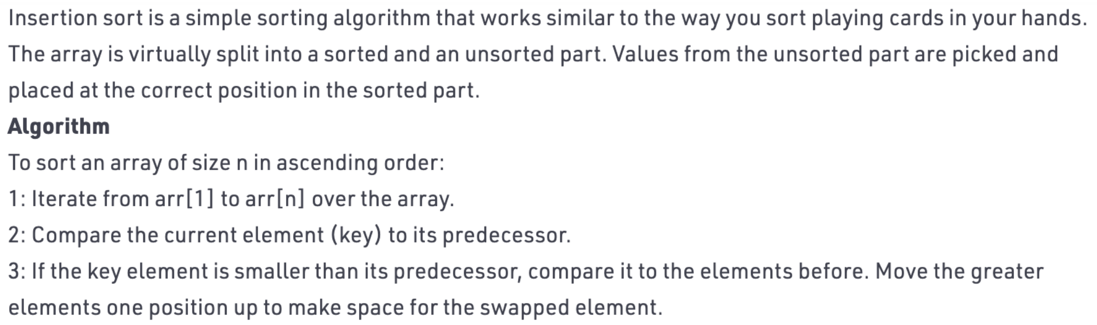
YZV102E/104E - Introduction to Programming for Data Science (Python) Lab 9

15.05.2025

Res. Asst. Barış Bilen (bilenb20@itu.edu.tr)
Res. Asst. Erhan Biçer (bicer21@itu.edu.tr)
Res. Asst. Püren Tap (tap23@itu.edu.tr)
Res. Asst. Sümeyye Öztürk (ozturks20@itu.edu.tr)
Res. Asst. Uğur Önal (onalug@itu.edu.tr)

1 Exercise 1

In this part, we will implement a sorting algorithm called ”insertion sort”. You can see the algorithm of the Insertion Sort in Figure 1



Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

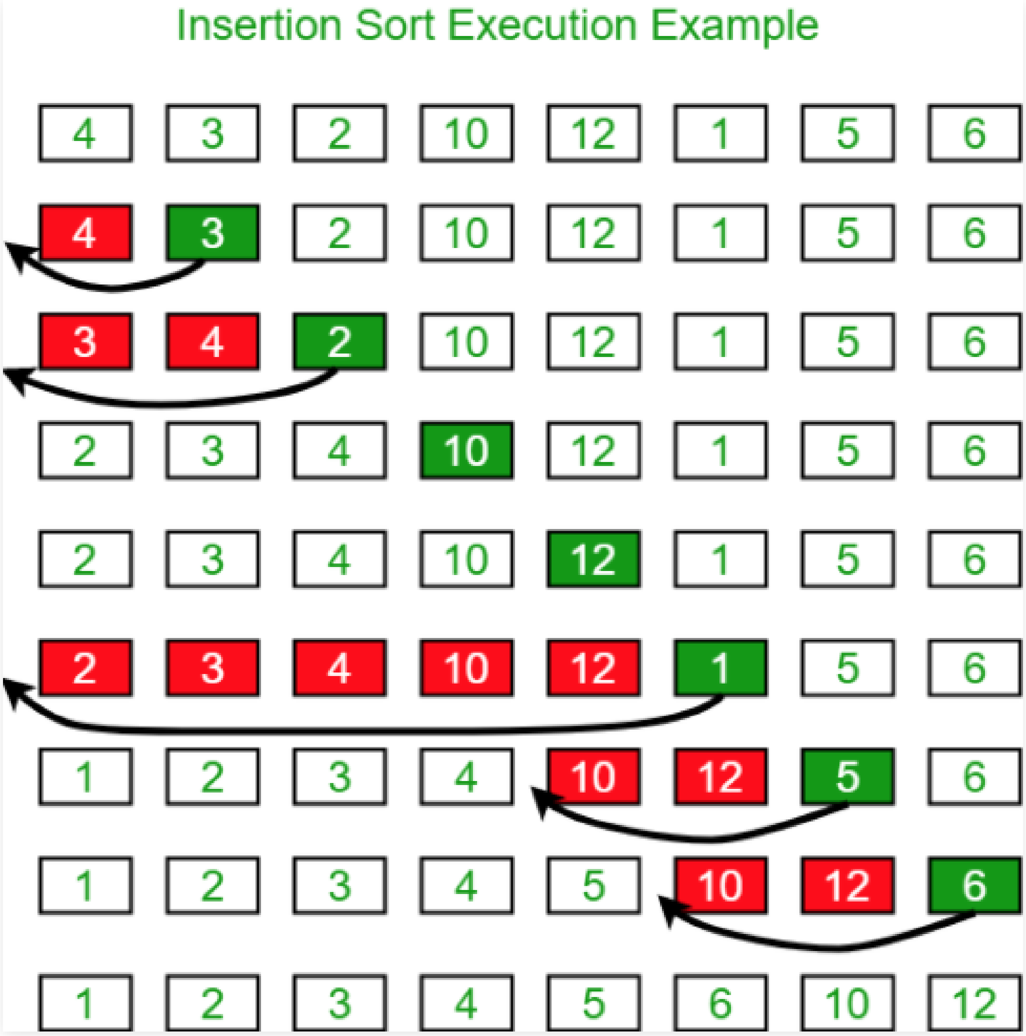
Algorithm

To sort an array of size n in ascending order:
1: Iterate from arr[1] to arr[n] over the array.
2: Compare the current element (key) to its predecessor.
3: If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Figure 1: Algorithm of Insertion Sort. Source: geeksforgeeks

You can see an example run of insertion sort in Figure 2.

Example:



4 3 2 10 12 1 5 6

4 3 2 10 12 1 5 6

3 4 2 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

1 2 3 4 10 12 5 6

1 2 3 4 5 10 12 6

1 2 3 4 5 6 10 12

Figure 2: Example Run of Insertion Sort. Source: geeksforgeeks

Create a random list and try to sort it with Insertion Sort.

1.1 Solution of Exercise 1

The solution is given in Code Snippet 1;

Code Snippet 1: Solution of Exercise 1

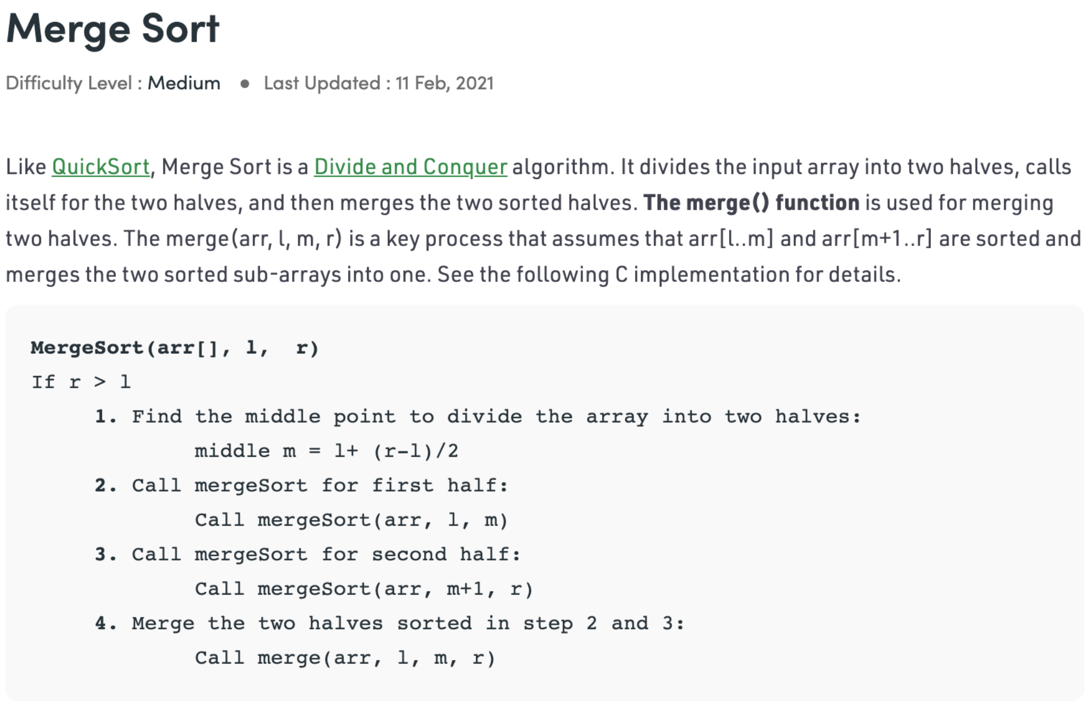
```python
import random


def insertion_sort(lst):
    for i in range(1, len(lst)):
        temp = lst[i]
        j = i - 1
        while j >= 0 and temp < lst[j]:
            lst[j + 1] = lst[j]
            j -= 1
        lst[j + 1] = temp


lst1 = [random.randint(1, 100) for _ in range(10)]
print(lst1)
insertion_sort(lst1)
print(lst1)
```

2 Exercise 2

In this part, we will implement a sorting algorithm called ”merge sort”. You can see the algorithm of the Merge Sort in Figure 3



Merge Sort

Difficulty Level : Medium • Last Updated : 11 Feb, 2021

Like QuickSort, Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. **The merge() function** is used for merging two halves. The merge(arr, l, m, r) is a key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one. See the following C implementation for details.

```
MergeSort(arr[], l,  r)
If r > l
     1. Find the middle point to divide the array into two halves:
             middle m = l+ (r-l)/2
     2. Call mergeSort for first half:
             Call mergeSort(arr, l, m)
     3. Call mergeSort for second half:
             Call mergeSort(arr, m+1, r)
     4. Merge the two halves sorted in step 2 and 3:
             Call merge(arr, l, m, r)
```

Figure 3: Algorithm of Merge Sort. Source: geeksforgeeks

You can see an example run of merge sort in Figure 4.

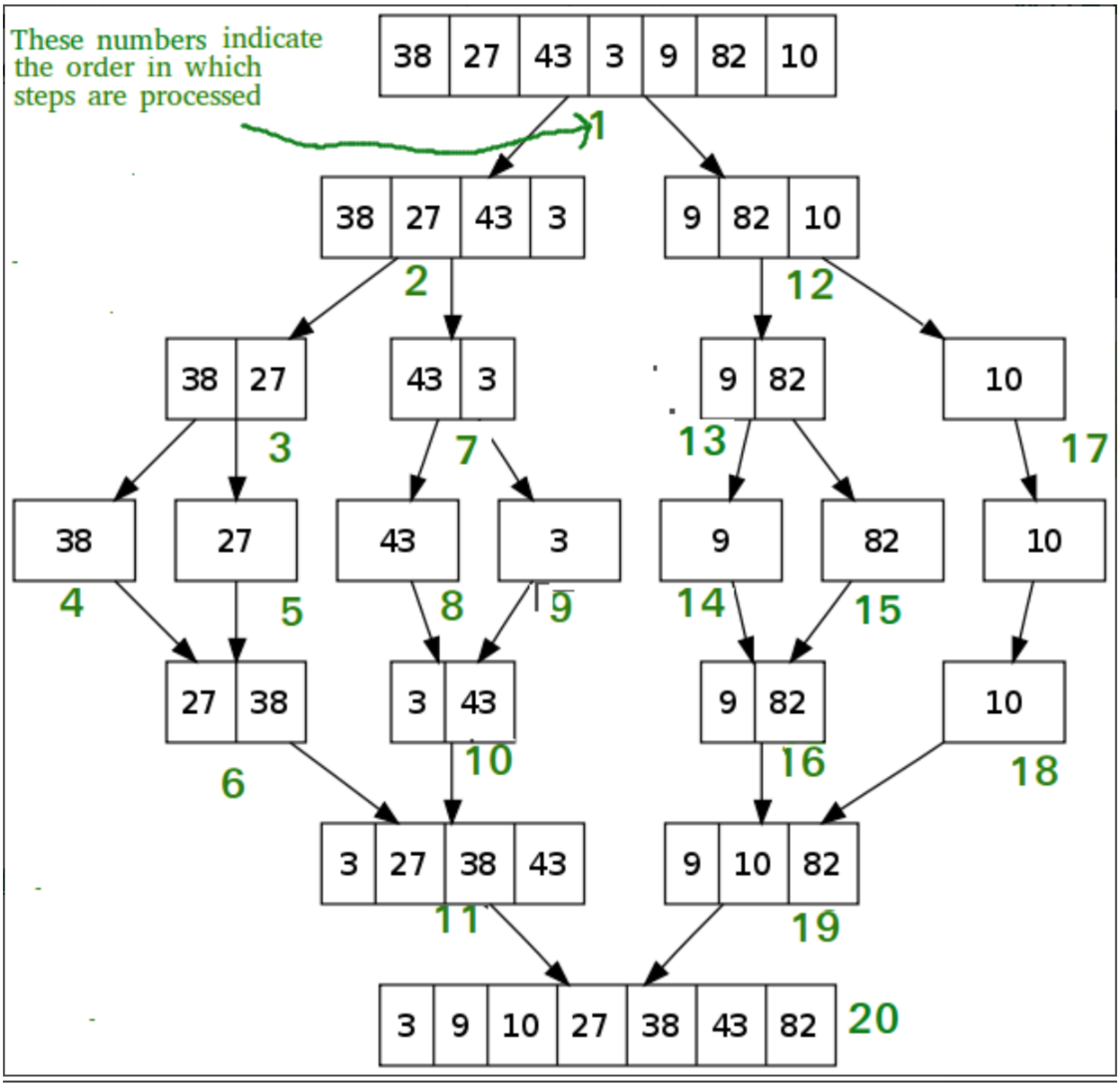


Figure 4: Example Run of Merge Sort. Source: geeksforgeeks

Create a random list and try to sort it with Merge Sort.

2.1 Solution of Exercise 2

The solution is given in Code Snippet 2;

Code Snippet 2: Solution of Exercise 2

```
import random


def merge_sort(lst):
    if len(lst) == 1:
        return lst

    middle = len(lst) // 2
    left = merge_sort(lst[:middle])
    right = merge_sort(lst[middle:])

    return merge(left, right)


def merge(left, right):
    merged_list = []

    i = 0
    j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            merged_list.append(left[i])
            i += 1
        else:
            merged_list.append((right[j]))
            j += 1

    merged_list.extend(left[i:])
    merged_list.extend(right[j:])

    return merged_list


lst1 = [random.randint(1, 100) for _ in range(10)]
print(lst1)
lst1 = merge_sort(lst1)
print(lst1)
```

3 Exercise 3

You will implement three functions that are;

1. Create a Python file that has a definition for a $Circle$ class. The $Circle$ class takes a radius parameter when it is initialized. $Circle$ class has two methods and they are $area()$ and $perimeter()$. In the $area()$ method, it is expected to calculate the area of the circle and return it. In the $perimeter()$ method, it is expected to calculate the perimeter of the circle and return it.

2. Create a Python file that has a definition for a $Square$ class. The $Square$ class takes an edge length parameter when it is initialized. The $Square$ class has two methods and they are $area()$ and $perimeter()$. In the $area()$ method, it is expected to calculate the area of the square and return it. In the $perimeter()$ method, it is expected to calculate the perimeter of the square and return it.

3. In another Python file import the $Circle$ and $Square$ classes and create two different instances for both classes. Print their areas and perimeters.

3.1 Solution of Exercise 3

The solution is given in Code Snippets 3, 4, 5;

Code Snippet 3: circle.py

```python
import math


class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * (self.radius ** 2)

    def perimeter(self):
        return 2 * math.pi * self.radius

```

Code Snippet 4: square.py

```python
class Square:
    def __init__(self, edge_length):
        self.edge_length = edge_length

    def area(self):
        return self.edge_length ** 2

    def perimeter(self):
        return 4 * self.edge_length

```

Code Snippet 5: exercise1.py

```python
from circle import Circle
from square import Square

circ1 = Circle(5.2)
circ2 = Circle(7)

sq1 = Square(5.2)
sq2 = Square(7)

print(f"The circle with a radius of {circ1.radius} - Area: {circ1.area()} Perimeter: {circ1.perimeter()}")
print(f"The circle with a radius of {circ2.radius} - Area: {circ2.area()} Perimeter: {circ2.perimeter()}")
print(f"The square with an edge length of {sq1.edge_length} - Area: {sq1.area()} Perimeter: {sq1.perimeter()}")
print(f"The square with an edge length of {sq2.edge_length} - Area: {sq2.area()} Perimeter: {sq2.perimeter()}")

```