DCC-IME-USP

Routo Terada
Sala 208-C Tel.: 3091 5918
e-mail rt@ime.usp.br

MAC122 Princípios de Desenvolvimento de Algoritmos - BM,BMA

Segundo Semestre de 2011

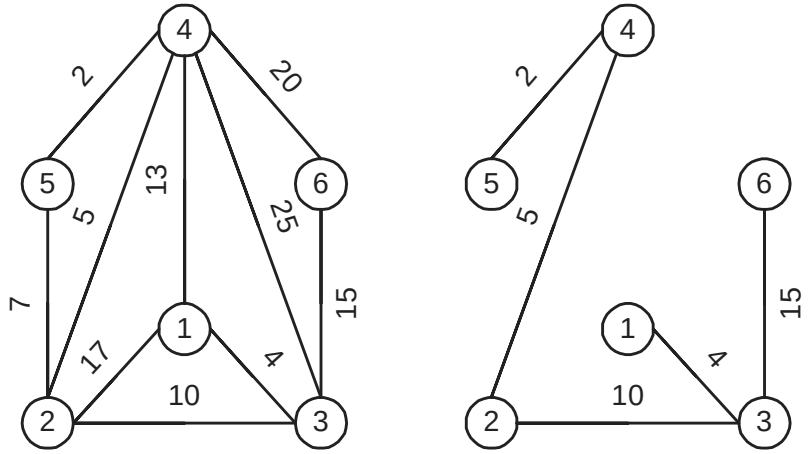
Exercício-Programa 2, Peso 1 Data de entrega: veja no paca.ime.usp.br.

Observações

- Este exercício é para ser feito *individualmente*.
- Entregue no sistema PACA UM ÚNICO arquivo contendo os arquivos seguintes, eventualmente comprimidos:
  - um arquivo chamado LEIA.ME (em formato .txt) com:
    - seu nome completo, e número USP,
    - os nomes dos arquivos inclusos com uma breve descrição de cada arquivo,
    - uma descrição sucinta de *como usar* o programa executável, *necessariamente* na linha-de-comando, i.e., SEM interface gráfica,
    - qual computador (Intel, AMD, ou outro) e qual sistema operacional (LINUX, UNIX, iOS, ou outro) foi usado,
    - instruções de como compilar o(s) arquivo(s) fonte(s).
  - o arquivo MAKE, se for o caso,
  - os arquivos do programa-fonte necessariamente em *linguagem ANSI-C*, compatível com o compilador *GCC*.
  - o programa *compilado*, i.e., incluir o código executável (se não incluir, a nota será zero!). Programa com algum erro de compilação receberá nota zero!
  - se for o caso, alguns arquivos de entrada e saída usados nos testes: arquivos com os dados de *entrada* chamados ENT1, ENT2, etc., e arquivos com os dados de *saída* correspondentes, chamados SAI1, SAI2, etc.
- O seu programa *deve* ser compilado pelo compilador GCC. Você *deve compilá-lo* com as opções (*flags*) `-ansi` e `-pedantic` (veja URL para documentação correspondente no paca.ime.usp.br)
- Coloque *comentários* em seu programa explicando o que cada etapa do programa significa! Isso será levado em conta na sua nota.
- Faça uma *saída clara*! Isso será levado em conta na sua nota.
- Não deixe para a *última hora*. Planeje investir 70 porcento do tempo total de dedicação em escrever o seu programa todo e simular o programa SEM computador (eliminando erros de lógica) ANTES de digitar e compilar no computador. Isso economiza muito tempo e energia.
- A nota será *diminuida* de um ponto a cada dia "corrido" de atraso na entrega.

ÁRVORE ESPALHADA MÍNIMA

Seja $G = (VG, aG)$ um grafo não-orientado com um valor real (positivo ou não), chamado custo, associado a cada aresta de G. Na Figura (a) a seguir vê-se um exemplo de grafo nestas condições.



Figuras (a) e (b)

Uma árvore espalhada é uma árvore não-orientada que conecta *todos* os vértices em VG; mais formalmente, é uma árvore da forma $E = (VG, aE)$ onde $aE \subseteq aG$. O custo de uma árvore espalhada é simplesmente a soma dos custos das suas arestas. O problema desta seção é a determinação de uma árvore espalhada de custo mínimo para G, ou simplesmente, uma árvore espalhada mínima para G. Na Figura (b) vê-se uma árvore espalhada mínima para o grafo na Figura (a).

Aplicações de árvores espalhadas (mínimas) são frequentes, por exemplo, em redes de circuitos elétricos, de estradas, de telecomunicação, de distrubuição de água, etc.. Algumas aplicações se baseiam na propriedade de que, dado $G = (VG, aG)$, uma árvore espalhada de G é um subgrafo minimal G' de G tal que $VG' = VG$ e G' é conexo (minimal no sentido de que não há um subgrafo de G' com menor número de arestas). Se os vértices de G representam, por exemplo, n cidades, e as arestas representam linhas de comunicação ou transporte ligando duas cidades, então o número mínimo de linhas necessárias para conectar as n cidades é $n - 1$; e as árvores espalhadas de G representam todas as soluções viáveis. Associadas às linhas temos, em geral, certos custos que representam, por exemplo, custos de transporte ou outros custos. Deseja-se então um conjunto de linhas conectando todas as n cidades e com o custo total, ou comprimento, mínimo. Se os custos associados às linhas forem positivos, o conjunto de linhas desejado deve formar uma árvore (se não fosse uma árvore, ter-se-ia um circuito e a exclusão de qualquer linha deste circuito formaria um conjunto ainda conectando todas as cidades mas com custo menor, o que contrariaria a hipótese). É claro então que deseja-se determinar uma árvore espalhada de custo mínimo.

ALGORITMO de KRUSKAL e o MÉTODO GULOSO

O *médoto guloso* aplicado a este problema de obtenção de uma árvore espalhada mínima sugere que a árvore seja construída aresta por aresta. A próxima aresta a ser escolhida deve satisfazer algum critério de otimização. O critério mais simples é o de escolher uma aresta que acarrete um incremento mínimo da soma dos custos das arestas já escolhidas. O algoritmo de Kruskal escolhe as arestas do grafo em ordem não-decrescente dos custos. O conjunto A das arestas já selecionadas é uma *floresta de árvores* que poderá ser transformada em uma árvore quando novas arestas forem incluídas. Na Figura (c) apresentamos a escolha das arestas pelo algoritmo de Kruskal, quando o grafo considerado é o da Figura (a). Note-se que durante o processo de escolha das arestas em ordem não-decrescente de custo, algumas arestas são rejeitadas porque a sua inclusão em A acarretaria a formação de um circuito, e isso não é desejável pois A deve permanecer sendo uma floresta. Note-se também que a árvore espalhada mínima da Figura (b) foi construída sem necessidade de se examinar todas as arestas do grafo original (no exemplo, as arestas $(1,2), (3,4)$, e $(4,6)$ não foram examinadas).

Coleção C (conjs. c/ vértices na mesma árvore)	aresta menor (v,w) extraída	custo	condição na linha (7) $v \in R_i, w \in R_j, i \neq j$
$\underbrace{\{1\}}_{R_1}, \underbrace{\{2\}}_{R_2}, \underbrace{\{3\}}_{R_3}, \underbrace{\{4\}}_{R_4}, \underbrace{\{5\}}_{R_5}, \underbrace{\{6\}}_{R_6}$	$(4,5)$	2	verdadeira
$\underbrace{\{1\}}_{R_1}, \underbrace{\{2\}}_{R_2}, \underbrace{\{3\}}_{R_3}, \underbrace{\{4,5\}}_{R_4}, \underbrace{\{6\}}_{R_6}$	$(1,3)$	4	verdadeira
$\underbrace{\{1,3\}}_{R_1}, \underbrace{\{2\}}_{R_2}, \underbrace{\{4,5\}}_{R_4}, \underbrace{\{6\}}_{R_6}$	$(2,4)$	5	verdadeira
$\underbrace{\{1,3\}}_{R_1}, \underbrace{\{2,4,5\}}_{R_2}, \underbrace{\{6\}}_{R_6}$	$(2,5)$	7	falsa
inalterada	$(2,3)$	10	verdadeira
$\underbrace{\{1,2,3,4,5\}}_{R_1}, \underbrace{\{6\}}_{R_6}$	$(1,4)$	13	falsa
inalterada	$(3,6)$	15	verdadeira
$\underbrace{\{1,2,3,4,5,6\}}_{R_1}$			
Coleção C final, um único conj.			

Figura (c) Ordem das arestas escolhidas pelo algoritmo de Kruskal do grafo na Figura (a)

Na Figura (d) vê-se o algoritmo de Kruskal em pseudo-código. O algoritmo constrói a árvore espalhada mínima chamada A, que é inicialmente vazia, através da linha (1). À medida que as arestas vão sendo selecionadas em ordem não-decrescente de custo, na linha (6), elas vão sendo removidas da *fila de prioridade F*. Se for determinado que uma dada aresta deve pertencer à árvore espalhada mínima, ela é incluida em A na linha (11); desta forma, a cada iteração das linhas 1-13, A contém na realidade várias árvores disjuntas (uma *floresta*). A coleção C, inicialmente vazia na linha (1), contém conjuntos disjuntos de vértices R_i; cada conjunto R_i é constituído por vértices que são conectados por uma *mesma árvore* em A, a cada iteração das linhas 1-13. Portanto, cada aresta (v,w) na linha (6) tal que v e w não estão conectados por uma mesma árvore em A satisfaz a condição nas linhas 7-8, e (v,w) deve estar na árvore espalhada mínima final.

- **Algoritmo ArvEspMin (Kruskal)**: para calcular árvore espalhada mínima.
- entrada: (VG,aG), um grafo não-orientado conexo, com vértices VG, e arestas aG com custos reais associados;
- saída: (A), uma árvore espalhada mínima para o grafo (VG,aG).

```
1 Árvore A ← vazia; coleção C ← vazia;
2 "construa uma fila de prioridade F contendo
3         todas as arestas em aG, com os custos associados";
4 para cada vértice v em VG faça Incl(v,C);  (* cada vértice constitui um conj. R_i *)
5 enqto |C|> 1 faça (* enquanto houver mais que um conj. em C *)
6       (v,w) ← ExMin(F); (* extrair de F a aresta de custo mínimo *)
7       se v ∈ R_i e w ∈ R_j estão em conjuntos distintos
8                 chamados R_i e R_j, em C
9       então
10          R_i ← R_i ∪ R_j  (* união de dois conjs. disjuntos, resultando R_i *)
11          Incl((v,w),A) (* incluir (v,w) na floresta A *)
12      fim-se
13 fim-enqto;
14 devolva (A)
```

Figura (d) Algoritmo de Kruskal para o cálculo de uma árvore espalhada mínima.

CERTIFICAÇÃO

Vamos estabelecer a certificação do algoritmo ArvEspMin no teorema seguinte.

- TEOREMA - O algoritmo ArvEspMin determina corretamente uma árvore espalhada mínima para um grafo (VG,aG) conexo com custos reais associados às arestas.
- DEMONSTRAÇÃO

É fácil de se ver que a execução do algoritmo sempre termina.

Vamos provar que a resposta A é realmente uma árvore espalhada mínima.

Suponha por absurdo que A não seja uma árvore espalhada de custo mínimo.

Seja $a_1, a_2, a_3, \ldots a_{n-1}$ a sequência das arestas incluídas em A na linha (11) pelo algoritmo de Kruskal.

Estas arestas estão em ordem não-decrescente de custo.

Seja A_m uma outra árvore espalhada mínima, distinta de A, que contém as arestas $a_1, a_2, a_3, \ldots a_{i-1}$ para o *maior* valor possível do índice i. A_m pode conter outras arestas do grafo (VG,aG).

$$\underbrace{a_1, a_2, a_3, \ldots a_{i-1},}_{\in A_m} \quad a_i, \ldots, a_{n-1}$$

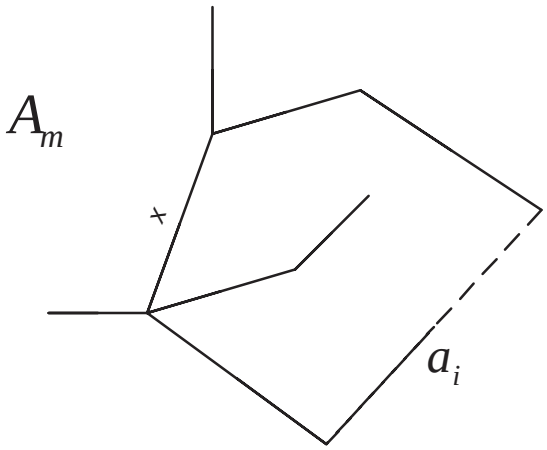


Figura (e)

Obviamente, $i - 1 \geq 1$ e como supomos que A não é mínima, $i < n$.

A inclusão de a_i em A_m faz formar um circuito em A_m (pois A_m é uma árvore espalhada) e este circuito deve conter uma aresta $x \neq a_j$ para todos os valores j tais que $1 \leq j \leq i$ (se não existisse tal x, então $a_1, a_2, a_3, \ldots a_{i-1}, a_i$ formariam um circuito em A_m). Veja a Figura (e).

Como x e $a_1, a_2, a_3, \ldots a_{i-1}$ pertencem a A_m, que não contém circuito, e como o algoritmo ArvEspMin inclui em A somente a aresta de custo mínimo que não resulte na formação de um circuito, podemos concluir que (custo de x) $\geq$ (custo de a_i) (senão, x seria incluído em A no lugar de a_i).

Se (custo de x) > (custo de a_i), então $A'_m = A_m - \{x\} \cup \{a_i\}$ é uma árvore espalhada de custo total menor do que o custo de A_m, o que seria uma contradição à hipótese de A_m ser mínima.

Portanto, deve-se ter (custo x) = (custo de a_i), o que faz de A'_m uma árvore espalhada mínima, e isso contradiz a hipótese de i ser o maior índice possível na sequência $a_1, a_2, a_3, \ldots a_{i-1}, a_i, \ldots a_{n-1}$.

Conclui-se então que A calculado pelo algoritmo é uma árvore espalhada de custo mínimo.

Q.E.D.

EP-Exercício-programa

Você deve implementar o Algoritmo de Kruskal. **Não** é necessário utilizar estruturas com ponteiros.

Um arquivo de entrada chamado ENTRA.txt com um grafo G, com no máximo $n = 100$ vértices e $O(n^2)$ arestas,.estará disponível para o seu programa LER do disco, no sistema paca.ime.usp.br. O seu programa deverá listar pelo menos as arestas escolhidas pelo algoritmo e os seus custos, mais ou menos como na Figura (c), à medida que o algoritmo é executado.

Cada linha do arquivo de entrada ENTRA.txt deve possuir uma sequência de três inteiros positivos separados por brancos, correspondentes a uma aresta com o seu custo associado.

1. O primeiro inteiro é um vértice da aresta,
2. o segundo inteiro é o outro vértice da aresta, e
3. o terceiro, é o custo associado a esta aresta.

Por exemplo, no grafo da Figura (a) as arestas $(4, 3)$ e $(5, 4)$ correspondem a uma linha do arquivo com os inteiros 4 3 25 5 4 2. Uma linha pode conter mais que uma ou duas arestas.