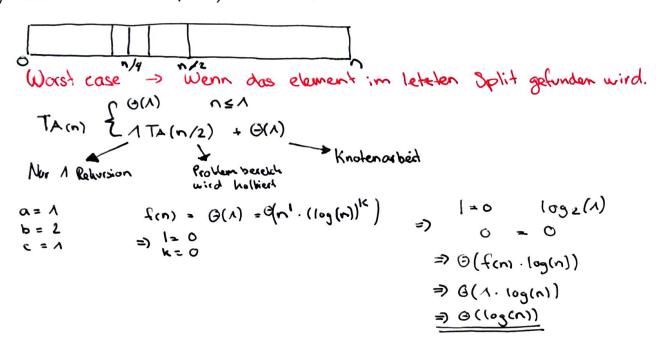
Graded homework - Binary search

Team-Berger-Nussbaum

1) Comment the code on the previous page with your understanding of it.

Line	Code	Comment	
1)	int binarySearch(int left , int right , E target) {	Function declaration	
2)	int mid = (left + right)/2;	Compute middle index	
3)	<pre>int comparison = comp.compare(target, a.get(mid));</pre>	Compare target with element at middle index	
		And save the comparison result.	
		(-1 = target < elem@middle)	
		(0 = target = elem@middle)	
		(1 = target > elem@middle)	
4)	if (right == left)	Check if right index is the same as left index	
5)	return (comparison <=0) ? left : right+1;	When comparison I smaller or equal to 0	Base case index overlap
		(target < elem@middle)	→ Decide whether left or right
		return left index, otherwise return right index +1	is correct position
6)	else if (comparison == 0 (comparison < 0 && left == mid))	When comparison is 0	Base case (same element is
		(target = elem@middle)	found OR left index is middle
		OR	index)
		comparison is smaller than 0	→ mid is the correct position
		(target < elem@middle) AND left index is the	
		same as middle index	
7)	return mid ;	Return middle index	
8)	else if (comparison < 0)	When comparison is smaller than 0	
		(target < elem@middle)	
9)	return binarySearch (left , mid -1, target);	Search in left part	Recursion in left part
10)	else		
11)	return binarySearch (mid +1, right , target);	Search in right part	Recursion in right part
12)	}		

2) Worst-time complexity of binary search



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

Bei DoublyLinked Lists ist das gamze gleich.

- -> Bei Dobbly LinkedLists Können wir im Unterschied Zur Singty Linked List auch von rechts her navigieren. Bringt uns bein Binary Search jedoch wicht viel, da das nachste gefragte Element wieder in der Mitte der beiden vorherigen Grenzen liegt. - Ob von links (Sturt) oder von rechts (Emde) navigiert wird, spiell heine Rolle.
- 4) Siehe README.MD & ../code "
- 5) Halbiere den Problembereich, bis das Problem trivial wird.