Optimized bubble sort

```
heights = [5, 1, 2, 4, 7, 3]

def bubble_sort(heights):
    n = len(heights)
    counter = 0

    for i in range(n - 1):
        # These are the passes
        for j in range(n - 1 - i):
            counter += 1
            if heights[j] > heights[j + 1]:
                # No of swaps
                heights[j], heights[j + 1] = heights[j + 1],
heights[j]

    print(counter)
    return heights

bubble_sort(heights)
```

```
15

[1, 2, 3, 4, 5, 7]
```

```
# Optimised bubble sort code


def optimised_bubble_sort(a):

    n = len(a)

    for i in range(n - 1):
        already_sorted = True

        for j in range(n - 1 - i):
            # Checking values
            if a[j] > a[j + 1]:
                # Swap
                a[j], a[j+1] = a[j+1], a[j]
                already_sorted = False

        # check for sorted or not
        if already_sorted == True:
            return a
```

```
heights = [5, 1, 2, 4, 7, 3]

li = [1, 2, 5, 3, 4]

optimised_bubble_sort(heights)

[1, 2, 3, 4, 5, 7]

optimised_bubble_sort(li)

[1, 2, 3, 4, 5]

bubble_sort(li)

10

[1, 2, 3, 4, 5]
```

Selection sort

```
a = [4, 5, 6, 7, 8, 2, 0]

# Find the minimum value in the list

def min_value(a):
    current_min = a[0]

    for i in a:
        if i < current_min:
            current_min = i
    return current_min

# c_m = 0

min_value(a)

0
```

```
# Code for Selection sort

def selection_sort(a):
    n = len(a)
```

```
    for i in range(n - 1):
        # Current_min
        current_min = i

        for j in range(i+1, n):
            # check for min value
            if a[j] < a[current_min]:
                current_min = j
        # Swap
        if current_min != i:
            a[current_min], a[i] = a[i], a[current_min]
    return a

li = [2, 8, 5, 3, 9, 4, 1]

selection_sort(li)
```

```
[1, 2, 3, 4, 5, 8, 9]
```

```
# HW: Find number of iteration



def selection_sort(a):
    n = len(a)

    for i in range(n - 1):
        # Current_min
        current_min = i

        print(a)
        print("-"*20)

        for j in range(i+1, n):
            # check for min value
            if a[j] < a[current_min]:
                current_min = j
        # Swap
        if current_min != i:
            a[current_min], a[i] = a[i], a[current_min]
    return a

li = [2, 8, 5, 3, 9, 4, 1]

selection_sort(li)
```

```
[2, 8, 5, 3, 9, 4, 1]
--------------------
[1, 8, 5, 3, 9, 4, 2]
--------------------
[1, 2, 5, 3, 9, 4, 8]
--------------------
```

```
[1, 2, 3, 5, 9, 4, 8]
--------------------
[1, 2, 3, 4, 9, 5, 8]
--------------------
[1, 2, 3, 4, 5, 9, 8]
--------------------

[1, 2, 3, 4, 5, 8, 9]
```

```
# HW: Do dry runs and send pictures on the group

# HW: Python inbuilt sort?
```

Insertion Sort

```
l = [4, 5, 1, 3, 2]

# Tim: Hybrid algorithm -> insertion + merge

def insertion_sort(a):
    n = len(a)

    for i in range(1, n):
        index_to_insert = i
        j = i - 1

        while j >= 0:
            # Move left and compare values
            if a[j] < a[index_to_insert]:
                break
            # In case we need swap, update values
            a[j], a[index_to_insert] = a[index_to_insert], a[j]
            index_to_insert = j
            j -= 1
    return a

insertion_sort(l)

[1, 2, 3, 4, 5]
```

```python
def insertion_sort(a):
    n = len(a)

    for i in range(1, n):
        index_to_insert = i
        j = i - 1

        while j >= 0:
            print(a)
            print("-" * 20)

            # Move left and compare values
            if a[j] < a[index_to_insert]:
                break
            # In case we need swap, update values
            a[j], a[index_to_insert] = a[index_to_insert], a[j]
            index_to_insert = j
            j -= 1
    return a

l = [4, 5, 1, 3, 2]

insertion_sort(l)
```

```
[4, 5, 1, 3, 2]
--------------------
[4, 5, 1, 3, 2]
--------------------
[4, 1, 5, 3, 2]
--------------------
[1, 4, 5, 3, 2]
--------------------
[1, 4, 3, 5, 2]
--------------------
[1, 3, 4, 5, 2]
--------------------
[1, 3, 4, 5, 2]
--------------------
[1, 3, 4, 2, 5]
--------------------
[1, 3, 2, 4, 5]
--------------------
[1, 2, 3, 4, 5]
--------------------

[1, 2, 3, 4, 5]
```

```
# No of times we have to run this loop
```