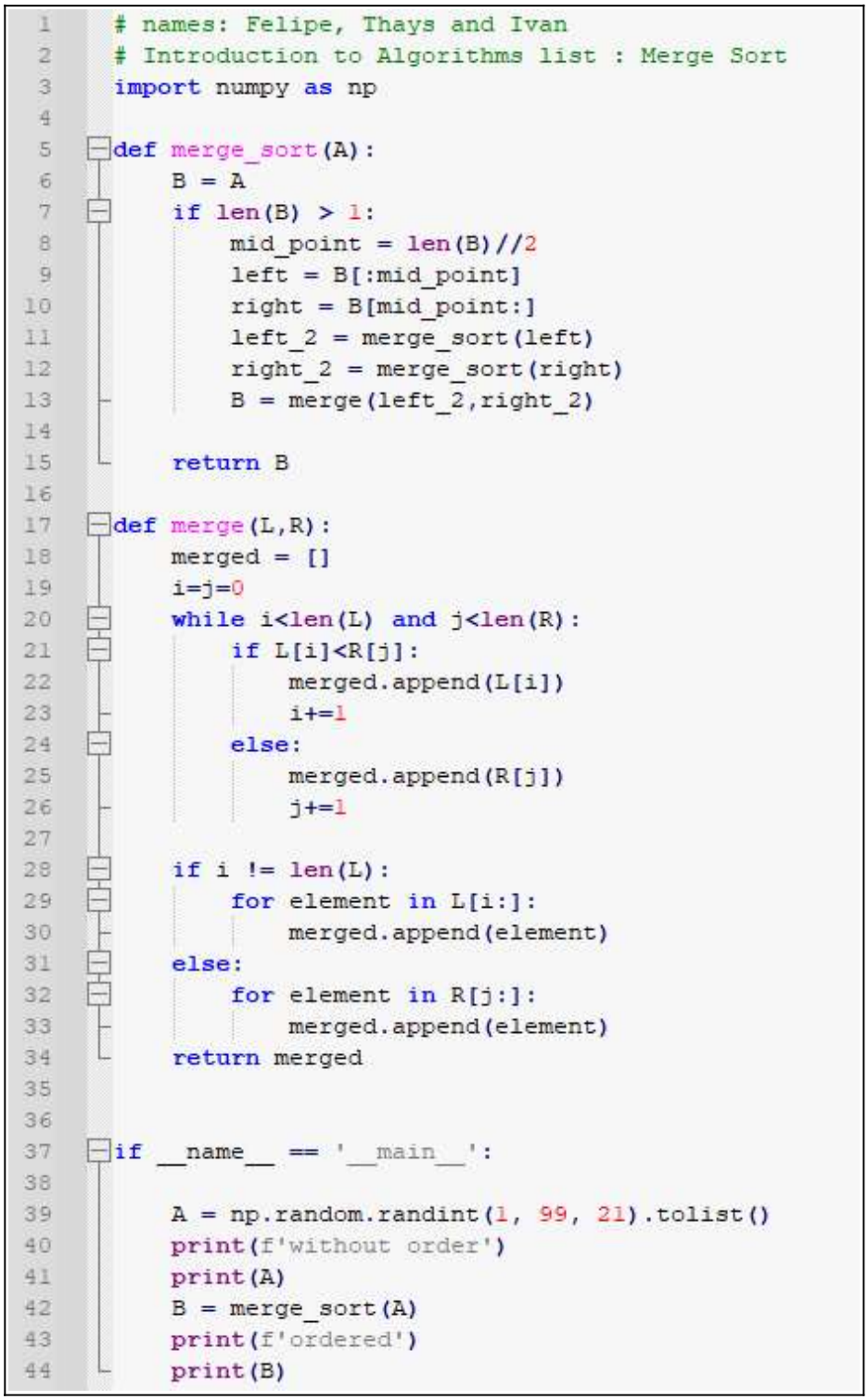
Federal University of Santa Catarina
Department of Automation and Systems Engineering
DAS-410047: Introduction to Algorithms

Instrutor: Eduardo Camponogara
Students: Felipe Alfredo Nack, Thays Da Cruz Franco, Ivan Michel Lehmkuhl de Souza

Mid-Term Exercises – Given Results

```python
# names: Felipe, Thays and Ivan
# Introduction to Algorithms list : Insertion Sort

import numpy as np

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


if __name__ == '__main__':

    C = np.random.randint(1,99,20)
    print(f'without order')
    print(C)
    insertion_sort(C)
    print(f'ordered')
    print(C)
```

Insertion Sort Algorithm

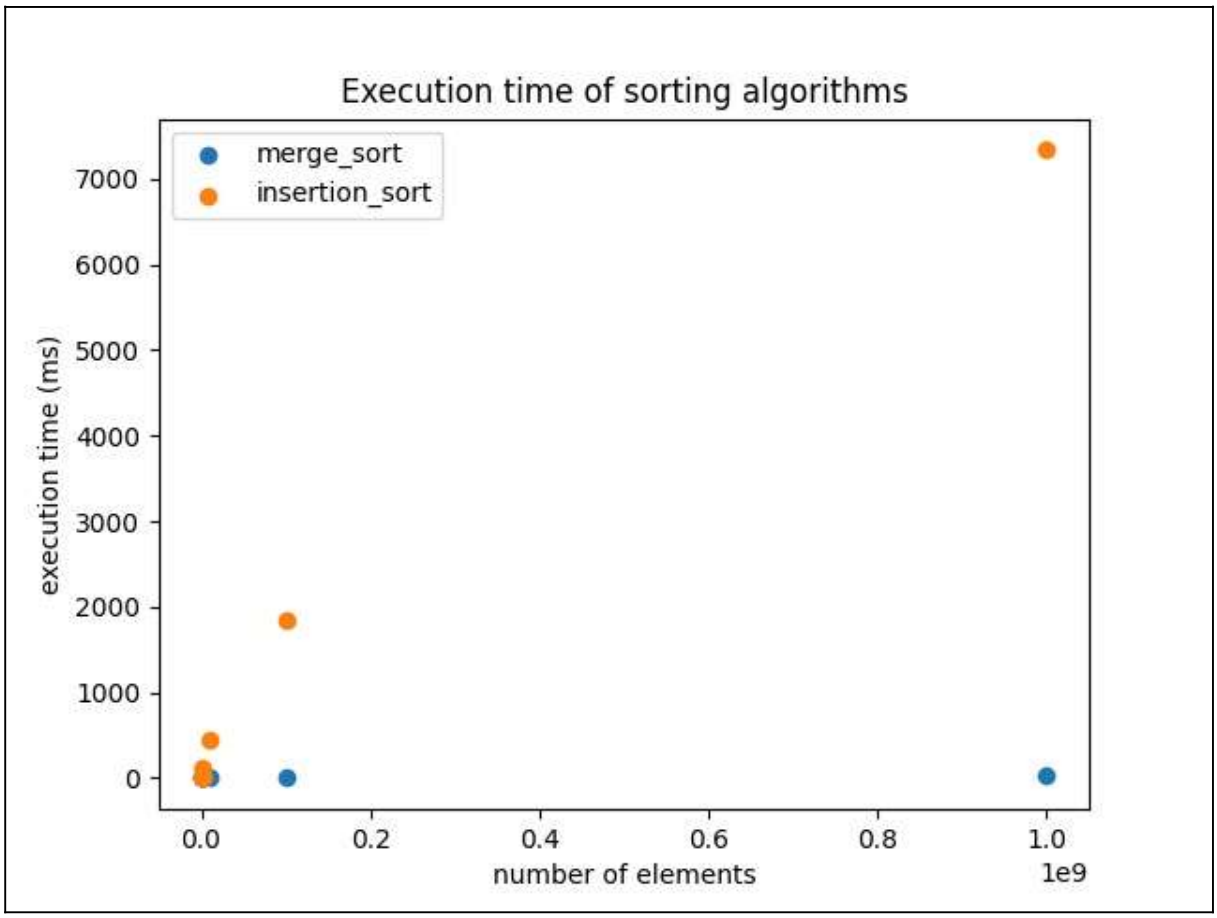


```python
# names: Felipe, Thays and Ivan
# Introduction to Algorithms list : Merge Sort
import numpy as np

def merge_sort(A):
    B = A
    if len(B) > 1:
        mid_point = len(B)//2
        left = B[:mid_point]
        right = B[mid_point:]
        left_2 = merge_sort(left)
        right_2 = merge_sort(right)
        B = merge(left_2,right_2)

    return B

def merge(L,R):
    merged = []
    i=j=0
    while i<len(L) and j<len(R):
        if L[i]<R[j]:
            merged.append(L[i])
            i+=1
        else:
            merged.append(R[j])
            j+=1

    if i != len(L):
        for element in L[i:]:
            merged.append(element)
    else:
        for element in R[j:]:
            merged.append(element)
    return merged


if __name__ == '__main__':

    A = np.random.randint(1, 99, 21).tolist()
    print(f'without order')
    print(A)
    B = merge_sort(A)
    print(f'ordered')
    print(B)
```

Merge Sort Algorithm

```
if __name__ == '__main__':

    exec_time_merge = []
    exec_time_insertion = []
    elements = []
    size_list = 10

    for i in range (10):
        size_list = size_list*2
        list_merge = np.random.randint(low=0,high=10**3,size=size_list)
        start = time.time()
        merge_sort(list_merge)
        final = 1000*(time.time() - start)
        exec_time_merge.append(final)
        elements.append(10**i)

    size_list = 10
    for i in range (10):
        size_list = size_list*2
        list_insertion = np.random.randint(low=0,high=10**3,size=size_list)
        start = time.time()
        insertion_sort(list_insertion)
        final = 1000*(time.time() - start)
        exec_time_insertion.append(final)
        #elements.append(10**i)

    plt.scatter(elements, exec_time_merge, label='merge_sort')
    plt.scatter(elements, exec_time_insertion, label='insertion_sort')
    plt.legend(['merge_sort', 'insertion_sort'])
    plt.ylabel('execution time (ms)')
    plt.xlabel('number of elements')
    plt.title('Execution time of sorting algorithms')
    plt.show()
```

Algorithm for performance comparison using inputs from n = 20 to n = 1 x 10^{9}



Plotting execution time results using Insertion Sort and Merge Sort