

Computer Science and Programming Lab Class 8

Task 1 *Sort algorithm – Merge sort (15 minutes)*

Write a function `merge_sort()` to sort an integer list in ascending order with the merge sort algorithm.

Task 2 *Sort algorithm – Permutation sort (15 minutes)*

Write a function `permutation_sort()` to sort an integer list in ascending order with permutation, i.e., to generate all possible orders of elements and choose the sorted one from all these orders.

Hint: Please implement this task with recursion!

Task 3 *Sort algorithm – Random sort (10 minutes)*

Write a function `random_sort()` to sort an integer list in ascending order randomly. In each iteration, you select a pair of elements and compare their values. If the two elements are in the wrong order, then switch them. If the whole list is sorted, terminate the program.

Task 4 *Sort algorithm – Comparison (10 minutes)*

Use the following function to generate a random integer series. Parameter "start" and "end" represents the range of integers while "length" marks the length of integer series.

```
import random
def random_int_list(start, stop, length):
    random_list = []
    for i in range(length):
        random_list.append(random.randint(start, stop))
    return random_list
```

Compare the execution time of all your own sort algorithm implementations with the **same** integer list generated by the above function. See how the execution time of these sorting algorithm rises as the length of integer list increases (length = 10, 100, 1000, 10000, etc.).

Task 5 *Sort algorithm for classes (15 minutes)*

Build a **Student** class which stores the **names** and the **IDs** of students. Implement the a sort algorithm which sorts a list `[s1, s2, s3, ..., sn]` **Student** objects by their names and IDs (The **Students** are first sorted by their names. If several students have the same names, then sort them by their IDs.). Try your code on the students `[(jim, 3), (jane, 2), (jim, 1)]`, where the first entry is the name and the second entry is the ID, respectively.

Task 6 *Sort algorithm for classes II (open)*

If you are finished with all other tasks and still have time, please implement a function that sorts rational number objects (the class was introduced in the lecture).