PROTIP:

Reviewed by Walrus

Julia Kozak, Nina Jiang, Diana Akhmedova
APCS pd08
HW46
2021-12-10
time Spent: 1.0 hrs

- method addInOrder takes a Superarray & newVal (integer)
- has a for loop that checks each element of the Superarray (use SuperArray.getO), then when it reaches the first value greater than newVal, it uses add-at-inde at that point.

This is what we got!
Very clear explanation.

- method arrange takes a Superarray
- creates a new instance of Superarray for temporary storage of terms in the Super array we're arranging
  - uses set() for temporary super array, get() for arranging SuperArray
- within a while loop that checks if temporary size is >0
  - find minimum term in temporary (set first term to min, then have a for loop check if each next term is less than min, in which case set it to min (and record min index))
  - set the arranging array's next term to the minimum
  - remove the min. from temporary

*possible failure: the method of finding the minimum is incorrect

Makes sense & works but might be slower than other methods.

example

SuperArray a

0	1	2	3	4
7	3	0	4	2

example
SuperArray b | 0 | 4 | 7 | 8 |

addInOrder(b,5)
b[0] > 5? false
b[1] > 5? false
b[2] > 5 true
↳ add-at-index(2,5) → b | 0 | 4 | 5 | 7 | 8 |

arrange(a) → temp. | 7 | 3 | 0 | 4 | 2 |
min = 7
↳ min = 3
↳ min = 0
a. | 0 | 3 | 0 | 4 | 2 |
temp. | 7 | 3 | 4 | 2 |
min = 7
↳ min = 3
↳ min = 2
a | 0 | 2 | 0 | 4 | 2 |
temp. | 7 | 3 | 4 |
min = 7
↳ min = 3
a | 0 | 2 | 3 | 4 | 2 |
temp | 7 | 4 |
min = 7
↳ min = 4
a | 0 | 2 | 3 | 4 | 2 |
temp | 7 |
min = 7
a | 0 | 2 | 3 | 4 | 7 |
temp []
↓ exit while loop

The examples are very clear!