Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Nicolás Lahsen (nflahsen@uc.cl)
Florencia Ferrer (fpferrer@uc.cl)
Lothar Droppelmann (ldroppelmann@uc.cl)

IIC2133 — Estructuras de datos y algoritmos — 2’ 2020

Ayudantía 2

1 Merge Sort

Merge Sort es un algoritmo de ordenación utilizando el método *divide and conquer*, que involucra la división recursiva del arreglo por la mitad, ordenándolas por separado y luego haciendo un *merge* de ellas.

function MergeSort(A,i,f):
  if $f - i \geq 1$ **then** ▷ Si es que puedo volver a dividir
    $m = \frac{i+f}{2}$
    A_1 = MergeSort($A, i, \lfloor m \rfloor$)
    A_2 = MergeSort($A, \lceil m \rceil, f$) ▷ Ordena los 2 subarreglos recursivamente
    A[i,f] = **Merge(**A_1**,**A_2**)** ▷ Uno los subarreglos ordenadamente
    return A[i,f]
  end if
end function

La funcion *Merge* recibe como parámetro los dos subarreglos ordenados, y devuelve la unión ordenada de ambos. Puedes asumir que *Merge* es correcta y toma tiempo $\Theta(n)$.

1. Determina la correctitud del algoritmo

2 Binary Search

Binary Search es un algoritmo de búsqueda que encuentra la posición de un valor en un arreglo ordenado.

```
function BinarySearch(A,min,max,value):
    if max < min then
        return false
    else
        mid = (max+min)//2
        if A[mid] > value then
            return BinarySearch(A, min, mid-1, value)
        else if A[mid] < value then
            return BinarySearch(A, mid+1, max, value)
        else
            return true
        end if
    end if
end function
```

1. Determina la correctitud del algoritmo
2. Calcula su complejidad

3 Teorema Maestro

*Basado en la explicación del curso Matemáticas Discretas IIC1253, Diéguez y Suárez, 2019.

El teorema maestro es un método matemático utilizado para calcular complejidades en algoritmos del estilo *Divide & Conquer*, en términos de la **notación asintótica**. Estos algoritmos se analizan a partir de un umbral n_0, a partir del cuál se resuelve recursivamente el problema (Desde n_0 en adelante, el problema se comporta de tal forma, mínima cantidad de elementos para dividir $= \frac{b}{b-1}$). Por lo general, estos algoritmos dividen el input en una constante b, y luego aproximan a un entero (utilizando $\lceil\ \rceil$ o $\lfloor\ \rfloor$), haciendo a_1 y a_2 llamadas recursivas en cada caso. Además, por lo general se realiza un procesamiento adicional, antes o después de las llamadas recursivas, que llamaremos $f(n)$, la cual hace $c \cdot n^d$ pasos. Siguiendo lo anterior, el teorema enuncia lo siguiente:

Si a_1, a_2, b, c, c_0, d, $\in \mathbb{R}^+$, y $b > 1$, entonces para una recurrencia de la forma:

$$T(n) = \begin{cases} c_0 & 0 \leq n < n_0 \\ a_1 \cdot T(\lceil \frac{n}{b} \rceil) + a_2 \cdot T(\lfloor \frac{n}{b} \rfloor) + c \cdot n^d & n \geq n_0 \end{cases}$$

se cumple que

$$T(n) \in \begin{cases} \Theta(n^d) & a_1 + a_2 < b^d \\ \Theta(n^d \cdot log(n)) & a_1 + a_2 = b^d \\ \Theta(n^{log_b(a_1+a_2)}) & a_1 + a_2 > b^d \end{cases}$$

Utilizando el teorema, calcule la complejidad asintótica de los siguientes algoritmos:

- Binary-Search
- MergeSort