COMPUTER PROGRAMMING
Unit 2 teaching guide
Analysis of algorithms. Efficiency. Sorting

Jon Ander Gómez Adrián, Marisa Llorens Agost
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
{jon,mllorens}@dsic.upv.es

January 25, 2018

1 Contents

2 Specific objectives

Upon completion of this subject the student should be able to ...

Objective 1 list the differences between theoretical (or a priori) analysis and empirical (or a posteriori) analysis.

Objective 2 enumerate the factors to take into account for both types of analysis: theoretical and experimental.

Objective 3 prepare various test cases to analyze experimentally the behavior of a program, whether he wrote it or not.

Objective 4 identify the input size of a problem from the full specification by an statement.

Objective 5 determine the input size of a problem from the source code that solves it.

Objective 6 express the concept of program step.

Objective 7 identify possible critical operations in the source code he has to analyze.

Objective 8 detect when an algorithm has significant instances, and if so, identify the best and worst cases.

Objective 9 get the expression that represents the temporal cost depending on the input size of the problem. If the program has some significant instances, students must obtain two temporal cost functions, one for the best case and another one for the worst case.

Objective 10 express, by using asymptotic notation, the order of growth of the temporal cost function of an algorithm, distinguishing best and worst cases when the problem presents significant instances.

Objective 11 get the temporal cost function of an iterative algorithm by applying the following three steps: 1) identify the input size of the problem, 2) determine whether there are significant instances, and 3) point out, at least, one critical operation that serves as a reference of program step.

Objective 12 get the temporal cost function of a recursive algorithm by applying the same steps 1 and 2 as in the previous objective, and deduce the recurrence relation that will allow him to obtain the temporary cost function.

Objective 13 programming a not-rapid sorting algorithm to sort a vector whose components are of an elementary data type. With not-rapid algorithm we refer to one of the following: selection sort, insertion sort or Bubble-Sort.

Objective 14 programming a fast sorting algorithm, concretely the Merge-Sort.

Objective 15 choose the most appropriate sorting algorithm depending on how data are distributed in the vector. For example, if the vector to sort is practically sorted, then, one algorithm should be used, if it is inversely sorted others algorithms must be used.

Objective 16 programming the Natural-Merge algorithm to obtain a new sorted vector from two previously sorted ones.

Objective 17 programming the binary search algorithm for searching within a sorted vector.

Objective 18 decide between implementing a sequential search algorithm, or the binary search algorithm with a prior sorting of the vector, depending on the expected number of searches on a vector. Knowing that in the second case the cost of the previous sorting must be added to the cost of planned searches.

3 References

- "Empezar a programar usando Java"
  Profesores de las asignaturas IIP y PRG
  Apuntes facilitados vía *PoliformaT*
  Chapters 12 and 13

- "Introducció a l'anàlisi i disseny d'algorismes"
  Ferri, F,J., Albert, F.V., Martín, G.
  Universitat de València, 1998
  Chapter 2
  Chapter 3, except subsections from 3.3.2 to 3.3.5
  Chapter 5, sections from 5.1 to 5.3, and subsection 5.4.1
  **** This book is specially recommended for people able to read and understand the Valencian language.*

- "Estructuras de datos en Java: compatible con Java 2"
  Mark Allen Weiss
  Ed. Addison Wesley, 2000 - 2006
  Chapter 5
  Chapter 8, sections from 8.1 to 8.3

- "Fundamentos de Algoritmia"
  G. Brassard y P. Bratley
  Pearson – Prentice Hall, 2001
  Chapter 2, sections from 2.1 to 2.6
  Chapter 4, sections from 4.1 to 4.4

- "Introduction to Algorithms" 2nd edition
  T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein
  The MIT Press – Cambridge, Massachusetts, 2007
  Chapter 1, 2, 3 and 4

4 Planning of each session

Duration of activities

	Inside classroom	Outside classroom
Before	–	2h
Session 1	1.5h	3h
Session 2	1.5h	3h
Session 3	1.5h	2h
Session 4	1.5h	2h
Session 5	1.5h	2h
Session 6	1.5h	2h
Session 7	1.5h	2h
Session 8	1.5h	2h
Session 9	1.5h	–
	13.5h	20h

Before the first session

Outside classroom activities (up to 2h)

- It is proposed to read chapter 12 (sections 12.1 up to 12.4) of "Empezar a programar usando Java", or paragraphs from 2.1 to 2.2 except paragraphs 2.2.2 and 2.2.3 of the book "Introducció a l'Anàlisi i disseny d'algorismes", F.J. Ferri et al.

- As complementary, we suggest reading the paragraphs 5.1 to 5.4 of Chapter 5 of the book "Data Structures in Java: Java compatible 2" by Mark Allen Weiss.

  Section 5.3 discusses a problem more complex than usual for the first year of computer programming. The intention is to help students to understand the importance of a good algorithmic solutions. It is not essential understanding the most efficient solutions presented in this book.

- "Introduction to Algorihtms" by T.H. Cormen et al, chapter 1 complete and paragraphs 2.1 and 2.2 of chapter 2.

First session

Classroom activities (1h 30')

Minutes	Activity
20'	The teacher explains. Introduction to the analysis of algorithms, basic concepts explanation, and a brief discussion of the factors that influence the behavior of the algorithms (intrinsic factors of the algorithms versus programming-environment factors). Students will have available a copy of the slides to take notes.
5'	Break.
20'	The teacher explains. Introduction to both types of analysis of algorithms, namely, theoretical analysis or *a priori*, and experimental analysis or *a posteriori*.
15'	Formulation of three different versions of the same algorithm, with the goal that the students count basic operations and observe the differences among the different versions.
20'	Deducting the temporal cost function by counting basic operations for the three different algorithms for solving the same problem. Concept of input size of the problem.
10'	Resolution of doubts.

Outside classroom activities (up to 3h)

- Review of basic concepts explained in class by reading a few paragraphs of chapters of books listed below. Concepts to be reviewed: analysis of algorithms, input size, resource consumption (CPU time, RAM), counting of basic operations.

- Study of new concepts such as: sample of input data or instance of the problem, program step, temporal cost function, asymptotic behavior, asymptotic notation.

- Review and expand knowledge by reading chaper 12 (sections 12.1 up to 12.5, specially section 12.5) of "Empezar a programar usando Java", or chapter 2 (except paragraph 2.2.2) of the book "Introducció a l'anàlisi i disseny d'algorismes" by F.J. Ferri et al.

- In addition, "Estructuras de Datos en Java" by Mark Allen Weiss, Addison-Wesley:

  - Review the reading of chapter 5, paragraphs 5.1, 5.2, 5.3* and 5.4
  - Recall that Section 5.3 discusses a more complex problem than usual in a first year of computer programming. Our intention is to help students to understand the importance of good algorithmic solutions. It is not essential to understand the most efficient solutions.

- You can extend your knowledge with the book "Fundamentos de Algoritmia" by G. Brassard and P. Bratley, Pearson/Prentice Hall:

  - Chapter 2, paragraphs 2.1, 2.2 and 2.3

- Complementary reading: "Introduction to Algorihtms" by T.H. Cormen et al, chapter 1 complete, paragraphs 2.1 and 2.2 of chapter 2, and paragraph 3.1 of chapter 3.

- Deliverable #2.1 – First phase of the puzzle 2 –

  Individually, each student has to analyze one of the following traversal algorithms on unidimensional arrays: (a) Calculating the average value of all components of a vector of real values (`double`), (b) Find the maximum value in a vector of integers (`int`), and (c) Computing the scalar product of two vectors of integers, both vectors must be of the same size.

  These exercises were studied in the topic on arrays of IIP from previous semester. Following, you have the source code of each algorithm.

Source code for deliverable #2.1

```java
class Deliverable1
{
    public static double average( double A[] )
    {
        double sum=0.0;
        for( int i=0; i < A.length; i++ ) {
            sum += A[i];
        }
        return sum / A.length;
    }
    public static int maximum( int V[] )
    {
        int max = V[0];
        for( int i=1; i < V.length; i++ ) {
            if ( V[i] > max ) max = V[i];
        }
        return max;
    }
    public static int scalarProduct( int A[], int B[] )
    {
        // Testing size equality: A.length == B.length
        if ( A.length != B.length ) {
            System.err.println( "Impossible computing the scalar product "
                              + "of two different vectors." );
            return 0;
        }

        int product=0;
        for( int i=0; i < A.length; i++ ) {
            product += A[i] * B[i];
        }
        return product;
    }
}
```

Source code for deliverable #2.1

Each member of a workgroup must analyze one of the three algorithms by completing the following steps:

1. express with his own words what is the input size of the problem,
2. count basic operations and try to obtain the temporal cost function, and
3. write down the questions that arise during reading and during the analysis of the algorithm.

- This deliverable must be submitted by email to the teacher before the beginning of the next section. And a hard copy must be available for the expert meeting –second phase of the puzzle 2– during the next session.

Second session

Classroom activities (1h 30')

Minutes	Activity
30'	Expert meeting. – Second phase of the puzzle 2 – Those students who have solved the same problem meet in groups of four or five members for sharing their solutions, helping themselves answering questions raised individually, interchanging ideas, and resolving together the problem in order to obtain a better solution. Teacher solves doubts to the expert groups.
20'	The teacher explains. Concept of input size. Concept of temporal cost function, or running time, depending on the input size. Critical instruction. Introduction to asymptotic notation.
30'	Workgroup meets. – Third phase of the puzzle 2 – Each member explains to the others members of the workgroup the algorithm he analyzed during ten minutes, and how the analysis was done. The improved solution generated in the expert meeting must be used. In the next classroom session will be performed a test where each member of a workgroup should analyze the two algorithms he didn't analyze in the previous phases.
10'	Resolution of doubts.

Outside classroom activities (up to 3h)

- Review of concepts and study of new ones, such as program step, elementary operation, or critical instruction, different cases analysis, best-case analysis and worst-case analysis, and efficiency. We recommend the reading of the sections of books which are detailed next:
  - Reviewing and extending knowledge by reading sections 12.4 and 12.5 of "Empezar a programar usando Java", or the section 2.2 of chapter 2, except 2.2.2 of the book "Introducció a l'anàlisi i disseny d'algorismes" by F.J. Ferri et al.
  - As an alternative, you can read sections 5.4, 5.5 and 5.6.1 of chapter 5 of the book "Estructuras de Datos en Java" by Mark Allen Weiss, Addison-Wesley.
  - And as another alternative, you can read sections 2.4, 2.5 and 2.6 of chapter 2 of the book "Fundamentos de Algoritmia" by G. Brassard and P. Bratley, Pearson/Prentice Hall.
  - Alternative and/or complementary reading "Introduction to Algorihtms" by T.H. Cormen et al, paragraphs 2.1 and 2.2 of chapter 2.
- Deliverable #2.2 – Individually, each student must analyze the linear search algorithm for integer vectors. To do that he must apply the following steps:
  1. Expressing, by using his own words, what is the input size of the problem, and obtain the expression that defines it.
  2. Pointing out three possible critical instructions at least, i.e., those elementary operations which may represent a program step.
  3. Determine if there are significant instances, i.e., if the algorithm executes a different number of elementary operations depending on input data for the same size of the input.
  4. Get the temporal cost function, $T(n)$, where n represents the input size of the problem. If the algorithm presents significant instances, then, the different cases analysis must be done, and they must be computed $T^b(n)$ and $T^w(n)$, for the best-case and worst-case respectively.
  5. Express the upper and lower bounds, by using asymptotic notation, of the behavior for each of the temporal cost functions obtained.
- Submit this deliverable by email to the teacher before the next session.

Third session

Classroom activities (1h 30')

Minutes	Activity
20'	Individual test. Deliverable #2.3 – Fourth phase of the puzzle 2 – Each student must analyze the two algorithms he didn't analyze in the previous phases of the puzzle 2, the ones his workgroup mates explained to him. See at the end of this document how the grade for the puzzle 2 is computed.
25'	The teacher explains. Definitive version of the analysis of the linear search algorithm, including the average-case analysis.
5'	Break.
30'	Resolution of some problems in order to show the significance of efficient algorithm design, and the effect of intrinsic factors of algorithms versus the programming environment factors.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Study of sorting algorithms by reading the sections of books detailed next:
  - Chapter 13 of "Empezar a programar usando Java".
  - From book "Introducció a l'anàlisi i disseny d'algorismes" by F.J. Ferri et al, chapter 5, paragraphs 5.1 to 5.3 both included.
  - Alternative for Spanish speakers, from book "Estructuras de Datos en Java" by Mark Allen Weiss, Addison-Wesley: chapter 8, paragraphs 8.1, 8.2 and 8.3
  - Complementary reading from book "Fundamentos de Algoritmia" by G. Brassard and P. Bratley, Pearson/Prentice Hall: chapter 2, paragraphs 2.4 and 2.7.2
  - Complementary reading from book "Introduction to Algorihtms" by T.H. Cormen et al.: chapters 1 and 2, and paragraph 3.1 of chapter 3.
- Deliverable #2.4 – Analysis of the selection sort algorithm applying the same steps were applied for deliverable #2.2.

  This deliverable must be submitted to the teacher by email before the beginning of the fifth session.

Fourth session

Classroom activities (1h 30')

Minutes	Activity
30'	The teacher explains. The sorting problem for data stored in vectors. Explanation of the sorting algorithms *"selection sort"* and *"insertion sort"*.
5'	Break.
20'	The teacher explains. Analysis of the insertion sort algorithm.
5'	Break.
20'	The teacher explains. Partial or complete sorting of a vector by using the *"Bubble-Sort"* algorithm.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Chapter 13 of "Empezar a programar usando Java".
- Book "Introducció a l'anàlisi i disseny d'algorismes" by F.J. Ferri et al, chapter 3, except paragraphs from 3.3.2 to 3.3.5.

- Alternative, “Estructuras de Datos en Java” by Mark Allen Weiss, Addison-Wesley, chapter 5, paragraph 5.6.2
- Alternative, “Fundamentos de Algoritmia” by G. Brassard and P. Bratley, Pearson/Prentice Hall, chapter 4, paragraphs 4.1, 4.2, 4.3 and 4.4. Except the parts dedicated to “Torres de Hanoi” and “Cálculo de determinantes” of section 4.4, and except paragraph 4.2.3 of section 4.2.
- Alternative and/or complementary reading “Introduction to Algorihtms” by T.H. Cormen et al, and paragraphs 4.1 of chapter 4.

Fifth session

Classroom activities (1h 30’)

Minutes	Activity
20’	Students analyze the algorithm for obtaining the n-th element of the sequence of Catalan numbers, defined as: $$C_0 = 1 \quad ; \quad C_{n+1} = \frac{2*(2n+1)}{n+2} C_n \quad \text{or} \quad C_n = \frac{2*(2n-1)}{n+1} C_{n-1}$$ Concretely the iterative version, see below the source code.
20’	The teacher analyzes the recursive version of the algorithm for computing the Catalan numbers.
5’	Break.
15’	The teacher analyzes the Bubble-Sort algorithm presented in the previous classroom session.
5’	Break.
15’	The teacher explains. Analysis of algorithms with a computational cost which is logarithmic with respect to the input size, such as (1) counting the number of digits of an integer value, and (2) computing the logarithm, truncated to an integer, of a positive number in any base greater than 1.
10’	Resolution of doubts.

Source code of the iterative version of *CatalanNumber(n)*

```
public static int iterative( int n )
{
    int c=1;

    for( int i=1; i <= n; i++ ) {
        c = (2*(2*i-1)*c)/(i+1);
    }

    return c;
}
```

Source code of the iterative version of *CatalanNumber(n)*

Source code of the recursive version of *CatalanNumber(n)*

```
public static int recursive( int n )
{
    if ( n == 0 )
        return 1;
    else
        return (2*(2*n-1)*recursive(n-1))/(n+1);
}
```

Source code of the recursive version of *CatalanNumber(n)*

Outside classroom activities (up to 2h)

Reviewing and extending knowledge.

- "Empezar a programar usando Java", chapter 12, sections 12.6 and 12.7.
- Book "Introducció a l'anàlisi i disseny d'algorismes" by F.J. Ferri et al, chapter 3, except paragraphs from 3.3.2 to 3.3.5.
- Book "Fundamentos de Algoritmia" by G. Brassard and P. Bratley, Pearson/Prentice Hall, chapter 4, paragraph 4.2.3
- Alternative and/or complementary reading "Introduction to Algorihtms" by T.H. Cormen et al, and paragraph 4.1 of chapter 4.
- Deliverable #2.5 – Individually, each student should analyze the recursive version of the algorithm for computing the factorial of n, and try to analyze the recursive version of the algorithm for computing the *Fibonacci* succession. Both algorithms are contained in the boxes below.

  This deliverable must be submitted to the teacher by email before the beginning of the sixth session.

Source code for $n!$

```

// We assume n is always greater than or equal to zero.
int factorial( int n )
{
    if ( n > 1 )
        return n*factorial(n-1);
    else
        return 1;
}
```

Source code for $n!$

Source code of *Fibonacci(n)*

```

// We assume n is always greater than or equal to zero.
int Fibonacci( int n )
{
    if ( n > 1 )
        return Fibonacci(n-1) + Fibonacci(n-2);
    else
        return n; // Fibonacci(0) = 0, Fibonacci(1) = 1
}
```

Source code of *Fibonacci(n)*

Sixth session

Classroom activities (1h 30')

Minutes	**Activity**
25'	The teacher explains. Analysis of recursive algorithms: *Fibonacci*.
5'	Break
25'	The teacher solves problems related to the analysis of recursive and iterative algorithms. Problems will be extracted from the problem set of this chapter.
5'	Break
20'	The teacher explains. The binary search algorithm.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Deliverable #2.6 – To be done by the workgroups.

  The workgroups must code a program which reads two sorted vectors of integers from two different files, and builds up a new vector containing the sorted fusion of the two sorted vectors. The Natural-Merge algorithm must be used.

- The workgroups must code the methods `sort()` and `naturalMerge()` by using the source code presented below as starting point.

Incomplete source code for the deliverable #2.6

```java

import java.util.*;
import java.io.*;

class Deliverable1_6
{
    public static void main( String args[] )
    {
        int A[] = loadFromFile( "fichero1.txt" );
        int B[] = loadFromFile( "fichero2.txt" );

        sort( A );
        sort( B );

        int C[] = naturalMerge( A, B );

        saveToFile( "fichero3.txt", C );
    }
    static int [] loadFromFile( String fileName )
    {
        int V[] = null;
        try {
            Scanner sf = new Scanner( new File( fileName ) );
            int n = sf.nextInt();
            V = new int [n];
            int i=0;
            while( sf.hasNext() ) {
                V[i++] = sf.nextInt();
            }
            sf.close();
        }
        catch( IOException ioe )
        {
            ioe.printStackTrace( System.err );
            System.exit(-1);
        }

        return V;
    }
    static void saveToFile( String fileName, int V[] )
    {
        try {
            PrintWriter pw = new PrintWriter( new File( fileName ) );
            pw.println( V.length );
            for( int i=0; i < V.length; i++ ) pw.print( V[i] + " " );
            pw.close();
        }
        catch( IOException ioe )
        {
            ioe.printStackTrace( System.err );
            System.exit(-1);
        }
    }
    static void sort( int V[] )
    {
    }
    static int [] naturalMerge( int A[], int B[] )
    {
    }
}
```

Incomplete source code for the deliverable #2.6

- For coding the `sort()` method you can choose one of the following algorithms: selection sort, insertion sort or Bubble-Sort.

- Example of operation. If the contents of files 1 and 2 is the following:

```
file1.txt
7
4 56 2 1 5 7 9
```

```
file2.txt
8
9 4 5 8 5 2 1 3
```

the program should create a new file (`file3.txt`) as follows:

```
file3.txt
15
1 1 2 2 3 4 4 5 5 5 7 8 9 9 56
```

- This deliverable consist in a complete Java program, so that it must be sent to the professor via electronic mail, including the name of the workgroup and the name of each member.

Seventh session

Classroom activities (1h 30')

Minutes	Activity
20'	The teacher solves doubts about the Natural-Merge algorithm and analyzes it.
5'	Break.
30'	The teacher explains the Merge-Sort algorithm in detail showing how the Natural-Merge is used. Students use their notebooks for coding the Merge-Sort algorithm and testing it in the classroom.
5'	Break.
20'	The teacher analyzes the Merge-Sort algorithm using the substitution method, useful for analyzing recursive algorithms.
10'	Resolution of doubts.

Outside classroom activities (up to 2h)

- Each workgroup codes the binary search algorithm and includes it in the program of the deliverable #2.6.
- Students solve problems from the problem set for this chapter. This activity can be performed individually or in group. Then, they have to formulate questions that the teacher will answer in the next classroom session.

Eighth session

Classroom activities (1h 30')

Minutes	Activity
25'	The teacher analyzes the binary search algorithm.
5'	Break.
60'	The remaining time is used for solving doubts and answering questions formulated by students, and for analyzing new algorithms.

Outside classroom activities (up to 2h)

- Students solve problems from the problem set for this chapter. This activity can be performed individually or in group. Then, they have to formulate questions that the teacher will answer in the next classroom session.

Ninth session

Classroom activities (1h 30')

Minutes	Activity
90'	Class dedicated to answer questions submitted by students and to analyze algorithms of any kind (iterative, recursive, sort, search, etc.), with pauses of 5 minutes every 20 minutes.

Outside classroom activities

- No activities are planned after the last session of this topic, check the teaching guide for the next topic, you will find activities planned to be performed before the first session.

5 Issues and problems that, minimally, should be studied in this topic

- The analysis of linear algorithms, such as the ones for vector traversal, and for searching a value in a vector.
- The analysis of algorithms which have different behavior depending on the distribution of input data for the same size of the input, by performing the worst-case analysis and the best-case analysis.
- The analysis of the Natural-Merge algorithm used to obtain a new sorted vector which is the fusion of two previously sorted vectors.
- Implementation and analysis of sorting algorithms, such as the *Selection Sort* algorithm, the *Insertion Sort* algorithm, the *Bubble-Sort* algorithm, and the *Merge-Sort* algorithm.
- Implementation and analysis of the binary search algorithm.

6 Deliverables

Id	Description	Modality	Submission date	Points
#2.1	Analysis of one of three iterative algorithms.	Individually	By email before the beginning of the second session.	20
#2.2	Analysis of the linear search algorithm.	Individually	By email before the beginning of the third session.	40
#2.3	Individual test of puzzle 2 using the algorithms of the deliverable #2.1	Individually	During the third session.	80
#2.4	Analysis of the selection sort algorithm.	Individually	By email before the beginning of the fifth session.	30
#2.5	Analysis of the recursive algorithms for computing $n!$ and $Fibonacci(n)$.	Individually	By email before the beginning of the sixth session.	30
#2.6	Implementation of the Natural-Merge algorithm and a sorting algorithm.	In group	By email before the beginning of the seventh session.	30

7 Qualification

The grade for all activities performed in this topic will be computed as the sum of the points of each deliverable. The grade of activities performed in group will be the same for each member of the workgroup. Particularly, the mark obtained in the puzzle 2, $MP2$, is computed as follows:

$$GM = (MOI_1 + MOI_2 + MOI_3)/3$$

$$MP2_i = (GM + MOI_i)/2$$

$$MP2_i = \begin{cases} MP2_i += 0,1 \cdot TP & \text{if } GM \geq 0,6 \cdot TP \\ MP2_i & \text{if } GM < 0,6 \cdot TP \end{cases}$$

where MOI_i is the mark obtained individually in puzzle 2 by the i-th member of the group, calculated as the sum of the marks of deliverables #2.1 and #2.3, GM is the group average mark, $MP2_i$ is the mark assigned to i-th member, and TP is the total amount of points which can be obtained in puzzle 2. Notice how $MP2_i$ is increased if the average grade is good enough.

The knowledge of this subject will be assessed in the quizzes and in the lab practices, the grade obtained in the deliverables described herein is used to measure the work of formation, and is part of the NAS (Grade of the Follow-up Activities, the Spanish acronym is used), which, in turn, as said in the evaluation rules, contributes to the final score with a 20%.