# Graded homework — Binary Search

Team - Berger - Nussbaum

2) Worst-time complexity of binary search


Worst case → Wenn das element im letzten Split gefunden wird.

$$T_A(n) \begin{cases} \Theta(1) & n \leq 1 \\ 1\, T_A(n/2) + \Theta(1) & \end{cases}$$

- Nur 1 Rekursion
- Problembereich wird halbiert
- Knotenarbeit

$a = 1$
$b = 2$
$c = 1$

$f(n) = \Theta(1) = \Theta(n^l \cdot (\log(n))^k)$
$\Rightarrow l = 0$
$k = 0$

$\Rightarrow \quad l = 0 \qquad \log_2(1)$
$\qquad\quad 0 = 0$

$\Rightarrow \Theta(f(n) \cdot \log(n))$
$\Rightarrow \Theta(1 \cdot \log(n))$
$\Rightarrow \Theta(\log(n))$

3) Die Zeitkomplexität steigt, wenn SinglyLinkedList verwendet werden.
→ Dies passiert deshalb, weil der Zugriff auf bestimmte Elemente (an bestimmten Positionen) nicht mehr konstant ist.
z.B. wenn die gewünschte Position = n/4 ist
→ Dann müssen vom 1. Element (n/4) - Schritte passieren um das Element zu betrachten

Bei DoublyLinked Lists ist das ganze gleich.
→ Bei DobblyLinkedLists können wir im Unterschied zur SinglyLinkedList auch von rechts her navigieren.
Bringt uns beim BinarySearch jedoch nicht viel, da das nächste gefragte Element wieder in der Mitte der beiden vorherigen Grenzen liegt.
→ Ob von links (Start) oder von rechts (Ende) navigiert wird, spielt keine Rolle.

4) Siehe README.MD & ../code ☺

5) Halbiere den Problembereich, bis das Problem trivial wird.