# ACH2024 ALGORITMOS E ESTRUTURAS DE DADOS II

## Semestre 2024-1 - Exercício prático – Caminho em grafo – t.94

**Contexto**: Seja um ambiente virtual representado por um grafo não-dirigido formado por cidades (vértices) e ligações rodoviárias entre elas (arestas), em que você precisa encontrar um caminho de uma determinada origem até um determinado destino contando com um montante limitado de recursos financeiros.

Cada rodovia possui um pedágio que precisa ser pago para poder transitar, cujo valor é descontado do montante total que você possui. Em contrapartida, em cada cidade há uma oferta de trabalho disponível, e você sabe de antemão quanto vai ganhar se for até lá. Mas para isso, claro, precisa ter dinheiro suficiente para visitá-la e então receber o pagamento.

**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 dirigido ponderado:

```
NO *caminho(int N, int A, int ijpeso[], int ganhos[], int início, int fim, int* dinheiro);
```

**Grafo de entrada:**

- Os parâmetros *N, A* e os vetores *ijpeso*[] e *ganhos*[] definem um grafo de *N* vértices numeradas de 1 a N e A arestas. As posições zero desses vetores não são utilizadas.
- Os vértices são numerados de 1 a *N* (portanto, crie um vetor de N+1 posições) e, quando visitados, oferecem um ganho cujo valor é dado pelo vetor *ganhos*[]. Não use a posição zero do vetor. O vértice 1 deve estar na posição 1, e o vértice N deve estar na posição N.
- As arestas são representadas pelo vetor *ijpeso* de tamanho 3 * *A*, contendo triplas na forma vértice origem *i*, vértice destino *j*, e valor do *pedágio*.

Por exemplo, dado *A*=2 e os vetores *ijpeso*[] ={1,2,3,4,5,6} e *ganhos*[] = {2,4,6,8,10,12}, 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* cidades (vértices numerados 1..*N*) oferece um ganho se visitada. A cidade 1 tem ganho 2, a cidade 2 tem ganho 4, e assim por diante.

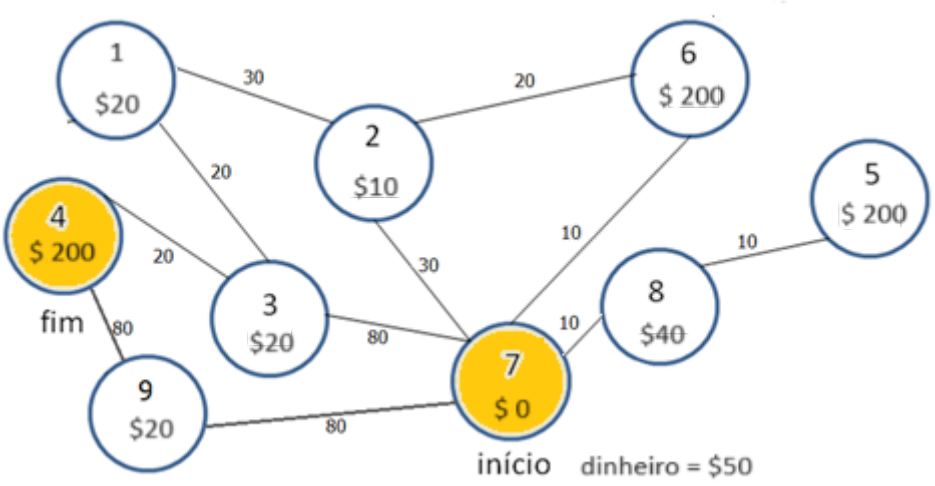
Para manipulação desta estrutura pelo EP, sugere-se assim que seja primeiramente criado um grafo (em listas de adjacências ou em matriz) a partir deste conjunto de parâmetros. SE CRIAR UM GRAFO EM VETOR, NÃO UTILIZE A POSIÇÃO ZERO OU SEU EP SERÁ INVALIDADO. O GRAFO NÃO PODE TER VÉRTICE ZERO.

**Outros parâmetros do problema**:

Além da especificação do grafo acima, são parte da assinatura da função os inteiros *início*, *fim* e a variável compartilhada **dinheiro*. Esta última é o montante inicialmente disponível. Este montante diminui ao percorrer o grafo (pois são deduzidos os pedágios) e aumenta ao atingir uma cidade (recebendo o ganho que ela oferece). Por definição, o ganho da cidade de início é sempre zero.

**Saída**: um ponteiro do tipo NO* representando qualquer caminho **simples válido** (i.e., de cidades adjacentes sem repetição) desde *início* até *fim* que possa ser cumprido **sem ficar com montante negativo**. Use o campo adj do tipo NO para armazenar cada vértice da sequência de resposta. A lista sempre começa com o vértice *início* e termina com o vértice *fim*. Qualquer outro formato invalida a resposta. O caminho precisa ser do tipo simples, isso é, você não pode visitar a mesma cidade duas vezes.

**Exemplos**: No grafo abaixo, partindo-se de um montante inicial de $50 na cidade 7 (em amarelo), não é possível transitar para as cidades 9 ou 3, cujos pedágios (de $80 cada) superam este valor. No entanto, é possível seguir para a cidade 2 (pagando $30 e então recebendo $10), depois para a cidade 1 (pagando $30 e recebendo $20), cidade 3 (pagando $20 e recebendo $20), e finalmente atingindo o destino na cidade 4 (pagando $20 e recebendo $200). Assim, uma resposta possível seria a lista ligada {7,2,1,3,4}. Outra solução possível seria, por exemplo, {7,6,2,1,3,4}.




Exemplos de soluções inválidas seriam {7,9,4} (porque falta dinheiro para chegar em 9), {7,1,3,4} (porque salta entre os vértices não adjacentes 7 e 1) ou {7,2,1,3} (porque não chega ao fim), dentre outras possibilidades.

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.
- Não use nó cabeça, sentinela, circularidade ou encadeamento duplo. É uma lista ligada simples do tipo NO fornecido.
- 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 um caminho simples de *início* até *fim*, ou seja, uma sequência ordenada de vértices adjacentes no grafo sem "saltar" de uma cidade para outra que não seja adjacente, e sem repetição.
- O percurso pressupõe que haja fundos em dinheiro para cada movimento. Se em qualquer ponto do trajeto o saldo ficar negativo, a resposta é inválida.

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 deve ser entregue.

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 **30 de abril**. 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 usando sua conta USP:

https://docs.google.com/spreadsheets/d/16ux2Wep_hrn18ZzT4AKQ5cNbLUV9QX6Qy9jDETxPMzw/edit?usp=sharing

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

O 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 em um arquivo trabalho.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 diversos tipos de 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 uma resposta válida, 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.