Problema 4

- Linear search $O(n)$

Mejor caso
Se busca elemento en la primera posición

$1 \rightarrow O(1)$

Peor caso
Pasa si el elemento no está en el arreglo o está en la última posición

$n \Rightarrow O(n)$

Caso promedio
Si existe una distribución uniforme entre n y el número esperado

$$E[x] \Rightarrow \frac{n+1}{2}$$

$$\Rightarrow O(n)$$

Binary search → recurrencia → $T(n) = T(\frac{n}{2}) + \Theta(1)$

$\Rightarrow T(n) = \Theta(\log n)$

Mejor caso:
Primera posición

$1 \rightarrow O(1)$

Peor caso:

$2^{k-1} < n \leq 2^k$

$k = [\log_2 n] + 1$

$\therefore k = O(\log n)$

Caso promedio
Si la posición objetivo es cualquiera con probabilidad uniforme o si el elemento no no está (ó similar)

$\Theta(\log n)$

Quick sort

Peor caso

$T(n) = T(n-1) + cn$

$T(0) = T(1)$

$T(n) = T(n-1) + cn$

$= T(n-2) + c(n-1) + cn$

$= T(n-2) + c(n-1) + cn$

$= T(1) + c\sum_{k=2}^{n} k$

$T(n) = c_0 + c\left(\frac{n(n+1)}{2} - 1\right) -$

$= \theta(n^2)$

Mejor caso

$T(n) = 2T(n/2) + cn$

$a = 2$

$b = 2$

$n^{\log_2 2} = n$

$T(n) = 2T(n/2) + f(n)$

$T(n) = \theta(n \log n)$

Caso promedio

$$T(n) = \frac{1}{n} \sum_{0}^{n-1} (T(k) + T(n-b-1)) + cn$$

$$T(n) = \Theta(n \log n)$$