MODULE 7 PROJECT: SEARCHING & SORTING

Learning Objectives:

CLO 3. Develop problem solving skills by implementing data structures.

CLO 4. Compare advantages and disadvantages of different data structure implementations.

Function Implementations

Files to Modify: algorithms.py

Directions:

1. Implement the function `linear_search(a, x)` which uses the linear search algorithm to return the index of element `x` if it is found in the `ArrayList` object `a`. If `x`, is not found in `a`, then the function returns -100.
2. Implement the function `binary_search(a, x)` which uses the binary search algorithm to return the index of element `x` if it is found in the *sorted* `ArrayList` object `a`. If `x`, is not found in `a`, then the function returns -100.
3. Implement the function `_merge(a0, a1, a)` which overwrites `ArrayList` object `a` by merging the elements of `ArrayList` object `a0` and `ArrayList` object `a1` in increasing order.
4. Implement the function `merge_sort(a)` which uses the merge-sort algorithm to sort the `ArrayList` object `a`.
5. Implement the helper function `_quick_sort_f(a, start, end)` which uses the quick-sort algorithm with the first element as pivot to sort the `ArrayList` object `a`.
6. Implement the helper function `_quick_sort_r(a, start, end)` which uses the quick-sort algorithm with a random element as pivot to sort the `ArrayList` object `a`.

   NOTE: You may introduce any additional helper functions your quick sort functions might need, as long as you do not change the parameters defined for each function.

SUBMISSION PROCESS

1. Submit your project to Repl.it
2. Submit to CodePost:
    - `algorithms.py`