


| Algorithmics | Student information   | Date  | Number of session |
|--------------|-----------------------|---|-------------------|
|              | UO: 296503            | 03/03/2025  | 3                 |
|              | Surname: Mulet Alonso |  Escuela de Ingeniería Informática<br>Universidad de Oviedo |                   |
|              | Name: Sergio          |   |                   |



## Activity 1. Divide and Conquer by subtraction

- Subtraction1:

It stops giving times at  $n = 8192$ , due to `stackOverflowError` as it is mentioned in the pdf of the practice “Subtraction1.java and Subtraction2.java classes have an approach by subtraction with  $a=1$ , which involves a large expenditure of stack memory”, it overflows.

- Subtraction2:

It stops exactly at the same size, it makes sense since it has the same problem as Subtraction1.

- How many years will take for Subtraction3 complete  $n=80$ ?

For calculating this we have to make the following operation:  $T(n) = a * T(n-1)$

The last measurable value I got for Subtraction3 is  $n=30$   $t=38705$ ms.

$$T(80) = 38705 * 2^{(80-30)} = 4.3578 * 10^{19} \text{ ms} = 1.380.931.845 \text{ years}$$

- Subtraction4

We have to implement a divide and conquer by subtraction algorithm with complexity  $O(n^3)$ , in my case I implemented an algorithm with  $a=1$ ;  $b=1$ ;  $k=2$ .

As  $a=1$  the complexity is  $O(n^{k+1})$  which is  $O(n^3)$

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

```
public static long rec4(int n) {
    long cont = 0;
    if (n <= 0)
        cont++;
    else {
        for(int i = 0; i < n*n; i++) { //O(n^2)
            cont++;
        }
        rec4(n - 1); //a=1; b=1; k=2 --> O(n^3)
    }
    return cont;
}
```

Times I got:

| n    | t(ms) |
|------|-------|
| 100  | 1     |
| 200  | 11    |
| 400  | 86    |
| 800  | 713   |
| 1600 | 5845  |
| 3200 | 47420 |
| 6400 | OoT   |

Times match the complexity, let's take for instance  $n=1600$ , with this size the algorithm finished at 5845 milliseconds, by multiplying the size by 2, I got a time which is more or less  $2^3$  times bigger ( $n=3200$ ;  $t = 47420$ ),  $47420/5845 = 8,11$ .

- Subtraction5:

We have to implement a divide and conquer by subtraction algorithm with complexity  $O(3^{n/2})$ , for that, we have to implement an algorithm with  $a=3$  and  $b=2$  (in my case  $k=0$ )

As  $a > 1$  the complexity is  $O(a^{n/b})$ .

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

```

public static long rec5(int n) {
    long cont = 0;
    if (n <= 0)
        cont++;
    else {
        cont++;
        rec5(n - 2);
        rec5(n - 2);
        rec5(n - 2);
    }
    return cont;
}

```

Times I got:

| n  | t(ms) |
|----|-------|
| 30 | 399   |
| 32 | 1181  |
| 34 | 5535  |
| 36 | 10763 |
| 38 | 32119 |
| 40 | OoT   |

For n=30 we get a time of 399ms, this match the complexity since as we increase n by 2, times are multiplied more or less by 3, and this is because n is divided by b, which is 2, so it is growing by one and then being multiplied by 3 make sense.

- How many years will take for Subtraction5 complete n=80?

For calculating this we have to make the following operation:  $T(n) = a * T(n-1)$

The last measurable value I got for Subtraction5 is n=38 t=32119ms.

$$T(80) = 32119 * 2^{(80-38)} = 1,4126 * 10^{17} \text{ ms} = 4.476.351,201472 \text{ years}$$

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

## Activity 2. Divide and conquer by division

- Division1

It has  $O(n)$  complexity, we don't get big times for a real measure because it is too fast, but we can appreciate how by multiplying the size by two, times get multiplied by two also, for instance, when  $n = 262144$   $t = 3$  and when  $n = 524288$   $t = 6$ .

At the end of execution, we get  $n = 4194304$   $t = 37$  and then  $n = 8388608$   $t = 74$ . This is the best case in which we see how times follow the complexity.

- Division2

It has  $O(n \log n)$  complexity, in this case times grow more or less as in Division1, maybe this is due to the waste of stack, which is  $O(\log n)$ .

For instance,  $n = 4194304$   $t = 1059$  and  $n = 8388608$   $t = 2155$ , which is slightly more than the double.

- Division3

It has  $O(n)$  complexity, and it is clearly shown by times.

For instance,  $n = 524288$   $t = 66$  and  $n = 1048576$   $t = 137$ . By increasing the size by two, the time increases also by two, then it follows the linear complexity.

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

- Division4

| n     | t(ms) |
|-------|-------|
| 1000  | 6     |
| 2000  | 27    |
| 4000  | 94    |
| 8000  | 349   |
| 16000 | 1447  |
| 32000 | 5858  |
| 64000 |       |

Times follow the complexity  $O(n^2)$  clearly

- Division5

| n     | t(ms) |
|-------|-------|
| 1000  | 26    |
| 2000  | 106   |
| 4000  | 427   |
| 8000  | 1750  |
| 16000 | 7090  |
| 32000 | 27632 |
| 64000 | OoT   |

In my case, I have implemented an algorithm with  $a=4$ ,  $b=2$  and  $k=1$ , so  $a > bk$ , then the complexity is  $O(n^{\log_b a}) \Rightarrow O(n^2)$

Times follow the quadratic complexity.

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

## Activity 3. Two basic examples.

- Vector

All the measurements are with 1000000 as number of repetitions.

Sum 1 (iterative)

| n     | t(ns) |
|-------|-------|
| 3     | 40    |
| 6     | 64    |
| 12    | 93    |
| 24    | 142   |
| 48    | 245   |
| 96    | 438   |
| 192   | 817   |
| 384   | 1575  |
| 768   | 3111  |
| 1536  | 6156  |
| 3072  | 12289 |
| 6144  | 24575 |
| 12288 | 49382 |
| 24576 | OoT   |

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

#### Sum2 (D&C by subtraction)

| n    | t(ns) |
|------|-------|
| 3    | 84    |
| 6    | 137   |
| 12   | 227   |
| 24   | 398   |
| 48   | 745   |
| 96   | 1461  |
| 192  | 2881  |
| 384  | 5976  |
| 768  | 13259 |
| 1536 | 27890 |
| 3072 | 55784 |
| 6144 | OoT   |

#### Sum3 (D&C by division)

| n    | t(ms) |
|------|-------|
| 3    | 90    |
| 6    | 195   |
| 12   | 404   |
| 24   | 825   |
| 48   | 1721  |
| 96   | 3403  |
| 192  | 7046  |
| 384  | 14269 |
| 768  | 28780 |
| 1536 | 53457 |

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

The three approaches follow the linear complexity, however it seems that for this specific problem, the iterative approach is the best of them. This could be because since it is not a big problem, using divide and conquer may be useless (in fact it is worse).

- Fibonacci

Fib1 (iterative solution  $O(n)$ )

| n  | t(ns) |
|----|-------|
| 10 | 93    |
| 11 | 87    |
| 12 | 94    |
| 13 | 98    |
| 14 | 106   |
| 15 | 108   |
| 20 | 134   |
| 40 | 229   |
| 59 | 300   |

Fib2(Iterative solution using a vector, dynamic programming)

| n  | t(ms) |
|----|-------|
| 10 | 111   |
| 11 | 122   |
| 12 | 130   |
| 13 | 132   |
| 14 | 145   |
| 15 | 149   |
| 20 | 184   |
| 40 | 370   |
| 59 | 482   |



| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

- Fib3(D&C by subtraction)

| n  | t(ns) |
|----|-------|
| 10 | 202   |
| 11 | 218   |
| 12 | 234   |
| 13 | 251   |
| 14 | 256   |
| 15 | 270   |
| 20 | 358   |
| 40 | 640   |
| 59 | 928   |

Fib4(D&C by subtraction)

| n  | t(ns) |
|----|-------|
| 10 | 2436  |
| 11 | 3970  |
| 12 | 7093  |
| 13 | 11013 |
| 14 | 17788 |
| 15 | 29146 |
| 16 | 45632 |
| 17 | OoT   |

For fib1, fib2 and fib3 is hard to see the linear complexity, as we are increasing size one by one and they are small values, we can see that times follows the complexity more or less from n=20 to n=40, for instance fib2 n=20 t=184 and n=20 t =370.

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

For this problem, as it happened with vector, the iterative solution is the best option.

Dynamic programming is faster than D&C by subtraction.

Fib4 is the worst case, as it follows an exponential complexity  $1,6^n$  and times show this perfectly, by increasing just by 1 the problem size, we are multiplexing the times by 1.6, for instance,  $n=11$   $t = 3970$  and  $n=12$   $t = 7093$ . We must try to avoid this type of complexity.

## 4. Petanque championship organization

First of all, the basic case is when  $n=2$ , we have to assign for instance Juan vs Pepe / Pepe vs Juan.

```
if (size == 2) {
    matches[start][0] = names[start + 1];
    matches[start + 1][0] = names[start];
    return;
}
```

To reach this base case, my approach is D&C by division, dividing the number of participants by 2 and organize the left part and the right part individually, that is, if I have 8 participants I dive first into first half and second half, so we have a problem size of 4, as this is not the basic case, I still have to dive each of them to a size of 2.

```
int mid = size / 2;
organize(start, mid);
organize(start + mid, mid);
```

Finally, after the recursive call, we have to combine the results.

```
for (int day = 0; day < mid; day++) {
    for (int i = 0; i < mid; i++) {
        int left = start + i;
        int right = start + mid + ((i + day) % mid);
        matches[left][day + mid - 1] = names[right];
        matches[right][day + mid - 1] = names[left];
    }
}
```

I have some trouble with the right part of the participants because it is easy for the first elements, but once you pass the middle it is more confusing.

| Algorithmics | Student information   | Date       | Number of session |
|--------------|-----------------------|------------|-------------------|
|              | UO: 296503            | 03/03/2025 | 3                 |
|              | Surname: Mulet Alonso |            |                   |
|              | Name: Sergio          |            |                   |

Finally, to calculate the complexity:

$a = 2$ ;  $b = 2$ ;  $k = 2$ , then  $\Rightarrow a < b^k$  then the complexity is  $O(n^k)$  which ends up being a quadratic complexity, which is quite good.

However, my time measurements don't reflect that.

| n     | t(ms) |
|-------|-------|
| 2     | 0     |
| 4     | 0     |
| 8     | 0     |
| 16    | 0     |
| 32    | 0     |
| 64    | 0     |
| 128   | 1     |
| 256   | 1     |
| 512   | 5     |
| 1024  | 19    |
| 2048  | 79    |
| 4096  | 330   |
| 8192  | 1555  |
| 16384 | 11044 |
| 32768 | 82890 |
| 65536 | OoM   |

OoM = out of memory (exception)

```
public class CalendarTimes {
    public static void main(String arg[]) throws IOException {
        long t1, t2;
        for(int n=2; n < Integer.MAX_VALUE; n*=2) {
            Calendar calendar = new Calendar(generateNames(n));
            t1 = System.currentTimeMillis();
            calendar.organize(0, n);
            t2 = System.currentTimeMillis();
            System.out.println("n=" + calendar.getNumberOfParticipants() + " **TIME=" + (t2 - t1));
        }
    }
}
```