Program performance & doubling hypothesis

It is very important that we understand the importance of program performance with regard to cost-efficiency and technological progress (improvement of algorithms). It is normal, that the running time of a program increases with the problem size (size of input or value of command-line argument). The question is, however, how much does the time necessary (Tn) increase? To answer this question, we can apply the doubling hypothesis. According to the doubling hypothesis, when doubling the problem size, it is observable that the time necessary to print all lines / execute the program increases by a constant factor (a). Hence, the goal of the doubling hypothesis is to elaborate a formula which allows us to calculate the time necessary for any problem size N.

This can be achieved by applying the general doubling hypothesis formula which is as follow:

$$T_N = aN^3$$

Tn = Time necessary to print all lines
a = constant factor of time increase
N = size of input

Be aware, the power of N depends on the algorithm complexity. This can be determined very easily by counting the number of loops the program code contains (check the lecture slides). The formula allows us to calculate the Time necessary for any problem size N.

To set up the formula we need to run our algorithm while doubling the input size a couple of times.

```
# Perform a doubling test for merge sort with 100 trials
starting with n = 64
n:      64 time: 0.062491 (n:n/2) ratio: -
n:     128 time: 0.137992 (n:n/2) ratio: 2.208207
n:     256 time: 0.347589 (n:n/2) ratio: 2.518897
n:     512 time: 0.676999 (n:n/2) ratio: 1.947701
n:    1024 time: 1.271950 (n:n/2) ratio: 1.878807
n:    2048 time: 2.806505 (n:n/2) ratio: 2.206460
n:    4096 time: 7.094551 (n:n/2) ratio: 2.527895
```

We see that the algorithm needs 7.09 seconds for an input size of 4096. We also see that the approx. doubling factor is equal to 2 (ratio). However, the constant factor a not only depends on the time and problem size as it is embedded within the general doubling hypothesis formula (see above). To derive the value of a we can simply insert the values we got for input size 4096. That means

7.09 = a * 4096 ^3

Solving this term results in a = 3.43 * 10^-11

Now we can set up the doubling hypothesis formula as follow:

Tn = 1.032 * 10 ^ -10 * N^3

Conclusion: We are now able to calculate the time necessary for any problem size N. And we are therefore able to identify performance improvement potential.