Lista de Ejercicios de Grafos

I. Parte I

University of KENT

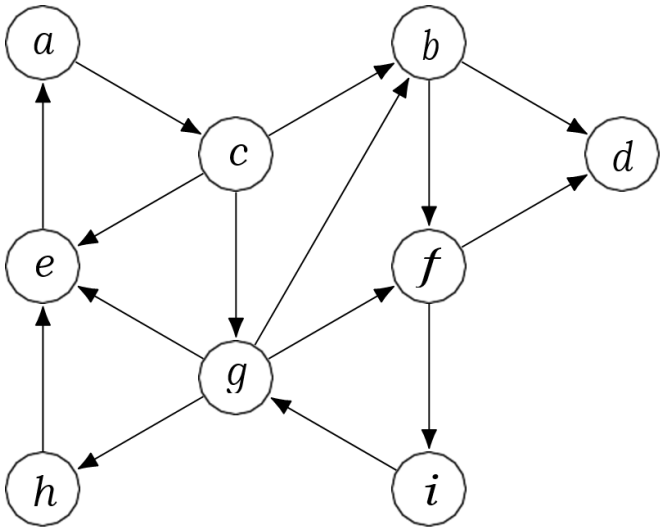
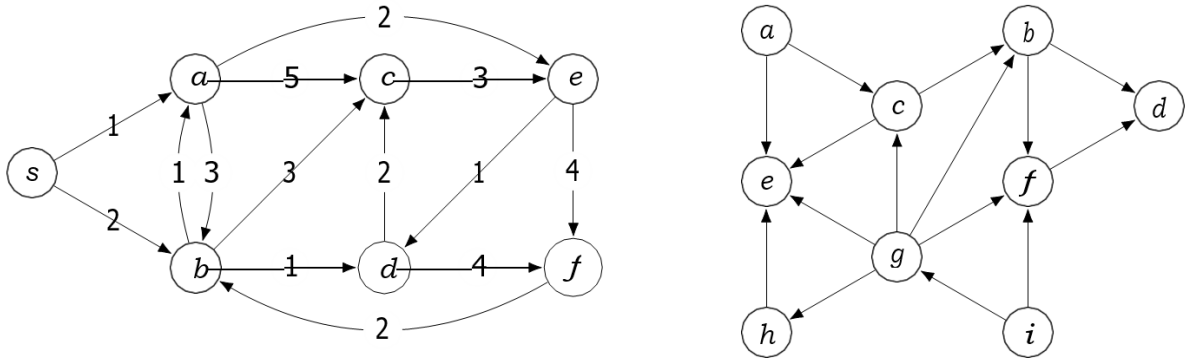


Fig. 1. An example graph.

1) For the graph given above in Figure 1, …
   a) …run a BFS starting in vertex a. Show the produced tree.
   b) …run a DFS starting in vertex a and identify tree, forward, back, and cross edges.

2) Consider a graph $G = (V, E)$ whose edges are marked red or blue. An alternating path is a path whose edges alternate in color (red, blue, red, …; or blue, red, blue …). The alternating distance from vertex u to vertex v is the length of the shortest alternating path from u to v. Determine in linear time the alternating distance from a source vertex s to all vertices.

3) In a given directed graph G, a vertex u is *reachable* from a vertex v if there is a directed path from v to u. Give an efficient algorithm to test whether G contains a vertex v such that every vertex u is reachable from v. State the time complexity of your algorithm.



(a) (b)

Fig. 1. Two example graphs.

4) For the graph given above in Figure 1(a), . . .
   a) . . . run a BFS starting in vertex s.
   b) . . . run a DFS starting in vertex s and identify tree, forward, back, and cross edges.
   c) . . . run Dijkstra's algorithm starting in vertex s.
   d) . . . create a minimum spanning tree (you can ignore the edge directions).

5) For the graph given above in Figure 1(b), run a topological sort.

6) Give a linear time algorithm to remove all the cycles in a directed graph $tt = (V, E)$. Removing a cycle means removing an edge of the cycle. If there are k cycles in tt, the algorithm should only remove at most $O(k)$ edges.

7) Consider a graph $tt = (V, E)$ whose edges are marked red or blue. An alternating path is a path whose edges alternate in colour (red, blue, red, . . . ; or blue, red, blue . . .). The alteranting distance from vertex u to vertex v is the length of the shortest alternating path from u to v. Determine in linear time the alternating distance from a source vertex s to all vertices.

8) You have given a weighted directed acyclic graph and two vertices s and t. Give an algorithm that finds the shortest path from s to t in linear time.

9) Suppose we are given the minimum spanning tree T of a given graph tt and a new edge $e = uv$ of weight w that we will add to tt. Give an $O(n)$ time algorithm to find the minimum spanning tree of the graph $tt + e$.

10) Let $tt = (V, E)$ be a weighted, directed graph with exactly one negative weight edge and no negative-weight cycles. Give an algorithm to find the shortest distance from s to all other vertices in V that has the same running time as Dijkstra.

11) In certain graph problems, vertices have can have weights instead of or in addition to the weights of edges.

Let $\omega(v)$ be the cost of vertex v, and $\omega(xy)$ the cost of the edge xy. This problem is concerned with finding the cheapest path between vertices a and b in a graph $tt = (V, E)$. The cost of a path is the sum of the costs of the edges and vertices encountered on the path.

a) Suppose that each edge in the graph has a weight of zero. Assume that $\omega(v) = 1$ for all vertices $v \in V$ (i. e., all vertices have the same cost). Give an efficient algorithm to find the cheapest path from a to b and its time complexity.
b) Now suppose that the vertex costs are not constant (but are all positive) and the edge costs remain as above. Give an efficient algorithm to find the cheapest path from a to b and its time complexity.
c) Now suppose that both the edge and vertex costs are not constant (but are all positive). Give an efficient algorithm to find the cheapest path from a to b and its time complexity.

12) We are given a weighted directed graph $tt = (V, E, \omega)$ with a source vertex $s \in V$, where $\omega: E \rightarrow R$ is the weight function. All weights are non-negative. We are also given a partition of the edges into "red" and "blue" edges. For all vertices $v \in V$, find the minimum cost of reaching v from s along a path that uses at most one red edge. Your algorithm should run in "Dijkstra time". Do not invent a new algorithm; apply an algorithm studied in class to an input other than *tt*. Your job is to construct the new input.

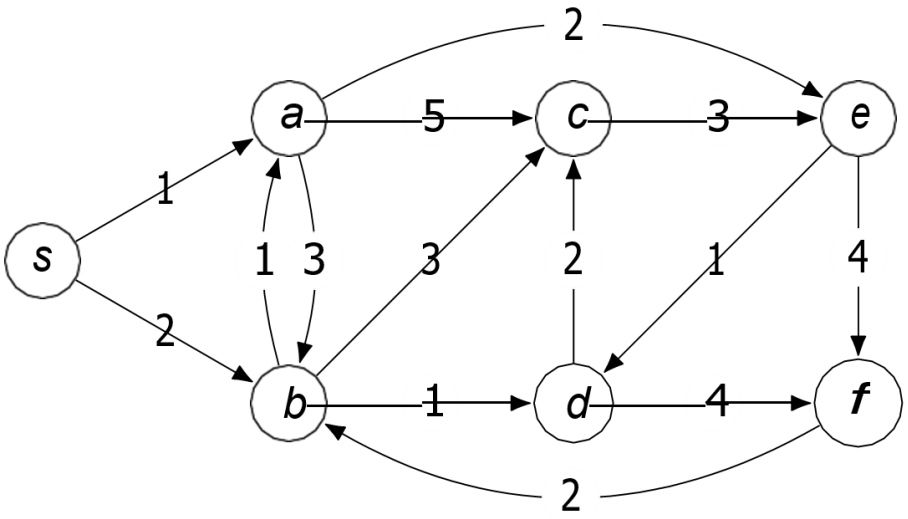


Fig. 1. An example graph.

13) For the graph given above in Figure 1, . . .

a) . . . run Dijkstra's algorithm starting in vertex *s*. Show the computed tree and distances.
b) . . . create a minimum spanning tree (you can ignore the edge directions).

14) You are given a weighted directed acyclic graph and two vertices *s* and *t*. Give an algorithm that finds the shortest path from *s* to *t* in linear time. Note that edge weight can be negative.

15) Let $tt = (V, E)$ be a weighted, directed graph with exactly one negative weight edge and no negative-weight cycles. Give an algorithm to find the shortest distance from *s* to all other vertices in *V* that has the same running time as Dijkstra's algorithm.

16) Sliding Puzzle Solver: Description. A sliding puzzle consists of an n n frame of numbered square tiles in

random order with one tile missing. The object of the puzzle is to place the tiles in order (see Figure 1) by making sliding moves that use the empty space.

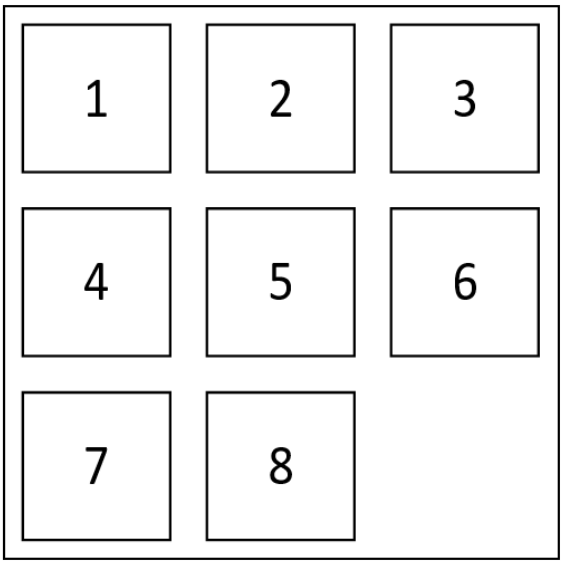


Fig. 1. A solved 3 × 3 sliding puzzle.

Write a simple program which solves these puzzles. Your program should be able to generate a random 3 3 puzzle and try to solve it. It should be able to find a solution using BFS and to find a solution using DFS. Print out the generated puzzle and (if it exists) the step by step solution to the standard output.

Note. Your program should be a simple terminal application. There will be no input. Please write your program in C/C++, Java, or C#. Keep the code simple, readable, and within one file. Do not use third party libraries. Submit the code file via email. Make sure the code contains your name at the beginning of the file.

17) Second Shortest Path: **Description.** In class, we discussed how to find the shortest path from a vertex to all other vertices using a BFS. In a given graph, there can be up to exponentially many paths. Describe an algorithm which finds the second shortest path in an unweighted graph, i. e., if you sort all paths by length, it is at position 2.

Note. In the case that there are multiple shortest paths, the path we are looking for has the same distance as the shortest path. It can differs by only one vertex or in all vertices except start and end vertex.

II. Parte II

IME-Universidad de Sao Paulo

1) De um algoritmo eficiente para: dado um grafo conexo *G* com *n* vertices, *m* arestas e pesos nao-negativos nas arestas, encontrar um conjunto de arestas cuja remoçao nao desconecta *G*, e que tenha peso maximo. Como sempre, o peso de um conjunto de arestas e a soma de seus pesos. Note que estamos falando de remoçao de arestas, sem remoçao de vertices.

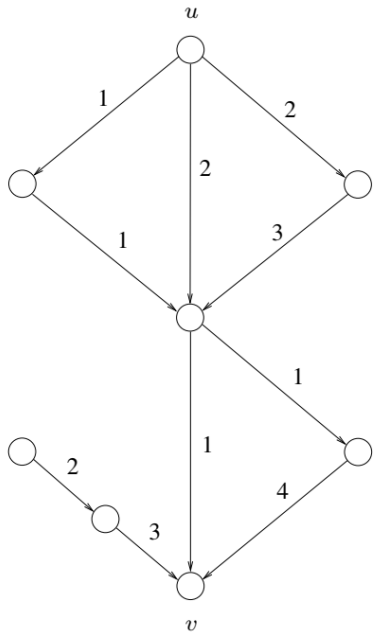
2) Seja $G = (V, A)$ um grafo orientado acıclico com pesos positivos nas arestas dados por $w : A \rightarrow R$.

O problema da contagem de caminhos e o seguinte: dados dois vértices u e v em V , encontrar o número de caminhos que conectam u a v.
O problema da contagem de caminhos mínimos e o seguinte: dados dois vértices u e v em V , encontrar o número de caminhos de comprimento mínimo que conectam u a v.

Por exemplo, no grafo abaixo existem 6 caminhos de u a v, dentre os quais somente 2 sao caminhos mínimos.

a. Descreva um algoritmo eficiente para o problema da *contagem de caminhos*. Annalise a sua complexidade.

b. Descreva um algoritmo eficiente para o problema da *contagem de caminhos mínimos*. Analise a sua complexidade.



III. Parte III

Universidad Federal de Uberlandia, Brasil

1) Defina los seguintes conceptos:

(a) Defina o que eum subgrafo.

(b) Defina o que eum grafo bipartido.

(c) Defina o que eum grafo conexo. E um desconexo?

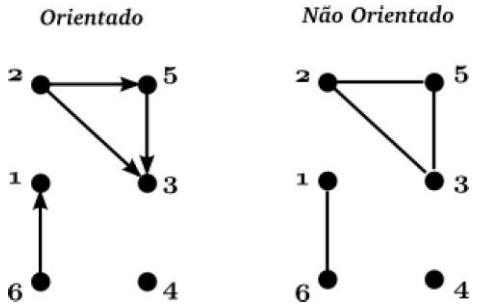
(d) O que sao grafos isomorfos? Desenhe um exemplo.

(e) Defina o que eum grafo Hamiltoniano.

(f) Defina o que eum grafo Euleriano.

2) Desenhe as versoes nao orientadas e orientadas do grafo $tt(V, E)$, onde $V = \{1, 2, 3, 4, 5, 6\}$ e $E = \{(2, 5), (6, 1), (5, 3), (2, 3)\}$.

3) Defina os grafos ilustrados abaixo



4) Defina e desenhe os grafos nao orientados completos com 4, 5 e 6 vertices.

5) De um exemplo de um grafo em que cada vertice e adjacente a dois outros vertices e cada aresta e adjacente a duas outras arestas

6) Quantas arestas tem um grafo com 3 vertices de grau 3 e um vetice de grau 5?

7) Em um grafo com n vertices e m arestas, qual a soma dos graus de todos os vetices? Observe que, em um grafo nao orientado, cada aresta soma 1 ao grau de cada vetice em que incide e cada aresta incide somente sobre dois vertices. Em um grafo orientado, por outro lado, cada aresta soma 1 ao grau de cada vertice em que incide, porem, cada aresta incide somente sobre um vetice.

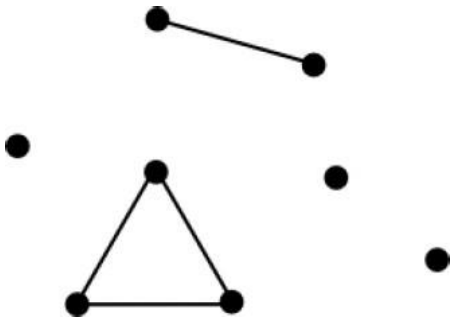
8) Sabendo que cada vetice tem pelo menos grau 3, qual o maior numero possıvel de vetices em um grafo com 35 arestas? Lembre-se que a soma dos graus dos vetices e igual a duas vezes o numero de arestas. Se cada aresta liga dois vetices terıamos 70 vetices de grau 1.

9) Quantas arestas possui um grafo completo com n vetices? E um grafo orientado com- pleto com n vertices?

10) Faça uma funcao para obter todos os nos adjacentes (vizinhos) a um nodo grafo, dado que o grafo e representado por uma **matriz de adjacencias**.

11) Faça uma funcao para obter todos os nos adjacentes (vizinhos) a um nodo grafo, dado que o grafo e representado por uma **lista de adjacencias**.

12) Quantas componentes conexas tem o seguinte grafo?



13) Descreva com suas palavras o funcionamento de um algoritmo de busca em profundi- dade. Dedois exemplos de aplicac¸ao real desse algoritmo.

14) Descreva com suas palavras o funcionamento de um algoritmo de busca em largura. Dedois exemplos de aplicacao real desse algoritmo.

15) Descreva com suas palavras o funcionamento de um algoritmo de busca pelo menor caminho. De dois exemplos de aplicacao real desse algoritmo.

16) Dado o digrafo $tt = (V, E)$ sendo $V = M, N, O, P, Q, R, S$ e

$$E = \{(M, S), (N, O), (P, R), (N, S), (O, M), (N, Q), (O, M),$$
$$(P, P), (S, M), (O, N), (S, M), (N, R), (P, M), (M, S)\}$$

(a) Especifique, caso exista, um caminho simples desde o vertice M ateo vertice S.

(b) Especifique, caso exista, um ciclo simples, envolvendo pelo menos 4 nos.

(c) O digrafo econexo ou nao conexo?

(d) Qual o grau dos vertices N e R.

(e) Represente o digrafo utilizando representaçao por lista de adjacencia.

(f) Represente o digrafo utilizando representaçao por matriz de adjacencia.

17) Implemente um algoritmo para verificar se um grafo eacıclico utilizando o algoritmo de busca em profundidade.

18) Escreva uma versao nao recursiva do algoritmo de busca em profundidade.

19) Exemplifique com algumas situaçoes de uso dos grafos e justifique.

20) Os Turistas Jensen, Leuzingner, Dufour e Medeiros se encontram em um bar de Paris e começam a conversar. As lınguas disponıveis sao o ingle, o franc, o portugu e o alemao. Jensen fala todas. Leuzingner nao fala apenas o portugues. Dufour fala franc e alemao. Medeiros fala ingle e portugu. Represente por meio de um digrafo todas as possibilidades de um deles dirigir a palavra a outro, sendo compreendido.

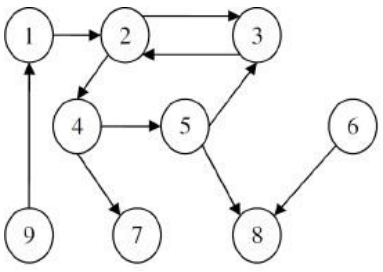
21) Voceusaria uma lista de adjacencia ou uma matriz de adjacência em cada um dos casos abaixo? Justifique sua escolha.

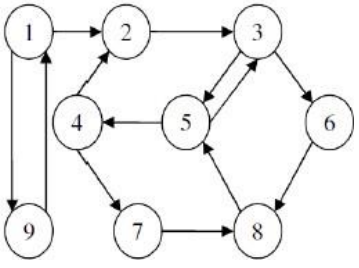
(a) O grafo tem 10.000 vertices e 20.000 arestas, e e importante usar tao pouco espac¸o quanto possıvel.

(b) O grafo tem 10.000 vertices e 20.000.000 arestas, e e espac¸o quanto possıvel. importante usar tao pouco

(c) Voce deve ter a aresta adjacente tao rapido quanto possıvel, sem se importar quanto espac¸o voce usa.

22) Dado os grafos abaixo, mostre o resultado da busca em largura e em profundidade.





23) Seja um grafo G cujos vertices sao os inteiros de 1 a 8 e os vertices adjacentes a cada vertice sao dados pela tabela abaixo:

Vértice	Vértices Adjacentes
1	2 3 4
2	1 3 4
3	1 2 4
4	1 2 3 6
5	6 7 8
6	4 5 7
7	5 6 8
8	5 7

(a) Desenhe o grafo G.

(b) Represente o grafo por meio de uma matriz de adjacencia.

(c) Represente o grafo por meio de uma lista de adjacencia.

24) Dada a matriz de adjacencias de uma grafo de N vertices, faça um algoritmo que deter- mine se esse grafo eorientado ou nao-orientado.

25) Por que, em uma matriz de adjacencias, verificar a existencia de uma aresta e$O(1)$.

26) Qual metodo usa mais espaço, listas de adjacencia ou matriz de adjacencia e porquè

27) Escreva um algoritmo que verifique se dois grafos G1 e G2 nao sao isomorfos com base no numero de vertices e arestas e, também, comparando a lista ordenada dos graus de seus vertices.

28) Escreva um algoritmo que recebe um caminho e verifica se ele eum ciclo

29) Escreva um algoritmo que recebe um caminho e verifica se ele eum ciclo simples

30) Considere a seguinte representacao de um grafo com 8 vertices e 9 arestas usando listas de adjacencias

A: E F B B: A
C: G D F D: H G C
E: A
F: A G C G: D F C
H: D

Mostre o resultado da busca em em largura e em profundidade a partir do vertice A. Mostre tambem a distancia de cada vertice ao vetice A.

31) Considere a seguinte representacao de um grafo usando listas de adjacencias:

A: F B B: A F
C: D I D: E C I
E: D J I F: A B
G: H
H: G
I: J E C D J: I E

Obtenha os componentes conectados de um grafo usando o algoritmo de busca em profundidade.

IV. Parte IV

Representação de Grafos e 2 Buscas em Grafos UNICAMP, Brasil

1) Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees?

2) The transpose of a directed graph G = (V, E) is the graph $G^T = (V, E^T)$, where E^T = (v, u) $\in$ V x V : (u, v) $\in$ E. Thus, G^T is G with all its edges reversed. Describe efficient algorithms for computing G^T from G, for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

3) The square of a directed graph G = (V, E) is the graph $G^2 = (V, E^2)$ such that (u, w) $\in E^2$ if and only if for some v $\in$ V, both (u, v) $\in$ E and (v, w) $\in$ E. That is, G^2 contains an edge between u and w whenever G contains a path with exactly two edges between u and w. Describe efficient algorithms for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. Analyze the running times of your algorithms.

4) When an adjacency-matrix representation is used, most graph algorithms require time $\Omega(V^2)$, but there are some exceptions. Show that determining whether a directed graph G contains a universal sink (a vertex with in-degree $|V| - 1$ and out-degree 0) can be determined in time $O(V)$, given an adjacency matrix for G.

5) Give a counterexample to the conjecture that if there is a path from u to v in a directed graph G, and if d[u] < d[v] in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

6) Show that a depth-first search of an undirected graph G can be used to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected com- ponents. More precisely, show how to modify depth-first search so that each vertex v is assigned an integer label cc[v] between 1 and k, where k is the number of connected components of G, such that cc[u] = cc[v] if and only if u and v are in the same connected component.

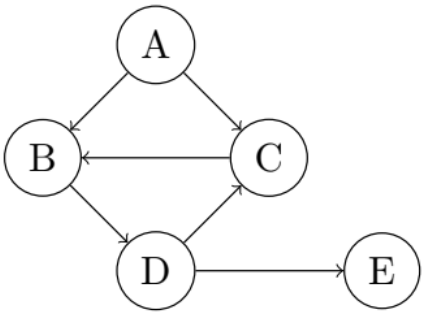
7) Give an algorithm that determines whether or not a given undirected graph G = (V, E) contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

8) Another way to perform topological sorting on a directed acyclic graph G = (V, E) is to repeate- dly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in time $O(V + E)$. What happens to this algorithm if G has cycles?

V. Parte V

University of Stanford, EEUU

1) Consider the following directed graph:

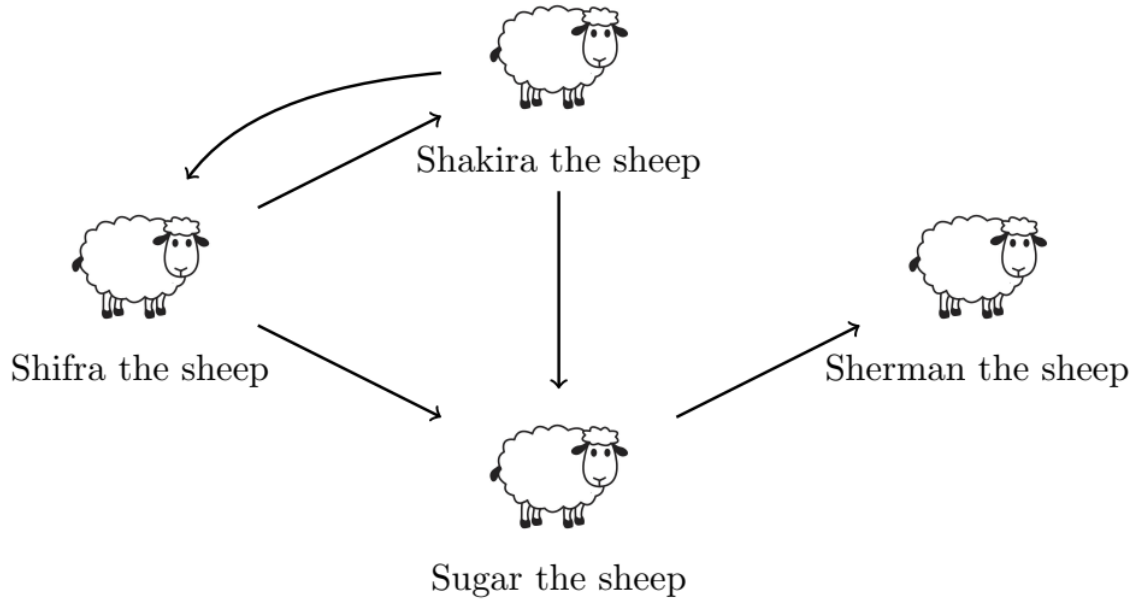


(a) Draw the DFS tree for this graph, starting from node A. Assume that DFS traverses nodes in alphabetic order. (That is, if it could go to either B or C, it will always choose B first).

(b) Draw the BFS tree for this graph, starting from node A. Assume that BFS traverses nodes in alphabetic order. (That is, if it could go to either B or C, it will always choose B first).

2) Go over the problems on the midterm that you didn't get. You don't have to turn anything in, but this problem set is intentionally a little bit shorter to give you time to go over the midterm. Midterms will be handed back and solutions will be posted on Monday.

3) You arrive on an island with n sheep. The sheep have developed a pretty sophisticated society, and have a social media platform called Baaaahtter (it's like Twitter but for sheep1). Some sheep follow other sheep on this platform. Being sheep, they believe and repeat anything that they hear. That is, they will re-post anything that any sheep they are following said. We can represent this by a graph, where (a) (b) means that (b) will re-post anything that (a) posted. For example, if the social dynamics on the island were:



the Sherman the Sheep follows Sugar the Sheep, and Sugar follows both Shakira and Shifra, and so on. This means that Sherman will re-post anything that Sugar posts, Sugar will re-post anything by Shifra and Shikira, and so on. (If there is a cycle then each sheep will only re-post a post once). For the parts below, let tt denote this graph on the n sheep. Let m denote the number of edges in tt.

(a) **(2 pt.)** Call a sheep a **source sheep** if anything that they post eventually gets re-posted by every other sheep on the island. In the example above, both Shifra and Shakira are source sheep. Prove that all source sheep are in the same strongly connected component of *tt*, and every sheep in that component is a source sheep.

[We are expecting: A short but rigorous proof.]

(b) **(4 pt.)** Suppose that there is at least one source sheep. Give an algorithm that runs in time $O(n + m)$ and finds a source sheep. You may use any algorithm we have seen in class as a subroutine.

[We are expecting: Pseudocode or an English description of your algorithm; a short justiftcation that it is correct; and short justiftcation of the running time. You may use any statement we have proved in class without re-proving it.]

(c) **(2 pt.)** Suppose that you don't know whether or not there is a single source sheep. Give an algorithm that runs in time $O(n + m)$ and either returns a source sheep or returns no source sheep. You may use any algorithm we have seen from class as a subroutine.

[We are expecting: Pseudocode or an English description of your algorithm, and an informal

justiftcation of why it is correct and runs in time $O(n + m)$.]

VI. Parte VI

Algoritmos elementares em grafos, UNICAMP, Brasil.

1) (CLRS 22.1-1) Dada uma representaçao por listas ligadas de um grafo orientado, quanto tempo leva para se calcular o grau de saida de cada vertice? E os graus de entrada?

2) (CLRS 22.1-5) O quadrado de um grafo orientado $tt = (V, E)$ e o grafo $tt^2 = (V, E^2)$ tal que $(u, w) \in E^2$ se e somente se para algum $v \in V$, tanto $(u, v) \in E$ como $(v, w) \in E$. Ou seja, o grafo tt^2 contem uma aresta (u, w) se em tt existe um caminho de u a v com exatamente duas arestas. Descreva algoritmos eficientes para determinar tt^2 a partir de tt tanto para as representaçoes por listas de adjacencias como para a por matriz de adjacencia. Analise a complexidade de tempo de seus algoritmos.

3) (CLRS 22.1-6) Quando uma representaçao por matriz de adjacencia e usada, a maioria dos algoritmos requer tempo $\Omega(V^2)$, mas ha excec¸oes. Mostre que determinar se um grafo orientado tt contem um sorvedouro universal – um vertice com grau de entrada $|V| - 1$ e grau de saida 0 – pode ser determinado em tempo $O(V)$, dada uma matriz de adjacencia para tt.

4) (CLRS 22.2-4) Argumente porque em uma busca em largura, o valor $d[u]$ atribuido a cada vertice e independente da ordem em que os vertices aparecem em cada lista de adjacencia. Descreva um exemplo msotrando que a arvore de busca em largura obtida pelo algoritmo BFS visto em aula pode depender da ordem em que vertices aparecem nas listas de adjacencias.

5) (CLRS 22.2-5) Descreva um grafo orientado $tt = (V, E)$ com um vertice origem s especificado e uma arvore orientada T tal que para cada vertice $v \in V$, o unico caminho de s a v em T e um caminho mais curto de s a v em tt, mas T nao pode ser obtida atraves de uma execuçao de BFS, independentemente da ordem em que os vertices aparecem em cada lista de adjacencia.

6) (CLRS 22.3-7) Mostre um contra-exemplo para a afirmaçao: se existe um caminho de u a v em um grafo orientado tt e se $d[u] < d[v]$ em uma busca em profundidade de tt, entao v e um descendente de u na floresta de busca em profundidade obtida.

7) (CLRS 22.4-2) Descreva um algoritmo linear que tem como entrada um grafo orientado acıclico $tt = (V, E)$ e dois vertices s e t, e determina o numero de caminhos de s a t. Note que so e preciso contar o numero de caminhos e nao lista-los.

8) (CLRS 22.4-3) Descreva um algoritmo que determina se um grafo nao-orientado $tt = (V, E)$ contem um ciclo. Seu algoritmo deve ter complexidade $O(V)$ independente de $|E|$.

9) (CLRS 22.4-5) Outra forma de obter uma ordenaçao topologica de uma grafo orientado acıclico $tt = (V, E)$ e repetidamente encontrar um vertice com grau de entrada 0, imprimi-lo e remove-lo juntamente com todas as arestas que saem deste do grafo. Explique como implementar esta ideia em tempo $O(V + E)$.

VII. Parte VII

PUC-RIO, Brasil

1) Responda as perguntas abaixo.

a) Em um grafo bipartido completo, todos os vértices de uma partição são ligados a todos vértices da outra. Qual é a altura da árvore gerada por uma BFS em um grafo bipartido completo? Qual é a altura da árvore gerada por uma DFS em um grafo bipartido completo?
b) Seja G um grafo direcionado e sejam u, v dois vértices de G. Assuma que ao executar uma DFS obtemos pre(u) < pre(v) < post(v) < post(u). Podemos afirmar que existe um caminho entre u e v no grafo? Por que?
c) Seja G um grafo direcionado e sejam u, v dois vértices de G. Assuma que ao executar uma DFS em G obtemos pre(u) < pre(v) < post(u) < post(v). O que podemos afirmar sobre a DFS? Por que?
d) Seja G um grafo fortemente conexo. Podemos afirmar que G não admite ordenação topológica? Por que?
e) Seja G um grafo que admite uma ordenação topológica O que podemos afirmar sobre as componentes fortemente conexas de G? Por que?

2) Seja G um grafo direcionado com pesos nas arestas, seja s um vértice de G e seja k um inteiro positivo. Como podemos encontrar os k vértices mais próximos de s? Com qual complexidade? Analise em função de k, m, n.
3) Seja G = (V, E) um grafo direcionado acíclico. [PUC-RIO, Laber, 2013]
   a) Quantas componentes fortemente conexas tem G? Por que?
   b) Ao executar uma DFS em G obtemos P os[v] para todo vértice v ∈ V. Seja O uma ordenação dos vértices em ordem decrescente destes valores de Pos. Podemos afirmar que O é uma ordenação topológica para G? Por que?
4) Considere o pseudocódigo abaixo que executa em um grafo n □ao direcionado G = (V, E) que tem n vértices e m arestas. [PUC-RIO, Laber, 2013]

```
1.  Para v ∈ V
2.      EXISTE(v) ← FALSE
3.      Para u ∈ Adj[v]
4.          Se existe uma caminho entre u e v no grafo G − uv
5.              EXISTE[v] ← TRUE
6.          Fim Se
7.      Fim Para
8.  Fim Para
```

   a) Analise a sua complexidade em função de n e m. Assuma que a existência de um caminho na linha 4 é verificada através de uma BFS.
   b) O que podemos afirmar para os vértices v tal que EXISTE[v]=TRUE ao término do algoritmo? E para os vértices v tal que EXISTE[v]=FALSE ao término do algoritmo?
5) Seja um grafo G = (V, E) não direcionado com pesos nas arestas e seja f uma aresta de G. [PUC-RIO, Laber, 2013]
   a) Assuma que os pesos das arestas são **positivos**. Explique com palavras como seria um algoritmo polinomial para encontrar a árvore geradora com menor peso dentre as árvores geradoras para G que não contém a aresta f. Analise a complexidade deste algoritmo.
   b) Para este item assuma que algumas arestas podem ter pesos negativos. Explique com palavras como seria um algoritmo polinomial para encontrar o subgrafo conexo gerador de G de peso mínimo.
6) Seja G = (V, E) um grafo não direcionado e seja uv uma aresta de G. Explique com palavras como seria um algoritmo polinomial para determinar se existe um ciclo no grafo que contém a aresta uv. Qual a complexidade deste procedimento?

7) Dado um grafo conexo e não direcionado G = (V, E), explique como seria um algoritmo para decidir se existe ou não uma aresta em E tal que sua remoção não desconecta o grafo. Analise a complexidade do algoritmo proposto. Soluções mais eficientes terão maior pontuação.
8) Seja G = (V, E) um grafo não direcionado, sem pesos, armazenado como uma lista de adjacências. Dado dois vértices u e v e um caminho P entre u e v responda os itens abaixo.
   a) Como seria um algoritmo polinomial para decidir se existe um caminho Q entre u e v que não tem arestas em comum com P? Discuta a complexidade do algoritmo proposto.
   b) Como seria um algoritmo polinomial para decidir se existe um caminho R entre u e v que tem no máximo uma aresta em comum com P? Discuta a complexidade do algoritmo proposto.
9) Considere um tabuleiro de xadrez (4x4) com as casas numeradas de 1 a 16. Seja G = (V, E) um grafo, aonde V = {1, ..., 16} e

$$E = \{ij | \text{ as casas } i \text{ e } j \text{ são adjacentes (separadas por uma reta) no tabuleiro}\}.$$

Além disso, assuma que o custo da aresta entre i e j é i × j.
   a) Desenhe G e compute uma árvore geradora mínima para G. Explique com palavras o algoritmo utilizado e analise sua complexidade.
   b) Encontre a árvore de caminhos mais curtos (de menor peso) entre o n □o 4 e os demais nós do grafo. Explique com palavras o algoritmo utilizado e analise sua complexidade.
10) Considere um grafo direcionado G = (V, E) representado por uma lista de adjacências. Explique como seria um algoritmo que recebe um vértice s ∈ V e devolve uma lista ordenada contendo as distancias de todos os vértices que são alcançáveis a partir de s. Analise a complexidade do algoritmo proposto. Quanto mais eficiente o algoritmo maior a pontuação.
11) Seja G = (V, E) um grafo direcionado com pesos nas arestas e sem ciclos negativos.
   a) Como seria um algoritmo para descobrir se existe um vértice u ∈ V tal que u alcança todos vértices de V e todos vértices de V alcançam u. Quanto mais eficiente (assintoticamente) o algoritmo maior a pontuação.
   b) Responda o item (a) assumindo que todos os pesos das arestas são números positivos.
   c) Responda o item (a) assumindo que todas as arestas tem peso 1.
12) Modifique o pseudo-código da DFS abaixo para ela calcular a altura da árvore DFS que ela produz para um grafo conexo não direcionado G = (V, E) ao ser chamada a partir de um nó s. Lembre que cada vértice de V é um nó da árvore DFS e u é pai de v na árvore se e somente se v é visitado a primeira vez a partir de u. Qual a complexidade do procedimento? A altura representa o comprimento do maior caminho no grafo que tem início em s?

```
Main
        DFS-VISIT(s)
DFS-VISIT(u)
1.      Marque u como visitado
2.      Para todo vértice v ∈ Adj(u)
3.          Se v não foi visitado
4.              DFS-VISIT(v)
```

13) Seja um grafo conexo e não direcionado G = (V, E). Uma aresta e ∈ G é uma ponte se G − e é desconexo. Explique com palavras como seria um algoritmo polinomial para determinar se existe um ciclo no grafo que contém a aresta uv. Qual a complexidade deste procedimento?
14) Seja G = (V, E) um grafo não direcionado. Um conjunto S de vértices é uma cobertura para o grafo G se toda aresta de E tem pelo menos uma extremidade que pertence a S. Considere o seguinte algoritmo para obter uma cobertura para G.

$S \leftarrow \emptyset$

Enquanto G contém arestas faça

$v \leftarrow$ vértice de maior grau em G. Em caso de empate escolha o vértice com menor número

$S \leftarrow S \cup v$

$G \leftarrow G - v$.

Fim Enquanto

a) O conjunto S obtido ao término do algoritmo é sempre uma cobertura para G? Por que?
b) Assuma que G é uma árvore. Mostre um exemplo em que o conjunto S obtido ao término do algoritmo é uma cobertura para G com número mínimo de vértices.
c) Assuma que G é uma árvore. Mostre um exemplo em que S não é uma cobertura para G com número mínimo de vértices.
d) Assuma que G é um caminho, ou seja, V = {1, . . . , n} e E = {{1, 2}, {2, 3}, . . . , {n − 1, n}}. O conjunto S obtido ao término do algoritmo ser □a sempre uma cobertura para G com o número mínimo de vértices? Por que?