ESTRUCTURA DE DATOS 1
Código ST0245

Laboratory practice No. 2: Big O notation

Juan Sebastián Díaz Osorio
Universidad Eafit
Medellín, Colombia
jsdiazo@eafit.edu.co

Liz Oriana Rodrigues Cruz
Universidad Eafit
Medellín, Colombia
lorodriguc@eafit.edu.co

3) Practice for final project defense presentation (next page)

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

3.1 It is answered in next point.

3.2 ***Merge sort algorithm***

```
public static void merge(int A[], int izq, int m, int der) {
    int i, j, k;
    int[] B = new int[A.length];
    for (i = izq; i <= der; i++) B[i] = A[i];

    i = izq;     j = m + 1;      k = izq;

    while (i <= m && j <= der) {
        if (B[i] <= B[j]) A[k++] = B[i++];
        else A[k++] = B[j++];
    }
    while (i <= m) {
        A[k++] = B[i++];
    }
}
```

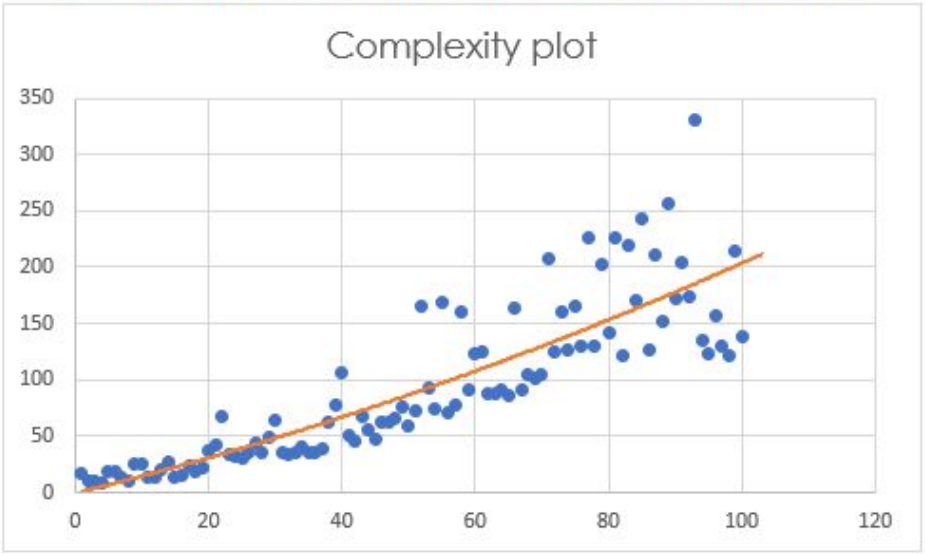
A[] is the array to sort

izq is the leftmost index

m is the middle index

der is the rightmost index

It seems a processor error, but theorically it is O(n*log n)



n (array.length)	Time (ms)
528083	17
904682	10
1854298	11
2030452	8
2171803	19
2682016	18
3045521	14
3114132	10
3472305	26
3504502	26
4350722	13
4611501	13
4827052	20
4936269	27
5039976	13
5238214	15
5873203	23
6488802	19
6865162	22
7101490	38

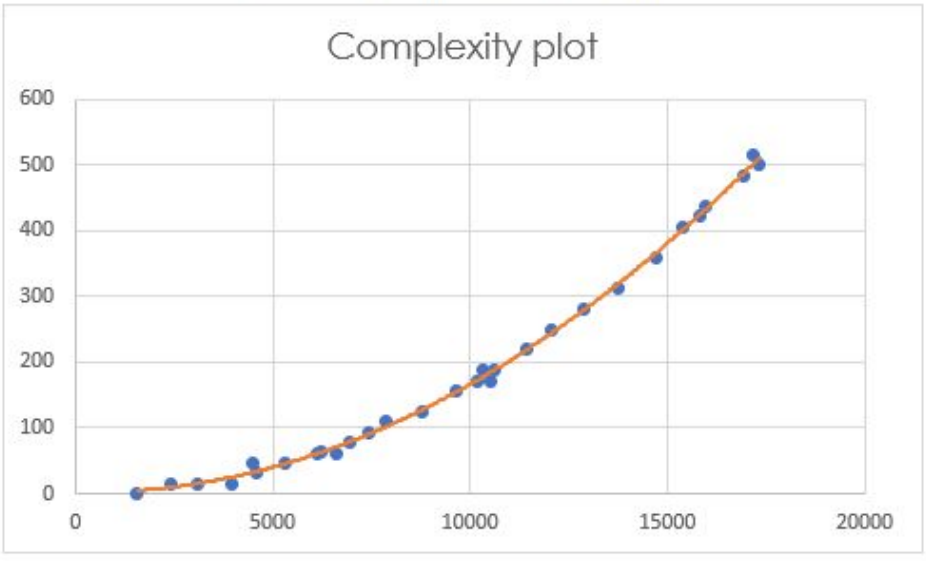
PhD. Mauricio Toro Bermúdez
Professor | School of Engineering | Informatics and Systems
Email: mtorobe@eafit.edu.co | Office: Building 19 – 627 Phone: (+57) (4) 261 95 00 Ext. 9473

Insertion sort algorithm

```java
public static int[] rsort(int[] arr) {
    return rsort(arr, 1);
}

public static int[] rsort(int[] arr, int start) {
    if (arr.length == start) {
        return arr;
    }
    return rsort(insert(arr, start), start + 1);
}

private static int[] insert(int[] arr, int start) {
    if (start == 0) {
        return arr;
    }

    if (arr[start - 1] > arr[start]) {
        arr[start] += arr[start - 1];
        arr[start - 1] = arr[start] - arr[start - 1];
        arr[start] -= arr[start - 1];
        return insert(arr, start - 1);
    }

    return arr;
}
```

arr[] is the array to sort

It seems O(n^2) and theorically it is



n (array.length)	Time (ms)
1553	0
2440	15
3071	16
3952	16
4511	47
4580	31
5324	47
6111	62
6230	63
6592	62
6929	78
7435	94
7858	109
8783	125
9650	156
10194	172
10315	188
10521	172
10631	187
11439	219

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

3.3 Merge sort is such a good algorithm for big numbers, because its complexity plot shows a slightly increment while n increases.

3.4 We can use merge sort in videogames when we manage a big amounts of data.

3.5 The answer depends on memory importance. There is memory problems when we try to manage less data with merge sort, though, because this algorithm uses recursion. In that case, we could use a different sort algorithm, like insertion sort.

3.6 CodingBat's maxSpan algorithm consists in going over an array length "x" from left to right until the array's length. At the same time the array is gone over from right to left. In this process, the algorithm compares if the value in the arrays positions in both operations is the same. Finally, the program prints the maximum span or range between the both equal values.

Examples:

1. **Array = [1, 2, 1, 1, 3], Array.length = 5**

Pos1 = 0 Pos2 = 4
¿ 1 = 3 ? X
Pos1 = 0 Pos2 = 3
¿ 1 = 1 ? V
(3 - 0) + 1
4 V
...
Pos1 = 2 Pos2 = 3
¿ 1 = 1 ? V
(3 - 2) + 1
2 X
...
Return 4

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

2. **Array = [1, 4, 2, 1, 4, 1, 4], Array.length = 7**

Pos1 = 0 Pos2 = 6
¿ 1 = 4 ? X
Pos1 = 0 Pos2 = 5
¿ 1 = 1 ? V
(5 - 0) + 1
6 V
...
Pos1 = 3 Pos2 = 5
¿ 1 = 1 ? V
(5 - 3) + 1
3 X
...
Return 6

3.7 *Array 2*

countEvens: T(n) = n + C_1, es O(n)

bigDiff: T(n) = n + C_1, es O(n)

centeredAverage: T(n) = n + C_1, es O(n)

sum13: T(n) = n + C_1, es O(n)

sum67: T(n) = n + C_1, es O(n)

Array 3

maxSpan: T(n) = ∑n(n - i) [i=0 to n] + C_1, es O(n^2)

fix34: T(n) = ∑n(n - 2 - i) [i=1 to n - 2] + C_1, es O(n^2)

fix45: T(n) = n^2 + C_1, es O(n^2)

canBalance: T(n) = ∑n(n - i) [i=0 to n] + C_1, es O(n^2)

linearIn: T(n) = n + C_1, es O(n)

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

3.8 ***Array 2***

countEvens: O(n) -> ***n*** is the **array length**

bigDiff: O(n) -> ***n*** is the **array length**

CenteredAverage: O(n) -> ***n*** is the **array length**

Sum13: O(n) -> ***n*** is the **array length**

Sum67**:** O(n) -> ***n*** is the **array length**

Array 3

maxSpan: $O(n^2)$ -> ***n*** is the **array length**

fix34: $O(n^2)$ -> ***n*** is the **array length**

fix45: $O(n^2)$ -> ***n*** is the **array length**

canBalance: $O(n^2)$ -> ***n*** is the **array length**

linearIn: O(n) -> ***n*** is the **inner[ ] length**

4) Practice for midterms

4.1 c)

4.2 d)

4.3 b)

4.4 b)

4.5 d)

4.6 a)

4.7 Answers:

4.7.1 T(n) = n - 1

4.7.2 O(n) = n

4.8 a)

4.9 d)

4.10

4.11 c)

4.12 b)

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

4.13 c)

4.14 a)

5) Recommended reading (optional)

5.1 This point was skipped.

6) Team work and gradual progress (optional)

We meet once only. This is the record: https://bit.ly/2MJuJlJ

This is the progress report with github commits and Kanban board screenshots https://bit.ly/34cJSBs

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