# EXERCÍCIOS DE ALGORITMOS EM GRAFOS
## 1o. SEMESTRE DE 2005

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
$$2^{\binom{n}{2}}/n! \leq g_n \leq 2^{\binom{n}{2}}. \tag{1}$$
{*Data de entrega*: 12/4/2005}
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* $G^c = \overline{G}$ 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 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 $\delta(G) \geq 2$.

---

*Date*: Versão de 17 de maio de 2007.

11. 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*: 12/4/2005}
12. (Proposição 1.2 de ISTG) Prove que um grafo $G$ é conexo se e somente se $\nabla(X) \neq \emptyset$ para todo subconjunto próprio e não-vazio $X$ de $V(G)$.
13. (E 1.39 de ISTG) Seja $G$ um grafo com $\delta(G) > (|V(G)|-2)/2$. Mostre que $G$ é conexo.
14. (E 1.40 de ISTG) Seja $G$ um grafo com $\delta(G) \geq 3$. Mostre que $G$ tem um circuito par.
15. Seja $G$ um grafo conexo com pelo menos dois vértices. Mostre que há dois vértices distintos $x$ e $y$ em $G$ tais que ambos $G - x$ e $G - y$ são conexos.
16. (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 $G$. {*Data de entrega*: 12/4/2005}
17. 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 $x$.) {*Data de entrega*: 12/4/2005}
18. 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 $G$.
19. 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.

20. Para cada inteiro $n > 0$, seja $P_n$ a probabilidade de o grafo aleatório com $n$ vértices e $m$ arestas ser conexo, onde

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

($\log n$ acima é na base e $= 2.718\ldots$). Naturalmente, 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 $L = \lim_{n\to\infty} P_n$ existe e $0 < L < 1$. Neste exercício, queremos estimar este limite.] {*Data de entrega*: 15/4/2005 (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.)}

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*: 19/4/2005}

22. 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 $G$ (os *blocos* de $G$ são os subgrafos de $G$ gerados pelas classes de equivalência da relação $e \sim f$ se e só se as arestas $e$ e $f$ são iguais ou pertencem a um mesmo circuito). 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.) (Entregue pelo Panda) {*Data de entrega*: 3/5/2005}

23. 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*: 3/5/2005}

24. 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. As árvores aleatórias que consideraremos são definidas assim:

▷ Considere o grafo completo $K_n$ sobre o conjunto de vértices $\{0, \ldots, n-1\}$. Podemos gerar uma árvore aleatória com raiz 0 sobre $\{0, \ldots, n-1\}$ executando uma busca aleatória em $K_n$ a partir de 0 (considere a busca aleatória discutida em sala). Seja $R_n$ uma tal árvore aleatória. Estime experimentalmente o valor esperado da altura $h(R_n)$ de $R_n$. **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*: 6/5/2005}

25. 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*: 10/5/2005}
26. 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*: 17/5/2005}
27. (*i*) Descreva o que ocorre quando executamos o algoritmo de Kosaraju em um DAG; mais precisamente, descreva como o algoritmo evolui ao processar o DAG (não descreva apenas a saída).
    (*ii*) Descreva o que ocorre quando executamos o algoritmo de Tarjan em um DAG.
    {*Data de entrega*: 20/5/2005}
28. 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*: 31/5/2005}

29. 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}
30. Sejam $G$ e $c$ como no Exercício 29. 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}
31. 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, isto é, uma orientação $\vec{G}$ tal que, para quaisquer $x$ e $y \in V(\vec{G})$, há um caminho dirigido de $x$ para $y$.
    {*Data de entrega*: exercício recomendado, sem data de entrega}
32. 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

$$p(v) \leq p(u) + c(u, v) \text{ para todo } (u, v) \in E(G). \tag{3}$$

    Ademais, dizemos que um arco $(u, v)$ é *justo* se vale a *igualdade* em (3). 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*: exercício recomendado, sem data de entrega}

33. 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 32). 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*: exercício recomendado, sem data de entrega}