respectivas funções.

Saída: um ponteiro do tipo NO* representando **o caminho de menor custo possível** de *início* até *fim*, i.e., um percurso válido de salas adjacentes que não inclua acesso inválido a salas trancadas sem que a chave tenha sido previamente obtida, se for o caso. Use o campo adj do tipo NO para armazenar cada vértice da sequência de resposta. O percurso pode ou não incluir a passagem pela chave. Se dois ou mais caminhos válidos possuírem custo mínimo, qualquer um deles pode ser retornado como resposta. O valor da sala *chave* pode eventualmente ser definido como sendo zero. Isso significa que não existe nenhuma chave naquele ambiente (mas ainda assim pode haver um ou mais caminhos possíveis que não dependam de chave). Note também que o caminho de custo mínimo pode ou não exigir a chave, ou seja, cabe ao seu algoritmo decidir se o melhor é pegar a chave ou não.

Exemplo: Supondo-se que no grafo abaixo o vértice vermelho (1) está trancado e os demais estão abertos, o custo mínimo para percorrer de início até fim é 100. São exemplos de soluções válidas {7,3,4} (que não precisou da chave e tem custo total 100) e {7,6,2,1,3,4} (que obteve a chave em 6 antes de chegar em 1, e também tem custo 100). São exemplos de soluções inválidas {7,2,1,3,4} (que apesar do custo mínimo atravessa a sala 1 sem ter obtido a chave), {7,6,3,4} que salta do vértice (6) para uma posição não-adjacente (3), {7,2,6} que não termina no estado final esperado, e {7,9,4}, que não possui custo mínimo.



Para ser considerada válida, a lista fornecida como resposta deve obrigatoriamente atender aos seguintes requisitos.

- A lista de resposta deve seguir o formato especificado no projeto, com um inteiro (adj) 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.
- Se não existe caminho possível, a lista fica vazia (NULL).
- Se a lista não for vazia (ou seja, se um caminho for encontrado), o primeiro nó contém obrigatoriamente o nro. do vértice *início*, e o último nó contém obrigatoriamente o nro. do vértice *fim*.
- Os elementos da lista representam 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* que esteja fechada só pode ser incluída no percurso se a sala *chave* tiver sido alcançada antes de *i*, ou seja, se o vértice *chave* ocorrer antes de *i* na sequência dada como resposta.
- A sala *chave*, por definição, nunca está trancada.
- Uma vez obtida a chave, assume-se que todas as salas que estavam trancadas estão agora acessíveis.

Tenha em mente que tudo que for necessário para executar a função solicitada deve obrigatoriamente estar no arquivo entregue, ou seu EP estará incompleto.

Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc. Apenas implemente a função solicitada.

A função *main*() serve apenas para seus testes particulares, e não precisa ser entregue. Caso você prefira mantê-la no corpo do programa, pede-se apenas que *main*() seja a última função do programa, ou seja, que não haja nenhum código abaixo dela.

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 (*typedef*) do modelo fornecido.

O EP pode ser desenvolvido individualmente ou em duplas, desde que estas sejam cadastradas até **quinta-feira 11 de maio**. Duplas formadas tardiamente não serão consideradas. Alunos que queiram trabalhar individualmente também devem se cadastrar. Por favor acesse o link abaixo e cadastre seus dados:

https://docs.google.com/spreadsheets/d/1Jfp_aF_2gKafmEDqpf7NDFXcC_w_fV3cYBJatUMqOwE/edit?usp=sharing

Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.

0 programa deve ser compilável no Codeblocks 13.12 sob Windows 10 ou superior. Será aplicado um desconto de 30% na nota do EP caso ele não seja imediatamente compilável nesta configuração.

O que/como entregar:

- A entrega será via upload no sistema e-disciplinas por **apenas um** integrante de cada dupla.
- Entregue apenas o código da função principal e das funções auxiliares que ela invoca.
- A extensão do arquivo deve ser .cpp - **favor não compactar**.
- Preencha as funções nroUSP1() e nroUSP2() do modelo disponível no sistema para que você seja identificado. Se o EP for individual, mantenha o valor do segundo nro. como zeros.
- Na primeira linha do código, escreva um comentário "//" com os nomes dos integrantes.

Prazos e penalidades:
O EP deve ser depositado no prazo definido na atividade cadastrada no sistema. Não serão aceitos EPs entregues depois do prazo, independentemente do motivo. Entregas no último dia são assim por conta e risco do aluno, e nenhum tipo de imprevisto de última hora (e.g., problemas de saúde, indisponibilidade de rede etc.) pode ser usado como justificativa para o atraso. O EP é uma atividade para ser desenvolvida ao longo de várias semanas, não nos últimos dias antes da entrega. É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema. Atrasos/falhas na submissão invalidam o trabalho realizado. Após o *upload*, verifique se você consegue abrir o arquivo que está no sistema, e certifique-se de que é a versão correta do programa.

Critérios de avaliação:
A função será testada com uma série de chamadas repetidas e consecutivas, com diversas listas de inteiros positivos como entrada. É assim importante assegurar que o seu programa funciona desta forma (por exemplo, chamando-o dentro de um laço *for*), e não apenas para um teste individual. Um teste é considerado correto se a lista fornecida representar um caminho de custo mínimo válido, ou incorreto em caso contrário. Erros de alocação de memória ou compilação invalidam o teste, assim como a ausência de funções auxiliares necessárias para a execução do programa. Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED2. Sua nota é parte integrante da 1ª. avaliação e *não é* passível de substituição.