Revista de Sistemas de Informação da FSMA
n. 9 (2012) pp. 3-15

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

Uma visão global sobre conteúdos e livros de referências de duas disciplinas de algoritmos

Sanderson L. Gonzaga de Oliveira

Resumo — **Neste artigo, descreve-se uma visão global sobre conteúdos e seus correspondentes livros de referências para ensino de duas disciplinas de algoritmos que podem ser oferecidas em cursos de graduação em Ciência da Computação, Engenharia da Computação, Sistemas de Informação, Licenciatura em Computação, Engenharia de Software e similares. Também são recomendadas obras complementares para essas disciplinas.**

Palavras chave — **ensino de algoritmos, ensino de estruturas de dados.**

I. INTRODUÇÃO

Algoritmos são um aspecto fundamental a ser aprendido por um estudante de computação. Não importa se ele deseja um emprego na indústria ou se deseja seguir carreira acadêmica: em ambos os casos ele terá que conviver (direta ou indiretamente) com a implementação de algoritmos para solução dos problemas computacionais que tiver que enfrentar. Isto posto, é fundamental que a formação destes alunos seja a mais completa possível. Para que o egresso de uma graduação em Ciência da Computação, Engenharia da Computação, Sistemas de Informação, Engenharia de Software e Licenciatura em Computação e similares tenha uma introdução adequada a algoritmos, sugerem-se os seguintes conteúdos de disciplinas de algoritmos:

(1) conteúdo básico sobre algoritmos, variáveis, constantes, tipos de variáveis, estruturas condicionais, estruturas de repetição, vetores, matrizes, registros, arquivos e ponteiros;

(2) tipos abstratos de dados, listas, filas, pilhas, filas de prioridade, recursividade, introdução à comparação de tempo de execução de algoritmos, algoritmos de ordenação, algoritmos de busca em vetor, *hashing* e tabelas *hash;*

(3) conceitos e propriedades matemáticas de árvores, árvores binárias e *n*-árias, percurso em árvores, árvores binárias de busca, árvores balanceadas e codificação de Huffman;

(4) *heaps*, ordenação externa, árvores rubro-negras, árvores B e suas variações, introdução à busca digital, introdução a grafos, algoritmos em grafos e algoritmos para casamento de cadeia de caracteres;

(5) exatidão de algoritmos, análise de algoritmos, notação assintótica, resolução de recorrências, paradigmas e técnicas de projetos de algoritmos, reduções polinomiais e NP-Completude.

A apresentação desses conteúdos em um curso de graduação é baseada em uma mescla das recomendações de currículos de referência da Sociedade Brasileira de Computação (SBC, 2012a) e Association for Computing Machinery (ACM, 2012) para os cursos citados. Sobre algoritmos e estruturas de dados, pode-se considerar que os currículos de referência dessas entidades são fundamentados em currículos da década de 1970 e 1980, principalmente baseados em autores como Knuth, Aho, Hopcroft e Ullman. Isso porque esses autores contribuíram para divulgar e consolidar diversos dos tópicos citados. Por outro lado, a adequação do processo de ensino às necessidades do mercado de computação (incluindo neste termo, as carreiras científicas) é garantida pela atualização constante que essas entidades realizam nesses currículos.

As recomendações da inclusão de conteúdos por essas entidades já pode ser considerado suficiente para a inclusão dos tópicos em um curso de graduação. Entretanto, neste texto, os tópicos relacionados aos itens 4 e 5 são especificados e detalhados. Por isso, justificativas para a inclusão de cada conteúdo são discutidas adiante.

S. L. Gonzaga de Oliveira é professor do Departamento de Ciência da Computação da Universidade Federal de Lavras, 35-3829-1545/1945 (e-mail: sanderson@dcc.ufla.br).

Consta no Currículo de Referência da SBC (2012a), que "a forma como esse conteúdo será trabalhado no curso, estabelecida pelo projeto didático-pedagógico, é tão ou mais importante que a simples distribuição de matérias em disciplinas". Ainda, nesse documento, é descrito que "dada a forte interdependência entre grade curricular e projeto didático-pedagógico, esses elementos devem ser desenvolvidos conjuntamente". No curso de graduação, sugere-se que a apresentação dos conteúdos siga a ordem dada, mesmo que de forma aproximada. Dependendo do nível de profundidade em que cada tópico é abordado, pode ser que nem todos os tópicos possam ser apresentados em uma disciplina semestral conforme apresentado. Por exemplo, podem-se selecionar os tópicos dos itens 2 e 3 que forem considerados mais pertinentes para serem apresentados em uma disciplina de um semestre ou podem-se apresentar os tópicos em dois semestres. Nesse último caso, as disciplinas podem dar ênfase nas implementações dos algoritmos.

É possível que grande parte desses itens sejam cobertos em cursos de graduação em Ciência da Computação. Por outro lado, cursos de graduação em Engenharia da Computação, Sistemas de Informação, Licenciatura em Computação, Engenharia de Software e similares podem cobrir boa parte desse conteúdo, dependendo de como se idealizou o perfil do egresso. Entende-se aqui que o egresso dos cursos em discussão não deve ter conhecimentos superficiais em algoritmos e estruturas de dados porque, claramente, constam no núcleo da implementação de um software. Como exemplo, um engenheiro de *software* deve conhecer em profundidade sobre análise de requisitos, padrões de projeto, projeto e arquitetura de sistemas, gerência de projetos, testes, qualidade de *software*, modelagem de negócios etc. Mas é importante entender que, para um engenheiro de *software*, o conhecimento de algoritmos e estruturas de dados também é relevante. Isso porque se espera que o engenheiro de *software* venha a trabalhar, principalmente, na "linha de produção de *software*", projetando sistemas e gerenciando equipes e, para isso, precisa ter conhecimento de *como* os técnicos desenvolvem o sistema para que seja de *boa* qualidade. O mesmo se pode afirmar sobre um curso de Sistemas de Informação, no qual se visa que o egresso venha a atuar na média ou alta gerência da empresa. Já o foco de um egresso de um curso de Ciência da Computação é de forma que possa atuar em quaisquer atividades em Tecnologia, Informação e Comunicação nas empresas.

Estas definições de carreiras voltadas para o mercado não impedem que exista a necessidade fundamental de um futuro pesquisador entender corretamente a ciência e a aplicação de algoritmos. Afinal, a carreira científica pressupõe a capacidade de resolver problemas computacionais para resolver problemas científicos complexos e é impossível resolver ambos sem os fundamentos básicos da computação, entre os quais incluem-se os algoritmos. É razoável supor que, em geral, um egresso que segue a carreira acadêmica tem condições de ser um bom profissional no mercado não acadêmico. As vantagens de apresentar os conteúdos para profissionais não acadêmicos já foi comentada e ainda será discutida adiante.

Os tópicos comentados nesta introdução e que serão detalhados no resto deste texto podem ser entendidos como os conceitos básicos necessários para a formação de um bom profissional em computação. Entretanto, a lista de tópicos não é exaustiva, podendo ser oferecidos tópicos adicionais, conforme a necessidade, percebida pelo corpo docente, do mercado em que o egresso atuará. Ainda, a compreensão pelos alunos dos conteúdos dos tópicos dos itens 4 e 5 pode ser importante ao contribuir para o entendimento de disciplinas subseqüentes, como:

- Otimização, com conteúdo provável sendo heurísticas, programação linear, *simulated annealing*, busca tabu, algoritmos genéticos, GRASP etc.;
- Inteligência Artificial, com conteúdo provável sendo noções de algoritmos de busca, métodos de MiniMax, métodos probabilísticos (redes bayesianas, modelos de Markov, teoria da decisão), aprendizado de máquina, redes neurais, lógica *fuzzy*, sistema especialistas etc.

Os tópicos listados podem ser avaliados nas provas anuais do Exame Nacional para Ingresso na Pós-Graduação em Computação (POSCOMP)[1]. Nota-se que o POSCOMP também pode direcionar os conteúdos que os cursos de graduação devem abordar para fornecer uma formação básica adequada a seus egressos que pretendem seguir a carreira acadêmica.

Na tabela 1, mostram-se as quantidades de questões relacionadas aos itens 2, 3, 4 e 5, para as cinco últimas provas do POSCOMP. Na tabela 1, mostra-se também a quantidade de questões relacionadas a possíveis disciplinas posteriores, em Otimização e Inteligência Artificial, sendo que as disciplinas 2, 3, 4 e 5 abordam tópicos necessários para o entendimento adequado dessas disciplinas. Para a contagem de questões mostrada na tabela 1, questões sobre grafos não foram incluídas porque foram consideradas como sendo lecionadas em uma disciplina de Matemática Discreta ou Teoria dos Grafos.

Com as sugestões apresentadas neste presente texto, não se tem a pretensão de esgotar o tema e, naturalmente, cada grupo docente busca adequar-se às realidades e características locais. Como já comentado, os conteúdos e seus níveis de profundidade em um curso de graduação dependem do perfil do profissional definido pelo corpo docente. Com isso, se o corpo docente deseja que o egresso esteja preparado para o ingresso em uma pós-graduação com bom conceito, os tópicos sobre algoritmos citados nos itens 2 a 5 podem ser considerados importantes.

[1] Para saber mais sobre o POSCOMP, sugere-se a leitura da página web da SBC (2012b).

Tabela 1: Quantidade de questões no POSCOMP relacionadas às disciplinas 2, 3, 4, 5 e suas disciplinas posteriores (Otimização e Inteligência Artificial – consideradas na coluna 6) para as cinco últimas provas do POSCOMP.

Ano	Disciplina				Percentual em relação ao número de questões da prova
	2 e 3	4	5	6	
2011	5	1	3	3	~13
2010	2	2	8	1	~17
2009	5	3	1	1	~13
2008	2	0	5	1	10
2007	3	0	4	3	10

Entre dezenas de disciplinas de graduação, os conteúdos citados nos itens 2, 3, 4 e 5 compuseram entre 10% a 17% nas provas POSCOMP dos últimos cinco anos, sendo que há dezenas de disciplinas em um curso de graduação. Podemos considerar que a cobrança nessas disciplinas tem sido proporcional às demais disciplinas de um curso de Ciência da Computação na prova POSCOMP.

Ainda, o conhecimento das técnicas fundamentais sobre algoritmos é relevante para os profissionais não acadêmicos. Isso se faz importante para que desenvolvam sistemas de informação de boa qualidade. Como exemplo, os sistemas de informação não devem apresentar erros ou problemas decorrentes de baixa eficiência. Esses problemas podem ser evitados se a equipe de desenvolvimento tiver conhecimento de técnicas e métodos já consolidados e bastante difundidos na área. Nesse sentido, não é incomum ocorrerem sistemas de informação com erros ou ineficientes. Esses sistemas de informação poderiam ter sido desenvolvidos de forma ainda melhor se os desenvolvedores tivessem sido treinados com os conceitos básicos sobre algoritmos citados nos itens 2 a 5.

Pode-se supor que o Brasil tenha potencial para se tornar uma potência mundial na produção de tecnologias computacionais e sistemas de informação e não ser, apenas, um consumidor de tecnologia da informação. Há exemplos atuais importantes em que excelentes ideias surgem na Universidade: claramente, uma formação apropriada deve ser fomentada.

O objetivo fundamental deste artigo é, então, expor informações para fundamentar uma discussão sobre o tema. Para tanto, este artigo está organizado da seguinte maneira: na seção II, abordam-se livros importantes para referências bibliográficas para os tópicos listados nos itens 4 e 5. Nas seções III e IV, abordam-se conteúdos possíveis para os itens 4 e 5, respectivamente. Em cada subseção das seções III e IV, há comentários sobre cada tópico, seus possíveis correspondentes livros de referências e os motivos pelo quais esses tópicos são importantes na formação básica em algoritmos. Na última subseção de cada seção III e IV, resumem-se as sugestões de livros-texto para cada tópico da seção e sugerem-se os livros-textos e complementares. As considerações finais são dadas na seção V.

II. LIVROS IMPORTANTES PARA REFERÊNCIA BIBLIOGRÁFICA PARA OS TÓPICOS LISTADOS NOS ITENS 4 E 5

Neste texto, são listados possíveis livros de referências para os tópicos listados nos itens 4 e 5. Para os conteúdos listados nos itens 4 e 5, a segunda edição do livro de Cormen et al. (2001) tem sido bastante utilizada no Brasil. Há uma versão em português dessa segunda edição, Cormen et al. (2002), mas tem sido menos utilizado que sua versão em inglês, aparentemente, por algumas questões na tradução.

Neste texto, será comentada a terceira edição dessa obra: Cormen et al. (2009). A obra de Cormen, Leiserson, Rivest e Stein (CLRS) tem sido a mais empregada nas melhores instituições do Brasil e do mundo no ensino de algoritmos porque mantém um equilíbrio entre *rigor matemático*, característica fundamental em cursos exigentes e *didática*, sendo que os autores têm o dom de escrever de forma precisa, clara e objetiva.

Além disso, a abrangência é umas das grandes vantagens dessa obra: os autores apresentam a maioria dos principais problemas computacionais, seus algoritmos e estruturas de dados. A obra desses autores tem sido listada como uma das mais citadas na Ciência da Computação por muitos anos. Ainda, desde a primeira edição, em 1990, reimpressões semestrais ou anuais têm sido lançadas, cada uma corrigindo os erros de versões anteriores, muitos indicados pelos milhares de leitores.

Com isso, ao se ler a obra de CLRS, tem-se bastante confiança de que todos os assuntos abordados estão corretos e tem-se uma base adequada sobre algoritmos. Ao final de cada um dos 35 capítulos, os autores incluíram descrições buscando fornecer ao leitor em quais publicações se pode encontrar o caminho adequado para se ter conhecimento do estado da arte nos tópicos abordados.

Por essas características, a obra de CLRS tem cada vez mais credibilidade na comunidade acadêmica e científica. Com isso, há professores que não pensam em outro livro-texto nessas disciplinas de algoritmos além da obra de CLRS. De fato, na média, a obra de CLRS pode ser considerada a mais adequada para uma introdução a algoritmos.

Entretanto, apesar da obra de CLRS ser excelente, procurando-se na rede, encontram-se algumas críticas como o pequeno número de exemplos e a falta de mais exercícios resolvidos. Além disso, há tópicos importantes, como no caso da ordenação externa, que não são abordados na obra. Por algum motivo, que não cabe aqui ser discutido, os autores não

puderam se aprofundar ainda mais (nas já 1292 páginas da terceira edição) em alguns tópicos ou incluir tópicos relevantes sobre algoritmos.

Ainda, apesar da excelente qualidade das apresentações dos tópicos, há livros que descrevem tópicos específicos de forma ainda melhor que em CLRS (2009), especialmente, em relação à didática, rigor formal, quantidade de exemplos e exercícios resolvidos. São desses livros e tópicos específicos que este texto busca citar a seguir.

Outro conjunto de obras clássicas que deve ser mencionado são os livros de Knuth (1997, 1981, 1998, 2001). Os assuntos abordados são esgotados pelo autor. Os assuntos descritos pelo autor não são triviais, mas, mesmo sendo assuntos complicados e abrangentes, o autor sabe escolher as palavras adequadas para ser sucinto e descrever apropriadamente os conceitos importantes sobre algoritmos e estruturas de dados.

Knuth realiza sua obra com essas características: com linguagem clara e precisa, sem excesso de tecnicidades, ele consegue abordar os assuntos com a profundidade adequada e ainda consegue ser engraçado, tendo incluído comentários bem humorados, o que faz com que o estudo da Ciência da Computação seja divertido com os livros de Knuth. Espero que o leitor tenha a oportunidade de estudar com as suas obras e divertir-se tanto quanto o responsável por estas notas. Apesar disso, há professores que não consideram os textos de Knuth didáticos o suficiente para serem utilizados na graduação. Entretanto, há tópicos que são melhor descritos nas obras de Knuth que em outros livros. Com isso, as obras de Knuth podem ser utilizadas na graduação em determinados tópicos.

Essas obras são ótimas para as disciplinas compostas pelos itens 4 e 5, discutidas neste texto. Entretanto, há outras obras que, pela forma com que foi apresentado determinado tópico, merecem ser consideradas para serem incluídas como referência principal ou complementar nessas disciplinas, sendo citadas a seguir neste artigo.

III. CONTEÚDO POSSÍVEL PARA UMA DISCIPLINA DE ALGORITMOS COMPOSTA PELO ITEM 4

Para uma disciplina de algoritmos composta pelo item 4 em um dos cursos de graduação em discussão, podem ser abordados os tópicos *heaps*, ordenação externa, árvores rubro-negras, árvores B e suas variações, introdução à busca digital, introdução a grafos, algoritmos em grafos e algoritmos para casamento de cadeia de caracteres. Esses conteúdos são comentados a seguir, bem como são listados os correspondentes livros de referências sugeridos para uso em cursos de graduação.

A. Heaps

Uma excelente maneira de se implementar filas por prioridades é por *heaps*. Williams (1964) desenvolveu esta estrutura de dados para o algoritmo *heapsort*. Não se deve confundir uma estrutura de dados *heap* com o termo *heap* em relação à alocação dinâmica de memória.

Estruturas de dados *heaps* são fundamentais em diversos algoritmos. Como exemplos, *heaps* são utilizadas em algoritmos em grafos (veja a subseção *F* desta seção). *Heaps* podem ser utilizadas na implementação do algoritmo de Prim para encontrar a árvore geradora mínima. Os algoritmos de busca A* (Hart et al., 1968) e SMA* (Russel, 1992) também utilizam filas de prioridades que podem ser implementadas por *heaps*.

O conhecimento de estruturas de dados básicas, como a *heap*, também é importante para desenvolvedores de protocolos de redes, por exemplo. Exemplo de utilização de *heaps* é no algoritmo de Dijkstra (1959), que por sua vez é base para protocolos de roteamento por estado de enlace. *Heaps* também podem ser utilizadas para gerenciamento de recursos como largura de banda em uma linha de transmissão (como conexões VoIP) em roteamento de redes. Muitos protocolos para redes locais (LAN - *local area networks*) utilizam filas de prioridade, que podem ser implementadas por *heaps*, na camada de acesso ao meio (MAC - *media access control*), em que VoIP e IPTV são exemplos. Para introdução a esses assuntos, veja livros de redes de computadores, como Tanenbaum (2003) ou Kurose e Ross (2006).

Para estudo das estruturas de dados *heap*, sugerem-se as seguintes obras:

- no sexto capítulo, CLRS (2009) apresentam didaticamente o *heapsort*, *heaps* e filas de prioridades,
- listas de prioridades e *heaps* são descritas objetivamente no sexto capítulo de Szwarcfiter e Markenzon (2009),
- uma apresentação objetiva e precisa de filas de prioridades e do *heapsort* é dada no capítulo 9 de Sedgewick (1998) e
- uma quarta abordagem que merece destaque por causa da boa didática é a subseção 4.3.1 de Skiena (2008), que é dedicada a *heaps*.

B. Ordenação externa

Considere que se deseja ordenar um conjunto de elementos. Pode ocorrer que em cursos de graduação em discussão, apresentam-se os algoritmos para esse problema em uma segunda disciplina de algoritmos. Exemplos desses algoritmos são ordenação por bolha, por inserção, por seleção, *quicksort*, *heapsort* e *mergesort*. Para se utilizar um desses algoritmos, todos os dados necessitam estar na memória principal. Quando a memória principal não comporta todos os dados a serem ordenados, um algoritmo de *ordenação externa* deve ser utilizado. Nesses algoritmos, somente uma parte dos dados são carregados para a memória principal a cada fase, enquanto outra parte permanece em uma memória dita *externa* (ou secundária), em que o disco rígido é um exemplo. Neste contexto, aqueles algoritmos podem ser chamados de algoritmos de ordenação interna.

Para um curso de Sistemas de Informação, no Currículo de Referência da SBC (2012a), é recomendado que se aborde com profundidade uma disciplina Pesquisa e Ordenação, na qual conste "algoritmos para pesquisa e *ordenação em memória* principal e *secundária*". Para um curso de Licenciatura em Computação, a recomendação é que o conteúdo seja apresentado com abrangência.

Algoritmos de ordenação externa podem ser de fundamental importância para desenvolvedores que venham a trabalhar com arquivos de tamanho grande, algo que tem se tornado cada vez mais comum nas organizações que priorizam o recolhimento das informações transacionais cotidianas. Também é importante apresentar algoritmos básicos de ordenação externa na graduação para que os alunos notem como algoritmos simples podem ser projetados com ideias engenhosas, mesmo para um problema que poderia parecer, a princípio, complexo. Outro benefício de se apresentar este conteúdo é o aprendizado de técnicas de soluções de problemas que podem ser extrapoladas pelos alunos para utilização em problemas reais que venham a enfrentar, tanto em uma carreira acadêmica quanto na indústria. A ordenação externa é utilizada em aplicações empresariais importantes, como em transações de banco de dados. Como exemplo, considere que um usuário solicita um subconjunto de colunas de uma tabela de determinado banco de dados. A solicitação é retornada em um arquivo. Nesse arquivo, pode-se ter registros duplicados e um algoritmo de ordenação externa pode ser utilizado para excluir os registros duplicados, antes de serem transmitidos ao usuário.

Alguns algoritmos básicos para a resolução do problema de ordenação externa que podem ser abordados em graduação são: intercalação balanceada de múltiplos caminhos, intercalação polifásica de múltiplos caminhos e seleção por substituição. Para este tópico, sugerem-se as seguintes obras:

- Ziviani (2011) apresenta uma boa descrição desses algoritmos em sua seção 4.2;
- uma apresentação ainda melhor é dada em Sedgewick (1998), na seção 11.3.

Outro algoritmo bastante simples e interessante para ser apresentado é a seleção natural, que apresenta uma pequena, mas significativa, alteração em relação ao seleção por substituição. É provável que somente Knuth (1998) tinha abordado este algoritmo em livro até 2011.

C. Árvores rubro-negras

Uma árvore rubro-negra é um tipo de árvore binária de busca aproximadamente balanceada. Enquanto uma árvore de busca binária de busca (que é apresentada, provavelmente, em uma segunda disciplina de algoritmos) apresenta $O(h)$ em suas operações básicas, em que h é a altura da árvore (consequentemente, a busca pode ser linear em árvores degeneradas); uma árvore rubro-negra apresenta $O(\lg n)$, em que n é o número de nodos da árvore, no pior caso, em suas operações básicas.

Pode-se dizer que as árvores rubro-negras são similares, em desempenho e características fundamentais, às árvores AVL, que são amplamente estudadas. Entretanto, o rebalanceamento de árvores rubro-negras é mais eficiente que o das árvores AVL. Por isso, justifica-se ensinar as árvores rubro-negras como forma eficiente de se implementar árvores balanceadas, inclusive para se implementar filas de prioridades. Por exemplo, as árvores rubro-negras são implementadas no Completely Fair Scheduler, que é o escalonador do núcleo do Linux a partir da versão 2.6.23 (veja, por exemplo, Jones, 2009, para detalhes).

Em outro exemplo, as árvores rubro-negras podem ser utilizadas em aplicações de tempo real por causa de seu desempenho. As árvores rubro-negras também podem ser representadas por árvores 2-4, que é um tipo de árvore B (veja a próxima subseção).

Para este tópico, sugerem-se as seguintes obras:

- as árvores rubro-negras são apresentadas de forma excelente em CLRS (2009), no capítulo 13,
- Szwarcfiter e Markenzon (2009) também apresentam uma boa descrição sobre árvores rubro-negras na subseção 5.4 e, ainda
- as árvores rubro-negras são muito bem descritas na seção 13.4 de Sedgewick (1998).

D. Árvores B e suas variações

Em uma árvore B, mantêm-se os dados ordenados de forma que acesso sequencial, inserções e exclusões são realizadas em tempo logarítmico. As variações dessas árvores são importantes nas implementações de sistemas gerenciadores de bancos de dados e nos sistemas de arquivos de sistemas operacionais, como exemplos de suas aplicações. No Currículo de Referência da SBC (2012a), é recomendado que se abordem as árvores B e B+ com profundidade para os cursos de Sistemas de Informação e Licenciatura em Computação.

Para este tópico, sugerem-se as seguintes obras:

- CLRS (2009) apresentam os conceitos da árvore B original, suas operações e seus pseudo-códigos, sendo as apresentações dadas de forma detalhada;
- uma abordagem adequadamente didática e precisa de árvores B e suas variações, árvores B* e árvores B+, é feita na subseção 7.1 de Drozdek (2002) e
- uma descrição objetiva e clara de árvores B também é dada em Sedgewick (1998) na seção 16.3, em que códigos em C++ são apresentados.

E. Introdução à busca digital: tries e árvore Patricia

As árvores básicas para busca digital são simples e adequadas para serem apresentadas em nível de graduação. Uma *trie* (de *retrieval*) é uma árvore em que os dados permanecem ordenados.

Uma *trie* é utilizada para armazenar conteúdos em que as chaves são, geralmente, *strings*. *Tries* são muito eficientes em aplicações de busca de chaves, como dicionários, por exemplo. Seu tempo de busca é proporcional ao tamanho da chave e ainda permite ajudar na busca de prefixos ou palavras similares, o que é muito útil em aparelhos com interface de teclado limitada, como celulares e outros dispositivos portáteis. Uma abordagem didática de *tries* é apresentada na subseção 7.2 de Drozdek (2002).

Numa árvore Patricia, um tipo especial de *trie*, cada nodo interno contém o índice do *bit* mais significativo a ser verificado para decidir para qual ramo seguir. Nesta árvore, percorrem-se os nodos de acordo com os *bits* da chave de busca e não de acordo com o resultado da comparação da chave inteira. *Strings*, conjunto de inteiros como endereços do protocolo da internet (em roteamento IP) ou sequências arbitrárias gerais de objetos em ordem lexicográfica são exemplos de chaves que podem ser utilizadas.

Para esses tópicos, sugerem-se as seguintes obras:

- introdução à busca digital, *tries* e árvore Patricia são muito bem descritas no capítulo 9 de Szwarcfiter e Markenzon (2009),
- (subseção 5.4 de) Ziviani (2011) por causa da apresentação didática e
- a descrição mais detalhada pode ser a apresentada por Sedgewick (1998) nas seções 15.2 e 15.3.

F. Algoritmos em grafos

O leitor pode observar que no Currículo de Referência da SBC (2012a), recomenda-se que se abordem noções em algoritmos em grafos em um curso de Licenciatura em Computação. Em relação a grafos, como os problemas básicos e seus algoritmos, podem ser citados:

- percorrer os vértices de uma grafo (busca em profundidade e busca em largura),
- árvore geradora mínima (algoritmos de Prim, Kruskal e Boruvka),
- caminho mínimo de um vértice para os demais (algoritmos de Dijkstra e Bellman-Ford),
- caminho mínimo entre todos os vértices (algoritmo de Floyd-Warshall) e
- fluxo máximo (algorimo de Ford-Fulkerson), um problema de fluxo em redes.

Percorrer vértices de um grafo ou buscar um determinado vértice é relevante em diversas situações, em que aplicações da Internet são exemplos, já que roteadores podem ser modelados como vértices e suas conexões as arestas. Outro exemplo é um jogo de computador, em que os jogadores são vértices e o campo de visão entre eles as arestas, um próximo "ataque" pode ser dado pela busca em largura. A busca em profundidade, projetado por *backtracking* (veja técnicas de construção de algoritmos, a seguir), pode ser um dos métodos de busca recursiva mais simples.

Considere que um método recebe um grafo qualquer e deve retornar *uma árvore* (um grafo acíclico) que contenha todos os vértices, sendo que o custo de percorrer todos os vértices é *mínimo*. Essa árvore pode ser chamada de *árvore geradora mínima*. Por exemplo, o problema de encontrar a árvore geradora mínima é comum em aplicações em Engenharia Elétrica.

Para ser apresentado aos alunos, o problema do caminho mínimo pode ser associado ao problema de roteamento em redes, por exemplo, na Internet. Sendo o algoritmo de Dijkstra, já comentado, base para protocolos de roteamento de redes por estado de enlace, este algoritmo é um exemplo em que o professor tem uma boa oportunidade de mostrar a importância de algoritmos que são executados no dia a dia pelos estudantes. Exemplos de protocolos de roteamento por estado de enlace incluem o Intermediate-System-to-Intermediate-System (IS-IS) e o Open Shortest Path First (OSPF). Para saber mais sobre o IS-IS, veja Tanenbaum (2007, p. 389) ou a Request for Comments (RFC) 1142, em http://tools.ietf.org/html/rfc1142. Para saber mais sobre o OSPF, veja Tanenbaum (2007, p. 483-488), Kurose e Ross (2006, p. 255-256) e a RFC 2328, em http://tools.ietf.org/html/rfc2328. As atualizações para IPv6 são especificadas na versão 3 do OSPF na RFC 5340 de 2008, em http://tools.ietf.org/html/rfc5340.

O problema de fluxo máximo é associado ao transporte de material de uma fonte para um sumidouro por uma rede (ou grafo). O uso do termo *rede* em vez de *grafo* é por causa das primeiras publicações sobre o tópico na década de 1950. O problema de fluxo máximo é base para aplicações diversas, como em segmentação de imagens e programação de rotas de linhas aéreas. Um exemplo simples que pode ser apresentado aos alunos e pode ser facilmente encontrado na rede é a aplicação de Barnett et al. (2007), que utilizaram conceitos de fluxo máximo na disposição de sensores em redes de vias para segurança urbana.

Há outros problemas, com seus algoritmos correspondentes, que poderiam ser abordados, mas podem formar conteúdo demasiado em uma disciplina de 60 horas (ou um pouco mais) em que, ainda, outros tópicos são abordados além dos algoritmos em grafos. Além disso, para se abordar algoritmos em grafos, precisam-se apresentar uma introdução a grafos, suas terminologias e formas de representação básicas: matriz de adjacências, listas de adjacências e matriz de incidências.

Para este tópico, sugerem-se as obras:

- o livro mais indicado para esses assuntos é CLRS (2009), pois as descrições são apresentadas de maneira didática, precisa e objetiva; em geral, CLRS (2009) dedicaram um capítulo para cada um dos problemas e seus algoritmos correspondentes, em que as descrições são didáticas e os algoritmos são bem apresentados e exemplificados;

- Sedgewick (2002) é um livro excelente, específico sobre algoritmos em grafos, em que o autor se aprofunda em cada algoritmo e apresenta-os na linguagem C++;
- outra descrição didática dos algoritmos deste tópico está na obra de Skiena (2008), nos capítulos 5, 6 e 11.

G. Casamento de cadeia de caracteres (string searching algorithms)

Nesses algoritmos, o objetivo é encontrar as ocorrências de um padrão (*string*) em um texto (*string* maior). Algumas aplicações desses algoritmos são em edição de texto, estudo de sequência de DNAs em biologia computacional e em recuperação da informação.

Para este tópico, sugerem-se as seguintes obras:

- CLRS (2009) apresentam boa descrição de algoritmos para este problema no capítulo 32;
- abordagens objetivas e precisas também são encontradas no capítulo 10 de Szwarcfiter e Markenzon (2009) e
- no capítulo 8 de Ziviani (2011).

H. Tabela com conteúdos e sugestões de livros-texto

Na tabela 2, resumem-se as sugestões de livros-texto para cada tópico que pode ser abordado em uma disciplina de algoritmos composta pelo item 4. Recomendam-se CLRS (2009) e Sedgewick (1998) como os principais livros-texto a serem utilizados.

Em sendo necessário utilizar três livros-texto em uma disciplina de graduação, como terceiro livro-texto, recomenda-se Drozdek (2002) por causa de sua descrição de árvore B e suas variações. Recomenda-se que Knuth (1998) seja a principal obra empregada como leitura complementar nessa disciplina.

Tabela 2: Recomendações de livro-texto e livros para leituras complementares para tópicos que podem ser abordados em uma disciplina de algoritmos composta pelo item 4.

Tópico	**Sugestão como principal livro-texto**	**Sugestão como livro-texto auxiliar**	**Segunda sugestão como livro-texto auxiliar**
Heaps	CLRS (2009)	Sedgewick (1998)	Szwarcfiter e Markenzon (2009)
Ordenação externa	Knuth (1998)	Sedgewick (1998)	Ziviani (2011)
Árvores rubro-negras	CLRS (2009)	Sedgewick (1998)	Szwarcfiter e Markenzon (2009)
Árvores B e suas variações	Drozdek (2002)	CLRS (2009)	Sedgewick (1998)
Introdução à busca digital	Sedgewick (1998)	Szwarcfiter e Markenzon (2009)	Ziviani (2011)
Introdução a grafos e algoritmos em grafos	CLRS (2009)	Sedgewick (2002)	Skiena (2008)
Casamento de cadeia de caracteres	CLRS (2009)	Ziviani (2011)	Szwarcfiter e Markenzon (2009)

IV. CONTEÚDO POSSÍVEL PARA UMA DISCIPLINA DE ALGORITMOS COMPOSTA PELO ITEM 5

Para uma disciplina de algoritmos composta pelo item 5 em um dos cursos de graduação em discussão, podem ser abordados os tópicos: exatidão de algoritmos, fundamentos de análise de algoritmos, notação assintótica, resolução de recorrências, paradigmas e técnicas de projetos de algoritmos, reduções polinomiais e NP-Completude. Essa disciplina é chamada, algumas vezes, de Projeto e Análise de Algoritmos em cursos de graduação.

Para um curso em Licenciatura em Computação, no Currículo de referência da SBC (2012a), recomenda-se que se aborde essa disciplina com profundidade. Já para um curso de Sistemas de Informação, a recomendação é apresentar os conteúdos com abrangência e a ementa recomendada é: "Desenvolvimento de algoritmos. Técnicas de projeto de algoritmos eficientes. Análise assintótica de limites de complexidade. Técnicas de prova de cotas inferiores. Exemplos de análise de algoritmos iterativos e recursivos. Programação dinâmica. Algoritmos probabilísticos. Complexidade Pessimista, Complexidade Média. Complexidade Mínima do problema, Classes de problemas: P, NP, NP-Completa." No Currículo de Referência da ACM (2012), a apresentação desses conteúdos é essencial em um curso de Engenharia de Software. Os conteúdos sugeridos neste texto são comentados a seguir, bem como são listados os

correspondentes livros de referências mais adequados para uso em cursos de graduação.

A. Revisão de matemática básica para a análise de algoritmos

Para a análise de algoritmos, um revisão de matemática discreta pode ser importante, especialmente devido a alguma eventual fragilidade de conhecimentos matemáticos dos alunos egressos do ensino médio. Para este tópico, sugerem-se as seguintes obras: Gersting (2004), Rosen (2007), Graham et al. (1995) por causa das apresentações didáticas e aprofundadas e, no apêndice A de CLRS (2009), encontra-se uma boa revisão sobre somatórios.

B. Exatidão de algoritmos

Para se apresentar um novo algoritmo, é necessário provar que o algoritmo retorna a saída desejada para o domínio de entrada e que sempre termina. A apresentação acadêmica dessa noção é importante para o egresso de uma das graduações citadas, permitindo que o mesmo seja capaz de validar as aplicações que desenvolve tanto no âmbito acadêmico quando no âmbito comercial.

Boas apresentações deste tópico são o capítulo 5 de Harel (2004), a seção 1.3 de Skiena (2008), em que há também provas de exatidão de algoritmos por várias partes do livro. Ainda, há de se citar a apresentação didática de CLRS (2009), na seção 2.1 sobre laços invariantes.

C. Análise de algoritmos

Há de se considerar que não basta que um problema computacional seja resolvido. O algoritmo deve ser eficiente de forma que a solução execute em tempo hábil para a aplicação a qual foi concebido. Para saber sobre eficiência em algoritmos, há de se estudar a análise de algoritmos, pois o problema representado por um algoritmo assintoticamente lento não pode ser superado (Goodrich e Tamassia, 2004), mesmo se o desempenho do sistema de computação for melhorado de forma significativa. Consequentemente, realizar a análise de algoritmos pode ser tão importante quanto realizar um bom projeto de algoritmo (veja a subseção *F* a seguir).

Um embasamento formal em análise de algoritmos é tópico fundamental para os egressos dos cursos mencionados, principalmente para aqueles que desejam desenvolver aplicativos em áreas de computação intensiva, que são muito comuns na atividade acadêmica e em áreas específicas do desenvolvimento comercial. O conhecimento de análise de algoritmos se faz importante, principalmente, para os profissionais que trabalharão em atividades relacionadas a disciplinas como Otimização, Inteligência Computacional e Processamento Digital de Imagens.

As melhores abordagens em análise de algoritmos são as de:

- CLRS (2009) abordam didaticamente o tópico, no capítulo 2;
- Goodrich e Tamassia (2004), no capítulo 1, apresentam mais detalhes que os elencados por CLRS (2009) e merece ser uma obra elegida como livro-texto ou, ao menos, como leitura complementar em relação à análise de algoritmos;
- outro texto interessante é o capítulo 4 de Brassard e Bratley (1996) por causa da apresentação didática e
- um quarto texto que pode ser utilizado é a subseção 1.2.10 de Knuth (1997), que apresenta o tópico com objetividade.

Muitas vezes, o custo de uma operação em determinada estrutura de dados é alto em determinada fase, mas esse custo pode ser *amortizado* na quantidade de operações executadas *no limite*. Em relação à *análise amortizada*:

- a obra de CLRS (2009) é a mais indicada, em que os autores apresentam a análise agregada, o método da contabilidade, o método potencial e mostram exemplo da aplicação dos métodos em tabelas dinâmicas no capítulo 17;
- Goodrich e Tamassia (2004) apresentam descrição didática de análise amortizada na seção 1.5 e
- Skiena (2008) pode ser uma opção para descrição sobre análise amortizada também por causa de sua apresentação didática.

D. Notação assintótica

A notação assintótica é intrinsecamente relacionada à análise de algoritmos, sendo importante para a compreensão dos limites da eficiência de um algoritmo. Este tópico também deve ser apresentado adequadamente e em detalhes em cursos de graduação que apresentam a análise de algoritmos. Para este tópico, sugerem-se as obras:

- a melhor e mais completa abordagem sobre notação assintótica é a obra de Graham et al. (1995) no capítulo 9;
- a descrição das notações O, Ω, Θ, o e ω de CLRS (2009), no capítulo 3, também é interessante, em que há um equilíbrio entre rigor formal e didática;
- Goodrich e Tamassia (2004) também apresentam uma descrição didática sobre notação assintótica na seção 1.2.

E. Resolução de recorrências

Em geral, o tempo de execução de um algoritmo recursivo é expresso por uma recorrência. Por causa da ampla utilização de algoritmos recursivos, torna-se importante o estudo sistemático da solução de recorrências.

Muitas vezes, um algoritmo recursivo apresenta desempenho computacional pior ou consome mais memória que sua versão não-recursiva. O conhecimento de como analisar se um algoritmo recursivo é eficiente ou não pode ser relevante nas diversas aplicações em sistemas de informação, acadêmicas, técnicas ou científicas que os egressos dos cursos desenvolverão.

Para este tópico, sugerem-se as obras:

- uma das melhores abordagens de resolução de recorrências é dada por Brassard e Bratley (1996) na seção 2.3 por causa da profundidade e didática da apresentação;
- outra excelente abordagem é a de Graham et al. (1995) na seção 7.3 por causa da apresentação objetiva e precisa;
- há apresentações didáticas sobre o tópico nas seções 7.1, 7.2 e 7.3 de Rosen (2007) e
- na seção 2.3 de Gersting (2004).

F. Paradigmas e técnicas de projeto de algoritmos

Geralmente, um estudante iniciante aprende a criar algoritmos de maneira ingênua. Com alguma experiência na criação de algoritmos, deve ser introduzido ao estudante algumas técnicas de projeto de algoritmos já bastante difundidas e que consistem de conceitos bastante simples.

Entre os paradigmas e técnicas de projetos de algoritmos mais básicos e empregados, podem ser citados:

- noções de força bruta (ou busca exaustiva);
- *backtracking* (ou tentativa e erro), uma variação de força bruta;
- *branch-and-bound*, uma variação de *backtracking*;
- recursividade;
- balanceamento;
- abordagem incremental;
- divisão e conquista;
- algoritmos gulosos;
- programação dinâmica e
- algoritmos aproximativos.

O conhecimento desses paradigmas e técnicas básicos de projeto de algoritmos pode acelerar o desenvolvimento adequado de sistemas de informação, evitando-se uma implementação ingênua, por força bruta, por exemplo. Para poder comparar diferentes abordagens, a análise de complexidade, citada na seção anterior, deve ter sido apresentada.

Há autores que consideram que os livros-texto devem apresentar um paradigma ou técnica de projeto de algoritmos em um capítulo e, em seguida, apresentar exemplos da aplicação de tal projeto na solução de problemas clássicos, por um ou mais algoritmos conhecidos. Em geral, nesses livros, os autores intitulam o capítulo com o nome do paradigma ou técnica de projeto de algoritmos, apresentam um ou dois parágrafos descrevendo o paradigma e, em seguida, descrevem alguns algoritmos que exemplificam o paradigma ou a técnica. CLRS (2009) seguem este esquema. Para exemplos de algoritmos pelos paradigmas ou técnicas, uma boa opção é a obra de CLRS (2009). Os autores dedicaram o capítulo 4 ao paradigma divisão e conquista, o capítulo 15 à programação dinâmica e o capítulo 16 a algoritmos gulosos.

Por outro lado, para a apresentação de paradigmas e técnicas de projeto de algoritmos, sugerem-se as seguintes obras:

- a abordagem de Ziviani (2011) é didática e objetiva, em que o autor dedicou o segundo capítulo para descrever os paradigmas e técnicas de projetos de algoritmos;
- a abordagem de Goodrich e Tamassia (2004), no capítulo 5, também é interessante porque aprofunda-se um tanto a mais nos paradigmas e técnicas de projetos de algoritmos, em relação aos demais livros; entretanto, a abrangência dos paradigmas e técnicas de projetos de algoritmos em Goodrich e Tamassia (2004) é menor em relação a Ziviani (2011);
- outra abordagem interessante sobre paradigmas e técnicas de projetos de algoritmos é a apresentada por Aho et al. (1974) porque os autores também descrevem os temas de maneira objetiva e didática.

Tratando-se especificamente sobre recursão, o estudo do capítulo 5 de Sedgewick (1998) é importante por causa da apresentação aprofundada no assunto. Ainda neste livro, merecem ser estudadas a seção 5.2, sobre divisão e conquista e, a seção 5.3, sobre programação dinâmica também por causa das apresentações aprofundadas.

Especificamente sobre a análise de algoritmos por divisão e conquista, sugerem-se as seguintes obras:

- uma boa introdução é dada no capítulo 4 de CLRS (2009), em que o método da substituição, o método da árvore de recursão e o método mestre são apresentados didaticamente. Ainda, há uma introdução ao método de Akra-Bazzi;
- Goodrich e Tamassia (2004), na subseção 5.2.1, também apresentam introdução didática à análise de equações recursivas, em que descrevem o tempo de execução de algoritmos por divisão e conquista;
- uma terceira opção didática sobre resolução de relações de recorrência decorrentes de algoritmos por divisão e conquista é a seção 7.3 de Rosen (2007).

G. NP-Completude

Nos cursos de graduação em discussão, deve-se abordar com certo grau de profundidade a teoria sobre NP-Completude porque os alunos precisam entender que os problemas são mais importantes que os algoritmos que os resolvem. Há diversos (milhares se variações forem consideradas) problemas importantes catalogados para os quais não se sabe se podem ser desenvolvidos algoritmos polinomiais; tampouco foi provado que há um limite inferior não polinomial para um algoritmo resolver algum desses problemas.

Surpreendentemente para alguns, esses problemas são bastante comuns, principalmente em áreas da Engenharia. O problema do caixeiro viajante pode ser um dos mais conhecidos. Imagine que o graduando, futuro egresso, venha a trabalhar em uma transportadora e seu gerente lhe peça um sistema de informação para fornecer a rota de custo mínimo para que os caminhões da empresa levem os produtos. Ademais, existe a restrição de que cada caminhão deve passar uma só vez por cada cidade e retornar à cidade de origem. Se for implementado um algoritmo ingênuo, com algumas dezenas de cidades, o desempenho do programa já poderá ser bastante ruim e o gerente pode vir a considerar se o desenvolvedor é um bom profissional. Se esse desenvolvedor vier a pedir ao gerente mais poder computacional, o gerente pode gostar menos ainda. Isso está, similar e divertidamente, descrito no primeiro capítulo de Garey e Johnson (1979).

PxNP é considerado um dos problemas não resolvidos mais importantes na Ciência da Computação e Matemática. Por exemplo, mesmo em Otimização, é comum encontrar bons programadores que trabalham em problemas NP-Completo sem conhecer adequadamente o básico dessa teoria.

Garey e Johnson (1979) é o livro clássico sobre NP-Completude e uma das obras mais citadas em Ciência da Computação. Similarmente como apresentado por Garey e Johnson (1979), as classes desta teoria devem ser definidas formalmente em termos de um modelo de computação. Particularmente, a máquina de Turing pode ser um modelo adequado para esse objetivo.

O primeiro capítulo de Garey e Johnson (1979) é uma introdução didática à NP-Completude. No segundo capítulo, os autores formalizam as classes principais em termos da máquina de Turing. Exemplos de provas de NP-Completude são dados no capítulo 3. No capítulo 4, os autores mostram como se pode utilizar a NP-Completude para analisar problemas. No capítulo 5, os autores descrevem conceitos sobre "NP-Dificibilidade". No Capítulo 6, os autores apresentam garantias de desempenho para algoritmos aproximativos. No capítulo 7, os autores abordam a teoria "além" da NP-Completude. No apêndice, os autores listam os problemas NP-Completos conhecidos até o período em que o livro foi escrito.

Outra obra recomendada para estudo de NP-Completude é Sudkamp (2007) por causa da apresentação didática dos temas. Ainda, outra excelente descrição aprofundada em que as classes básicas da NP-Completude são definidas em termos de máquinas de Turing é a apresentada por Aho et al. (1974) no capítulo 10. Para uma abordagem didática sobre NP-Completude sem formalizar as classes básicas em termos de algum modelo de computação, recomenda-se o capítulo 34 de CLRS (2009).

Conteúdos importantes relacionados à NP-Completude são:

- conceitos básicos de reduções polinomiais, em que a descrição apresentada por Manber (1989) é adequadamente didática;
- provas de NP-Completude, em que CLRS (2009) e Manber (1989) apresentam provas de forma didática e simples;
- demonstração do teorema de Cook-Levin, em que as apresentações de Sudkamp (2007), Garey e Johnson (1979) e Aho et al. (1974) são as mas didáticas e formais.

H. Tabela com conteúdos e sugestões de livros-texto

Nas tabelas 3 e 4, resumem-se as sugestões de livro-texto para cada tópico descrito que pode ser abordado nesta seção. Na tabela 3, são listados os tópicos e os livros sugeridos para exatidão de algoritmos, revisão de matemática básica e fundamentos sobre análise de algoritmos, análise amortizada, notação assintótica e resolução de recorrências.

Tabela 3: Recomendações de livro-texto e obras complementares para uma primeira parde dos tópicos que podem ser abordado em uma disciplina de algoritmos composta pelos itens comentados na seção IV.

Tópico	**Sugestão como principal livro-texto**	**Sugestão como livro-texto auxiliar**	**Segunda sugestão como livro-texto auxiliar**
Exatidão de algoritmos	Harel (2004)	Skiena (2004)	CLRS (2009)
Revisão de matemática básica	Gersting (2004)	Rosen (2007)	Graham et al. (1995)
Fundamentos sobre análise de algoritmos	CLRS (2009)	Goodrich e Tamassia (2004)	Brassard e Bratley (1996)
Análise amortizada	CLRS (2009)	Goodrich e Tamassia (2004)	Skiena (2008)
Notação assintótica	CLRS (2009)	Graham et al. (1995)	Goodrich e Tamassia (2004)
Resolução de recorrências	Brassard e Bratley (1996)	Graham et al. (1995)	Rosen (2007)

Na tabela 4, são listados os tópicos e os livros sugeridos para paradigmas e técnicas de projeto de algoritmos, resolução de recorrências decorrentes de algoritmos por divisão e conquista e a teoria sobre NP-Completude. Em sendo necessário utilizar três livros-texto em uma disciplina de graduação, recomendam-se CLRS (2009), Goodrich e Tamassia (2004) e Ziviani (2011) como os principais livros-texto em uma disciplina de algoritmos abrangendo os tópicos comentados nesta seção. Ainda, recomendam-se como principais livros a serem utilizados como leitura complementar Garey e Johnson (1979), por causa da excelente descrição sobre NP-Completude e Graham et al. (1995), por causa da qualidade das descrições de revisão de matemática básica, notações assintóticas e resolução de recorrências.

Tabela 4: Recomendações de livro-texto e obras complementares para uma segunda parte de tópicos que podem ser abordados em uma disciplina de algoritmos composta pelos itens comentados na seção IV.

Tópico	**Sugestão como principal livro-texto**	**Sugestão como livro-texto auxiliar**	**Segunda sugestão como livro-texto auxiliar**
Paradigmas e técnicas de projetos de algoritmos	Ziviani (2011)	Goodrich e Tamassia (2004)	Aho et al. (1974)
Resolução de recorrências decorrentes de algoritmos por divisão e conquista	CLRS (2009)	Goodrich e Tamassia (2004)	Rosen (2007)
NP-Completude: teoria e apresentação das classes básicas	Garey e Johnson (1979)	Sudkamp (2007)	Aho et al. (1974)
NP-Completude: conceitos básicos de reduções polinomiais	Manber (1989)	Garey e Johnson (1979)	Sudkamp (2007)
Provas de NP-Completude	CLRS (2009)	Manber (1989)	Garey e Johnson (1979)
NP-Completude: prova do teorema de Cook-Levin	Sudkamp (2007)	Garey e Johnson (1979)	Aho et al. (1974)

V. CONSIDERAÇÕES FINAIS

Ao se recomendar a aquisição de livro-texto para disciplina de algoritmos composta pelos itens 4 e 5, de fato, CLRS (2009) é o mais indicado. Entretanto, esta obra pode não ser considerada a melhor opção de estudo para todos os tópicos descritos. Em tópicos específicos, há ótimas descrições em outros livros.

Na prática, há necessidade de se utilizar dois ou mais livros-texto em cada uma das duas disciplinas. Ainda, pode haver necessidade de serem utilizados mais de um livro em cada tópico abordado.

Também se recomenda o emprego das obras de Sedgewick (1998, 2002) para uma disciplina de algoritmo composta pelo item 4. O autor apresenta de forma detalhada e com implementações em C++ a maioria dos tópicos citados.

Para uma disciplina de algoritmos composta pelo item 4, a obra de Drozdek (2002) é opção adequada em tópicos específicos. Ainda, as obras de Szwarcfiter e Markenzon (2009) e Ziviani são opções adequadas para serem utilizadas nessa disciplina.

Também há opções ao se considerar maneiras específicas de apresentação ou a apresentação de tópicos específicos. Em relação à didática, sem exigências de complicações com formalismos, as obras de Ziviani (2005, 2011) apresentam vantagens. O autor domina a arte de apresentar um conteúdo de forma simples e objetiva. Uma grande vantagem das obras de Ziviani são as implementações em Pascal e C (Ziviani, 2011) e Java e C++ (Ziviani, 2005). As obras de Ziviani (2005, 2011) são bastante utilizadas nas graduações no Brasil.

Merecem destaque também as obras de Manber (1989) e Goodrich e Tamassia (2004). Particularmente, a obra de Goodrich e Tamassia (2004) pode ser utilizada como um dos livros-texto em uma disciplina de algoritmos, em que são apresentados os tópicos elencados no item 5. Aho et al. (1974), Garey e Johnson (1979), Brassard e Bratley (1996), Graham et al. (1995), Gersting (2004), Rosen (2007), Sudkamp (2007), Skiena (2008) e os quatro volumes de Knuth também devem ser considerados para fazerem parte da lista de livros de referências nas disciplinas de algoritmos discutidas neste texto.

O autor deste texto começou a criar notas de aulas desde que iniciou a lecionar essas disciplinas em graduações em Ciência da Computação e Sistemas de Informação. Os textos evoluíram para um livro e foram publicados em Gonzaga de Oliveira (2011), em que é descrita a maioria dos tópicos abordados nos itens 4 e 5. Este livro pode ser utilizado como leitura complementar para disciplinas de algoritmos compostas pelos itens 4 e 5.

Livros não citados podem não ser tão conhecidos, podem não ser muito utilizados entre os professores brasileiros ou são desconhecidos pelo autor. Ainda, pode ocorrer que livros não citados possam não ser tão adequados quanto outros livros para serem utilizados a determinados conteúdos. Isso porque, claramente, em cada livro, há o estilo do autor ou há uma busca de atender a determinado grupo de estudantes. Por

exemplo, há livros que não apresentam (quase) nenhuma equação porque isso pode afastar um grupo de leitores. Nesse exemplo, um autor pode explicar detalhada e longamente o tópico, com muitos exemplos, sem generalizações. Um livro em que não se aborda formalizações matemáticas pode ter um apelo comercial grande, ou seja, livros assim podem ser mais apreciados para um entendimento superficial do assunto em relação aos livros formais, isto é, que apresentam os conceitos em linguagem matemática, de forma abstrata e aprofundada. Isso depende do estilo e dos resultados esperados pelo autor.

Considera-se que ambas as abordagens são válidas. As abordagens *didáticas*, com bastante explicação em todos os assuntos, são importantes para uma primeira leitura. Por outro lado, saber mais detalhes sobre o assunto pode não ser possível nesses livros e uma segunda leitura pode ser enfadonha por causa da quantidade de texto inserido pelo autor. Já as abordagens *formais*, em que os tópicos são apresentados em linguagem matemática, de maneira abstrata e aprofundada, com generalizações e ainda busca-se melhorar o raciocínio lógico do aluno, podem ser difíceis de serem entendidos pelos alunos de graduação em uma primeira leitura. Esses livros podem ser utilizados após explicações didáticas em sala de aula. Por outro lado, nesses livros, adquire-se compreensão de detalhes importantes sobre os conceitos, fazendo com que sejam entendidos apropriadamente.

Dos livros citados como os três mais indicados para cada assunto, levou-se em consideração um *equilíbrio* entre a didática e rigor matemático. É nesse equilíbrio que um livro como o de CLRS (2009) se destaca.

As descrições deste texto, os tópicos sugeridos em disciplinas de algoritmos, seus livros de referências e os comentários em relação aos livros, representam a opinião do autor. Entretanto, o autor espera que possam servir como início de trabalho para professores em fase de elaboração de ementas e preparação dessas disciplinas.

VI. AGRADECIMENTOS

Agradecimentos à Fundação de Amparo à Pesquisa do Estado de Minas Gerais – FAPEMIG e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq pelo apoio financeiro a projetos do autor. O autor também agradece ao corpo editorial da Revista de Sistemas de Informação da Faculdade Salesiana Maria Auxiliadora: somente com as ideias e indicações do corpo editorial da revista é que o texto poderia alcançar o conteúdo atual. Agradece-se também aos professores Antonio Maria Pereira de Resende, Denilson Alves Pereira e Luiz Henrique Andrade Correia pelas revisões de trechos específicos.

VII. REFERÊNCIAS

1. A. V. Aho, J. E. Hopcroft, J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Reading: Addison-Wesley Publishing Company, 1974.
2. Association for Computing Machinery. Curricula Recommendations, 2012. Disponível em: http://www.acm.org/education/curricula-recommendations. Acessado em: 7 de março de 2012.
3. L. B. Anderson, R. J. Atwell, D. S. Barnett, R. L. Bovey, Application of the Maximum Flow Problem to Sensor Placement on Urban Road Networks for Homeland Security, Homeland Security Affairs. v. 3, n. 3, 2007. Disponível em: http://www.hsaj.org/?article=3.3.4. Acessado em: 25 de março de 2011.
4. G. Brassard, P. Bratley, *Fundamentals of Algorithmics*. Englewood Clifs: Prentice-Hall, 1996.
5. T. H. Cormen, C. E. Leiserson, R. L. Rivest, *Introduction to Algorithms*, Cambridge: The MIT Press, 1990.
6. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge: The MIT Press, 2001.
7. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Algoritmos: Teoria e Pratica*, Rio de Janeiro: Campus, 2002.
8. T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge: The MIT Press, 2009.
9. E. W. Dijkstra, A note on two problems in connexion with graphs. Numerische Mathematik, v. 1, p. 269–271, 1959.
10. A. Drozdek, *Estruturas de dados e algoritmos em C++*. São Paulo: Pioneira Thomson Learning, 2005.
11. M. R. Garey, D. S. Johnson, *Computers and intractability: A guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
12. J. L. Gersting, *Fundamentos Matemáticos para a Ciência da Computação: Um tratamento moderno de Matemática Discreta, 5a. ed*. Rio de Janeiro: LTC, 2004.
13. S. L. Gonzaga de Oliveira, *Algoritmos e seus fundamentos*, Lavras: Editora UFLA, 2011.
14. M. T. Goodrich, R. Tamassia, *Projeto de Algoritmos: Fundamentos, análise e exemplos da Internet*. Porto Alegre: Bookman, 2004.
15. R. L. Graham, D. E. Knuth, O. Patashnik, *Matemática Concreta: Fundamentos para a Ciência da Computação*, 2a. ed. Rio de Janeiro: LTC, 1995.
16. P. E. Hart, N. J. Nilsson, B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics SSC4* **4** (2): 100–107, 1968.
17. M. T. Jones, Inside the Linux 2.6 Completely Fair Scheduler: Providing fair access to CPUs since 2.6.23. developerWorks. Disponível em: http://www.ibm.com/developerworks/linux/library/l-completely-fair-scheduler. Acessado em: 25 de março de 2009.
18. D. E. Knuth, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms*, 2nd ed. Reading: Addison-Wesley Publishing Company, 1981.
19. D. E. Knuth, *The Art of Computer Programming, Volume 1, Fundamental Algorithms, 2nd ed.* Reading: Addison-Wesley Publishing Company, 1997.
20. D. E. Knuth, *The Art of Computer Programming, Volume 3, Sorting and Searching, 2nd. ed.* Reading: Addison-Wesley Publishing Company, 1998.
21. D. E. Knuth, The *Art of Computer Programming, Volume 4, Combinatorial Algorithm*s. Reading: Addison-Wesley Publishing Company, 2001.

22. J. E. Kurose, K. W. Ross, *Redes de Computadores e a Internet*. 3a. ed. São Paulo: Pearson, 2006.

23. U. Manber, *Introduction to Algorithms: A Creative Approach*. Reading: Addison-Wesley Publishing Company, 1989.

24. K. H. Rosen, *Matemática Discreta e suas aplicações*. São Paulo: Mc-Graw-Hill, Tradução da 6a. edição em inglês, 2007.

25. S. Russell, Efficient memory-bounded search methods. In Proceedings of the 10th European Conference on Artificial intelligence (Vienna, Austria). B. Neumann, Ed. John Wiley & Sons, New York, NY, 1-5, 1992.

26. R. Sedgewick, *Algorithms in C++, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching, 3th e*d. Reading: Addison-Wesley Publishing Company, 1998.

27. R. Sedgewick, *Algorithms in C++, Part 5: Graph Algorithms, 3th ed*. Reading: Addison-Wesley Publishing Company, 2002.

28. S. S. Skiena, *The Algorithm Design Manual, 2nd ed*. Berlin: Springer, 2008.

29. Sociedade Brasileira de Computação. Currículo de Referência para os cursos de Sistema de Informação, Licenciatura em Computação, Ciência da Computação e Engenharia da Computação, 2012. Disponível no endereço dado por: http://www.sbc.org.br/index.php?option=com_jdownloads &Itemid=195&task=viewcategory&catid=36. Acessado em: 7 de março de 2012.

30. Sociedade Brasileira de Computação. POSCOMP. Disponível em: http://www.sbc.org.br/index.php?option=com_content&view=category& layout=blog&id=237&Itemid=182. Acessado em: 21 de fevereiro de 2012.

31. T. A. Sudkamp, *Languages and Machines: An Introduction to the Theory of Computer Science, 3rd e*d. Reading: Addison-Wesley Publishing Company, 2006.

32. J. L. Szwarcfiter, L. Markenzon, *Estruturas de dados e seus algoritmos, 2a. ed*. Rio de Janeiro: LTC, 2009.

33. A. S. Tanenbaum, *Redes de Computadores*, tradução da 4a. ed. Rio de Janeiro: Campus, 2003.

34. J. W. J. Williams, "Algorithm 232 (heapsort)". Communications of the ACM, vol. 7, pp. 347–348, 1964.

35. N. Ziviani, *Projeto de Algoritmos com implementações em Java e C++*. São Paulo: Cengage Learning, 2005.

36. N. Ziviani, *Projeto de Algoritmos com implementações em PASCAL e C*, 3a. ed. São Paulo: Cengage Learning, 2011.