EXERCÍCIOS DE ALGORITMOS EM GRAFOS
1o. SEMESTRE DE 2007

1. Desenhe todos os grafos não-isomorfos com n vértices para todo $0 \leq n \leq 4$. (Por exemplo, para $n = 2$, você deve desenhar 2 grafos.)
2. Uma *floresta* é um grafo acíclico (isto é, que não contém circuitos). Uma *árvore* é uma floresta conexa. Desenhe todas as árvores com conjunto de vértices $[n] = \{1, \ldots, n\}$ para todo $0 \leq n \leq 4$. (Por exemplo, para $n = 3$, você deve desenhar 3 árvores.)
3. Seja g_n ($n \geq 0$) o número total de grafos não-isomorfos com n vértices (por exemplo, $g_2 = 2$). Mostre que
$$2^{\binom{n}{2}}/n! \leq g_n \leq 2^{\binom{n}{2}}. \tag{1}$$
4. [Desafio] Desenvolva um sistema computacional para determinar ou estimar g_n para valores grandes de n. $\{$*Data de entrega*: em aberto$\}$
5. (Exemplo 1.15 de ISTG) Duas arestas de um grafo G são *adjacentes* se têm uma ponta comum. Essa relação de adjacência define o *grafo das arestas* de G. Se G' denota o grafo das arestas de G, então $V(G) = A(G)$ e cada aresta de G' é um par ab em que a e b são arestas adjacentes de G. Faça uma figura do grafo das arestas de um K_3 e de um $K_{1,3}$ (o grafo completo com 3 vértices e o grafo bipartido completo com classes de vértices de cardinalidades 1 e 3).

   Faça uma figura do grafo das arestas de um K_4. Quantos vértices e quantas arestas tem o grafo das arestas de um K_n? Seja G o grafo das arestas de um K_5. Desenhe o *complemento* $G^{\mathrm{c}} = \overline{G}$ de G (o grafo com conjunto de vértices $V(G)$, com dois vértices distintos x e y de $V(G)$ adjacentes neste grafo se e só se x e y não são adjacentes em G).
6. (E 1.17 de ISTG) Mostre que o grafo das arestas de um K_5 é isomorfo ao complemento do grafo de Petersen.
7. (Proposição 1.1 de ISTG) Prove que, em todo grafo, a soma dos graus dos vértices é igual ao dobro do número de arestas. Ou seja, mostre que todo grafo (V, A) satisfaz a identidade $\sum_{v \in V} g(v) = 2|A|$.
8. (E 1.20 de ISTG) Mostre que todo grafo tem um número par de vértices de grau ímpar.
9. (E 1.21 de ISTG) Mostre que todo grafo com dois ou mais vértices tem pelo menos dois vértices de mesmo grau.
10. Mostre que um grafo é 2-colorível se e só se ele não contém um circuito de comprimento ímpar. $\{$*Data de entrega*: 16/3/2007$\}$
11. Descreva precisamente um algoritmo eficiente para decidir se um dado grafo é biparticionável. $\{$*Data de entrega*: 16/3/2007$\}$

Date: Versão de 1 de junho de 2007.

12. Gere vários grafos aleatórios com n vértices e m arestas e estime a probabilidade de eles serem conexos. Use os seguintes valores de m e n grandes:
    (i) $m = \lceil (1/4)n \log n \rceil$,
    (ii) $m = \lceil (1/2)n \log n \rceil$,
    (iii) $m = \lceil n \log n \rceil$,
    (iv) $m = \lceil 2n \log n \rceil$.
    Acima, escrevemos log para o logaritmo natural ln. {*Data de entrega*: 23/3/2007}
13. Mostre que todo grafo G com pelo menos um vértice contém um caminho de comprimento $\delta(G)$. Mostre que todo grafo G contém um circuito de comprimento pelo menos $\delta(G) + 1$, desde que $\delta(G) \geq 2$ ($\delta(G) = \min_v \deg(v)$, onde o mínimo é tomado sobre todo $v \in V(G)$ e $\deg(v)$ denota o grau de v). {*Data de entrega*: 23/3/2007}
14. Seja G um grafo conexo. Considere a família $\mathcal{F}$ dos caminhos de comprimento máximo em G.
    (i) Prove que quaisquer dois caminhos em $\mathcal{F}$ têm um vértice em comum.
    (ii) Suponha agora que os caminhos em $\mathcal{F}$ têm comprimento ímpar. Prove que quaisquer dois caminhos em $\mathcal{F}$ têm pelo menos dois vértices em comum.
15. (E 1.39 de ISTG) Seja G um grafo com $\delta(G) > (|V(G)| - 2)/2$. Mostre que G é conexo.
16. (E 1.40 de ISTG) Seja G um grafo com $\delta(G) \geq 3$. Mostre que G tem um circuito par.
17. (E 1.41 de ISTG) Mostre que todo grafo G satisfaz a desigualdade $|E(G)| \geq |V(G)| - c(G)$, onde $c(G)$ é o número de componentes de G.
18. Seja G um grafo conexo e T uma árvore geradora de G obtida pela execução de uma busca de profundidade em G. Dê um argumento preciso para provar que toda arestas em $A(G) \setminus A(T)$ é da forma $\{x, y\}$ com x um ancestral de y em T. (Lembre que T tem uma raiz natural, a saber, o vértice r a partir do qual a busca em profundidade foi disparada. Dizemos que x é um ancestral de y em T se o único caminho de r a y em T passa por x.)
19. Sejam G um grafo e v e w dois vértices de G. Mostre que se existe em G um passeio com extremos v e w, então existe uma caminho com extremos v e w em G.
20. Desenhe o grafo com 8 vértices e 10 arestas dado por

```
0:  7  5  2
1:  7
2:  6  0
3:  5  4
4:  5  7  3  6
5:  0  4  3
```

```
6:  4  2
7:  0  1  4
```

Simule a execução de `dfsR(G, {0,0})` (busca em profundidade a partir do vértice `0`). Determine os vetores `pre[]` e `st[]` após a execução desta chamada. Classifique todas os pares ordenados (a, b) com $ab \in A(G)$ nos quatro tipos (i) tipo árvore, (ii) tipo pai, (iii) tipo ascendente, (iv) tipo descendente. $\{$*Data de entrega*: 30/3/2007$\}$

21. Seja G um grafo 2-aresta-conexo. Prove que para quaisquer dois vértices distintos x e y de G, existem dois caminhos P_1 e P_2 de x a y que são disjuntos nas arestas (isto é, $E(P_1) \cap E(P_2) = \emptyset$). $\{$*Data de entrega*: 30/3/2007$\}$
22. Faça o Exercício 18.42 do Sedgewick.
23. Faça o Exercício 18.43 do Sedgewick. $\{$*Data de entrega*: 10/4/2007$\}$
24. Seja $G = (V, E)$ um grafo. Defina a relação $\sim$ sobre E como segue: para todo e e $f \in E$, $e \sim f$ se $e = f$ ou existe um circuito em G que contém ambos e e f. Prove que $\sim$ é uma relação de equivalência sobre $E(G)$. $\{$*Data de entrega*: 13/4/2007$\}$
25. Escreva um algoritmo que, dado um DAG D, determina o comprimento máximo de um caminho dirigido em D.
26. Seja G um grafo dirigido. O *fecho transitivo* $F = F(G)$ de G é o grafo dirigido com $V(F) = V(G)$ e com o arco (x, y) presente em $E(F)$ se e só se y é acessível a partir de x por um caminho dirigido em G. Descreva um algoritmo que, dado um DAG D (um grafo dirigido acíclico), computa o fecho transitivo $F = F(D)$ de D. Seu algoritmo deve ter complexidade de tempo $O(n + M)$, onde $n = |V(D)|$ e $M = |E(F)|$. $\{$*Data de entrega*: 8/5/2007$\}$
27. Escreva um programa que recebe como entrada um grafo dirigido G e tem como saída os componentes fortemente conexos de G. O seu programa deve também ter como saída o tamanho (número de vértices) dos componentes fortemente conexos, ordenados de forma decrescente. Seu programa deve ser eficiente, e deve receber como entrada os grafos gerados pelos programas `g_roget_simples` e `g_random`; veja

`http://www.ime.usp.br/~yoshi/2007i/mac328/Exercicios/SGB/ROGET`

e

`http://www.ime.usp.br/~yoshi/2007i/mac328/Exercicios/SGB/RANDOM`

No caso de `g_random`, os grafos de teste serão gerados com a opção `-d` (para gerar grafos dirigidos). Ainda no caso de `g_random`, experimente os seguintes valores de m e n grandes (como sempre, m é o número de arcos e n é o número de vértices):

(i) $m = \lfloor 0.9n \rfloor$,
(ii) $m = \lfloor 0.95n \rfloor$,
(iii) $m = n$,
(iv) $m = \lfloor 1.05n \rfloor$,
(v) $m = \lfloor 1.1n \rfloor$,

Seu programa deve admitir a opção `-t` para imprimir somente o tamanho dos componentes fortemente conexos em ordem decrescente (esta opção é útil para estudar os casos em que n é grande).

Você deve implementar os algoritmos de Kosaraju e de Tarjan. Faça uma comparação empírica da eficiência dos dois algoritmos. {*Data de entrega*: 11/5/2007}

28. Use os programas que você escreveu no Exercício 27 para estudar a "estrutura de componentes (fortemente conexos)" dos grafos dirigidos aleatórios dados no enunciado daquele exercício, isto é, gere grafos com os valores de m dados em (i)–(v) daquele exercício e valores grandes de n e procure descobrir se a "cara" do grafo muda conforme m cresce em relação a n (como dado em (i)–(v)). {*Data de entrega*: 27/5/2007}
29. Use os programas que você escreveu no Exercício 27 para estudar para quais valores de m é provável que os grafos dirigidos aleatórios com n vértices (n grande) e m arcos é fortemente conexo. Naturalmente, a resposta aqui é uma função de n (isto é, $m = m(n)$). [*Sugestão.* Lembre do Exercício 12.] {*Data de entrega*: 27/5/2007}
30. O algoritmo de Kosaraju para a determinação dos componentes fortemente conexos de um grafo dirigido envolve o cômputo do reverso de $\vec{G}$; depois usamos o *reverso* da ordem dada por `post[]`. Uma sugestão natural é usar o grafo original $\vec{G}$ (*sem* revertê-lo) e usar então a ordem dada por `post[]` *sem* revertê-la. Esta idéia funciona? Prove ou dê um contra-exemplo. {*Data de entrega*: 22/5/2007}
31. Seja G um grafo e $c\colon E(G) \to \mathbb{R}$ uma função custo definida nas arestas de G. Seja T uma árvore geradora de G. Para $e \in E(T)$, sejam T_1^e e T_2^e os dois componentes de $T - e$. Prove que são equivalentes:
    (i) T é uma árvore geradora de custo mínimo em G.
    (ii) Toda aresta $e \in E(T)$ tem peso mínimo dentre as arestas em $E(V(T_1^e), V(T_2^e))$.

    {*Data de entrega*: 1/6/2007}
32. Sejam G e c como no Exercício 31. Seja T uma árvore geradora de G. Lembre que se $e \in E(G) \setminus E(T)$, então o grafo $T + e$ contém um único circuito, chamado de *circuito fundamental de* e *em relação a* T, freqüentemente denotado por $C(e, T)$. Prove que são equivalentes:
    (i) T é uma árvore geradora de custo mínimo em G.
    (ii) Toda $e \in E(G) \setminus E(T)$ tem peso máximo dentre as arestas em $C(e, T)$.

    {*Data de entrega*: 1/6/2007}
33. Seja $G = (V, E)$ um grafo. Podemos definir um grafo dirigido $\vec{G} = (V, \vec{E})$ a partir de G, atribuindo para cada aresta $e = \{x, y\}$ de G uma de suas duas possíveis orientações (formalmente, temos $(x, y) \in \vec{E}$ ou $(y, x) \in \vec{E}$, mas não ambos). Podemos chamar um tal $\vec{G}$ de uma *orientação* de G. Suponha que G tem duas árvores geradoras T_1 e T_2

disjuntas nas arestas, isto é, T_1 e T_2 são subárvores de G com $V(T_1) = V(T_2) = V(G)$ e $E(T_1) \cap E(T_2) = \emptyset$. Mostre que G admite uma orientação $\vec{G}$ fortemente conexa.

34. Vale a recíproca do Exercício 33? Isto é, se um grafo G admite uma orientação $\vec{G}$ fortemente conexa, então há árvores T_1 e T_2 como no enunciado daquele exercício?

35. Seja G um grafo dirigido e $c\colon E(G) \to \mathbb{R}$ um função custo definida nos arcos de G. Por simplicidade, suponha que o custo de todo arco é não-negativo. Fixe $s \in V(G)$, e suponha que todo vértice de G é acessível a partir de s. Para todo $x \in V(G)$, escrevemos $d_c(s,x)$ para o custo mínimo dos caminhos dirigidos de s a x. Um *potencial* p *para* (G,c) é uma função $p\colon V(G) \to \mathbb{R}$ tal que

$$p(v) \leq p(u) + c(u,v) \text{ para todo } (u,v) \in E(G). \tag{2}$$

Ademais, dizemos que um arco (u,v) é *justo* se vale a *igualdade* em (2). Suponha que um potencial p satisfaz as seguintes duas propriedades:
(a) $p(s) = 0$,
(b) todo vértice $x \in V(G)$ é acessível a partir de s por um caminho que só contém arcos justos.

Prove os seguintes fatos:
(i) Para todo $x \in V(G)$, vale que $p(x) \leq d_c(s,x)$. [*Observação.* Para este item, a única hipótese necessária sobre o potencial p é que $p(s) \leq 0$.]
(ii) Para todo $x \in V(G)$, vale que $d_c(s,x) \leq p(x)$. [*Sugestão.* Seja (P_k) a seguinte asserção: *se x é acessível a partir de s através de um caminho com k arcos, todos eles justos, então* $d_c(s,x) \leq p(x)$. Prove que (P_k) vale para todo k por indução em k.]

Conclua que $d_c(s,x) = p(x)$ para todo vértice x de G. {*Data de entrega*: 12/6/2007}

36. Considere a implementação do algoritmo de Dijkstra visto em sala (Programa 21.1):

```
#define GRAPHpfs GRAPHspt
#define P (wt[v] + t->wt)
void GRAPHpfs(Graph G, int s, int st[], double wt[])
  { int v, w;  link t;
    PQinit(); priority = wt;
    for (v = 0; v < G->V; v++)
      { st[v] = -1; wt[v] = maxWT; PQinsert(v); }
    wt[s] = 0.0; PQdec(s);
    while (!PQempty())
      if (wt[v = PQdelmin()] != maxWT)
        for (t = G->adj[v]; t != NULL; t = t->next)
          if (P < wt[w = t->v])
            { wt[w] = P; PQdec(w); st[w] = v; }
  }
```

Seja G o grafo de 5 vértices e 10 arcos com custos nos arcos dado por

```
0: 1(5) 3(10)
1: 2(2) 3(3) 4(9)
2: 0(7) 4(6)
3: 1(2) 4(1)
4: 2(4)
```

Na notação acima, por exemplo, `0: 1(5)` significa que há um arco de `0` para `1` de custo `5`.

(i) Execute o Programa 21.1 no grafo G dado acima, com $s = 0$, ilustrando a evolução do algoritmo através de diagramas convenientes (uns 6 diagramas bastam).

(ii) Determine um potencial $p\colon V(G) \to \mathbb{R}$ para certificar que você encontrou as distâncias de 0 a todos os outros vértices em G em (i) acima corretamente (veja a Questão 35). Você deve dizer explicitamente por que sua função p é um potencial e por que ela de fato certifica as distâncias encontradas.

(iii) Descreva um algoritmo de complexidade $O(n + m)$ que poderia ser executado após o Programa 21.1 para verificar que de fato as distâncias a partir de s foram determinadas corretamente. Por simplicidade, suponha que todos os vértices de G são acessíveis a partir de s. (Como você pode se livrar dessa hipótese?)

{*Data de entrega*: 12/6/2007}

37. Simule o método de Ford–Fulkerson na rede a seguir:

```
0: 1(2) 2(3)
1: 3(3) 4(1)
2: 3(1) 4(1)
3: 5(2)
4: 5(3)
```

Na notação acima, por exemplo, `0: 1(2)` significa que há um arco de `0` para `1` com capacidade `2`. Mostre a evolução do fluxo desde o fluxo nulo até o fluxo máximo. Exiba um corte mínimo que certifica que seu fluxo final é de fato um fluxo máximo. {*Data de entrega*: 15/6/2007}

38. Escreva um programa para encontrar emparelhamentos de cardinalidade máxima em grafos bipartidos. O seu programa *deve ser baseado na redução deste problema ao problema do fluxo máximo*. Seu programa deve receber como entrada os grafos gerados pelos programas `g_random` e `g_livros_simples` (note que estes programas admitem a opção `-b`). {*Data de entrega*: 22/6/2007}

39. Use o programa `g_random` para gerar grafos bipartidos aleatórios $G(n, n; m)$ com n vértices em cada classe da bipartição e m arestas e estude para quais valores de $m = m(n)$ é provável que tal grafo tenha um *emparelhamento perfeito* (considere valores grandes de n) {*Data de entrega*: 22/6/2007}

40. Novamente usando o programa `g_random`, estude o tamanho dos emparelhamentos de cardinalidade máxima em grafos aleatórios bipartidos $G(n, n; m)$ para valores de $m = m(n)$ inferiores ao valor que você estabeleceu no Exercício 39 (faça gráficos etc). Estude os valores de m próximos de n com certo cuidado. (Considere valores grandes de n.) $\{$*Data de entrega*: 22/6/2007$\}$