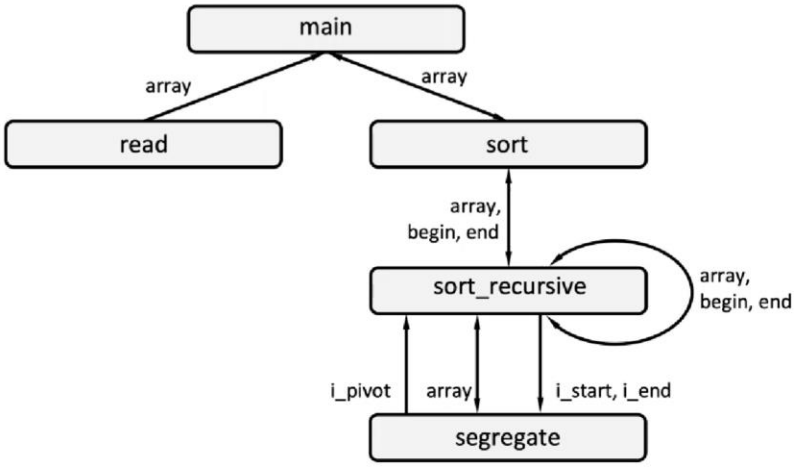
Braeden Pope

Modularization Design

Segregation Sort Design Analysis

Structure Chart



Function	Coupling	Cohesion
read	Simple. Array is the only variable exchanged, and no verification exists.	Strong. It reads the given file name into an array, and that's all.
sort	Simple. An array is passed from main, passed to sort_recursive, and then returned to sort and back to main.	Strong. It only calls the sort_recursive function.
sort_recursive	Interactive. The communication between sort_recursive and segregate, as well as with the recursive instances of sort_recursive call for frequent information exchange.	Extraneous. It calls itself to sort recursively, but also uses the segregate function to find the new pivot for each recursive iteration.
segregate	Interactive. There is frequent information exchanged between segregate and sort_recursive due to sort_recursive's recursive nature.	Extraneous. It sorts the array based on the given variables, but also find the new pivot value for sort_recursive.

Psuedocode

Pseudocode

```
sort(array)
   sort_recursive(array, 0, array.length)
```

Pseudocode

```
sort_recursive(array, i_begin, i_end)
   IF i_end - i_begin < 1 OR i_end < 0
      RETURN

   i_pivot ← segregate(array, i_begin, i_end)

   sort_recursive(array, i_begin, i_pivot -1)
   sort_recursive(array, i_pivot+1, i_end)
```

Pseudocode

```
segregate(array, i_start, i_end)
   IF i_begin = i_end
      RETURN i_begin

   i_pivot ← (i_begin + i_end) / 2
   i_up ← i_begin
   i_down ← i_end

   WHILE i_up < i_down
      WHILE i_up < i_down AND array[i_up] < array[i_pivot]
         i_up ++
      WHILE i_up < i_down AND array[i_down]  ≥ array[i_pivot]
         i_down  --

      IF i_up < i_down
         IF i_down = i_pivot
            i_pivot ← i_up
         ELSE-IF i_UP = i_pivot
            i_pivot ← i_down
         SWAP array[i_up], array[i_down]

   SWAP array[i_up], array[i_pivot]
   RETURN i_up
```

Algorithmic Efficiency: $O(n^2)$, as the larger the array is (or the more values it contains), the more recursions required to perform the function, exponentially increasing the time it takes for the program to run.

Test Cases

Test Case	Input	Expected Output
Empty	[]	[]
Singular	[8]	[8]
Small Sorted	[2,4]	[2,4]
Small Unsorted	[4,2]	[2,4]
Sorted Even	[5,7,9,10]	[5,7,9,10]
Reversed Even	[10,9,7,5]	[5,7,9,10]
Sorted Odd	[5,7,9,10,13]	[5,7,9,10,13]
Reversed Odd	[13,10,9,7,5]	[5,7,9,10,13]
Duplicates	[3,7,3,5,7]	[3,3,5,7,7]

Program Traces

Segregate Function Test Case: Small Unsorted, [4,2]

Line	array	i_begin	i_end	i_pivot	i_up	i_down
1	[4,2]	0	2	/	/	/
3	[4,2]	0	2	1	/	/
4	[4,2]	0	2	1	0	2
5	[4,2]	0	2	1	0	2
7	[4,2]	0	2	1	0	2
9	[4,2]	0	2	1	0	2
10	[4,2]	0	2	1	0	2
11	[4,2]	0	2	1	0	2