# Práctica 4

Juan Llinares Mauri
74011239E

## 1. Práctica 4 - Complejidad temporal: Cálculo analítico (II)

### 1.1. Ejercicio 2

```
void abstracto(unsigned n) {
    if (n > 1) {
        for (unsigned i = 1; i < n - 1; i++)
            cout << "*";
        cout << endl;

        for (unsigned i = 0; i < 4; i++)
            abstracto(n / 2);
    }
}
```

Este método no contiene mejor y peor caso. Realiza un printeo de asteriscos hasta $n-1$ y realiza una llamada recursiva con $\frac{n}{2}$ como parámetro. La recursión termina cuando $n <= 1$. Tenemos entonces que:

$$T(n) = \begin{cases} 1, & n \leq 1 \\ n - 1 + 4T(\frac{n-1}{2}), & n > 1 \end{cases}$$

Usando sustitución, podemos resolver la ecuación de la siguiente manera:

$$f(n) = n-1+4T(\frac{n-1}{2}) = (n-1)-1+4*4T(\frac{(n-1)-1}{2}) = n-2+16T(\frac{n-2}{2}) = ...$$

$$= n - k + 4kT(\frac{n-k}{2})$$

Para obtener $T(1)$, la variable $k$ deberá: $1 = \frac{n-k}{2} \rightarrow 2 = n - k \rightarrow k = n - 2$. Entonces: $T(n) = n-(n-2)+4(n-2)T(\frac{n-(n-2)}{2}) = 2+(4n-8)T(1) = 2+4n-8 = 4n - 6$. Tenemos así que la complejidad temporal de este algoritmo es de $O(n)$.