Ejercicios y problemas (ALGBIO)

Sandra Mingo Ramírez

2024/25

Ejercicio 1: Sort the following functions according to their growth:
$n, \sqrt{n}, n^{1.5}, n^2, n \log n, n \log \log n, n \log^2 n, 2/n, 2^n, 2^{n/2}, 37, n^2 \log n, n^3$

Para ordenar las funciones utilizando la notación O, hay que ver qué ocurre cuando n tiende a infinito. De forma ascendente:
$2/n, 37, \sqrt{n}, n, n \log \log n, n \log n, n \log^2 n, n^{1.5}, n^2, n^2 \log n, n^3, 2^{n/2}, 2^n$

Ejercicio 2: **Python** A classical example of a Divide and Conquer algorithm is binary search over ordered tables, which is essentially done according to the following pseudocode:

```
def bin_search(key, l_ints):
   m = len(l_ints)//2
   if key == l_ints[m]:
      return m
   elif key < l_ints[m]:
      search key on l_ints ip to index m-1 # left table search
   else:
      search key on l_ints from index m+1 # right table search
```

Expand the pseudocode into a correct recursive Python function.

```
def bin_search(key, l_ints):
   m = len(l_ints)//2
   if key == l_ints[m]:
      return m
   elif key < l_ints[m]:
      bin_search(key, l_ints[:m]
   else:
      bin_search(key, l_ints[m + 1:]
```

Ejercicio 3: Identify a proper key operation for `bin_search`. How many key comparisons are performed at most on the table [1, 2, 3, 4, 5, 6, 7] in successful searches? And in unsuccessful ones?

La operación clave es la comparación `key == l_ints`. En el mejor de los casos, solo se ejecuta una vez, pero en el peor de los casos, 3 veces.