PUC-Rio
Departamento de Informática
Profs. Marcus Vinicius S. Poggi de Aragão
Período: 2008.2
Horário: 2as-feiras e 4as-feiras de 17-19 horas
18 de novembro de 2008

ESTRUTURAS DISCRETAS (INF 1631)

2^a Lista de Exercícios

Procure ser conciso e preciso nas suas argumentações.

GRAFOS

1. O que é um grafo ? Defina um grafo orientado. Defina um grafo não-orientado.
2. Representação de grafos.
   (a) Explique porque a lista de arestas de um grafo não define completamente o grafo.
   (b) Defina a **Matriz de Ajacência** de um grafo. Desenhe um grafo não-orientado de 5 vértices e apresente a matriz de adjacência correspondente. Faça o mesmo para um grafo orientado.
   (c) Defina **Lista de Adjacência** de um vértice. usando os grafos desenhados no item anterior.
   (d) O que é o grau de um vértice em um grafo. Exemplifique nos grafos do item b).
3. Desenhe todos os grafos não-orientados de 5 vértices onde todos os vértices tem grau menor ou igual a 2 (ou seja, têm duas ou menos arestas incidentes).
4. O que é um caminho em um grafo orientado ? Defina. E em um grafo não-orientado. Defina ciclos (ou circuitos) nos dois tipos de grafos.
5. Considere o problema de se encontrar todos os vértices que cuja distância do vértice 1 é exatamente K (arestas). Ou seja a distância do caminho mais curto, em número de arestas, do vértice até um tal vértice é K.

   Teorema 1 *(K): Sabe-se encontrar todos os vértices de $G = (V, E)$ que cuja distância do vértice 1 é exatamente K.*

   (a) Prove este teorema por indução matemática.
   (b) Descreva o algoritmo resultante dessa prova. Como este algoritmo é conhecido na literatura ?
   (c) Mostre como este algoritmo obtém todos os vértices que estão a distância 2 ($K = 2$) no grafo: $V = \{1, 2, 3, 4, 5, 6\}$, $E = \{(1, 2), (1, 3), (3, 5), (3, 6), (3, 4), (2, 4), (4, 5), (4, 6)\}$
6. Considere o grafo do item c) da questão 5.

(a) Defina um *ciclo Euleriano*.

(b) Este grafo possui *ciclo Euleriano* ? Por que ?

(c) No caso afirmativo, apresente um ciclo Euleriano.

7. O algoritmo conhecido como *Busca em Profundidade* tem que objetivo quando aplicado a um grafo ? Qual outro algoritmo também cumpre esse objetivo. Aplique o algoritmo de *Busca em Profundidade*, utilizando a ordem lexicográfica para desempates, no grafo do item c) da questão 5 começando pelo vértice 1.

8. Defina caminho-mais-curto entre um par de vértices em um grafo orientado.

9. Seja um grafo orientado $G = (V, A)$ e o problema de encontrar o caminho mais curto entre um vértice

   (a) Escreva o algoritmo de Ford-Bellman para encontrar o caminho-mais-curto de s a todos os demais em V. Observe que este algoritmo pode ser obtido através da prova por indução matemática do teorema abaixo.

   Teorema 2 *(K): Sabe-se obter a distância mínima de s a todos os vértices de V onde os caminhos podem ter no máximo K arcos.*

   (b) Escreva o algoritmo de Dijkstra para encontrar o caminho-mais-curto de s a todos os demais em V. Observe que este algoritmo pode ser obtido através da prova por indução matemática do teorema abaixo.

   Teorema 3 *(K): Sabe-se determinar o K-ésimo vértice mais próximo a s e sua respectiva distância mínima de s.*

   (c) Aplique os algoritmos acima no grafo $G = (V, A)$ abaixo fazendo $s = 1$. $V = \{1, 2, 3, 4, 5, 6\}$, $A = \{(1, 2), (1, 3), (3, 5), (3, 6), (3, 4), (2, 4), (4, 5), (4, 6)\}$, com distâncias $d_{1,2} = 1, d_{1,3} = 2$, $d_{3,5} = 4, d_{3,6} = 2, d_{3,4} = 2, d_{2,4} = 2, d_{4,5} = 2, d_{4,6} = 3$.

10. Defina uma **árvore** em um grafo. O que é uma **floresta** ? O que é uma **árvore geradora** de um grafo ? Que algoritmo pode ser utilizado para encontrar uma árvore geradora qualquer em um grafo. Aplique este algoritmo no grafo do item c) da questão 5. O que seria uma árvore geradora de peso mínimo ?

11. Seja um grafo conexo $G = (V, E)$, não-orientado, e w_e, para $e \in E$ os pesos, onde $V = \{1, 2, \ldots, n\}$ associados às arestas de G. Seja o problema de encontrar a árvore geradora de peso **máximo** de G.

    (a) Escreva o algoritmo pode ser obtido através da prova do teorema abaixo por indução matemática ? Apresente o algoritmo e explique.

    Teorema 4 *(K): Sabe-se encontrar uma árvore de $G = (V, E)$, que contém o vértice 1 ($1 \in V$), que possui K vértices e possui peso máximo.*

    (b) Escreva o algoritmo pode ser obtido através da prova do teorema abaixo por indução matemática ? Apresente o algoritmo e explique.

    Teorema 5 *(K): Sabe-se encontrar a floresta $F = (V, E^K)$ de peso máximo de $G = (V, E)$ que possui K arestas, isto é $|E^K| = K$.*

(c) Aplique os algoritmos acima no grafo $G = (V, E)$. $V = \{1, 2, 3, 4, 5, 6\}$, $E = \{(1, 2), (1, 3), (3, 5), (3, 6), (3, 4), (2, 4), (4, 5), (4, 6)\}$, com pesos $w_{1,2} = 1, w_{1,3} = 2$, $w_{3,5} = 4, w_{3,6} = 2, w_{3,4} = 2$, $w_{2,4} = 2, w_{4,5} = 2, w_{4,6} = 3$.

12. Prove por indução e dê o algoritmo recursivo que resulta da sua prova.

    Teorema: Todos os vértices de uma qualquer árvore podem ser coloridos (sem ter vértices adjacentes com uma mesma cor) com duas cores. Uma árvore é um grafo não orientado $G = (V, E)$ sem ciclos, conexo, e onde $|E| = |V| - 1$.

    Dicas: use indução simples nos vértices e observe que qualquer árvore pode ser construída a partir de um vértice sozinho, com a inserção sucessiva de um vértice conectado a uma aresta.

    Você deve escrever o algoritmo para colorir uma árvore qualquer com duas cores.

13. Seja $G = (V, E)$ um grafo **orientado** e acíclico, então existe uma renumeração dos seus vértices tal que todos os vértices que podem ser atingidos a partir de um vértice v, $v \in V$, estão renumerados com valores superiores a v.

    Prove por indução o teorema abaixo e dê o algoritmo recursivo que resulta da sua prova.

    Teorema: Todo grafo conexo e acíclico possui pelo menos um vértice com grau de entrada (número de arcos que chegam em um vértice) igual a ZERO.

    Dicas: use indução simples nos vértices. Utilize $n = 2$ como caso base e mostre que o teorema vale para esta caso por exaustão.

    Você deve escrever um algoritmo que encontre uma renumeração dos vértices que satisfaça a propriedade descrita no início da questão. A entrada do algoritmo é um grafo, i.e. um conjunto de vértices e um conjunto de arcos. A saída é a renumeração dos vértices, i.e. por exemplo: os vértices 1, 2, 3, 4 e 5 devem ser renumerados como 2, 4, 1, 5, 3.

14. Seja $G = (V, E)$ um grafo **não-orientado**. Deseja-se decidir se G é um grafo bi-partido ou não.

    Um grafo é bi-partido se existe uma partição de V, V_1 e V_2 ($V_1 \bigcap V_2 = \emptyset$, $V_1 \bigcup V_2 = V$) tal que $\forall\ (u, v) \in E$, tem-se que $u \in V_1$ e $v \in V_2$ ou vice-versa. Isto é, não existe aresta entre vértices de V_1 ou V_2.

    Caso G seja bi-partido, deseja-se obter a partição V_1 e V_2 de V.

    Elabore o caso base e a hipótese indutiva para a prova, por indução, de que sabe-se encontrar um tal conjunto. Descreva o algoritmo resultante de sua prova.

15. Seja um grafo $G = (V, E)$, conexo e orientado, onde as distâncias dos arcos é dada por $l_e \geq 0$, $e \in E$. Seja também um vértice s, $s \in V$. Considere agora o problema de encontrar o caminho mais curto do vértice s aos demais vértices do grafo, $v \in V - \{s\}$. Seja o seguinte teorema:

    Teorema 6 *(K): Sabe-se determinar o K-ésimo vértice mais próximo do vértice s, v_k, e a sua respectiva distância de s até v_k, $d(v_k)$.*

    Observe que um vértice ser o K-ésimo mais próximo de um vértice s é uma condição definitiva e permanente.

    A prova do teorema acima por indução matemática simples leva ao algoritmo de Dijkstra para encontrar o caminho mais curto entre um vértice *fonte* e os demais vértices de um grafo orientado $G = (V, E)$ onde a distância associada a um arco $e \in E$ é dada por $l_e = l_{vw}$ onde v e w são o vértice de partida e de chegada do arco e, respectivamente. (Lembre que $n = |V|$ e $m = |E|$).

Algoritmo Dijkstra (s - *fonte*)

Passo 0: *Inicialização*

Seja S o conjunto de vértices com caminho mais curto a partir de s determinado, e $\overline{S}$ seu complemento ($\overline{S} = V - S$).

$S \leftarrow \emptyset$

$d(i) \leftarrow +\infty \ \forall\, i \in V$;

$d(s) \leftarrow 0$; $pred(s) \leftarrow 0$; K = 1;

Passo 1: *Iteração*

Enquanto $K \leq |V|$ ($S \subset V$) faça

1.1 Encontre $v \in \overline{S}$ t.q. $d(v) = \min\limits_{w \in \overline{S}} d(w)$

1.2 $S \leftarrow S \bigcup \{v\}$; $\overline{S} \leftarrow \overline{S} \setminus \{v\}$;

1.3 Para todo $w \in \Gamma^+(v)$
Se $d(v) + l_{vw} < d(w)$
então $d(w) \leftarrow d(v) + l_{vw}$; $pred(w) \leftarrow v$;

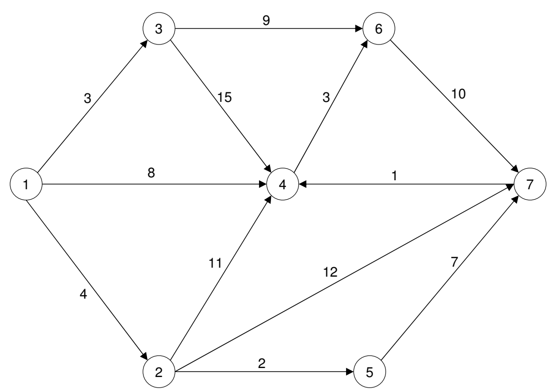
1.4 K = K + 1;

(a) Faça a relação entre a prova do teorema por indução matemática SIMPLES (caso base e passo indutivo) e o algoritmo acima.

(b) Argumente que o algoritmo acima determina $d(v_k)$ para todo v_k, para $K = 1, \ldots, n = |V|$.

(c) Aplique o algoritmo acima no grafo abaixo onde o vértice s é o vértice 1 e preencha tabela.

(d) Modifique os pesos dos arcos, que agora podem ser negativos, de forma que o algoritmo de Dijkstra não encontre corretamente os caminhos mais curtos.



16. Seja um grafo conexo $G = (V, E)$, **orientado**, e $d_{(u,v)}$, para $(u, v) \in E$ as distâncias associadas aos arcos de G. Seja o problema de encontrar o caminho mais curto do vértice 1 a todos os demais. Considere o algoritmo abaixo:

K	v_k	$d(v_1)$	$d(v_2)$	$d(v_3)$	$d(v_4)$	$d(v_5)$	$d(v_6)$	$d(v_7)$
1								
2								
3								
4								
5								
6								
7								

Passo 1 *Inicialização: Seja $D^K(v)$ a distância do menor caminho do vértice 1 ao vértice v, para todo v em V, que utiliza no máximo K arcos. Faça $D^0(v) = +\infty$ para todo v em V tal que $v \neq 1$, e $D^0(1) = 0$. Faça também $K = 1$.*

Passo 2 *Iteração: Faça $D^K(v) = D^{K-1}(v)$ para todo v em V.*

Para todo arco (w, y) em E teste se $D^{K-1}(w) + d_{(w,y)} < D^K(y)$ e em caso afirmativo faça $D^K(y) = D^{K-1}(w) + d_{(w,y)}$.

Faça $K = K + 1$.

Passo 3 *Parada: Se $K = |V|$ PARE. A distância do caminho mais curto do vértice 1 ao vértice v é dada por $D^{|V|-1}(v)$ para todo v em V. SENÃO, repita o passo anterior.*

Condidere também o teorema que se segue:

Teorema 7 *(K): Sabe-se encontrar a distância total do caminho mais curto, com no MÁXIMO K arcos, do vértice 1 ao vértice v, para todo vértice v em V.*

(a) Execute o algoritmo acima no grafo da questão de 1 c), assumindo que os arcos permitem passar nos dois sentidos. Faça uma tabela para D onde cada linha corresponde a um valor de K e cada coluna a um vértice v. Preencha esta tabela.

	v_1	v_2	v_3	v_4	v_5	v_6
D^0						
D^1						
D^2						
D^3						
D^4						
D^5						

(b) Explique como a prova por indução matemática do teorema acima aparece no algoritmo acima.

(c) Por que o algoritmo só pára quando $K = |V| - 1$? Quando os caminhos mais curtos seriam obtidos para $K < |V| - 1$? Quando não ?

(d) Este algoritmo funcionaria se existissem arcos com distâncias negativas ? Seria sempre possível encontrar o caminho mais curto sob estas condições ? Por que ?