

Team Frogs (Ruby Friedman, Ivina Wang, Samantha Hua)
APCS
HW46 -- Wrap the Wrapper
2021-12-09
Time Spent: 1.0 hrs

Reviewed by: SWAG SWASH

1st strategy:

Start by creating a "placeholder" array that contains all the values of Salay, our instance of SuperArray.

Use a for loop with a increment i that starts at 0 and increases by one for each iteration as long as i is less than `Salay.length`.

Use a helper function called `minIndex` which uses a for loop to find the index of the minimum value in a given array.

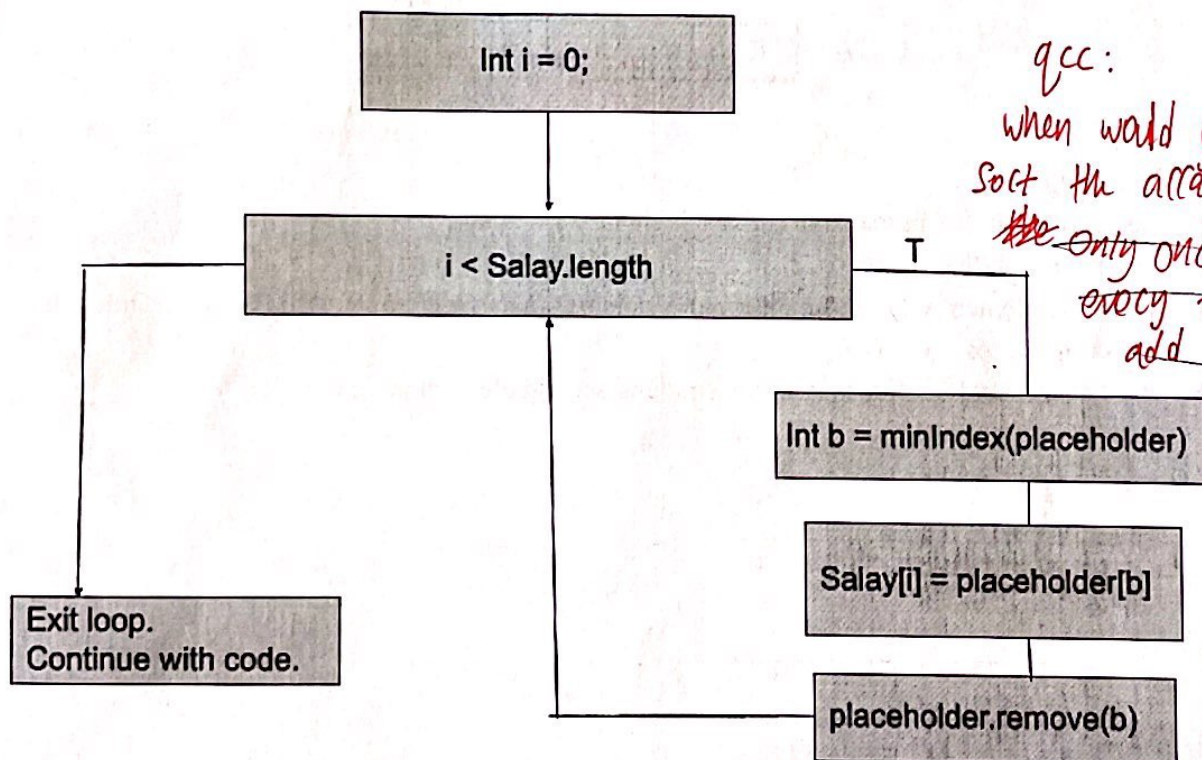
Find the `minIndex` of placeholder and replace `Salay[i]` with the value of `placeholder[minIndex]`.

Remove the `minIndex` from array placeholder using the `remove` method.

Continue looping through the for loop, now there will be a new `minIndex` in placeholder because the old value was removed.

Continue to append the new values of min index on to `Salay` in increasing order.

Placeholder has the same values as the inputted array, but does NOT point to the same array.



gcc:
when would you
sort the array?
~~the~~ only once or
every time you
add something
never

Salay

5	3	7	20	51
---	---	---	----	----

 i starts at 0
placeholder

5	3	7	20	51
---	---	---	----	----

↳ minIndex → 3 (sets salary[0] to minIndex)

Salay

3	3	7	20	51
---	---	---	----	----

placeholder

5	3	7	20	51
---	--------------	---	----	----

↳ removes the minIndex(3)

i > 1 → i = 1

placeholder

5	7	20	51
---	---	----	----

↳ minIndex → 5

Salay

3	5	7	20	51
---	---	---	----	----

sets index[1] = 5 & removes 5 from placeholder

placeholder

7	20	51
---	----	----

etc.

2nd strategy

An outer for loop that iterates through each element in Salay

An inner for loop that compares each element to the one that comes before it

- If the element with the smaller index is larger than the element with the larger index, they swap places
- This process continues until the first and second element are compared

Foo mega

SuperArray Salay

[51,5,7,2,28,75,4]

```
for (int i=1; i < Salay_size; i++) {
    for (int j=i; j > 0; j--) {
        If statement to compare the value
        at the given index (something like
        salary.get(i) < salary.get(i-1) )
        a old value var (to save one
        of the values when swapping)
        Swap certain values of Salay
    }
}
```


Salary

51	5	7	2	28	75	4
----	---	---	---	----	----	---

$i=1$ \swarrow compared these, (1st for loop)
and $51 > 5 \rightarrow$ swap values

Salary

5	51	7	2	28	75	4
---	----	---	---	----	----	---

 \swarrow nothing swapped from the 2nd loop yet

$i=2$ \swarrow compare $51 < 7 \rightarrow$ swap values

Salary

5	7	51	2	28	75	4
---	---	----	---	----	----	---

$i=3$ $j=3$ \swarrow compare $51 < 2 \rightarrow$ swap values

Salary

5	7	2	51	28	75	4
---	---	---	----	----	----	---

 $i=3$ $j=2$

\swarrow second for loop goes backwards to check if the previous elements are in order

$7 > 2 \rightarrow$ swap values

Salary

5	2	7	51	28	75	4
---	---	---	----	----	----	---

 $i=3$ $j=1$

\swarrow compares $5 > 2 \rightarrow$ swap

Salary

2	5	7	51	28	75	4
---	---	---	----	----	----	---

 $i=3$ $j=0 \rightarrow$ stops 2nd for loop

continues back to first for loop and $51 < 28$, etc.

{ }

add(3)

{ 3 }

add(6)

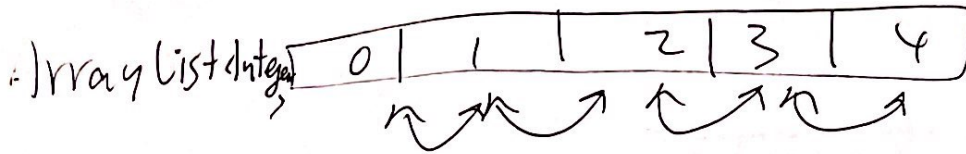
{ 3, 5 }

add(4)

{ 3, 4, 5 }

the sorting should return the index at which the new thing should be added

sorted or Not



Compare these values

compareTo will return -1 if the value on the right is larger than the value on the left.

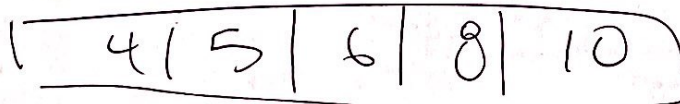
~~add Binary~~ add Binary

ArrayList
<Integer>

empty array - element should just be added



one item in the array - must compare the newVal to the singular value
↳ either it goes before it
↳ or after it

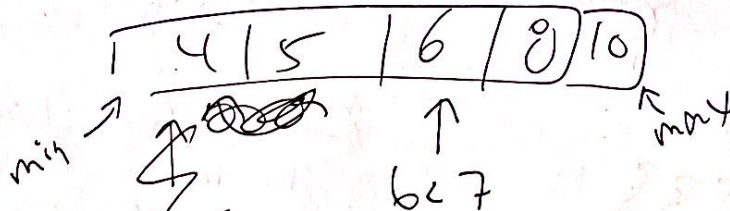
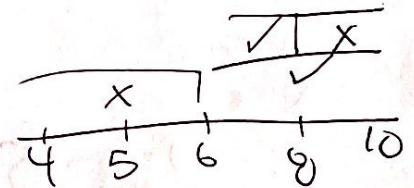


newVal = 7

min = 0

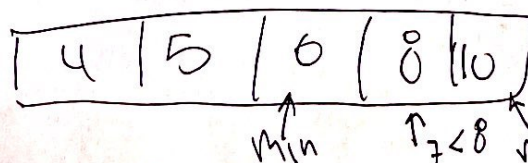
max = 4 → index of last value in the array

$$\text{mid} = \frac{(4-0)}{2} = 2$$



so we only look at the top half and min becomes mid + 1

then



so we decrease the max (max = mid - 1)

add Binary (cont.)

4	5	6	8	10
---	---	---	---	----

min mid max

~~mid~~

$6 < 7$ so $\text{min} = \text{mid} + 1$

4	5	6	8	10
---	---	---	---	----

min mid max

$\text{min} = \text{max}$ so we add newVal at mid

4	5	6	7	8	10
---	---	---	---	---	----

add Linear

2	5	9	15	17
---	---	---	----	----

~~newVal~~ newVal = 10

$2 < 10$

$5 < 10$

$9 < 10$

$10 < 15$!!! add newVal at $i = 2$

min = 0

max = 10

$$mid = \frac{(min + max)}{2}$$

while {

min ++

max ++

}

min = updated

max = updated

a = 5

b = 9

c = a + b

c = 14

mid needs to be updated inside the while loop because ~~it is not updated~~
~~it is not updated~~
~~it is not updated~~
min + max are looking at the values of those variables.