Exercício programa 1 - MAC0315 1º Sem. 2014

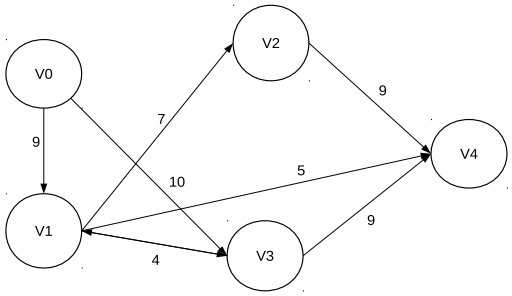
Data de entrega: **27/06/14** pelo PACA ou pelo email montanhe@usp.br

Instruções

- O exercício-programa deverá ser feito individualmente.
- Haverá controle de cópias (totais ou parciais) e caso estas sejam detectadas os envolvidos terão nota zero no programa.
- O EP deve ser feito em linguagem C, C++ ou python.
- Você pode utilizar qualquer compilador e sistema, mas garanta ao final que ele compila gcc 4.6.4. Se estiver com dúvidas sobre isso, fale com o monitor.
- EPs atrasados não serão aceitos. Não deixe para entregar muito em cima da hora.
- Você deverá entregar apenas os arquivos com o código fonte do seu programa, não é necessário entregar resultados de testes ou relatórios.
- Serão levados em conta na correção organização e comentários, portanto você deve usar funções no seu programa.
- Cuidado com divisões por zero.
- EPs que não compilarem serão zerados.
- Boa sorte e divirta-se!

Problemas de transporte

Um problema frequente no planejamento logístico de empresas é determinar a melhor rota para entrega de seus produtos. Neste exercício programa você deve implementar o algoritmo simplex para redes e utiliza-lo na solução de problemas de custo mínimo. Como você verá, o simplex para redes é uma especialização do simplex revisado e mantém muitas características deste algoritmo. Para entender o contexto onde a aplicação do simplex em redes é interessante considere a figura a seguir Cada V representa um terminal e o valor nas arestas corresponde ao custo em reais para escoar uma unidade de produto. Note que as arestas são orientadas, isto é, é possível ir de $V0$ para $V1$ mas não é possível ir de $V1$ para $V0$. Em nosso exemplo assim e ao longo do exercício as arestas do grafo são ilimitadas(ou seja, podem transportar a quantidade que quisermos de produtos). Suponha agora que desejamos levar 10 unidades de produtos do terminal $V0$ para $V4$. Como deve ser feito o escoamento? O simplex em redes nos ajuda a responder este tipo de pergunta. Uma formulação típica para este cenário é



$$
\begin{aligned}
\min z \quad &= \quad 9x_{01} + 10x_{03} + \ldots + 9x_{34} \\
s.a & \\
& \qquad x_{01} + x_{03} = 10 \\
& \qquad -x_{01} + x_{12} + x_{13} + x_{14} - x_{31} = 0 \\
& \qquad x_{24} - x_{12} = 0 \\
& \qquad -x_{03} - x_{13} + x_{31} + x_{34} = 0 \\
& \qquad -x_{14} - x_{24} - x_{34} = -10 \\
& \qquad x_{ij} \geq 0
\end{aligned}
$$

onde x_{ij} é a quantidade de produtos levados do terminal i para j. Para chegar a esta formulação precisamos assumir algumas hipóteses que consideramos válidas ao longo do exercício:

- O terminal 0 dispõe da quantidade de produtos que o terminal 4 demanda.
- Os terminais intermediários não demandam unidades do produto
- O terminal 4 é capaz de absorver toda sua demanda.

Uma forma interessante de representar as restrições nos terminais é usando matrizes. Neste caso

$$
A = \begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\
0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & -1 & -1 & 0 & -1
\end{pmatrix}, b = \begin{pmatrix}
10 \\ 0 \\ 0 \\ 0 \\ -10
\end{pmatrix},
$$

$$
\mathbf{c} = \begin{pmatrix} 9 & 10 & 7 & 4 & 5 & 9 & 4 & 9 \end{pmatrix}
$$

e o problema se escreve

$$\begin{aligned} \min z &= c^T x \\ s.a & \\ & Ax = b \\ & x \geq 0 \end{aligned}$$

1 Sua tarefa

Neste exercício programa você deve implementar o simplex em redes para resolver problemas da forma 1. Sua implementação deve ser dividida em pelo menos três módulos. Leitura de dados, obteção de solução inicial e simplex para redes.

1.1 Entrada de dados

Na leitura de dados seu programa deve ser capaz de receber um arquivo chamado *problema.dat* que tem a seguinte estrutura

```
# nós
nó origem
nó destino
# produto escoado
vértice_origem vértice_destino custo_aresta
.
.
.
vértice_origem vértice_destino custo_aresta
```

Desta forma o arquivo de entrada para o problema da seção anterior é

```
5
0
4
10
0 1 9
0 3 10
1 2 7
1 3 4
1 4 5
2 4 9
3 1 4
3 4 9
```

Esta estrutura diz respeito apenas à entrada dos dados e não como eles serão armazenados em seu programa para os cálculos. Você deve escolher uma estrutura adequada. Você pode usar uma matriz para armazenar os dados mas isso fará com que você carregue muitos dados inúteis(por exemplo um número considerável de zeros). Pense um pouco sobre como armazenar o grafo, caso tenha dúvida fale com o monitor.

1.2 Inicialização

Você deve implementar o algoritmo de inicialização descrito na seção Termination and Initialization do capítulo 19 do livro texto(Chvátal). Para isso você deve ser capaz de gerar vértices e arestas artificiais em seu grafo além de ter um algoritmo que encontra árvores. Ao implementar o algoritmo descrito no livro você será capaz de decidir se um problema de transporte é viável ou não. Seu módulo de inicialização deve ser capaz de reconhecer o caso em que o problema é inviável e terminar o programa sem iniciar o simplex.

1.3 Simplex para redes

O simplex para redes é mais simples do que o simplex revisado. Por exemplo, ele não requer a solução de sistemas lineares a cada iteração como o caso geral ou ainda o cálculo de custos reduzidos pode ser feito sem o uso de matrizes. Você encontra um esboço dos passos do simplex para redes na seção 7.3 do livro Introduction to linear optimization do professor Dimitris Bertsimas(Há algumas cópias na biblioteca). O livro texto também descreve em detalhes o simplex para redes no capítulo 19. Você deve implementar uma função que recebe além da estrutura do problema(custos, origem, destino e estrutura de restrições) uma árvore inicial para iniciar o algoritmo. Esta árvore inicial deve ser obtida com o módulo de inicialização. Ao final seu programa deve devolver um vetor com a solução para o problema além de indicar o custo mínimo para transferência. Você pode optar por mostrar a solução na tela criar um arquivo de saida com as informações.

2 Testes e Critério de avaliação

Como teste inicial do seu programa resolva o problema descrito primeira seção. Você pode gerar outras instâncias simples e comparar seus resultados com aqueles obtidos pela função glpk do octave por exemplo. O tempo de execução não contará pontos nesse EP e os pontos mais importantes na avaliação serão

- Divisão do seu programa em módulos.
- Implementação correta da inicialização.
- Implementação correta do simplex para redes.

Para testar seu código o monitor irá criar instâncias de diferentes tamanhos e algumas inviáveis. Em todos os casos vamos considerar arquivos de entrada como descritos na subseção entrada de dados.