ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: Big O notation

Ana Sofía Gutiérrez
Universidad Eafit
Medellín, Colombia
asgutierrt@eafit.edu.co

Santiago Hidalgo Ocampo
Universidad Eafit
Medellín, Colombia
shidalgoo1@eafit.edu.co

3. Practice for the final project defense presentation

3.1.1. Insertion Sort

n (Array Size)	Time (miliseconds)
100000	1566
115000	1802
130000	2307
145000	2888
160000	3518
175000	4231
190000	5757
205000	6643
220000	7544
235000	8508
250000	8636
265000	9714
280000	10855
295000	12057
310000	16223
325000	16983
340000	16893
355000	20181
370000	22534
385000	20606
400000	22634

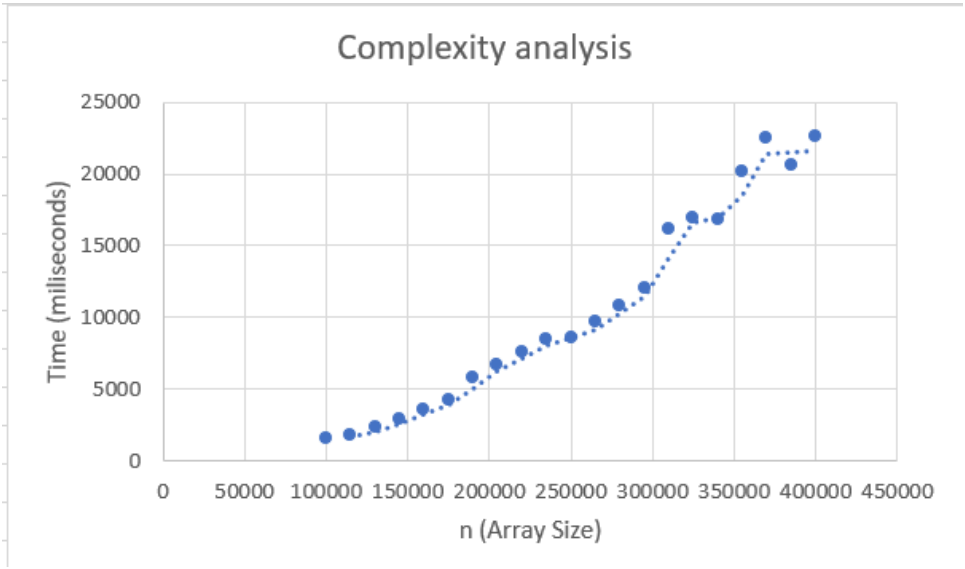
3.1.2. Merge Sort

n (Array Size)	Time (miliseconds)
100000	0
115000	0
130000	15
145000	16
160000	15
175000	16
190000	15
205000	16
220000	31
235000	16
250000	16
265000	31
280000	32
295000	31
310000	31
325000	31
340000	31
355000	31
370000	47
385000	31
400000	31

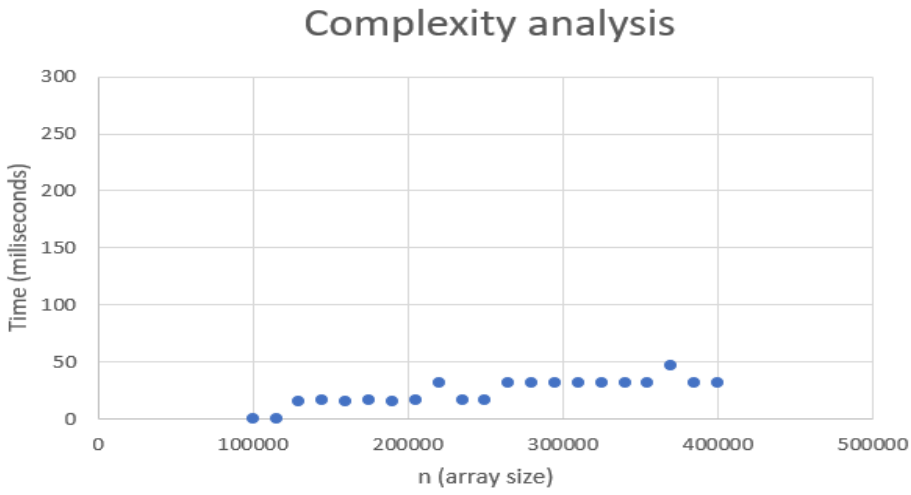
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.2

Insertion Sort:



Merge Sort:



3.3

Merge Sort is much more efficient than Insertion Sort. Merge Sort has a complexity of O(n*Log (n)), on the other hand, insertion Sort has a polynomial complexity $O(n^2)$. Insertion Sort is not appropriate for a database with millions of elements because its algorithmic complexity leads to a high time cost.

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

3.4
Max Span

If the leftmost value is equal to the rightmost value and the array has at least 1 element, then the interval is the length of the given array, otherwise the interval (span) is the length of the array-1.

3.5

Array 2

- **CountEvens**
  T(n) = C_1+ n*(C_2+ C_3+C_4) + C5
  T(n) is O (C_1+ n*(C_2+ C_3+C_4) + C5)
  T(n) is O (n* (C_2+ C_3+C_4)) by addition rule
  T(n) is O(n) by multiplication rule
- **Sum13**
  T(n) = (C1*n) +C2
  T(n) is O((C1*n) +C2)
  T(n) is O(C1*n), by the addition rule
  T(n) is O(n) by multiplication rule
- **Lucky13**
  T(n) = (C1*n) + C2
  T(n) is O((C1*n) + C2)
  T(n) is O(C1*n), by the addition rule
  T(n) is O(n), by multiplication rule
- **No14**
  T(n) = (C1*n) + C2
  T(n) is O((C1*n) + C2)
  T(n) is O(C1*n), by the addition rule
  T(n) is O(n), by multiplication rule
- **Has22**
  T(n) = (C1*n) + C2
  T(n) is O((C1*n) + C2)
  T(n) is O(C1*n), by the addition rule
  T(n) is O(n), by multiplication rule

Array 3

- **MaxSpan**
  T(n) = C
  T(n) is O(C)
  T(n) is O(1)

- **CanBalance**
  T(n) = C*n^2 + C1*n +C2
  T(n) is O(C*n^2 + C1*n +C2)

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

T(n) is O(C*n^2+C1*n), by the addition rule
T(n) is O(C*n^2), by the addition rule
T(n) is O(n^2), by multiplication rule

- **SeriesUp**
  T(n) = C*n^2 + C1*n +C2
  T(n) is O(C*n^2 + C1*n +C2)
  T(n) is O(C*n^2+C1*n), by the addition rule
  T(n) is O(C*n^2), by the addition rule
  T(n) is O(n^2), by multiplication rule

- **CountClumps**
  T(n) = C*n^2 + C1*n +C2
  T(n) is O(C*n^2 + C1*n +C2)
  T(n) is O(C*n^2+C1*n), by the addition rule
  T(n) is O(C*n^2), by the addition rule
  T(n) is O(n^2), by multiplication rule

- **Fix34**
  T(n) = C*n^2 + C1*n +C2
  T(n) is O(C*n^2 + C1*n +C2)
  T(n) is O(C*n^2+C1*n), by the addition rule
  T(n) is O(C*n^2), by the addition rule
  T(n) is O(n^2), by multiplication rule

3.6

Array2:

- **CountEvens:** n is the length of the array
- **Sum13:** n is the length of the array.
- **Lucky13:** n is the length of the array.
- **No14:** n is the length of the array.
- **Has22:** n is the length of the array.

Array3:

- **MaxSpan:** n does not exist, complexity is constant
- **CanBalance:** n is the length of the array.
- **SeriesUp:** n is the positive integer given.
- **CountClumps:** n is the length of the array.
- **Fix34:** n is the length of the array.

4) Practice for midterms

4.1 c) O(n+m)
4.2 d) O(m*n)
4.3 b) O(ancho)

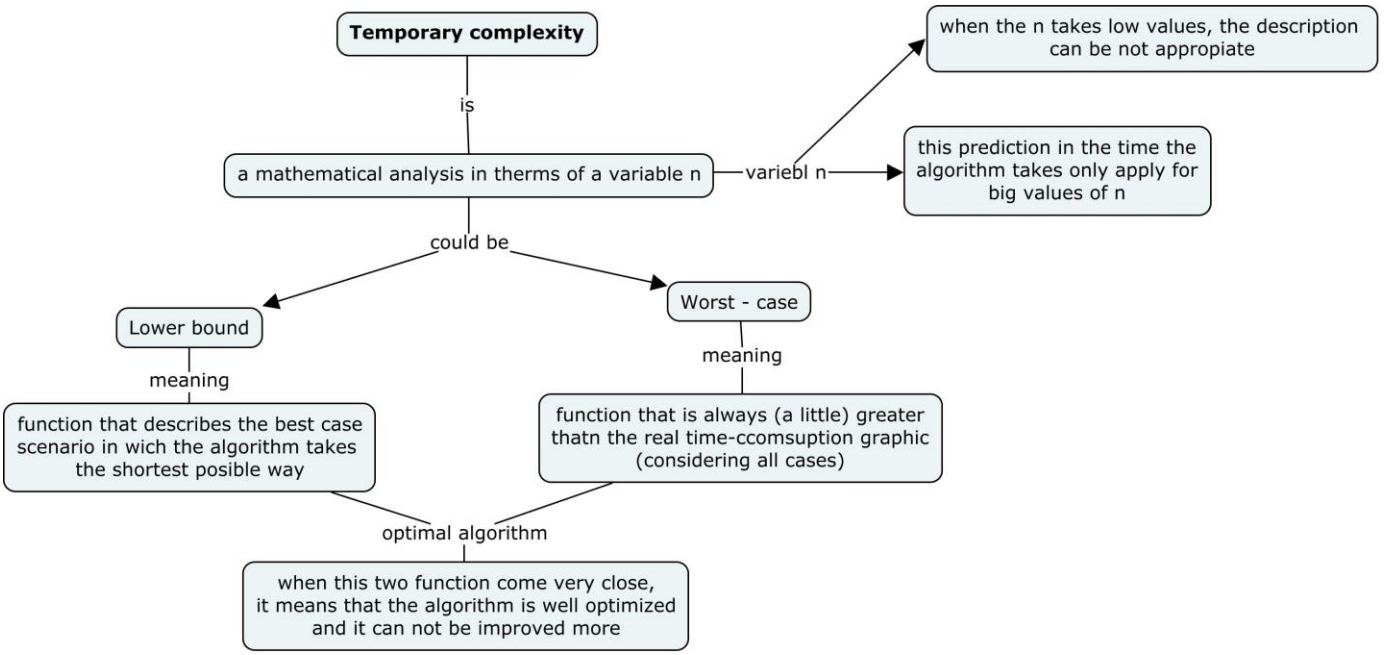
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473

4.4 *b) O(n3)*
4.5 *d) O(n2)*
4.6 a) T(n)=T(n−1)+C
4.7
4.7.1 T (n) = c + T (n - 1)
4.7.2 O (n).
4.8 a) It executes T (n) = C + T (n - 1) steps, which is O (n).
4.9 d) Executes more than n × m steps.
4.10 c) Executes less than n*logn steps //complexity 4.10: O(n)
4.11 c) Execute T (n) = T (n − 1) + T (n − 2) + c steps.
4.12 b) O(m × n × log(n) + n × m^2 + n^2 × log(n) + m^3)
4.13 c) T(n) = 2T(n/2) + n
4.14 a) O(n^3 + n(log(log(m)) + m × √m)

5) Introduction to algorithm´s analysis and design by R.C.T Lee et al.

Temporary complexity as an algorithm´s performance time analysis:
A mathematical analysis of the number of operations is made and written in terms of the size n of the problem, generally conditioned in such a way that O (g (n)) is somewhat greater than the real time of execution at all values of n.
Another important factor is the constant that accompanies the n (C_n*n); it is usually withdrawn since it loses importance as n grows, but it is not correct to say that in all cases a problem O(n) is faster (or less complex) that one O (n^3), for a small n, the times may have a different behavior and this is mostly determined by the constants C_n.
For different problems such as sorting, several solution algorithms like quick sort, insert sort and merge sort are presented, the difference between them being the time it takes to execute the task for which they were created and, according to what we seek to solve, the best will be the one that performs the least operations.
Another way to measure how good an algorithm is, is considering its level of complexity, if in the shortest time considered it performs fewer operations then it is simple, otherwise it is difficult. The way to express this mathematically is known as lower bound and is denoted by Ω. The smallest number of operations that the algorithm performs in certain conditions is analyzed, excluding trivial cases, such as performing only one operation or Ω(1); the most significant lower bound is selected and, citing the author "An algorithm is optimal if its temporal complexity(O) is equal to the lower bound (Ω) of this problem; then, it is no longer possible to improve even more the lower bound nor the algorithm "

PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473



PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627
Phone: (+57) (4) 261 95 00 Ext. 9473