Technische Hochschule Ingolstadt
Prof. Dr.-Ing. Richard Membarth

29. March 2023

Algorithms and Data Structures
Assignment 2

Submission deadline for the exercises: 02. April 2023

2.1 Order of Growth

a) The purpose of this exercise is to measure the execution time of functions in order to draw conclusions on the runtime behavior. The file `runtime.py` includes the bytecode for the following three functions:

- `def fun1(n)`
- `def fun2(n)`
- `def fun3(n)`

The functions expect the input size as argument. In order to call the functions in the bytecode, we need to use the *marshal* module to extract the binary of the bytecode. The binary can be executed using `exec` and a dictionary with global variables is provided. In the dictionary, we need to set the keys `fun` and `size` to define the function to execute and its input size:

```
import marshal
fun_bin = marshal.loads(fun_bytes)
# call fun2 with input size 1000
fun_dict = {"fun": 2, "size": 1000}
exec(fun_bin, fun_dict)
```

Write a test program that calls those functions with a reasonable number of different input sizes. Measure the execution time of each function dependent on the input size. For time measurement, you can use the following code:

```
import time
start = time.time()
exec(fun_bin, fun_dict)
end = time.time()
exec_time = end - start
```

Store the execution times and input sizes to `plot_x` and `plot_y`. You can plot the data using the `matplotlib` module in Python (you might need to install matplotlib first):

```
import matplotlib.pyplot as plt
plt.scatter(plot_x, plot_y)
plt.show()
```

Alternatively, you can plot the data manually.
Provide the plots for `fun1()`, `fun2()`, and `fun3()` as well as an estimation of the runtime complexity of each function using the $\mathcal{O}$-notation.