Diagram

Description automatically generated

get\_filename:

Cohesion: Strong, only gets the user’s input and returns it

Coupling: Trivial, no interchange exists

read\_file:

Cohesion: Strong, only reads the file and returns the raw data

Coupling: Simple, only receives one parameter, that parameter is just a string.

sort\_list:

Cohesion: Strong, only sorts the raw data into a sorted list

Coupling: Simple, receives the unsorted list as a parameter and returns the sorted list

display\_sorted\_list:

Cohesion: Strong, only displays the sorted list

Coupling: Simple, receives one parameter and that’s the sorted list

***Algorithmic Efficiency:***

get\_filename():

GET filename

RETURN filename

Efficiency: O(1)

read\_file( filename ):

READ filename 🡪 raw\_data

RETURN raw\_data

Efficiency: O(1)

sort\_list( raw\_data ):

size = raw\_data.length

source 🡨 raw\_data

dest = empty list, same size as “size”

num = 2

WHILE num > 1 // O(n) #1

num = 0

begin1 = 0

WHILE begin1 < size //O(n) NESTED: O(n^2) #2

end1 🡨 begin1 + 1

WHILE end1 < size AND src[end1 – 1] < OR = src[end1] #3 // O(n) NESTED O(n^3)

end1 += 1

begin2 🡨 end1

IF begin2 < size #4

end2 🡨 begin2 + 1

ELSE #5

end2 🡨 begin2

WHILE end2 < size AND src[end2 – 1] < OR =src[end2] #6// O(n) NESTED O(n^3)

end2 += 1

num += 1

combine(src, des, begin1, begin2, end2)

begin1 🡨 end2

SWAP src and des pointers #7

Estimated Overall Efficiency: O(2n^3)

combine(source, destination, begin1, begin2, end2)

end1 🡨 begin2

FOR i 🡨 begin1 + begin2 // O(n)

IF begin1 < end1 AND

begin2 = end2 OR source[begin1] < source[begin2]

destination[i] 🡨 source[begin1]

begin1 += 1

ELSE

destination[i] 🡨 source[begin2]

begin 2 += 1

RETURN destination

Efficiency: O(n)

display\_sorted\_list( destination )

PUT destination

Efficiency: O(1)

***Test Cases:***

1: [31, 72, 32, 10, 95, 50, 25, 18]

2: [-10, 13, 98, 35, 14, 30, 22]

3: [10, 11, 12, 13, 14, 15]

4: []

5: [0, 0, 0, 0, 0, 0]

6: [14, 77, 18, 6, 89, 25, 22, 19]

***Trace: Test Case 6***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | begin1 | begin2 | src[end1] | src[end -1] | src[end2] | src[end2 – 1] | des |
| 1. | - | - | - | - |  |  |  |
| 2. | 0 | - | - | - |  |  |  |
| 3. | 0 | - | 77 | 14 |  |  |  |
| 4 | 0 | - | 77 | 14 |  |  |  |
| 5. | 0 | 3 | 77 | 14 | 6 | 18 |  |
| 6. | 0 | 3 | 77 | 14 | 6 | 18 |  |
| 7. | 0 | 3 | 77 | 14 | 6 | 18 | [14,18,77,6] |
| 2. | 1 | - | - | - |  |  |  |
| 3. | 1 | - | 89 | 6 |  |  |  |
| 4 | 1 | - | 89 | 6 |  |  |  |
| 5. | 1 | 4 | 89 | 6 | 22 | 25 |  |
| 6. | 1 | 4 | 89 | 6 | 22 | 25 |  |
| 7. | 1 | 4 | 89 | 6 | 22 | 25 | [14, 18, 77, 6, 25, 89 ] |
| 2. | 2 | - | - | - |  |  |  |
| 3. | 2 | - | 22 | 19 |  |  |  |
| 4 | 2 | - | 22 | 19 |  |  |  |
| 5. | 2 | 5 | 22 | 19 | 22 | 25 |  |
| 6. | 2 | 5 | 22 | 19 | 22 | 25 |  |
| 7. | 2 | 5 | 22 | 19 | 22 | 25 | [14, 18, 77, 6, 25, 89, 19, 22] |
| 2. | 0 | - | - | - |  |  |  |
| 3. | 0 | - | 77 | 18 |  |  |  |
| 4 | 0 | - | 77 | 18 |  |  |  |
| 5. | 0 | - | 77 | 18 | 89 | 25 |  |
| 6. | 0 | - | 77 | 18 | 89 | 25 |  |
| 7. | 0 | - | 77 | 18 | 89 | 25 | [6, 14, 18, 25, 77, 89, 19, 22] |
| 2. | 1 | - | - | - |  |  |  |
| 3. | 1 | - | 89 | 77 |  |  |  |
| 4 | 1 | - | 89 | 77 |  |  |  |
| 5. | 1 | - | 89 | 77 | 22 | 19 |  |
| 6. | 1 | - | 89 | 77 | 22 | 19 |  |
| 7. | 1 | - | 89 | 77 | 22 | 19 | [6, 14, 18, 19, 22, 25, 77, 89] |