EXERCÍCIOS DE ALGORITMOS EM GRAFOS
1o. SEMESTRE DE 2004

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.) {*Data de entrega*: 9/3/2004}
2. Seja g_n ($n \geq 0$) o número total de grafos não-isomorfos com n vértices (por exemplo, $g_2 = 2$). Mostre que

$$2^{\binom{n}{2}}/n! \leq g_n \leq 2^{\binom{n}{2}}. \tag{1}$$

   {*Data de entrega*: 9/3/2004}
3. [Desafio] Desenvolva um sistema computacional para determinar ou estimar g_n para valores grandes de n. {*Data de entrega*: em aberto}
4. 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.
   {*Data de entrega*: 12/3/2004}
5. Determine o vertebrado correspondente à função

$$f = \begin{pmatrix} 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ a\ b\ c\ d\ e\ f \\ 0\ b\ f\ e\ c\ 3\ d\ 4\ 8\ 4\ 7\ 4\ 2\ e\ e\ 9 \end{pmatrix} \tag{2}$$

   {*Data de entrega*: 16/3/2004}
6. Prove que todo grafo G com 6 vértices contém um clique com 3 vértices ou um conjunto independente com 3 vértices; isto é, $\omega(G) \geq 3$ ou $\alpha(G) \geq 3$. Mostre que esta asserção é falsa para grafos com 5 vértices; isto é, existem grafos H com 5 vértices para os quais temos $\omega(H) < 3$ e $\alpha(H) < 3$. {*Data de entrega*: 19/3/2004}
7. Estude as implementações para o tipo abstrato de dados `Graph` dadas nos programas 17.3 e 17.6. Implemente a função `GRAPHremoveE` para a implementação com listas ligadas. {*Data de entrega*: 23/03/2004}
8. Para cada inteiro $n > 0$, seja P_n a probabilidade de o grafo aleatório com n vértices e m arestas ser conexo, onde

$$m = \left\lfloor \frac{1}{2} n \log n \right\rfloor, \tag{3}$$

Date: Versão de 10 de maio de 2005.

onde escrevemos $\log n$ para $\log_e n$. (Aqui, consideramos todos os $\binom{\binom{n}{2}}{m}$ grafos n vértices e m arestas equiprováveis.) Estime P_n para n grande através de experimentos computacionais. [*Observação.* Sabe-se que $\lim_{n\to\infty} P_n$ existe. Neste exercício, queremos estimar este limite.] {*Data de entrega*: 26/3/2004 (O Panda estará de pé em breve para você entregar sua solução, que deve consistir de seu(s) programa(s) e um pequeno relatório com suas conclusões.)}

9. 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 $E(G)\setminus E(T)$ 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.) {*Data de entrega*: 30/3/2004}
10. 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*: 2/4/2004}
11. Escreva um programa que, dado um grafo G como entrada, determina os componentes conexos de G, os vértices de corte de G, e os blocos de G. Para descrever os componentes e os blocos de G, o seu programa pode simplesmente imprimir o conjunto de vértices presentes nestes componentes e blocos. O seu programa deve ter complexidade de tempo $O(n + m)$, onde, como de usual, n é o número de vértices em G e m é o número de arestas em G. Experimente executar *o seu programa* nestes grafos: g1.gr, g2.gr, g3.gr, e nos grafos `book("anna",10,0,15,20,1,1,0)` e `book("homer",561,0,1,24,1,1,0)` do Stanford GraphBase. (Note que os grafos g2.gr e g3.gr são um tanto grandes.) [**Adiado por 3 dias!** {***Data de entrega*: 16/4/2004**}]
12. Familiarize-se com a plataforma *Stanford GraphBase* de Knuth. Para tanto, leia, pelo menos em algum detalhe, os Capítulos 1 a 4 do livro *Stanford GraphBase*. Uma ótima fonte local é uma página de uma edição de mac328 mantida pelo professor Coelho. Finalmente, você deve entregar o seguinte: escreva um programa baseado no `queen.w` que gera *três* grafos interessantes. Você deve descrever os seus grafos em um pequeno texto, explicando a razão de sua escolha. Seu programa deve imprimir uma descrição de seus grafos, possivelmente salientando as suas características que você achou interessantes. (Entregue pelo Panda.) {*Data de entrega*: 20/4/2004}
13. Neste exercício, trataremos de árvores aleatórias. O objetivo é determinar experimentalmente a *altura esperada* de certas árvores aleatórias. Você deve executar seus experimentos para valores de n razoavelmente grandes, onde n é o número de vértices nas árvores aleatórias em questão.

(a) Seja $\mathcal{T}(n)$ a família das árvores sobre o conjunto de vértices $[n] = \{1, \ldots, n\}$. Considere $\mathcal{T}(n)$ como um espaço de probabilidade, atribuindo para todo elemento de $T \in \mathcal{T}(n)$ a mesma probabilidade. Considere o vértice 1 como raiz de todos os $T \in \mathcal{T}(n)$; note que, fazendo isso, podemos definir a *altura* $h(T)$ de T em relação à raiz 1. Estime experimentalmente o valor esperado de $h(T)$. [*Sugestão.* Para gerar um T, use a prova de Joyal para a fórmula de Cayley.] **[A entrega deste item fica adiada para o dia 30/4/2004]**

(b) Consideremos agora o grafo completo K_n sobre o conjunto de vértices $[n]$. Podemos gerar uma árvore aleatória com raiz 1 sobre $[n]$ executando uma busca aleatória em K_n a partir de 1 (considere a busca aleatória discutida em sala). Seja R_n uma tal árvore aleatória. Estime experimentalmente o valor esperado de $h(R_n)$. **Use a seguinte versão do Programa 18.10**:

```
void rfs(Graph G, Edge e)
  { link t; int v, w;
    RQput(e);
    while (!RQempty())
      if (pre[(e = RQget()).w] == -1)
        { link t;
          pre[e.w] = cnt++; st[e.w] = e.v;
          for (t = G->adj[e.w]; t != NULL; t = t->next)
            if (pre[v = t->v] == -1)
                RQput(EDGE(e.w, v));
        }
  }
```

O programa acima é como o Programa 18.8 do Sedgewick (apenas troquei os `QUEUE` por `RQ` (*random queue*)). Note que, basicamente, o Programa 18.9 está para o Programa 18.8, como o Programa 18.10 está para o programa acima.

{*Data de entrega*: 23/4/2004}

14. 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. Prove que existe uma orientação fortemente conexa de G se e só se G é conexo e não tem arestas de corte. {*Data de entrega*: 27/4/2004}
15. Seja $G = (V, E)$ um grafo dirigido, onde $V = \{v_1, \ldots, v_n\}$, e seja $A = (a_{ij})$ a matriz de adjacência para G com $a_{ij} = 1$ se $(i, j) \in E$ e $a_{ij} = 0$ se $(i, j) \notin E$. (Apenas para maior clareza, suponha que G não tem laços, de forma que $a_{ii} = 0$ para todo i.) O que é a entrada ij da matriz A^2? (Aqui A^2 é o quadrado usual da matriz A.) Em geral, o

que é a entrada ij da k-ésima potência A^k de A ($k \geq 0$)? {*Data de entrega*: 11/5/2004}

16. Escreva um algoritmo que, dado um DAG D, determina o comprimento máximo de um caminho dirigido em D (basta entregar um pseudocódigo em papel). {*Data de entrega*: 18/05/2004}
17. (i) Descreva o que ocorre quando executamos o algoritmo de Kosaraju em um DAG.
    (ii) Descreva o que ocorre quando executamos o algoritmo de Tarjan em um DAG.
    {*Data de entrega*: 21/5/2004}
18. 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 o 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*: 25/5/2004}
19. 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*: exercício recomendado, sem data de entrega}
20. Sejam G e c como no Exercício 19. 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*: exercício recomendado, sem data de entrega}
21. Suponha que os custos $c(e)$ ($e \in E(G)$) das arestas de um grafo G são todos inteiros, com $1 \leq c(e) \leq n = |V(G)|$. O quão rápido podemos encontrar uma árvore geradora mínima de (G, c)? {*Data de entrega*: 1/6/2004}
22. Estude as implementações do algoritmo de Kruskal e do algoritmo de (Jarník–)Prim (com heaps binários) em `miles_span`, do Stanford GraphBase. As estatísticas relatadas na Seção 71 daquele programa foram compiladas com uma máquina Sun Sparc 2 (bastante antiga!). Compile estatísticas para máquinas Intel (ou Sun) mais modernas. Faça uma pequena discussão com base nos dados que você obtiver. Se você quiser experimentar estas implementações em mais instâncias, vá em frente! {*Data de entrega*: 4/5/2004}