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) | int comparison = comp.compare(target, a.get(mid)); | 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  (target < elem@middle) return left index, otherwise return right index +1 | Base case index overlap  🡪 Decide whether left or right is correct position |
| 6) | else if (comparison == 0 || (comparison < 0 && left == mid)) | When comparison is 0 (target = elem@middle) OR comparison is smaller than 0 (target < elem@middle) AND left index is the same as middle index | Base case (same element is found OR left index is middle index)  🡪 mid is the correct position |
| 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) | } |  |  |