2009

Universidade Federal de Lavras

Alexandre Abreu
João Daher Neto
Roberto Lourenço

[TRABALHO PRÁTICO DE TÉCNICAS DE PROGRAMAÇÃO]

Implementação do algoritmo de “menor caminho” em grafos, denominado Bellman-Ford, incluindo a simulação de execução em grafos e o desenvolvimento de uma aplicação para tal.

Sumário

Descrição

O Algoritmo de Bellman-Ford é um algoritmo de busca de caminho mínimo em um grafo ponderado, ou seja, cujas arestas têm peso, inclusive negativo. O Algoritmo de Dijkstra resolve o mesmo problema, em um tempo menor, porém exige que todas as arestas tenham pesos positivos. Portanto, o algoritmo de Bellman-Ford é normalmente usado apenas quando existem arestas de peso negativo. O algoritmo resolve, pois, problemas de caminho mais curto de única origem.

Esse algoritmo usa a técnica de relaxar as arestas, diminuindo progressivamente uma estimativa no peso de um caminho mais curto da origem até cada vértice, até alcançar o peso real do caminho mais curto.

Objetivo

O presente trabalho tem o intuito de especificar e exemplificar o funcionamento e a utilidade do algoritmo de Bellman-Ford. Para tal, uma breve definição do algoritmo em sí é apresentada, assim como algumas de suas utilidades em um contexto real e seguido da explicação da aplicação desenvolvida, dos relatórios e a conclusão.

A escpecificação e exemplificação, aborda sutilmente o funcionamento e o porquê do uso do Bellman-Ford, assim como situações reais onde alguns problemas são atacados utilizando o referido algoritmo(algumas vezes com versão um pouco modificadas).

Em seguida, encontra-se a descrição da aplicação desenvolvida, onde será exposto algumas classes e suas relações, métodos, e os formatos dos arquivos utilizados para a entrada e saída de dados. Uma orientação para a utilização de tal aplicação, também será abordada.

Finalmente, será apresentado um relatório onde resultados da aplicação em grafos contendo, no mínimo, vinte vértices. Findando, pois, com uma conclusão geral.

Aplicações

Serviço móvel de satélite

Serviço móvel de satélite (MSS) é um sistema de satélites que utiliza terminais portáteis terrestre. MSS terminais pode ser montado em um navio, um avião, ou um automóvel. MSS terminais pode mesmo ser transportada por um indivíduo.

A aplicação mais promissora do serviço móvel de satélite é telefones portáteis via satélite que permitirá serviço de telefone em qualquer parte do globo. Os serviços móveis de satélites são possíveis graças aos Low Earth Orbit satélites.

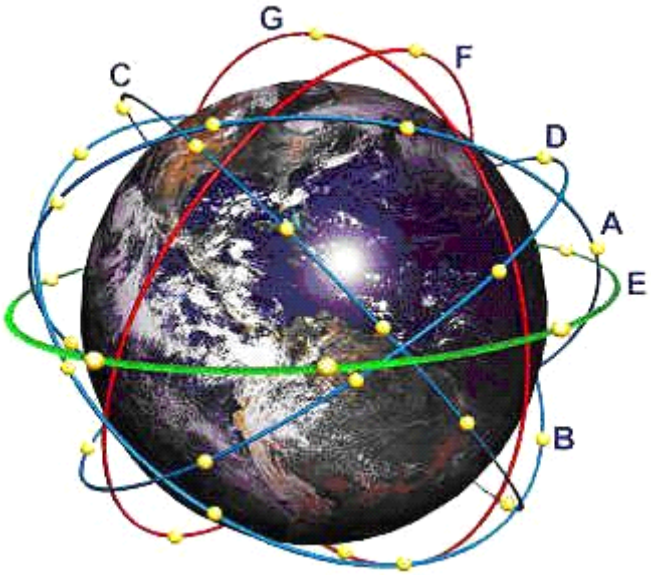
Low Earth Orbit (LEO) remete para um satélite que orbita a Terra em altitudes entre (muito aproximadamente) 200 milhas e 930 milhas. Estes satélites devem viajar muito rapidamente para resistir a atração da gravidade - cerca de 17,000 milhas por hora. Devido a isto, Lowe Earth Orbit satellies pode órbita do planeta em menos de 90 minutos.

Este sistema exigem várias dezenas de satélites, que operam em órbitas polares, para fornecer cobertura de todo o planeta.

Low Earth Orbit satélites são usados para aplicações onde um curto Round Trip Time (RTT) é muito importante. LEO satélites têm uma expectativa de vida típica de serviço de cinco a sete anos.

Para o gerenciamento dessa rede de satélites LEO, são usados basicamento dois protocolos de roteamento. O Bellman-Ford (Extended) e um outro protocolo denominado: Esforçado. Que é um novo protocolo que tem sido proposto como adequado para uso em redes LEO. Estes protocolos são comparados através de simulação computacional em duas das propostas LEO sistemas (Globalstar e Iridium), sob diferentes intensidades de tráfego.

Foram feitos comparativo com pacote de medidas de atraso, convergência, velocidade e protocolo gerais. Os resultados encontrados em ambos os protocolos foram aproximadamente equivalentes, embora o protocolo Esforçado tinha uma carga muito maior e demonstrou maior instabilidade na rede atualização períodos. Por exemplo, ao estado estacionário end-to-end os atrasos ficaram dentro de alguns milisegundos, os casos Esforçados apresentaram um aumento de 764% em tempo superior a convergência, com um aumento de 149% nas despesas gerais. Durante todos os casos, o Esforçado exigida uma média de 72,1% a mais do que o Bellman-Ford para realizar o mesmo trabalho. Esforçado foi prejudicado pela sua forte correlação entre os dados de tráfego e protocolo gerais.



Roteamento em redes Ad hoc

O algoritmo de Bellman-Ford sempre teve uma grande aplicabilidade em gerenciamento de redes, desde as mais simples até as mais complexas. Como é o caso de redes Ad hoc. Entretanto é interessante que sejam apresentadas, inicialmente, algumas das características do funcionamento dessas redes que influenciaram nas decisões adotadas pelos algoritmos.

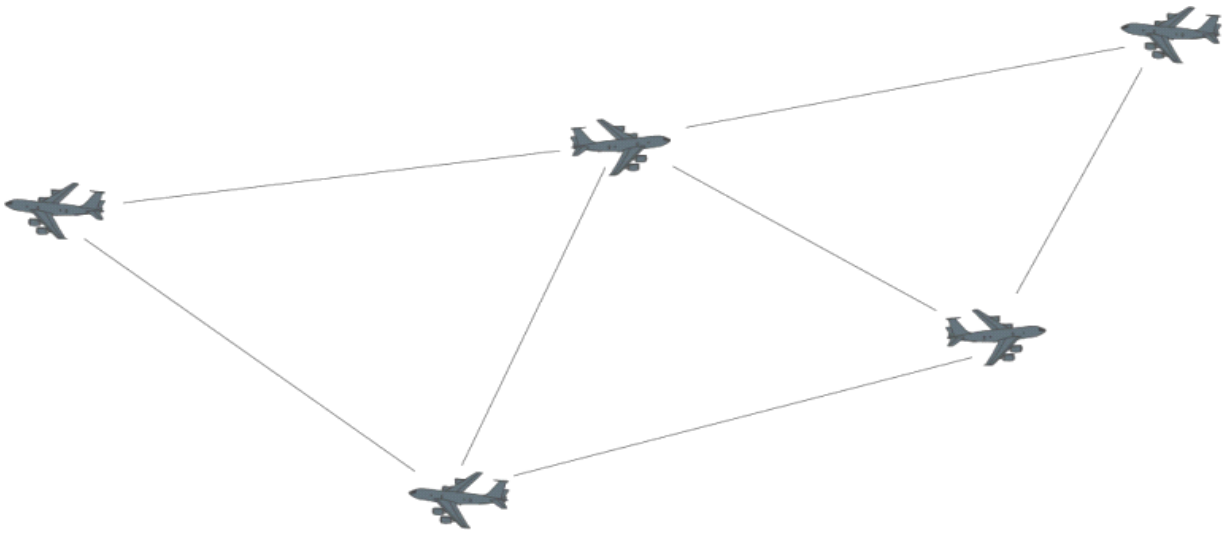
[Hass1998] Apresenta em seu estudo três características principais para redes ad hoc:

- Inexistência de uma entidade central: ao contrário de uma rede celular, onde são encontradas entidades que realizam o controle e o da rede (estações base, registro de localização, etc.), nas redes Ad hoc não existe nenhuma infraestrutura previamente estabelecida nem entidades que controlem o seu funcionamento;
- Rápidas alterações topológicas: já que se trata de uma rede que não possui uma infraestrutura básica montada e é composta por equipamentos móveis, não existe uma topologia bem definida durante o seu período de funcionamento;
- Todas as comunicações são realizadas através do meio sem fio: a comunicação através da rádio difusão torna a comunicação pouco confiável e extremamente vulnerável.

O DSDV (Destination-Sequenced Distance-Vector) [Perkins1994] é um protocolo pró-ativo baseado no mecanismo clássico de roteamento Bellman-Ford, com a eliminação de loops nas rotas. Em cada nó existe uma tabela com todos os possíveis destinos dentro da rede, e o número de saltos até cada um deles. Cada entrada nesta tabela é marcada com um número de sequência determinado pelo nó destino. Este número tem a função de distinguir rotas antigas das novas, evitando a formação de loops.

Para realizar a atualização, o DSDV dispõe de dois tipos de pacotes full dump, no qual todas as informações de roteamento são transmitidas, e os pacotes incrementais, onde apenas existe uma complementação da informação enviada no último full dump.

Cada mensagem, para a criação de novas rotas, inclui o endereço do destino, o número de saltos até o destino, o número de sequência da informação original sobre o destino e um número de sequência para esta mensagem. O DSDV utiliza sempre a rota com o número de sequência mais recente e no caso de duas atualizações possuírem o mesmo número de sequência, a rota com menor número de saltos será a escolhida.



Processo de Planejamento Militar (PPM)

O processo de planejamento militar (PPM) é uma tarefa árdua para os Oficiais Fuzileiros Navais. Planejar uma boa rota ou uma boa linha de ação (LA) para os seus grupos de desembarque não é trivial. O Oficial Fuzileiro Naval deve estar sempre atento a vários fatores relevantes a um PPM como objetivos a serem atingidos, elevações e propriedades intrínsecas ao terreno, a possível presença de inimigos, o espaço disponível para suas manobras, a viabilidade de pedir apoio de fogo, dentre muitos outros. Por isso, em meio de tantas variáveis e dados, o Oficial tende a se confundir na hora de ponderar esses dados, escolhendo uma LA não muito boa.

Na tentativa de amenizar toda essa complexidade de informações, se fez necessário criar uma ferramenta a qual buscasse tais informações para os Oficiais de maneira precisa, rápida e simples. Tal ferramenta recebeu o nome de Ferramenta de Apoio a Decisão (FAD). Esta aplicação ajuda na elaboração do PPM informando em tempo real, os dados que o Oficial precisa. Ela serve para realizar diversos cálculos, exibindo seus dados e informações de maneira numérica ou gráfica.

Tais cálculos baseiam-se, em sua maioria, nos algoritmos que envolvem grafos,pois essa foi a melhor forma de representarmos a disposição dos Acidentes Capitais(AC) e das suas Vias de Acessos (VA) pela área de operação (AOp) escolhida para o exercício. Para tal, cada AC foi atribuído a um nó do grafo enquanto cada VA foi associada a uma aresta.

Com esse grafo montado computacionalmente, incrementado com algoritmos matemáticos avançados nas áreas de teoria dos grafos e de computação gráfica, resolvemos diversas questões essenciais que auxiliam ao PPM.

Para estudar uma dada missão o comandante de um escalão segue um processo de planejamento militar (PPM) constituído de etapas e fases bem definidas.

• EXAME DA SITUAÇÃO: Análise dos fatores da situação para chegar à decisão;

• DESENVOLVIMENTO DO PLANO DE AÇÃO: Demonstração de como a será posta em execução e a ELABORAÇÃO DA DIRETIVA: Formalização da decisão sob a forma de documento;

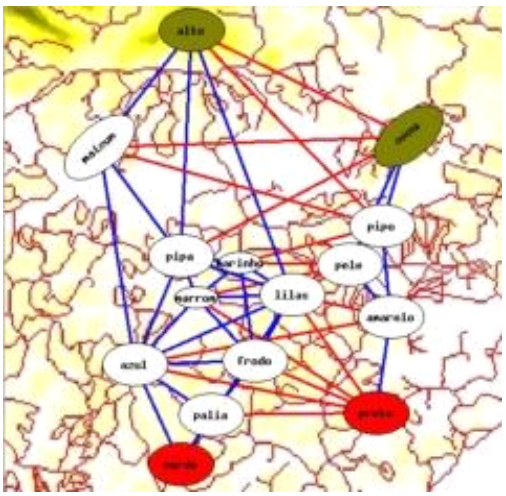
• CONTROLE DA AÇÃO PLANEJADA: Assegura que a operação se desenvolva como planejado e a revisa, se necessário.

A etapa de EXAME DA SITUAÇÃO consiste de uma extensa coleta de dados, seguida de uma análise sistemática que envolve a missão, o inimigo, o terreno, os meios e o tempo disponível; informações que o comandante leva em consideração para tomar suas decisões.

A missão já deverá ter sido estudada pelo comandante e alguns dados relevantes para nosso trabalho serão inseridos no sistema (objetivos, valor de tropa e tempo limite para a chegada nos objetivos, entre outros), através de uma interface apropriada.

De forma semelhante, o inimigo já deverá ter sido estudado pelo oficial de inteligência e suas possibilidades deverão também fazer parte dos dados de entrada do sistema.

Para ilustrar uma possível aplicação dessa ferramenta vamos mostrar um exemplo, que irá consistir em uma análise de duas possíveis linhas de ação (LAs) na região de Itaoca, ES. A escolha dos ACs foi feita de forma genérica em algumas elevações, não representando os reais posicionamento de valor militar das mesma. Os dados das duas LAs serão então analisados para determinar a mais adequada.



Grafo sobre a AOp em Itaoca, ES.

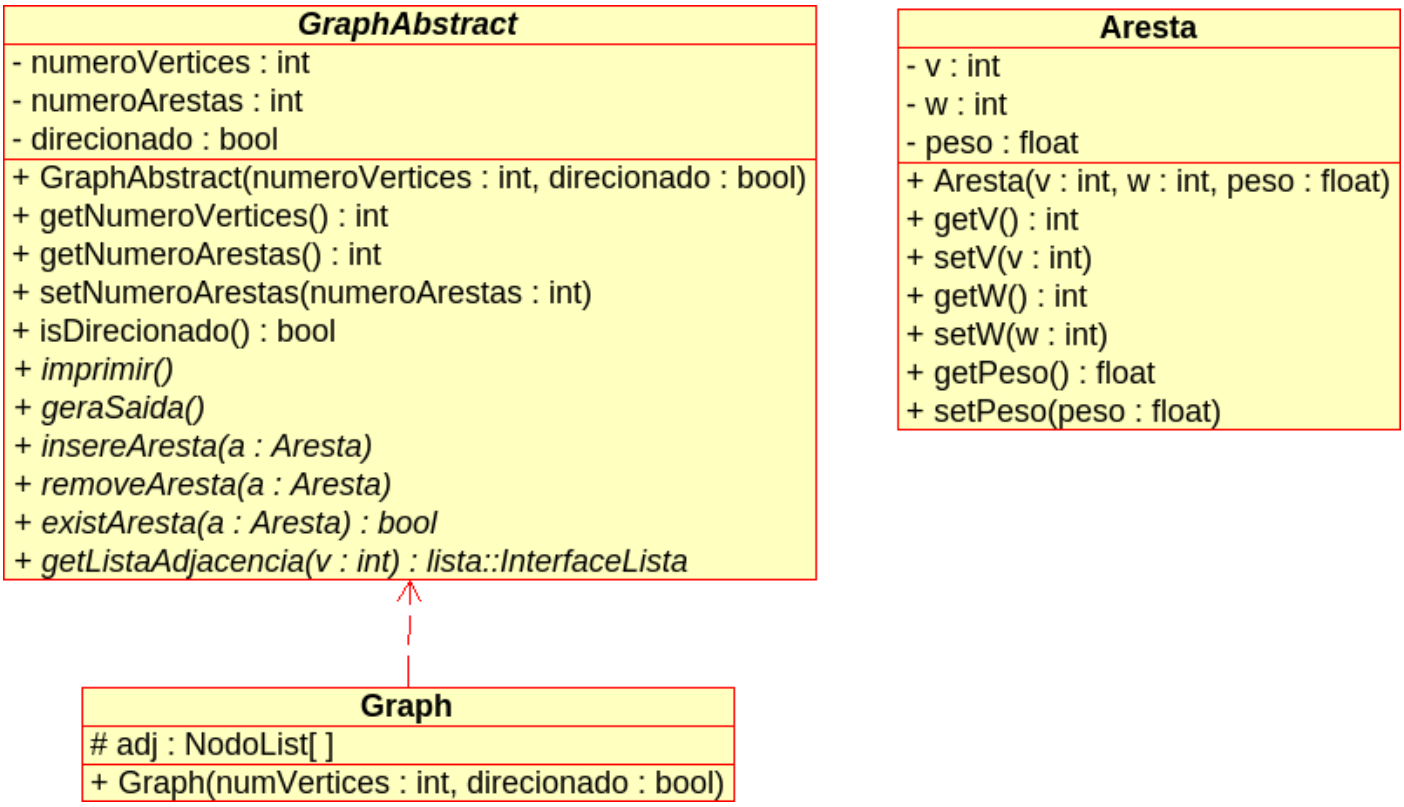
Conforme dito anteriormente, a Ferramenta de Apoio a Decisão é um instrumento de avaliação de métricas militares. Ela permite examinar uma dada situação em uma AOp, avaliando características e valores para fins de comparação. Cabe ao Oficial-Aluno decidir os critérios e então escolher o melhor caminho para os seus grupos de desembarque.

Note que a Ferramenta de Apoio a Decisão é uma aplicação que pode ser usada de várias formas. Por exemplo, poderíamos usá-la para treinar Oficiais-Alunos de modo a mostrar, de forma visual e quantitativa, se o planejamento deles não está coerente ou se está adequado e exeqüível. Ou ainda, poderíamos também utilizar a Ferramenta de
Apoio a Decisão à disposição dos Instrutores a fim de explorar a parte gráfica da aplicação e basear suas aulas em cima de simulações feitas sobre a AOp.

No Brasil, existe uma certa carência em relação a aplicações voltadas para o treinamento e ensino como a mostrada neste trabalho. Como o ensino e o treinamento prático exigem investimentos financeiros altos e consomem muito tempo com preparação e outros detalhes administrativos que, no geral, não contribuem diretamente com a formação do Oficial-Aluno. Com isso, um número cada vez maior de aplicações de avaliação, treinamento e simulação vêm sendo utilizadas, tanto para o ensino como para aprimoramento de doutrinas militares. A exemplo disso, essa ferramenta vem sendo testada e utilizada pelos Instrutores e Oficiais-Alunos no Curso de Aperfeiçoamento dos Oficiais (CAO) do Centro de Instrução Almirante Sylvio de Camargo (CIASC).

Descrição da Aplicação

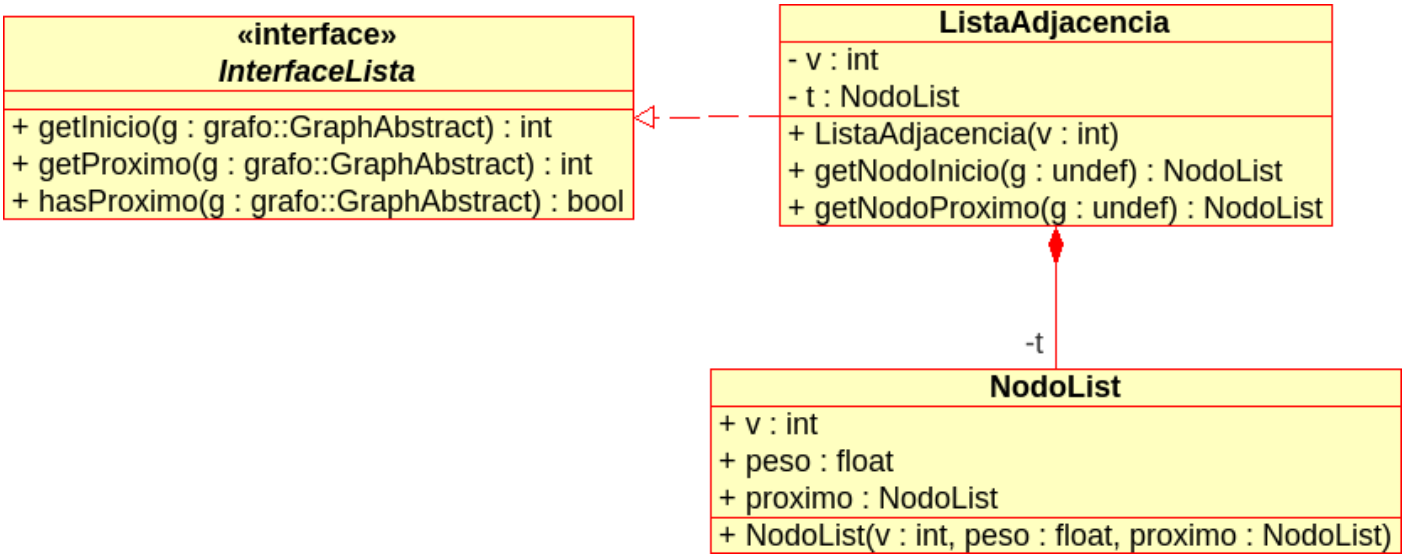
Para se realizar o estudo do algoritmo de Bellman-Ford, foi implementada a estrutura de dados grafo, utilizando lista de adjacência. Abaixo está o diagrama das classes que compõem esta estrutura:



Observando a estrutura de um grafo que utiliza matriz de adjacência e um que utiliza lista, pode-se encontrar atributos e métodos que são implementados da mesma forma. Devido a isso foi implementada uma classe abstrata chamada GraphAbstract, que contém estes atributos e métodos comuns às duas formas de implementação de uma grafo. Além disso há métodos que apesar de comuns as duas representações possuem implementações diferentes, portando estes métodos devem ser escritos pelas classes especificas.

Então, como foi implementado um grafo por lista de adjacencia, criou-se a classe Graph, que herda GraphAbstract e implementa seus métodos abstratos para usar lista de adjacencia. Graph utiliza a classe Aresta, que modela a aresta de grafo ponderado.

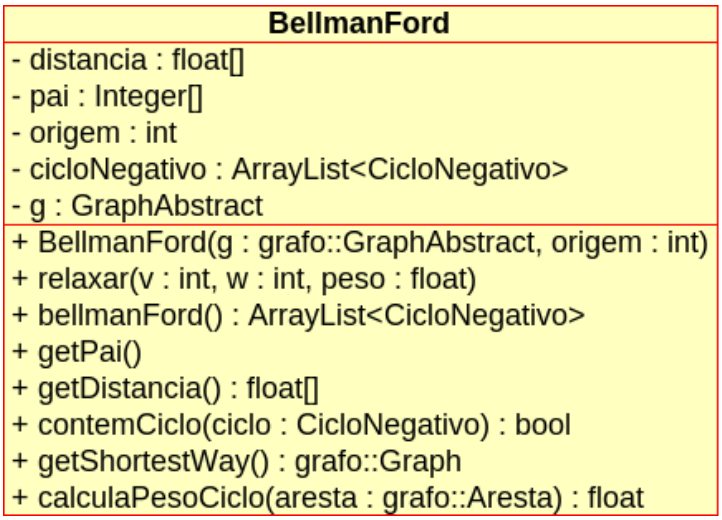
A lista que modela a lista de adjacencia para cada vértice do grafo foi modelada e implementada da seguinte maneira:

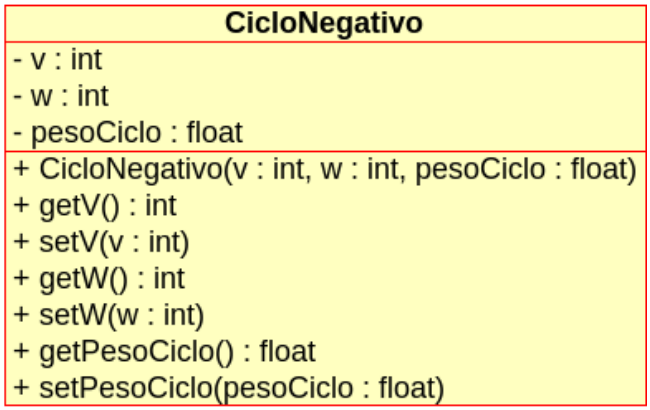


A implementação de uma lista de adjacencia para um grafo que utiliza matriz é diferente da implementação de um grafo q utiliza lista. Assim sendo, foi criada a interface, InterfaceLista, com o intúito de obrigar que as ambas as implementações mantenham os mesmos padrões dos métodos.

A classe lista Adjacência, como já dito, implementa a interface InterfaceLista e seus métodos abstratos. Ela precisa de uma classe que modela cada nó da lista, onde esta classe é a NodoList, que possui seus atributos como públicos, para facilitar o uso dentro do código.

Agora que definiu como a estrutura de dados grafo foi implementada, será mostrado como o algoritmo de Bellman-Ford foi construido:





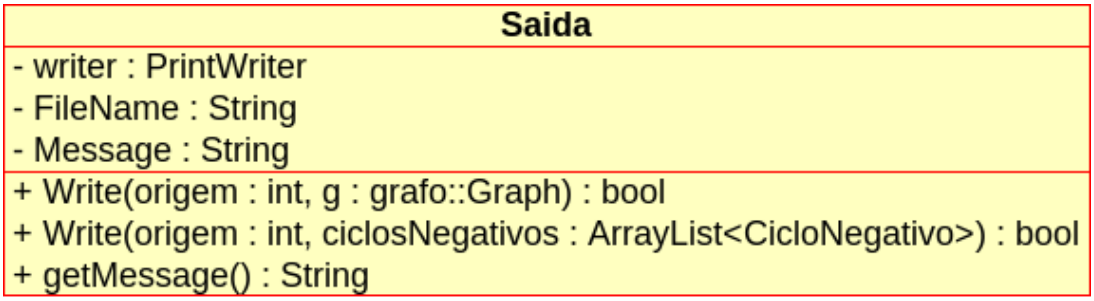
A classe CicloNegativo modela o ciclo negativo em uma grafo. Ela é utilizada pelo algoritmo quando algum ciclo negativo é achado no grafo.

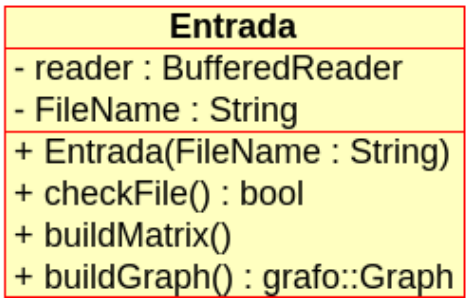
O algoritmo é implementado na classe Bellman-Ford, que possui as estruturas que guardam os predecessores e as distâncias, após rodar o algoritmo. A rotina de inicialização foi construida no contrutor da classe, assim, quando é instanciada a classe BellmanFord e passado o grafo por parâmetro, ocorre a inicialização dos vetores de predecessores e distâncias.

Quando desejar executar o algoritmo deve utilizar o método bellmanFord(), que possui uma chamada ao método relaxar, executando assim a primeira parte dos passos. Após esta chamada, o método verifica se há ciclos negativos; havendo ele retorna uma ArrayList contendo os objetos clico negativo, caso contrário nada é retornado.

Para obter o caminho mais curto é utilizado o método getShortestWay() que retornará um grafo, que é um subgrafo do passado no construtor. Os outros métodos são acessores para os atributos ou utilizados internamente na classe.

A entrada e saida da aplicação é realizada pelas seguintes classes:





Entrada é a classe que gera um grafo ou uma matriz de adjacência a patir de um arquivo texto que possui a seguinte forma:

4
1 0 34 5 10000
2 12 0 21 10000
3 10000 10000 10000 12
4 5 45 3 0

A primeira linha representa o número de vértices, e as linhas seguinte estão na estrutura de matriz de adjacencia.

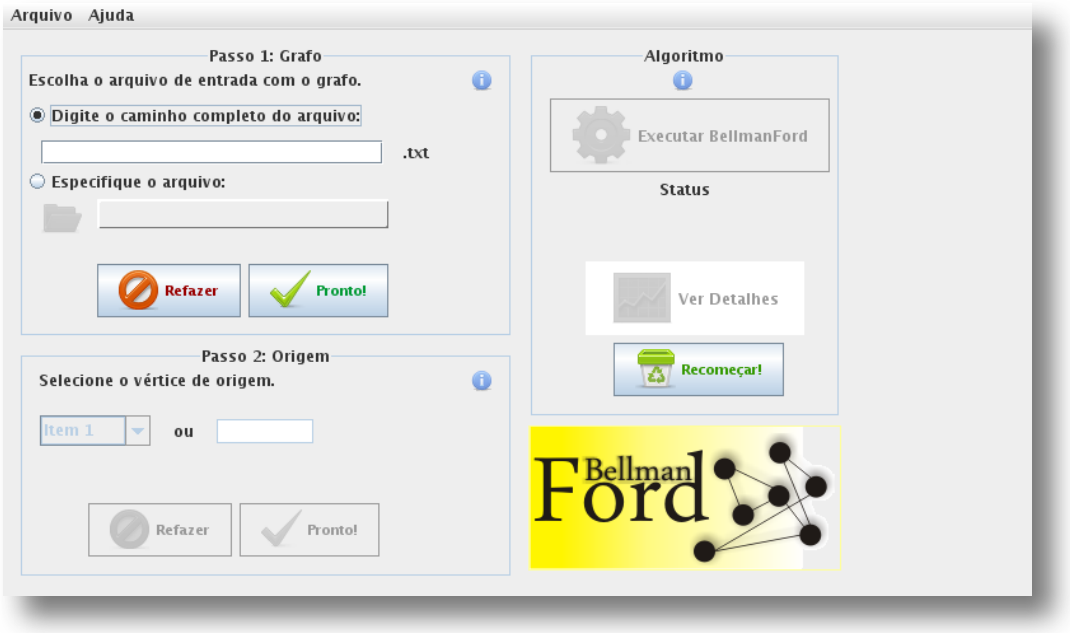
Após rodar a aplicação, esta gerará um arquivo de saída, em que para isso, utiliza a classe Saida. Esta tem dois métodos, onde um é passado um grafo, caso a aplicação não ache um ou mais ciclos negativos no grafo analisado. Se pelo menos um ciclo negativo for encontrado, é utilizado então o segundo método, que imprime o ciclo negativo. Ambas as saidas são demonstradas a baixo:

Sem ciclos negativos	Com ciclos negativos
Origem: 3 1: 2: (Vetice: 5 Peso: 7.0) 3: (Vetice: 2 Peso: 6.0) 4: (Vetice: 3 Peso: 6.0) 5: (Vetice: 1 Peso: -4.0)	Origem: 6 Ciclo de peso negativo: <5 , 6> <Peso: -3.0> Ciclo de peso negativo: <6 , 7> <Peso: -1.0>

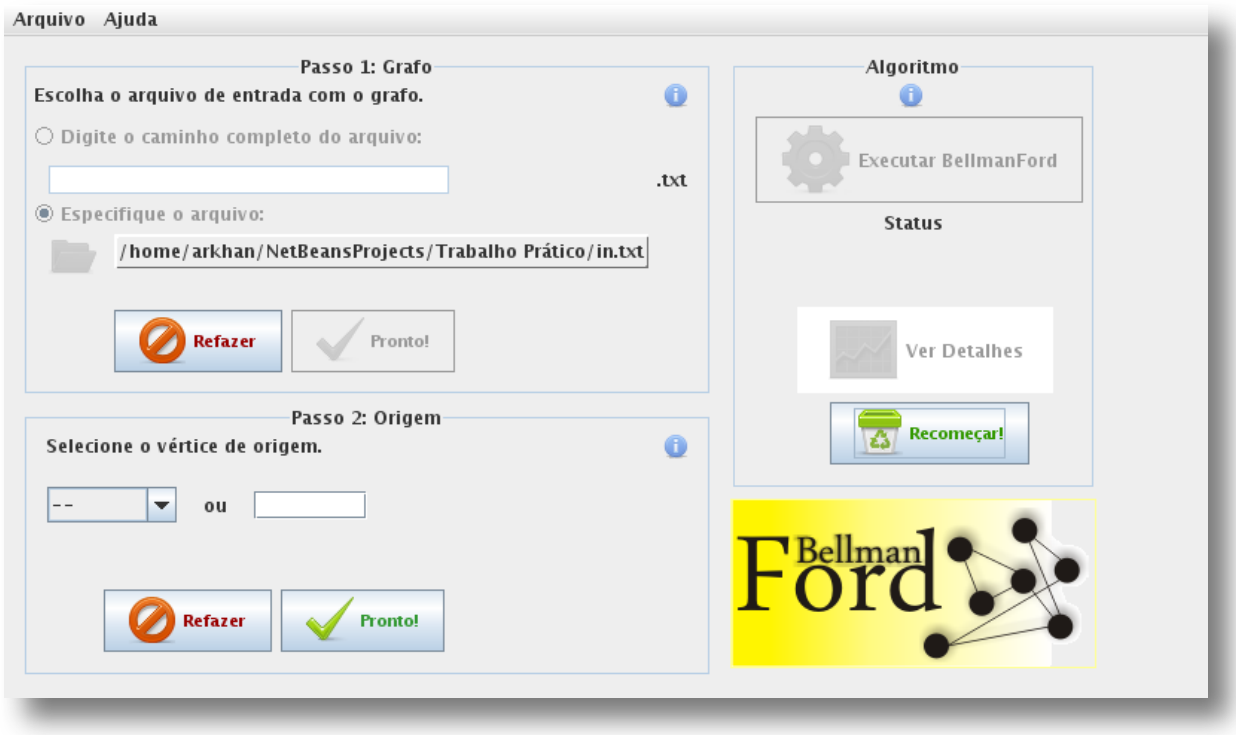
Estas são as principais classes utilizadas pela aplicação, agora será apresentada a interface de utilização, que é bem simples e intuitiva.

Composta por passos, a primeira coisa a se fazer é selecionar o arquivo que contém o grafo. É possível entrar com o endereço do arquivo ou selecioná-lo, para definir qual maneira, basta marcar as opções: “Digite o caminho completo do arquivo” ou “Especifique o arquivo”.

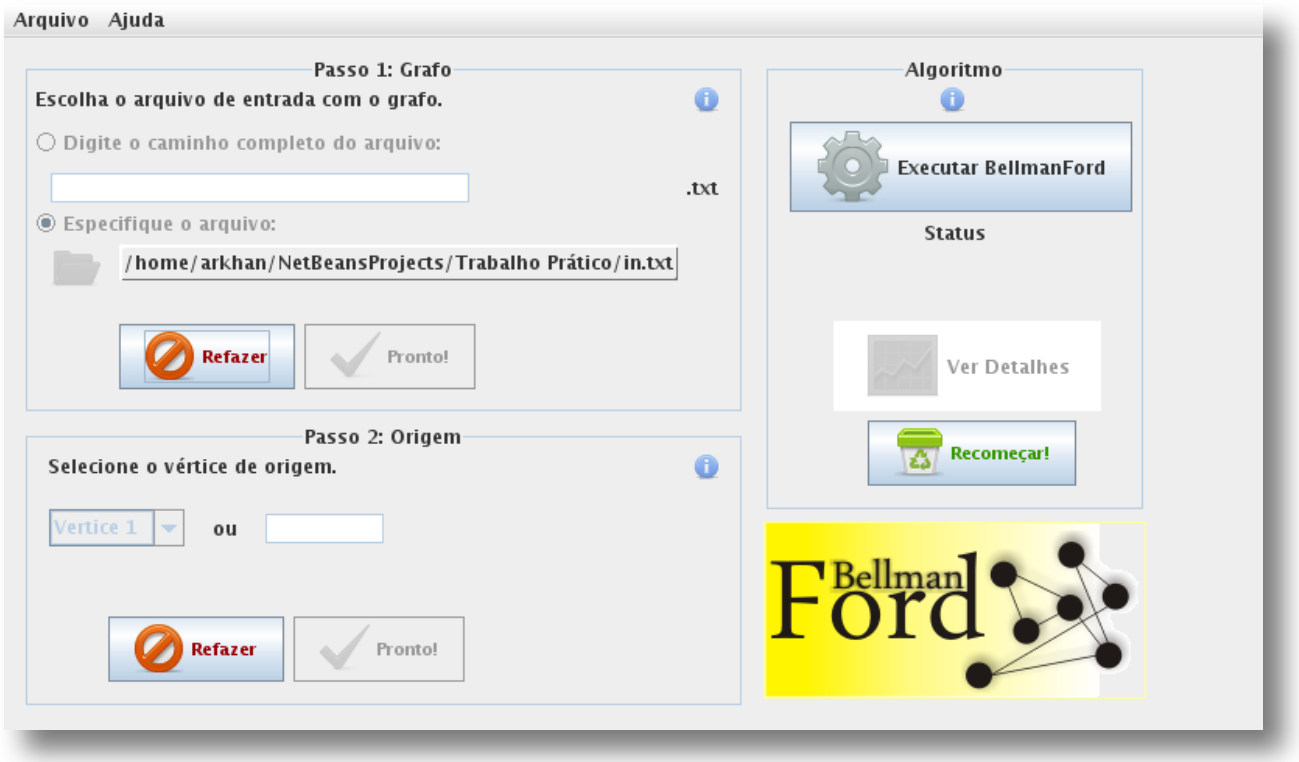
Selecionado o arquivo clique me pronto, caso queira selecionar outro arquivo clique em Refazer. O botão azul com um i dentro, traz informações sobre o arquivo de entrada.



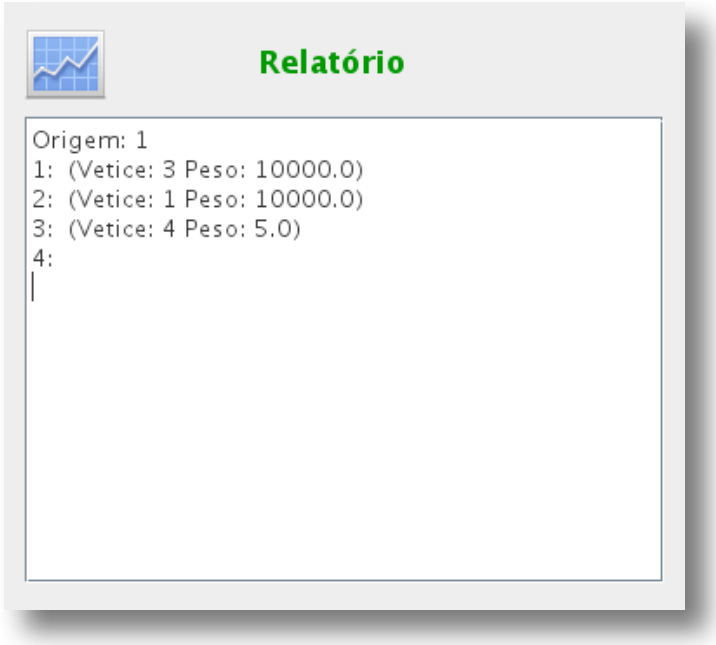
Realizado o passo 1, passa-se então ao 2, onde é selecionado o vértice de origem. Da mesma forma, o botão azul com um i traz mais informações, mas desta vez a respeito da escolha da origem.



O terceiro passo é a execução do algoritmo. Clicando em Executar BellmanFord a aplicação roda o algoritmo e gera a saida, em Status aparecem informações sobre o resultado, mostrando se há ou não ciclos negativos.



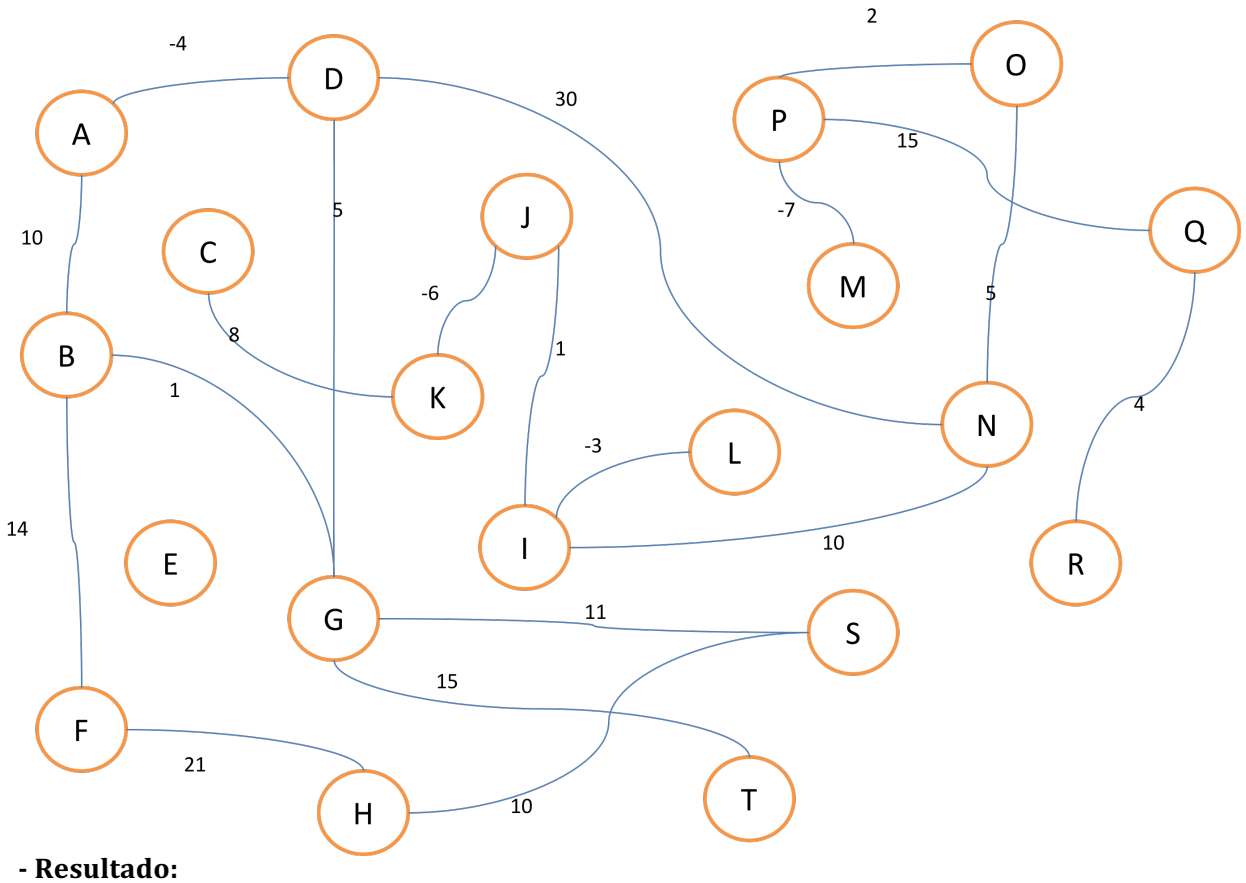
Para recuperar o resultado da execução, clique em Ver Detalhes, onde aparecerá, no mesmo formato do arquivo de saida, o resultado da análise realizada pelo algoritmo



Relatório de Resultados

Grafo 01:

- **Matriz de Ajdacência:** arquivo texto “graph01.txt”.
- **Características:** possui vários ciclos negativos; grafo não direcionado.
- **Representação Gráfica:**

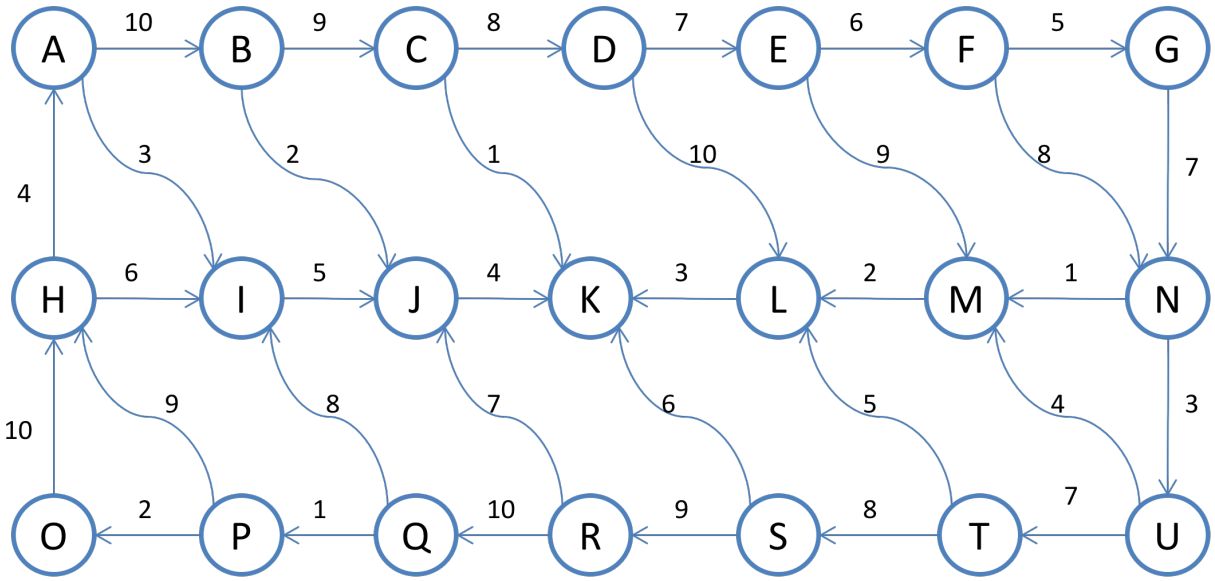


- **Resultado:**

```
Origem: 6
Ciclo de peso negativo: <0 , 3> <Peso: -8.0>
Ciclo de peso negativo: <1 , 5> <Peso: 28.0>
Ciclo de peso negativo: <3 , 6> <Peso: 10.0>
Ciclo de peso negativo: <8 , 11> <Peso: -6.0>
Ciclo de peso negativo: <9 , 10> <Peso: -12.0>
Ciclo de peso negativo: <9 , 8> <Peso: 2.0>
Ciclo de peso negativo: <12 , 15> <Peso: -14.0>
Ciclo de peso negativo: <12 , 4> <Peso: 20000.0>
Ciclo de peso negativo: <13 , 3> <Peso: 60.0>
Ciclo de peso negativo: <14 , 13> <Peso: 10.0>
```

Grafo 02:

- **Matriz de Ajdacência:** arquivo texto "graph02.txt".
- **Características:** não possui ciclos negativos; grafo direcionado
- **Representação Gráfica:**

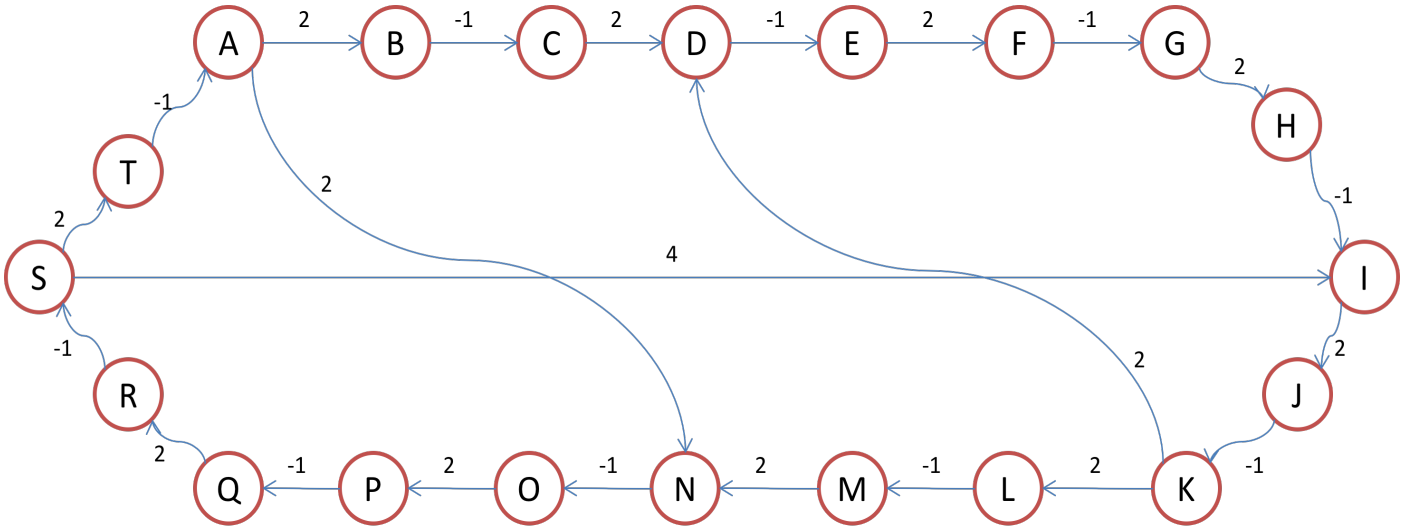


- **Resultados:**

```
Origem: 1
2:  (Vertice: 1 Peso: 10.0)
3:  (Vertice: 2 Peso: 19.0)
4:  (Vertice: 3 Peso: 27.0)
5:  (Vertice: 4 Peso: 34.0)
6:  (Vertice: 5 Peso: 40.0)
7:  (Vertice: 6 Peso: 45.0)
8:  (Vertice: 16 Peso: 95.0)
9:  (Vertice: 1 Peso: 3.0)
10:  (Vertice: 9 Peso: 8.0)
11:  (Vertice: 10 Peso: 12.0)
12:  (Vertice: 4 Peso: 37.0)
13:  (Vertice: 5 Peso: 43.0)
14:  (Vertice: 6 Peso: 48.0)
15:  (Vertice: 16 Peso: 88.0)
16:  (Vertice: 17 Peso: 86.0)
17:  (Vertice: 18 Peso: 85.0)
18:  (Vertice: 19 Peso: 75.0)
19:  (Vertice: 20 Peso: 66.0)
20:  (Vertice: 21 Peso: 58.0)
21:  (Vertice: 14 Peso: 51.0)
```

Grafo 03:

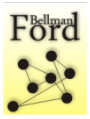
- **Matriz de Ajdacência:** arquivo texto "graph03.txt".
- **Características:** não possui ciclos negativos; grafo direcionado.
- **Representação Gráfica:**



- **Resultados:**

Origem: 1
2: (Vertice: 1 Peso: 2.0)
3: (Vertice: 2 Peso: 1.0)
4: (Vertice: 3 Peso: 3.0)
5: (Vertice: 4 Peso: 2.0)
6: (Vertice: 5 Peso: 4.0)
7: (Vertice: 6 Peso: 3.0)
8: (Vertice: 7 Peso: 5.0)
9: (Vertice: 8 Peso: 4.0)
10: (Vertice: 9 Peso: 6.0)
11: (Vertice: 10 Peso: 5.0)
12: (Vertice: 11 Peso: 7.0)
13: (Vertice: 12 Peso: 6.0)
14: (Vertice: 1 Peso: 2.0)
15: (Vertice: 14 Peso: 1.0)
16: (Vertice: 15 Peso: 3.0)
17: (Vertice: 16 Peso: 2.0)
18: (Vertice: 17 Peso: 4.0)
19: (Vertice: 18 Peso: 3.0)
20: (Vertice: 19 Peso: 5.0)

Bibliografia



- Maiedanis,J. UNICAMP 2003

- http://pt.tech-faq.com/low-earth-orbit.shtml