finds the position of a target value within a [sorted array](https://en.wikipedia.org/wiki/Sorted_array);

Binary search runs in at worst [logarithmic time](https://en.wikipedia.org/wiki/Time_complexity#Logarithmic_time), making *O*(log *n*) comparisons;

Binary search takes constant (*O*(1)) space, meaning that the space taken by the algorithm is the same for any number of elements in the array;

Given an array *A* of *n* elements with values or [records](https://en.wikipedia.org/wiki/Record_(computer_science)) *A*0, *A*1, ..., *An*−1, sorted such that *A*0 ≤ *A*1 ≤ ... ≤ *An*−1, and target value *T*, the following [subroutine](https://en.wikipedia.org/wiki/Subroutine) uses binary search to find the index of *T* in *A*.[[7]](https://en.wikipedia.org/wiki/Binary_search_algorithm#cite_note-FOOTNOTEKnuth1998%C2%A76.2.1_(%22Searching_an_ordered_table%22),_subsection_%22Algorithm_B%22-7)

1. Set *L* to 0 and *R* to *n* − 1.
2. If *L* > *R*, the search terminates as unsuccessful.
3. Set *m* (the position of the middle element) to the [floor](https://en.wikipedia.org/wiki/Floor_and_ceiling_functions), or the greatest integer less than (*L* + *R*) / 2.
4. If *Am* < *T*, set *L* to *m* + 1 and go to step 2.
5. If *Am* > *T*, set *R* to *m* − 1 and go to step 2.
6. Now *Am* = *T*, the search is done; return *m*.