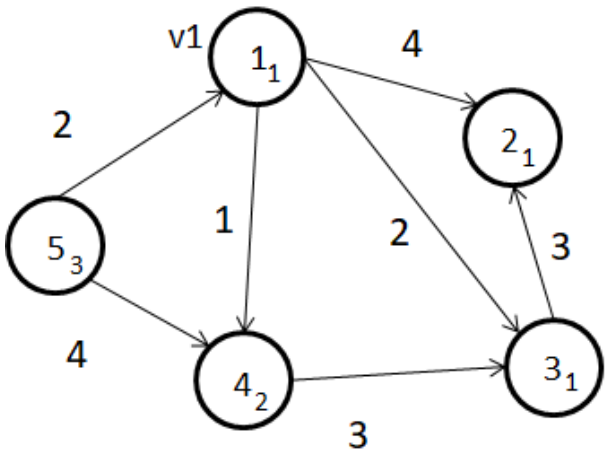
ACH2024 ALGORITMOS E ESTRUTURAS DE DADOS II

Semestre 2022-1 - Exercício prático – caminho mais curto e de custo mínimo

Estagiário PAE: Wesley Ramos dos Santos wesley.ramos.santos@usp.br

1. Seja um grafo ponderado dirigido em lista de adjacências **g** de **V** vértices numerados de 0 até **V**-1 representando localidades (vértices) e suas conexões (arestas), no qual a ponderação das arestas (int **custo**) representa o custo para viajar de uma localidade para outra. As localidades possuem também um campo int **tipo** indicando o tipo de atração oferecida. Por exemplo, o tipo 1 pode representar uma cidade litorânea, o tipo 2 pode ser uma localidade no interior etc.
2. O objetivo do trabalho é implementar de forma correta e completa a função *EncontrarCaminho* conforme modelo fornecido. A assinatura da função é a seguinte:

   NO* EncontrarCaminho(VERTICE* g, int V, int v1, int t)
3. A função recebe como entrada um ponteiro de grafo dirigido ***g** de **V** vértices numerados de 0 até **V**-1, um vértice inicial **v1** e o tipo **t** de vértice a ser localizado, e retorna uma lista ligada contendo um caminho que vai de **v1** até um vértice **v2** a ser localizado, tal que **v2** (a) seja do tipo **t** (2) seja o vértice daquele tipo que está o mais próximo possível da origem **v1**, e (3) acarrete o menor custo possível a partir da origem **v1**.
4. Havendo empate, ou seja, se houver mais de um vértice v2 à mesma distância da origem **v1** e com o mesmo custo, o algoritmo pode retornar qualquer uma das respostas válidas.
5. A lista a ser retornada deverá portanto começar em **v1** e terminar em **v2** (que será obrigatoriamente um vértice do tipo **t**) seguindo um caminho válido e de menor custo possível entre eles.
6. Exemplo: considere o objetivo de viajar de **v1** até uma localidade de tipo **t**=1 mais próxima, e com menor custo possível no grafo a seguir, em que os vértices são numerados de 1 a 5 e acompanhados de um subscrito indicando seu tipo **t**:



7. Neste exemplo, há dois vértices (candidatos a v2) que são de tipo 1 que estão à mesma distância da origem, ou seja, os vértices 2 e 3. O vértice a ser escolhido é entretanto v2=3, pois o custo para alcançá-lo é menor. A resposta do algoritmo é assim o caminho de custo mínimo <1, 3>.
8. Note entretanto que este exemplo é um caso muito simples, constituindo um caminho de apenas dois vértices, e que a solução deve considerar casos arbitrariamente complexos. É necessário primeiramente identificar quais os vértices do tipo desejado que estão mais próximos da origem, e depois computar o caminho de custo mínimo para cada um de modo a selecionar a resposta correta.

9. Restrições de implementação:
   - Sua solução deve necessariamente contemplar o uso de um ou mais **algoritmos de busca em grafos** implementados em listas de adjacências, acompanhado das respectivas declarações *typedef* utilizadas – mas não altere a definição do grafo (tipos VERTICE e NO) ou seu programa não vai conseguir tratar os grafos fornecidos como teste, que possuem exatamente os campos declarados no código exemplo.
   - Não defina variáveis globais.
   - Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc.
10. O EP pode ser desenvolvido individualmente ou em duplas, desde que estas sejam cadastradas até **domingo 2 de maio** no link a seguir. Alunos que queiram trabalhar individualmente também devem se cadastrar, e duplas formadas tardiamente não serão consideradas. Por favor acesse o link abaixo usando seu email USP (solicitações de acesso com outros endereços serão ignoradas):

    https://docs.google.com/spreadsheets/d/1_wy0Uko3vX8GE07G1WP4X6X6-yRdu6eCpLdmmRvU7EM/edit?usp=sharing

    Importante: anote o número do seu grupo para informá-lo no código a ser entregue.
11. A função implementada deve estar inteiramente contida em um arquivo trabalho.cpp, incluindo todo código necessário para executá-la, declarações *typedef* etc.. Note no entanto que para testar a sua implementação e garantir sua correção você provavelmente terá de criar várias outras funções auxiliares (e.g., entrada de dados, exibição etc.) que não serão avaliadas. E que não devem ser entregues

O que/como entregar:

- 0 programa deve ser compilável no Codeblocks 13.12 sob Windows 7 ou superior. Será aplicado um **desconto de 50%** na nota do EP caso ele não seja **prontamente** compilável nesta configuração.
- Entregue um arquivo trabalho.cpp (exatamente com este nome, sem compactação) contendo a função do EP e todas as rotinas que ela invoca, inclusive as declarações *typedef* etc.
- A entrega será via *upload* no sistema edisciplinas por **apenas um** integrante de cada dupla.
- Preencha no código as declarações *nroUSP1*, *nroUSP2* e *grupo* para que você seja identificado. Se o EP for individual, mantenha o valor do segundo nro. como zeros.

Prazos:
O EP deve ser depositado no prazo definido na atividade cadastrada no sistema. Não serão aceitos EPs entregues depois do prazo ou incompletos, 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 no último dia da entrega.

É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema (ou seja, sugere-se fazer *download* novamente para ter certeza). Atrasos/falhas na submissão invalidam o trabalho realizado, assim como entregas de versões erradas ou incompletas. Certifique-se também de que a versão entregue é a correta *antes* do prazo final. Não serão aceitas substituições.

Avaliação:

O objetivo do exercício é o de gerar sequências de ações **corretas**, que não necessariamente precisam ser eficientes ou "elegantes". Ou seja, basta que o programa forneça uma resposta correta qualquer para cada estado de teste fornecido. Não existe assim solução "meio" correta: ou a função faz o que foi proposto, ou não faz. Erros de execução e de alocação de memória (muito comuns!) também invalidam o teste, assim como a ausência de eventuais funções auxiliares necessárias para a execução do programa.

O EP será testado com uma série de chamadas fornecendo-se sempre estados iniciais válidos como entrada. Cada teste sem erro de execução e que retorne a resposta correta vale um ponto. Para qualquer resultado diferente do esperado, passa-se ao próximo teste sem pontuar.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED2. Sua nota é parte integrante da média final da disciplina e *não é* passível de substituição. Problemas com EPs – principalmente erros de execução e plágio – são a principal causa de reprovação na disciplina. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.

Não está funcionando? Use comandos *printf* para saber por onde seu programa está passando e os valores atuais das variáveis, ou os recursos de *debug* do *CodeBlocs*. O propósito do exercício é justamente o de encontrar erros por conta própria – não espere ajuda para isso. Bom trabalho!