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

An Overview on Subjects and Reference Literature for Two Algorithm Classes

Sanderson L. Gonzaga de Oliveira

Abstract — **In this paper, an overview on subjects and its matching reference books for the teaching of two algorithm classes in undergraduate courses on Computer Science, Computer Engineering, Information Systems, Computer Science Teaching, Software Engineering and other similar ones is given. Complementary literature for these subjects are also recommended.**

Keywords — **algorithm teaching, data structure teaching.**

I. INTRODUCTION

ALgorithms are a fundamental subject for a Computer Science student to learn. No matter whether he intends to go into Graduate School or to find a job in the ID industry, he will have to work (either directly or indirectly) with the implementation of algorithms to solve computational problems he will have to deal with.

Therefore, it is of the utmost importance that those students learn this subject thoroughly. In order for a graduate in Computer Science, Computer Engineering, Software Engineering and Computer Science Teachers School to have an adequate introduction to algorithms, the following subjects on algorithms re suggested:

(1) basic information on algorithms, variables, constants, variable types, conditional structures, loop structures, arrays and matrices, records, files and pointers;

(2) abstract data types, lists, queues, stacks, priority queues, recursion, algorithms execution time comparison, sorting algorithms, array search algorithms, hashing and hash tables;

(3) concepts and mathematical properties of trees, binary and –n-ary trees, tree traversal methods, binary search trees, balanced trees and Huffman code;

(4) heaps and external sorting, red-black trees, B-trees and its variations, introduction to digital search, introduction to graphs, algorithms for graphs and string matching algorithms;

(5) algorithms exactitude, algorithm analysis, asymptotic notation, recurrence resolution, paradigms and techniques for the project of algorithms, polynomial reduction and NP-Completeness.

The choice of these subjects is based on a mix (detailed extension) of recommendations coming from the reference curricula from the Brazilian Computer Science Society (SBC, 2012a) and Association for Computing Machinery (ACM, 2012) for the courses we are studying.

When discussing algorithms and data structures, one may consider the fact that these reference curricula are dated from 1970 and 1980, especially on the works of authors such as Knuth, Aho, Hopcroft and Ullman. This is due to the fact that these authors were major contributors in the consolidation of several of the topics at hand. On the other hand there is a need to adequate the teaching to the IT market needs (including the scientific careers), which will be guaranteed solely by the constant update that these societies perform on their curricula.

A recommendation from these entities may be considered enough to include a topic on a graduation course. Nevertheless, in this paper the topics concerning items 4 and 5 are specified and detailed. Hence, justifications for the inclusion of each subject matter are discussed throughout the text.

S. L. Gonzaga de Oliveira is a full professor at the Computer Science Department at the Federal University of Lavras, Phone: +55-35-3829-1545/1945 (e-mail: sanderson@dcc.ufla.br).

In the SBC Reference Curriculum (2012ª) it is stated that "the way this content will be taught in a course is established by the didactic and pedagogical project and is at least as important as the simple distribution of topics into subjects". Besides, this document states that "given the strong dependence between the curricular grid and the didactic and pedagogical project, these elements should be created together". At the undergraduation level, it is suggested that the topic presentation follows (at least approximately) the given order. Depending on the depth to which each topic is taught, it is possible that some of the topics may be absent from a single semester subject. For instance, we can either select the topics of items 2 and 3 that are more pertinent in a single semester subject or present them in two semesters. In the latter, the subjects can emphasize algorithm implementation.

It is possible that some of these items are covered in Computer Science undergraduation courses. On the other hand, Computer Engineering, Information Systems, Computer Science Teaching and Software Engineering should cover most of the items, depending on the graduate profile intended.

Graduates from Computer Science courses must not have only a superficial understanding of algorithms and data structures, because they are the foundation of a software implementation. For instance, a software engineer must know deeply requisite analysis, project patterns and systems architecture, testing, software quality and business modeling, among other topics. Nevertheless, for a software engineer, knowledge on data structures is also relevant, because he will probably will also work on software development, create the systems projects and managing teams, which imply on knowledge on how the technician develop the systems in order to achieve high quality.

The same can be said about a graduate from an Information Systems course, which intends for its graduate course to work in corporate either middle or high management. For Computer Science graduates, the need for this knowledge is more obvious, given that their course provide skills to them to work in all activities related o IT and communications in corporations.

The focus of the previous paragraphs on market oriented careers does not mean that a scientific researcher does not need to understand deeply the science and the application of algorithms. After all, the scientific career is based on the ability to solve complex scientific and computational problems and it is impossible to solve them without the basic knowledge on Computer Science, among which we must include algorithms and data structures.

It is reasonable to assume that a graduate that follows a scientific career is able to be a good non academic professional. The advantages in the presentation of these contents to non academic professionals have already been discussed and will be further explored in this document.

The topics presented in this introduction (which will be subject to further detailing in this document) may be understood as the basic subjects necessary for the graduate to become a good IT professional. Nevertheless, we do not claim that this list is exhaustive and other additional topics may be included in this list, according to the needs perceived by instructors from analysis on the market where the graduates will work.

Nevertheless, understanding the topics in items 4 and 5 may offer an important contribution to the understanding of the following subjects, such as:

- Optimization whose probable topics are heuristics, linear programming, simulated annealing, taboo search, genetic algorithms, GRASP etc.;
- Artificial intelligence, whose probable topics are notions on search algorithms, MiniMax methods, probabilistic methods (bayesian networks, Markov models, decision theory), machine learning, neural networks, fuzzy logic, expert systems, etc.

The topics listed are also evaluated in the yearly tests in the National Exam for Graduate Studies (POSCOMP)[1]. It can be seen that the POSCOMP exam can guide the choice of contents in undergraduate courses, for it shows what are the fundamental concepts a bachelor must know in order to follow an academic career.

In table 1 the number of questions related to topics 2, 3, 4 and 5, in the five last POSCOMP exams can be seen. In this table, the number of questions related to the following subjects, such as Optimization and Artificial Intelligence can also be seen. As stated before, classes with the subjects 2, 3, 4 and 5 cover the basic topics that are necessary for the understanding of these two subjects. In the count shown in table 1, questions of graphs were not included because they were considered as belonging to the disciplines Discrete Mathematics or Graph Theory.

The suggestions we make in this paper do not exhaust the discussion and, obviously, each faculty must adapt the curriculum to its local and geographical reality. As we commented before, the topics and the depth to which it is taught depend on the profile of the graduate as defined by faculty. As such, if the faculty intends for the graduate to be well prepared for a high level academic career, the topics mentioned in items 2 to 5 are of utmost importance.

Among dozens of undergraduate subjects, the content included in items 2, 3, 4 and 5 have comprised between 10% and 17% in the POSCOMP exams for the last five year. We can consider without much of a leap of faith that the concentration on those topics is proportional to the importance they have in Computer Science courses.

[1] To know more about POSCOMP, we suggest visiting SBC's web page (2012b).

Table 1: Amount of questions at the POSCOMP related to subjects 2, 3, 4, 5 and the folowing ones (Optimization and Artificial Inteligence – considered in columns 6) for the last 5 years.

Year	Subject				Percent when compared to total number of questions in the exam
	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

Besides, the knowledge on the fundamental techniques on algorithms is very important for non academic professionals to develop high quality computer systems. For instance, computer systems should not have problems coming from low efficiency implementations and these problems can be avoided (if not completely) if the development team knows deeply the techniques and methods already established in the field. It is not uncommon for those problems to arise and surely many of them could have been prevented either they were better developed or the developers in charge had been trained with the basic knowledge on the items 2 to 5.

We can assume that Brazil has the potential to become a world power in the production of information systems and computational techniques, shedding the status of IT consumer. There are many current examples in which excellent ideas arise in the academia and clearly an appropriate foundation must be fomented.

Therefore, the fundamental goal of this paper is to give some information to substantiate a discussion on the issue. In order to achieve this goal, this paper is organized as follows: in section II the important books for bibliographic references for the topics included in items 4 and 5 are discussed. In sections III and IV, the possible contents for the subjects on items 4 and 5, are respectively discussed.

In each subsection of sections III and IV there are comments on each topic, their corresponding references and the reasons why those topics are important for basic algorithm studies. In the last subsection of each of those sections, the suggestions on reference books for each topic in the section and suggested textbooks and complementary literature are summarized. Final thoughts are expressed in section V.

II. IMPORTANT BOOKS FOR THE BIBLIOGRAPHY OF TOPICS LISTED IN ITEMS 4 AND 5

In this section the possible reference books for topics listed in sections 4 and 5 are discussed. For those, the second edition of the book written by Cormen et al (2001) has been widely used in Brazil. There is a Portuguese version of this book but it is less used than its English counterpart, apparently for translation issues.

In this paper, we will comment on the third edition of this book: Cormen et al. (2009). This book, written by Cormen, Leiserson, Rivest e Stein (CLRS), has been the most used in the best universities in Brazil and balances well mathematical rigor (a fundamental aspect of demanding courses) and didactic, for the writers have the gift of writing precisely, clearly and objectively.

Besides, its scope is a major advantage since this book covers most of the main computational problems, its algorithms and data structures. It has been listed as one of the most quoted reference in Computer Science for many years and since its first edition it has been reprinted either every six months or every year, each time with some corrections many of them pointed out by its thousands of readers.

Therefore, when we read CRLS opus, we can be pretty confident that all the subjects described are correct and we have a strong foundation on algorithms. In the end of each of it 35 chapters the authors included descriptions in order to provide the curious reader with references to publications in which he can find knowledge on the state of the art of each of the approached topics.

Given those characteristics, CRLS opus has each day increased its standing in the academia. Therefore, there are professors that do not even consider textbooks in those subjects other than CRLS. In fact, in average the CLRS book can be considered as the most adequate for an introduction to algorithms.

Nevertheless, in spite of the excellence of this book, if we search the Internet we can find some criticism on it, especially on the small number of examples and the lack of solved exercises. Besides, there are important topics such as external sorting that are not described in this book. For some reason that is outside the scope of this paper, the authors could not go any deeper (in the already 1292 page long book) in some topics or include other relevant topics on algorithms.

Yet, in spite of the excellent quality of its presentation, there are books that describe some topics better than CLRS, especially when considering didactic, formal rigor amount of examples and solved exercises. These are the books in specific topics that this paper describes.

Another set of already classic books that must be mentioned are the ones written by Knuth (1997, 1981, 1998, 2001). The discussed subjects are exhausted by the author. The explained subjects are not trivial but in spite of their complication and scope, the author knows how to choose the correct words in

order to be brief and yet describe appropriately all important concepts on algorithms and data structures.

Knuth wrote his books with clear and precise language, not being excessively technical and managed to approach all subjects with the adequate depth yet is still funny, having included several humorous remarks which render some amusement to the study of Computer Science. I hope that the reader is granted the opportunity to study his books and amuse himself as much as myself. In spite of that, there are teachers that do not consider Knuth's books approachable enough for the typical undergraduate student. Nevertheless, there are topics that are better described in his books than in others, so that his books may be used in undergraduate courses on some specific topics.

These books are excellent for the subjects listed in items 4 and 5. Nevertheless, there are other books that should be considered for the main or auxiliary bibliography on those topics. Those are the references discussed in the next sections.

III. POSSIBLE CONTENT FOR A SUBJECT ON ALGORITHMS LISTED IN ITEM 4

For a discipline on algorithms as listed in item 4 in one of the mentioned undergraduate courses, one may include heaps, external sorting, red black trees, B trees and its variations, introduction to digital search , introduction to graphs and its algorithms and string matching algorithms.

These topics are commented in the following subsections. For each one of them the suggested corresponding reference literature for an undergraduate subject is included.

A. Heaps

Heaps are an excellent way to implement priority queues. Williams (1964) developed this data structure for the heapsort algorithm. It is important not to confuse the heap data structure with the term heap used when discussing dynamic memory allocation.

Heaps are important for several algorithms. For instance, heaps are used in graph algorithms (see subsection F below). They can also be found in the implementation of the Prim algorithm to find the minimum spanning tree. The A* (Hart et al., 1968) and SMA* (Russel, 1992) search algorithms also used priority queues that can be implemented using heaps.

Knowledge on fundamental data structures such as a heap is also important for those who will develop network protocols. An example of their usage is Dijkstra's algorithm (1959) which is the basis for link state routing. Heaps can also be used for transmission line (such as VoIP connections) bandwidth management in network routing. Many protocols for Local Area Networks (LAN) use priority queues (that can be implemented using heaps) in their medium access layer (MAC - *media access control*) - VoIP and IPTV are concrete examples of this. For an introduction on these subjects, please refer to computer network books such as Tanenbaum (2003) or Kurose and Ross (2006).

In order to study the heap data structure, the following references are recommended:

- In its sixth chapter, CLRS (2009) presents in a didactical way heapsort, heaps and priority queues;
- Priority queues and heaps are described in a succint way in the sixth chapter of Szwarcfiter and Markenzon (2009);
- an objective and precise presentation of priority queues and heapsort is given in chapter 9 of Sedgewick (1998);
- a forth approach that should be highlighted because of its good didactic is subsection 4.3.1 in Skiena (2008), which is fully dedicated to heaps.

B. External Sorting

Consider the situation where we want to sort a set of elements, a problem that is sometimes presented in a second subject on algorithms. Examples of theses algorithms are Bubble Sort, Insertion Sort, Selection Sort, quicksort, heapsort and mergesort.

In order to use one of those algorithms, all dataset needs to be in main memory. When the main memory is not large enough to hold the entire data set being ordered, an external sorting algorithm must be used. In this algorithm, only a fragment of the dataset is loaded into main memory at each phase, while another part remains in external (or secondary) memory storage, such as hard disks. In this context, the algorithms mentioned before could be called internal sorting algorithms.

For an Information Systems course, in the SBC Reference Curriculum (2012a), it is recommended to have a subject in which Sorting and Searching is deeply taught. Besides, it should be taught the concepts on "searching and sorting algorithms both in main and secondary memory". For a course on Computer Science teaching, the recommendation is to completely teach these subjects.

External sorting algorithms can be very important for developers that must work with huge datasets, something that has become common in organizations that prioritize gathering data from day to day transactions.

It is also important to teach the basic external sorting algorithms in undergraduate courses so that the students can realize that simple algorithms can breed ingenuous solutions to problems that can be deemed as complex. Besides, the students can then use those basic problem solution concepts to create new solutions for the problems they will have to solve when pursuing either an academic or an IT industry career.

External sorting is used in important business applications such as in databases transactions. For instance, consider the situation where a user requests a subset of columns from a certain table from a specific database. The request is granted

through a file, in which there may be duplicated records. Hence, an external sorting algorithm can be used to delete the duplicate records before sending them to the user.

Some basic external sorting algorithms that can be taught at the undergraduate levels are: balanced multi-way merging, polyphase merge sort and replacement selection. For this topic, the following bibliography is suggested:

- Ziviani (2011) has a good description on those algorithms in its section 4.2;
- an even better presentation is given in section 11.3 of Sedgewick (1998).

Another algorithm that is quite simple and very interesting is the natural selection that shows a small but significant change to replacement selection. It is probable that the only book that has already described this algorithm up to 2011 is Knuth (1998).

C. Red-blacktTrees

A red-black tree is a binary search tree that is approximately balanced. A binary search tree (which will be presented probably in the second subject on algorithms) shows *O(h)* time in all its basic operations , in which *h* is the height of the tree. Hence, the search can become linear in degenerate trees. Meanwhile, a red black tree shows *O(log n)* time for all its operations, in which *n* is the number of nodes in the tree, in the worst case.

Red-black trees can be said to have performance and basic characteristics similar to those of AVL trees, which are widely studied. Nevertheless, the re-balancing of red black trees is more efficient and so they can be taught as a more efficient way to implement balanced trees (and even priority queues). For instance, red black trees are used at the Completely Fair Scheduler, which is the scheduler of the Linux core starting from version 2.6.23 (see, for instance, Jones, 2009, for details).

Red-black trees can also be used in real time applications because of their performance. Besides, they can also be represented by 2-4 trees, which are a type of B Tree (see the following subsection).

For this topic, the following bibliography is suggested:

- Red-black trees are presented in an excellent way in chapter 13 of CLRS (2009),
- Szwarcfiter and Markenzon (2009) also present a good description of red-black trees in its subsection 5.4;
- there is a very good description of red-black trees in section 13.4 of Sedgewick (1998).

D. B-trees and its variations

In a B-Tree the data are sorted in such a way that sequential access, insertions and deletions are performed in logarithmic time. Its variations are important in the implementation of both database management systems and operational systems file systems.

In the SBC Reference Curriculum (2012a) it is recommended that B and B+ trees shall be taught deeply in the Information Systems and Computer Science Teacher courses.

For this topic, the following bibliography is suggested:

- CLRS (2009) presents the concepts of the original B-tree, its operations and pseudo-codes in a most detailed way;
- an extremely didactic approach of B-trees and its B* and B+ variations is made at subsection 7.1 of Drozdek (2002);
- a clear and concise description of B-trees is also given in section 16.3 of Sedgewick (1998), in which the codes in C++ are also presented.

E. Introduction to digital search: tries and Patricia trees

Basic trees for digital search are simple and amenable to undergraduate level teaching. A trie (from retrieval) is a tree in which the data remain ordered.

A trie is used to store contents whose keys are usually strings. They are very efficient in the search for keys, such as in dictionaries. Their search time is proportional to the size of the key and can also be helpful in the search for prefixes and similar words, what can be very useful in devices with limited keyboard interface, such as cell phones and other portable devices. A didactic presentation of tries is available at subsection 7.2 of Drozdek (2002).

In a Patricia tree, which is a special kind of trie, each internal node stores the index of the most significant bit to be verified in order to decide which branch to follow in a search operation. Namely, in this tree, we follow all the branches according to the bits of the search key and not according to the result of the full key comparison. Strings, integer sets such as IP addresses and arbitrary generic sequence of objects in lexicographical order are examples of keys that can be used with this kind of tree.

For these topics, we suggest the references:

- Introduction to digital search, tries and Patricia trees are well described in chapter 9 of Szwarcfiter and Markenzon (2009),
- subsection 5.4 of Ziviani (2011) because of its didactic presentation;
- the most detailed explanation is the one given by Sedgewick (1998) in sections 15.2 and 15.3.

F. Algorithms for Graphs

The reader can observe that in the SBC reference curriculum (2012a) it is recommended that notions of algorithms for graphs shall be given in Computer Science Teacher courses. The basic problems connected to graphs are:

- Visit all vertices in a graph (depth-first and breadth-first search),
- Minimum spanning tree (Prim, Kruskal and Boruvka algorithms),
- Minimum path from one vertex to the others (Dijkstra and Bellman-Ford algorithms),
- Minimum path between all vertices (Floyd-Warshall algorithms)
- Maximum flow (Ford-Fulkerson algorithms), a problem from the area of network flow.

Visiting all vertices in a graph or searching for a specific vertex is relevant in several situations, such as for Internet applications, given that routers can be modeled as vertices and their connections as edges. Another example is a computer game in which players are vertices and the field of vision between them the edges – in this case, an attack can be planned using breadth-first search. Depth-first search designed using backtracking (see algorithms construction techniques) can be one of the simpler recursive search methods.

Consider that a method receives a graph and must return a tree (an acyclic graph) that contains all vertices and the cost to visit all vertices is minimized. This tree can be called minimum spanning tree and is used, for instance, in applications in the field of Electrical Engineering.

In order to be presented to students, the minimum path problem can be associated with network routing, such as in the Internet. Dijkstra's algorithm, already mentioned, is the basis for link state network routing, so it is an example in which the teacher has a great opportunity to stress the importance of algorithms that are a part of the students' daily life. Examples of routing protocols based on link state include Intermediate-System-to-Intermediate-System (IS-IS) and Open Shortest Path First (OSPF). To know about IS-IS, see Tanenbaum (2007, p. 389) or the Request for Comments (RFC) 1142, at http://tools.ietf.org/html/rfc1142. To know more about OSPF, see Tanenbaum (2007, p. 483-488), Kurose and Ross (2006, p. 255-256) and the RFC 2328, at http://tools.ietf.org/html/rfc2328. The updates for IPv6 are specified at the version 3 of OSPF at the RFC 5340 from 2008, available at http://tools.ietf.org/html/rfc5340.

The maximum flow problem is associated with the problem of taking something from a source to a sink in a network (or a graph). Using the term network instead of graph comes from the first publications in this field in the 1950s. The problem of maximum flow is the foundation for several applications such as image segmentation and flight programming for air transportation companies. A simple example that might be presented to students is the application of Barnett et al. (2007) that used the concepts of maximum flow in the positioning of networked sensors in the streets for urban security purposes.

There are other problems with their corresponding algorithms that could be taught in such a class, but they might render the content to large for a 60h subject in which other topics also need to be taught. Besides, in order to approach algorithms for graphs one needs to introduce their terminology, basic representation forms, adjacency matrices and list, and incidence matrix.

For this topic, the suggested bibliography is:

- the best book for these subjects is CLRS (2009), because its descriptions are precise, objective and presented in a most didactic way; in general, CLRS (2009) has dedicated a chapter for each of the problems and their corresponding algorithms, in which the student will find a nice presentation of the algorithms with complete examples;
- Sedgewick (2002) is an excellent book, specific on graph algorithms, in which the author goes deep into each algorithm, presenting them in C++;
- another didactic description of the algorithms comprised in this topic in Skiena (2008), in the chapters 5, 6 and 11.

G. String searching algorithms

The goal of these algorithms is to find the occurrences of a pattern (string) in a text. Applications of these algorithms include text editing, DNA sequence analysis and information retrieval.

For this topic, the suggested bibliography is:

- CLRS (2009) has a good description for the algorithms that can be used to solve this problem in its chapter 32;
- Objective and precise descriptions can also be found in chapter 10 of Szwarcfiter and Markenzon (2009) and
- Chapter 8 of Ziviani (2011), which is a good description of those algorithms.

H. Table with contents and suggestions of textbooks

In table 2 the textbook suggestions for each topic that should be taught in a subject composed by item 4 are summarized. CLRS (2009) and Sedgewick (1998) are recommended as the main textbooks for this subject.

In case it is necessary to use three textbooks, we suggest Drozdek (2002) because of its description of B-trees and its variations. Knuth (1998) should be the main complementary literature for this subject.

Table 2: Textbook and complementary reading recommendations for the topics that should be taught in the subject composed by item 4.

Topic	**Textbook**	**Auxiliary textbook**	**Second auxiliary textbook**
Heaps	CLRS (2009)	Sedgewick (1998)	Szwarcfiter and Markenzon (2009)
External sorting	Knuth (1998)	Sedgewick (1998)	Ziviani (2011)
Red black trees	CLRS (2009)	Sedgewick (1998)	Szwarcfiter and Markenzon (2009)
B Trees and its variations	Drozdek (2002)	CLRS (2009)	Sedgewick (1998)
Introduction to digital search	Sedgewick (1998)	Szwarcfiter and Markenzon (2009)	Ziviani (2011)
Introduction to graphs and its algorithms	CLRS (2009)	Sedgewick (2002)	Skiena (2008)
String matching algorithms	CLRS (2009)	Ziviani (2011)	Szwarcfiter and Markenzon (2009)

IV. POSSIBLE CONTENTS FOR A SUBJECT ON ALGORITHMS AS LISTED IN ITEM 5

For a subject composed by item 5 in an undergraduate course, the topics that can be taught are: algorithm exactitude, algorithm analysis, asymptotic notation, recurrence solving, paradigms and techniques for algorithm project, polynomial reduction and NP-Completeness. This subject is sometimes called Design and Analysis of Algorithms.

In the SBC reference curriculum (2012a) it is recommended that this subject be taught in depth at Computer Science Teacher courses. For information systems courses, all the content must be presented and the recommended syllabus: "Algorithm development; Techniques for the design of algorithms, asymptotic analysis of complexity limits, techniques to prove inferior limits, analysis of recursive algorithms, dynamic programming, probabilistic algorithms, pessimistic, minimal and average problem complexity, classes of problems: P, NP, and NP-complete". In the ACM reference curriculum (2012) the presentation of these items is essential in a software engineering course.

The contents suggested in this paper are commented ahead and the corresponding textbooks that are more adequate to an undergraduation course are suggested.

A. Basic math review for algorithm analysis

A review of discrete math may be important to foster the learning of algorithm analysis, what is especially true given the low level of knowledge many students entering high level education possess. For this topic, the following books are suggested: Gersting (2004), Rosen (2007), Graham et al. (1995) because of its didactic presentation. In the appendix A of CLRS (2009), there is a good review on summations.

B. Algorithm exactitude

In order to present a new algorithm, it is necessary to prove that it always returns the expected output for the input domain and also that it will always finish. Academic presentation of this notion is important in the courses that interest us, allowing for the validation of the developed applications, both in the academic and in the IT industry realm.

Good presentations of these topics are available at the chapter 5 of Harel (2004), section 13 of Skiena (2008), a book that contains many proofs of exactitude for different algorithms spread through many sections. We also need to mention the didactic presentation of CLRS (2009) about invariant loops in its section 2.1.

C. Algorithm Analysis

It is often not enough for a computational problem to be solved – it must be solved in an efficient way so that the program execution finishes quickly enough for the application in its domain. In order to understand algorithm efficiency, we must study algorithm analysis, because a problem represented by an algorithm asymptotically slow may not be useful (Goodrich and Tamassia, 2004), even if the hardware is much improved. Therefore, performing algorithm analysis may be deemed as important as solving the problem.

A formal foundation on algorithm analysis is a fundamental topic for the graduates in all mentioned courses, especially those that intend to develop applications in computationally intensive domains that are very common in the academia and not so scarce in the IT industry. The knowledge on algorithm analysis is especially important for those that will work in fields such as optimization, artificial intelligence and digital image processing.

The best bibliography for this topic is:

- CLRS (2009) have a didactic presentation of the topic in its chapter 2;
- Goodrich and Tamassia (2004), in its chapter 1, present more detail than those in CLRS (2009) and thus deserve to be either a textbook or a complementary reading in this topic;
- Another interesting text is chapter 4 in Brassard and Bratley (1996) for its didactics.
- A forth text tthat can be used is subsection 1.2.10 of Knuth (1997), in which this topic is presented objectively.

Often, the cost of an operation in a certain data structure is high in a certain phase, but this cost can be amortized in the amount of operations executed when they grow a lot. The literature on amortized analysis is:

- CLRS (2009) is the most indicated for the authors present aggregate analysis, accounting method, potential method e show examples of applications of all methods in dynamics tables in chapter 17;
- Goodrich and Tamassia (2004) present a didactic description of amortized analysis in its section 1.5;
- Skiena (2008) can be an option also because of its didactic presentation.

D. Asymptotic Notation

Asymptotic notation in deeply connected to algorithm analysis, being very important to understand the efficiency limits of an algorithm. This topic must be presented adequately and in details in undergraduate courses that include algorithm analysis. For this topic, the following literature is suggested:

- The best description on this topic is Graham et al. (1995) , in its chapter 9;
- The description of notations O, Ω, Θ, o e ω in chapter 3 of CLRS (2009) is also interesting and quite balanced between formal rigor and ease of understanding;
- Goodrich and Tamassia (2004) also present a didactic description on this topic in its section 1.2.

E. Recurrence resolution

Usually, execution time of a recursive algorithm is expressed as a recurrence. Since recursive algorithms are widely used, it is important to include a systematic study of recurrence resolution in an undergraduate subject.

Recursive algorithms often present worse computation performance or use more memory than its iterative counterpart. Knowing how to determine whether or not a recursive algorithm is efficient may be relevant in many applications in information systems, academic, technical or scientific that will be created by professionals.

For this topic, the following literature is suggested:

- One of the best approaches on he topic is given by section 2.3 of Brassard and Bratley (1996), because of its depth and didactic;
- Section 7.3 of Graham et al. (1995) is also excellent due to its objectivity and precision;
- There are didactic presentations on this topic ins sections 7.1, 7.2 and 7.3 of Rosen (2007) e
- Section 2.3 of Gersting (2004), also because of its didactic presentation.

F. Paradigms and techniques in algorithm design

Usually, a beginner student learns to create algorithms in a naive way. After he acquires some experience on the field, some well known and quite simple techniques for algorithm design must be presented to the student.

Among the important paradigms and techniques for algorithm designs, we can mention the following:

- Brute force (or exhaustive search)
- backtracking, a variation of brute force;
- branch-and-bound, a variation of backtracking;
- recursion;
- balancing;
- incremental approach;
- divide and conquer;
- greedy algorithms;
- dynamic programming;
- approximate algorithms;

Knowledge of this paradigms and basic techniques may speed up the adequate development of information systems, avoiding a naive implementation. In order to adequately compare the different approaches, complexity analysis, described before, must have been presented beforehand.

There are authors that consider that textbooks must present a paradigm or technique of algorithm design in a chapter and then present applications and examples of the technique to solve classical problems. Usually, in those books the title of the chapter is equal to the name of the paradigm or technique, and its structure follows the rule of a brief introduction followed by some algorithms that exemplify it. For instance, CLRS (2009) follow this scheme. The authors have dedicated its chapter 4 to the paradigm of divide and conquer its chapter 15 to dynamic programming and its chapter 16 to greedy algorithms. Nevertheless, there are other books whose

approach one may consider superior and those are the ones recommended:

- Ziviani (2011) describes this topic in a didactic and objective way in its chapter 2, which is fully dedicated to paradigms and techniques for algorithm designs.
- Goodrich and Tamassia (2004) present this topic in its chapter 5 in an interesting fashion, for going deeper than the other books in this topic. Nevertheless, its coverage is less wide than the previously mentioned book.
- Another interesting approach for this topic is the one presented by Aho et al. (1974) because the authors also present this topic in a precise and didactic way.

If one wants to talk specifically about recursion, the study of chapter 5 of Sedgewick (1998) is important because of its depth on the subject. Besides, in this book there are also interesting sections on divide and conquer (5.2) and dynamic programming (5.3) that have thorough and deep presentations.

The following books are commendable on the issue of the divide and conquer paradigm:

- A good introduction is given in chapter 4 of CLRS (2009), in which the substitution method, the recursion tree method and the master methods are presented in a didactic way, besides having comments on Akra-Bazzi method;
- In subsection 5.2.1 of Goodrich and Tamassia (2004), there is also an introduction to the analysis of recursive algorithms, in which they describe the execution time of divide and conquer algorithms;
- A third option that is quite didactic on the resolution of recurrences that arise in the divide and conquer algorithms is section 7.3 of Rosen (2007).

G. NP-Completeness

In the undergraduate courses mentioned, the theory on NP-Completeness must be approached with some depth, because the students must understand that the problems are more important than the algorithms that solve them. There are several (even thousands, if we account for variations) important problems that have been described for which there are neither known polynomial algorithms nor proof of an inferior non polynomial limit of time for an algorithm to solve them.

Surprisingly for some, these problems are quite common in the Engineering field. The traveling salesman problem is probably the best known of them and it accounts for a situation in which the minimum cost traversing each graph vertex must be found. For this situation, imagine a professional asked to solve this problem in a transportation company. If a naive algorithm is used, the time required to solve it will be so high that any manager above the developer of the algorithm will probably have some doubts about his/her ability as a professional, especially if the developer requests more potent hardware to solve the problem. This situation is described in an amusing way in the first chapter of Garey and Johnson (1979).

PxNP is considered to be one of the most important unsolved problems in Computer Science and Mathematics. For instance, even in optimization, it is common to find programmers that work in NP-Hard problems without knowing adequately the fundamentals of this theory.

Garey and Johnson (1979) is the classical book that is most often quoted on NP-Completeness and one of the most referenced books in Computer Science. Just as is presented by Garey and Johnson (1979), the classes in this theory must be defined formally in terms of a computing model, and the Turing machine may be particularly adequate for this purpose.

The first chapter of this book is a comprehensible introduction to NP-Completeness. In the second chapter, the authors formalize the main classes according to the Turing machine. Example proofs of NP-Completeness are given in chapter 3, whereas the authors show how to use it to analyze problems in chapter 4. In chapter 5 the authors describe concepts on NP-hardness and in chapter 6 they show dome guarantees of performance for approximate algorithms. In chapter 7, they approach the theory beyond NP-Completeness and in the appendix they describe the NP-Complete problems known at the time of their writing.

Another book recommend for the study of NP-Completeness is Sudkamp (2007) because its presentation is very didactic. Besides, another excellent desciption on the basic classes of NP-Completeness according to Turing machines is given in chapter 10 of Aho et al. (1974). For a didactic approach on NP-Completeness without any formalization according to computing models, the recommended reading is chapter 34 of CLRS (2009).

The important topic related to NP-Completeness are:

- basic concepts of polynomial reductions, on which a good didactic presentation is given by Manber (1989);
- proofs of NP-Completeness, for which CLRS (2009) and Manber (1989) are adequately simple and easy to understand;
- demonstration of the theorem of Cook-Levin, for which the references Sudkamp (2007), Garey and Johnson (1979) and Aho et al. (1974) re pointer for references because of their formality and simplicity.

H. Table with contents and suggestions of textbooks

In tables 3 and 4 the textbook suggestions for each topic described in this section are summarized. In table 3 the topics and books suggested for algorithm exactitude, basic math

review, algorithm analysis fundamentals, amortized analysis, asymptotic notation and recurrence resolution are listed.

Table 3: Textbook and complementary reading recommendations for the first part of the topics that should be taught in a subject comprising the items discussed in section IV.

Topic	Textbook	Auxiliary Textbook	Second Auxiliary Textbook
Algorithm exactitude	Harel (2004)	Skiena (2004)	CLRS (2009)
Basic math review	Gersting (2004)	Rosen (2007)	Graham et al. (1995)
Fundamental of algorithm analysis	CLRS (2009)	Goodrich and Tamassia (2004)	Brassard and Bratley (1996)
Amortized analysis	CLRS (2009)	Goodrich and Tamassia (2004)	Skiena (2008)
Asymptotic notation	CLRS (2009)	Graham et al. (1995)	Goodrich and Tamassia (2004)
Recurrence resolution	Brassard and Bratley (1996)	Graham et al. (1995)	Rosen (2007)

In table 4 the topics and books suggested for paradigms and techniques for algorithm design, recurrence resolution that come from divide and conquer algorithms and the theory of NP-Completeness are listed.

If it is necessary to use three textbooks in an undergraduate subject, CLRS (2009), Goodrich and Tamassia (2004) and Ziviani (2011) are recommended as the main textbooks for the topics discussed in this section. As complementary reading the books by i) Garey e Johnson (1979) because of its excellent description of NP-Completeness and ii) Graham et al. (1995), because of the quality of its review of basic math, asymptotic notations and recurrence resolution are also recommended.

Table 4: Textbook and complementary reading recommendations for the second part of the topics that should be taught in a subject comprising the items discussed in section IV.

Topic	Textbook	Auxiliary Textbook	Second Auxiliary Textbook
Paradigms and techniques for algorithm design	Ziviani (2011)	Goodrich and Tamassia (2004)	Aho et al. (1974)
Resdolution of recurrences that come from divide and conquer algorithms	CLRS (2009)	Goodrich and Tamassia (2004)	Rosen (2007)
NP-Compleness: theory and presentation of basic classes	Garey e Johnson (1979)	Sudkamp (2007)	Aho et al. (1974)
NP-Compleness:: basic concepts of polynomial reductions	Manber (1989)	Garey and Johnson (1979)	Sudkamp (2007)
Proofs of NP-Compleness	CLRS (2009)	Manber (1989)	Garey and Johnson (1979)
NP-Compleness: proof of the Cook-Levin theorem	Sudkamp (2007)	Garey and Johnson (1979)	Aho et al. (1974)

V. FINAL THOUGHTS

If one needs to recommend a single book for the disciplines that include items 4 and 5, indeed CLRS (2009) is the best option. Nevertheless, this reference is not the best study option for all the topics described. In several specific topics, there are great descriptions in other books.

Actually, it is necessary to use two or more textbooks in bothdisciplines. Indeed, there may be the need to use more than one book in each topic discussed.

The books Sedgewick (1998, 2002) are also recommended for a subject that includes the item 4. The author presents most of the topics in this item in a detailed way with implementations in C++.

For this item, Drozdek (2002) is also an adequate option in specific topics. Besides, Szwarcfiter and Markenzon (2009) and Ziviani (2011) are also adequate alternative options.

There are also other options to consider when thinking about different presentation styles or specific topics. When thinking about didactic, with no demands on formalism complications, the books by Ziviani (2005, 2011) show advantages. The author knows how to present contents in a simple and objective way. One of the main advantages of these books is its implementations in Pascal and C (Ziviani, 2011) or Java and C++ (Ziviani, 2005), which may be the cause for being one of the most widely used in undergraduate courses in Brazil.

Other books that deserve praise are Manber (1989) and Goodrich and Tamassia (2004). The latter, in particular, can be used as a textbook in an algorithm subject tin which the topics describes in item 5 are taught. Aho et al. (1974), Garey and Johnson (1979), Brassard and Bratley (1996), Graham et al. (1995), Gersting (2004), Rosen (2007), Sudkamp (2007), Skiena (2008) and the four volumes of Knuth also must be considered to be part of the list of complementary reading of the subjects discussed in this text.

I started to create class notes since I started to teach classes on algorithms both in Computer Science and Information Systems courses. The notes have become a book, Gonzaga de Oliveira (2011), in which I describe most topics included in items 4 e 5. This book can be used as a complementary reading for the disciplines comprised of items 4 and 5.

The books not mentioned in this paper may not be very known, nor widely used by Brazilian professors or simply unknown to the author. It may also happen that books not mentioned here are not as adequate for some specific topics as the ones mentioned. That may be due to the author's style or for his intention to help a specific group of students. For instance, there are books that do not include almost any equations because this may scare off some readers. In this example, an author may explain the topic in a long and detailed way, with many examples without generalizations. A book that does not contain many mathematical formalizations may have a large commercial appeal, that is, they may be appreciated by those seeking a superficial understanding on the subject (more than those that are more formal, use more mathematical language in a deep and abstract way). This will depend on the style and expected results of each author.

Both approaches are valid – the more didactic are more efficient for a first reading. On the other hand, it may be impossible to achieve a deeper understanding with those books and a second reading may be boring due to the large amount of text they possess. The formal approaches, on the other hand, in which the topics are presented more formally with generalizations and in which the student ability for thinking logically is developed, may be difficult for undergraduate students in their first reading, but may be useful after didactic explanations in classrooms. Nevertheless, these books help the students understand important concepts on the topic at hand, allowing them to achieve proper learning of algorithms.

When choosing the three most adequate books, a balance between ease of understanding and mathematical rigor was one of the main goals. When considering this balance, books like CLRS (2009) stand out.

The descriptions in this paper, the topics suggested for algorithm subjects and their textbooks represent the author's opinion. Nevertheless, I expect that the descriptions can be useful as the basis of a discussion between professors when creating the syllabus for such classes.

VI. ACKNOWLEGMENTS

I would like to thank the Research Support Foundation for the State of Minas Gerais – FAPEMIG and to the National Scientific Development Council– CNPq for the financial support. The author would also like to thank the editorial board for the Salesian Journal on Information Systems whose ideas and indications helped this paper arrive to its current state. I would also like to thank professors Antonio Maria Pereira de Resende, Denilson Alves Pereira and Luiz Henrique Andrade Correia for their reviews on specific parts.

VII. REFERENCES

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.