Section 9

explain insertion sort algorithm

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

```
Insert(int[] sorted, int value)
  initialize i to 0
  WHILE value < sorted[i]
    set i to i + 1
  WHILE i < sorted.length
    set temp to sorted[i]
    set sorted[i] to value
    set value to temp
    set i to i + 1
  append value to sorted

InsertionSort(int[] input)
  LET sorted = New Empty Array
  sorted[0] = input[0]
  FOR i from 1 up to input.length
    Insert(sorted, input[i])
  return sorted
```

2 23>8

i=2

S[2] X => error

insert value insorted arr

insert an arr of int to sort

Section 10

Trace 1 , [8,4,23,42,16,15]

insertionSorrt([8,4,23,42,16,15])

sorted = [] empty list

append index[0] to sorted

step 1

sorted =[8]

0 1 2 3 4 5

[8,4,23,42,16,15]

index	input[i]	value	i	temp	sorted.lenght
1	4	4	0	8	1
		8	1		

for i=1 to n

insert (sorted ,value at index 1)

4>8 --> false ,

i < sorted.length --> true

set temp =8

sorted[i]=4

value=8

i +=1 =>1

sorted =[4,8]

step 2

index	input [i]	value	i	temp	sorted.lenght
1	23	23	0		2
2			1		
			2		

23 > 4 && 23 && 8 -> true

end with error index out of range => exit

add if statement to stop loop

i <sorted.length -> false

sorted =[4,8,23]

Section 11

step 3

sorted =[4,8,23]

index	input [i]	value	i	temp	sorted.lenght
1	42	23	0		3
2		42	1		
3			2		

sorted =[4,8,23,42]

16 >23 --> false

i=2< 4 --> true

index	input [i]	value	i	temp	sorted.lenght
	16	16	2	16	4
		23		23	
		15			

sorted =[4,8,16,42]

sorted =[4,8,16,23,42]

step 4

sorted =[4,8,16,23,42]

sorted =[4,8,15,16,23,42]

Section 12

- example 2 , Reverse-sorted: [20,18,12,8,5,-2]

step 1

i=1

sorted=[20]

sorted=[18,20]

insertionsort

i	input[i]
1	18

=>

insert first loop

i	value	sorted [i]
0	18 >	20

false =>

insert second loop

i	len sorted	temp	value	sorted [i]
0 <	1	20	18	20
0			20	18
1				20

[20,18,12,8,5,-2]

step 2

i=2

sorted=[18,20]

sorted=[12,18,20]

i	input[i]
2	12

=>

i	value	sorted [i]
0	12 >	18

false =>

i	len sorted	temp	value	sorted [i]
0 <	2	18	18	12
1 <	2	20	20	18
2 <	2			20

step 3

i=3

sorted=[12,18,20]

sorted=[8,12,18,20]

i	input[i]
3	8

=>

i	value	sorted [i]
0	8 >	12

false =>

i	len sorted	temp	value	sorted [i]
0 <	3	12	12	8
1 <	3	18	18	12
2 <	3	20	20	18
3 <	3			20

step 4

i=4

sorted=[5,8,12,18,20]

step5

i=5

sorted=[-2,5,8,12,18,20]

last values

5	20	20	18
5	6		20

Section 14

- Few uniques: [5,12,7,5,5,7]

step 1

i=1

sorted=[5]

sorted=[5,12]

insertionsort

i	input[i]
1	12

=>

insert first loop

i	value	sorted [i]
0	12 >	5
1	12	breack

=>

insert second loop

i	len sorted	temp	value	sorted [i]
1 <	1			
	false			

step2

i=2

sorted=[5,12]

sorted=[5,7,12]

i	input[i]
2	7

=>

i	value	sorted [i]
0	7 >	5
1	7 >	12

false =>

i	len sorted	temp	value	sorted [i]
1 <	2	12	12	7
2 <	2			12

step 3 step 4

i=3 i=4

sorted=[5,7,12]

sorted=[5,5,5,7,12]

step 5

i=5

sorted=[5,5,5,7,12]

i	input[i]
5	7

=>

i	value	sorted [i]
0	7 >	5
1	7 >	5
2	7 >	5
3	7 >	7

false =>

i	len sorted	temp	value	sorted [i]
3 <	5	7	7	7
4 <	5	12	12	7
5 <	5			12

sorted=[5,5,5,7,7,12]

Section 15

- Nearly-sorted: [2,3,5,7,13,11]

step1 sorted=[2]

i=1

step2 sorted=[2,3]

i=2

step 3 sorted=[2,3,5]

i=3

step 4 ,5 sorted=[2,3,5,7]

i=4 ,5

sorted=[2,3,5,7,11,13]

time complexity for insert()
is o(n)
n is the length of the sorted_list. The function iterates over the list to find the correct position to insert the value

time complexity for InsertionSort
o(n^2)
resulting in nested loops. The outer loop iterates over each element, and the inner loop performs the insertion by calling the insert function.

space complexity is o(n)

abdelrahman elatrash