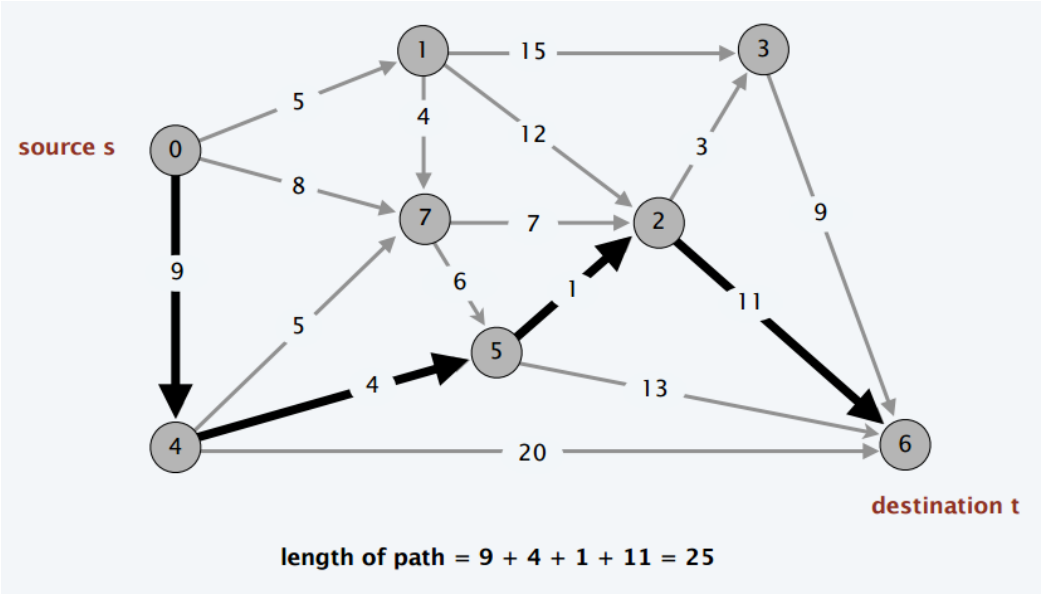
Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Ciência da Computação
Projeto e Análise de Algoritmos – 2º Semestre de 2020
Profª. Raquel Mini

TRABALHO PRÁTICO
VALOR: 15 PONTOS

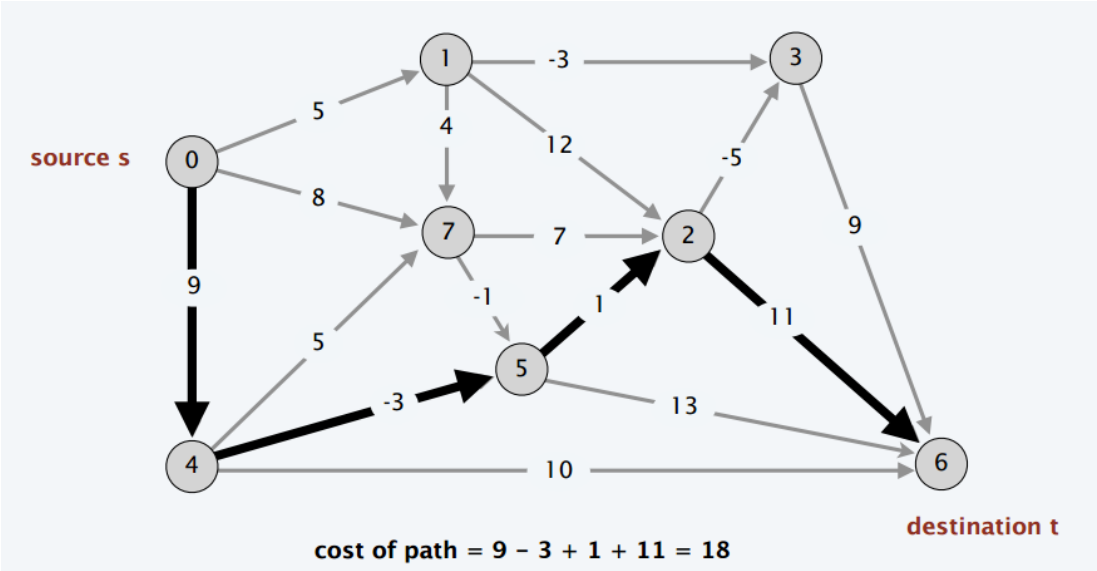
Este trabalho pode ser feito **em dupla** e deverá ser feito nas linguagens C ou C++. As respostas devem ser enviadas em um arquivo zipado contendo os seguintes arquivos: fonte (.cpp, .h) e documentação do trabalho (.pdf). Documentação entregue em formato diferente de pdf não será aceita. O código entregue deve estar comentado. O arquivo zipado deverá ser postado no Canvas até o dia **02 de novembro de 2020**.

CAMINHO MAIS CURTO EM UM GRAFO COM ARESTA DE PESO NEGATIVO

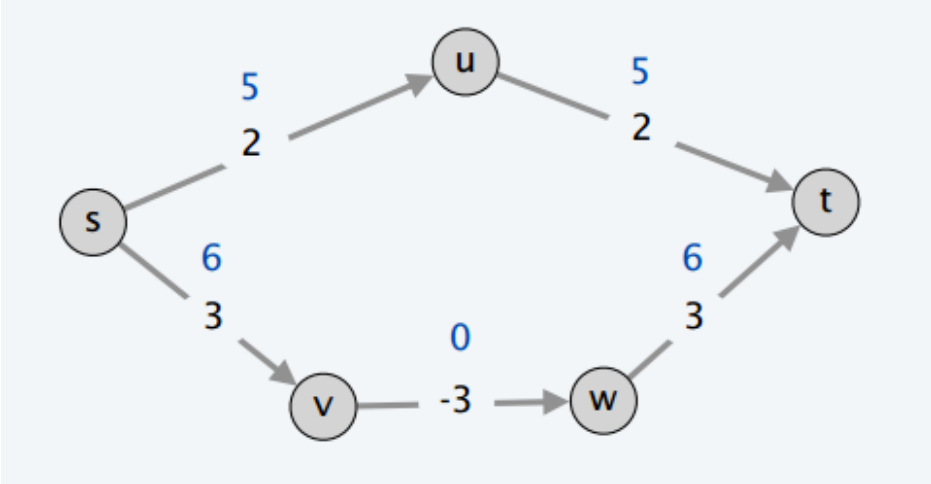
Seja G=(V,E) um grafo direcionado. Suponha que cada aresta $(i,j) \in E$ tem um peso associado c_{ij}. O peso c_{ij} representa o custo para ir diretamente do nó *i* para o nó *j* em um grafo. O Algoritmo de Dijkstra pode ser utilizado para encontrar o menor caminho entre dois vértices em um grafo com peso positivo nas arestas conforme ilustrado na figura abaixo na qual o menor caminho entre os vértices *s* e *t* é destacado.



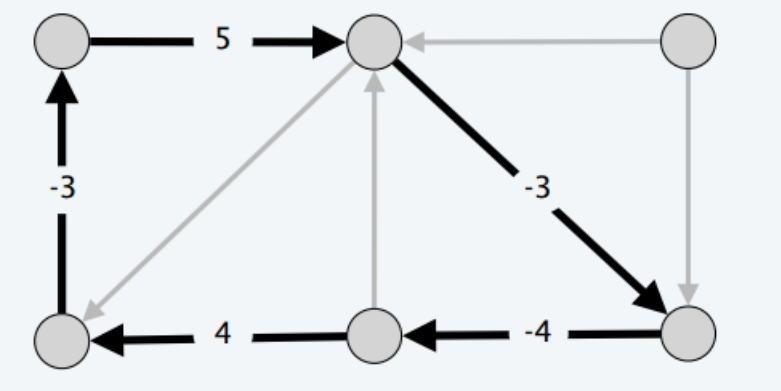
Entretanto, existem problemas mais complexos no qual precisamos encontrar o caminho mais curto entre dois vértices e existem arestas com peso negativo conforme ilustrado na figura abaixo.



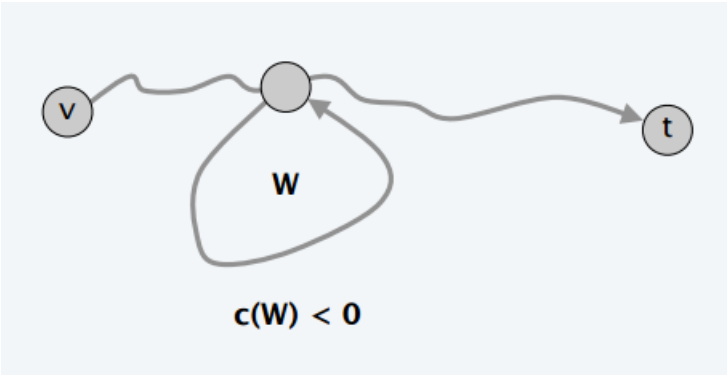
Por exemplo, considere o caso no qual os nós representam agentes de um sistema financeiro, e c_{ij} represente o custo da transação quando compramos do agente *i* e então imediatamente vendemos para o agente *j*. Neste caso, um caminho no grafo representa uma sucessão de transações e aresta com peso negativo representa uma transação com lucro. Para resolver esse problema, não basta inserir um peso constante em todas as arestas de forma a torná-las positivas. Veja na figura abaixo que o menor caminho entre *s* e *t* é *s*-*v*-*w*-*t*. Entretanto, ao incrementar em 3 unidades o peso de todas as arestas, o menor caminho seria *s*-*u*-*t*, o que nos leva a uma resposta incorreta. Portanto, o algoritmo que deve ser projetado para lidar com arestas de peso negativo deve ser mais flexível e decentralizado que o Algoritmo de Dijkstra.



Outra questão que merece destaque é a existência de ciclos negativos conforme ilustrado na figura abaixo.



Na aplicação do sistema financeiro, um ciclo negativo corresponde a uma sequência de transações com lucro, o que nos leva à seguinte situação: compramos de i_1, vendemos para i_2, compramos de i_2, vendemos para i_3, e assim por diante, finalmente chegando novamente em i_1 com lucro. Se o caminho a partir de um vértice *s* até um vértice *t* contém um ciclo negativo, então não existe um menor caminho entre *s* e *t,* pois poderíamos passar pelo ciclo negativo infinitas vezes, diminuindo cada vez mais o comprimento do menor caminho entre *s* e *t* conforme ilustrado abaixo.



Portanto, é razoável considerar o problema de encontrar o menor caminho entre dois vértices considerando que o grafo possua arestas com peso negativo, mas que não possua ciclos negativos.

O objetivo do trabalho é a implementação da solução para os seguintes problemas:

1. Algoritmo guloso para resolver o problema do caminho mais curto em um grafo com peso não negativo (Algoritmo de Dijkstra).
2. Algoritmo para determinar se existe um ciclo negativo em um grafo.
3. Algoritmo de programação dinâmica para resolver o problema do caminho mais curto em um grafo com aresta negativa, mas sem ciclo negativo.

A documentação de cada parte do trabalho deverá conter no mínimo os seguintes tópicos:

1) **Introdução**: informações gerais sobre o problema a ser tratado, o que vai ser feito no trabalho, os objetivos, visão geral sobre o funcionamento do programa.

2) **Implementação**: descrição sobre a implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas, o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Além disso, devem ser fornecidas informações técnicas sobre o ambiente computacional utilizado (sistema operacional, compilador, IDE) e também sobre como executar o seu programa.
3) **Análise de Complexidade**: análise de complexidade do pior e do melhor caso de todas as funções do programa e também do programa principal. Essa análise pode ser feita de forma mais detalhada linha por linha, somando-se as complexidades ou de forma mais geral, explicando a complexidade da função como um todo. De qualquer forma, tem que ficar claro qual é a operação relevante e também as configurações de entrada que levam ao pior e ao melhor caso.
4) **Testes**: descrever os testes realizados, mostrando a saída do programa além de eventuais análises e comparações que foram solicitadas no enunciado.
5) **Conclusão**: comentários gerais sobre o trabalho e as principais dificuldades encontradas na implementação.
6) **Bibliografia**: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da internet se for o caso
7) **Anexos**: listagem dos programas.