DAT600-Spring 2026: Assignment 1

Code of Conduct

The principle is trust, participation and collaboration for better learning.

So please, do your best to learn and don't try to cheat. It is allowed and encourage to collaborate with others but for the purpose of learning, not just copy-pasting. You are allowed to consult and get inspired by external resources, but you should mention them in your work.

NB! You may be asked to explain your work to the student assistants and/or the course responsible. Typically via a random selection, or some other relevant reasons such as the permission to use a good quality work as a good example, or of course in case of suspicion of plagiarism .

Delivery

Write a short and concise report in which you solve the following tasks. Add code sections when ever necessary, and a link to your GitHub repository if you find it more convenient.

Assignment – 1: Compulsory

Sorting Algorithms and Running Times

Task 1 counting the steps

Implement the Four algorithms listed in Table 1 in any programming language that you prefer (I will be using python notebooks). Then vary the size of the input and record the number of steps. Plot the number of steps as a function of the input size (n) to confirm that the plotted functions match the asymptotic running time shown in the Table.

Algorithm	Worst-case running time
Insertion-sort	$\Theta(n^2)$
Merge-sort	$\Theta(n\ lg\ n)$
Heap-sort	$O(n\ lg\ n)$
Quicksort	$\Theta(n^2)$

Task 2 Compare true execution time

Choose one or several of the listed algorithm and implement them in two different programming languages. For example python and C or Go or C# or whatever language that you like. Run a test by varying the input size and measuring the execution time on your computer. Comment the differences.

Task 3 Basic proofs

Show that $\frac{n^2}{\lg n} = o(n^2)$

Show that $n^2 \neq o(n^2)$

Hint: use the the definitions

Task 4 Divide and Conquer Analysis

Master Theorem (from Introduction to Algorithms)

Let $T(n)$ be defined by the recurrence:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

where $a \geq 1$, $b > 1$, and $f(n)$ is an asymptotically positive function. Then:

- **Case 1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
- **Case 2:** If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
- **Case 3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some $c < 1$ and sufficiently large n, then $T(n) = \Theta(f(n))$.

Using the master theorem, determine the runtime complexity of each to the following relations.

- R1: $T(n) = 16T\left(\frac{n}{4}\right) + n$

Task 4

Let $T(n) = 4T(n/2) + n$ be a recursive relation. Using the substitution method to determine the runtime complexity of $T(n)$ which inductive hypotheses hold for $T(n)$?

1. $T(n) \leq c.n^2$ where $c > 0$
2. $T(n) \geq c.n^2$ where $c > 0$
3. $T(n) \leq (c.n^2 - b.n)$ where $c > 0$ and $b > 0$

Bonus Task

Find the runtime complexity of this recursive relation using the substitution method or a more complete version of the master theorem.

$T(n) = 2T\left(\frac{n}{2}\right) + nlogn$

Master theorem (a more complete version)

$T(n) = a\,T(n/b) + f(n)$:

Case 1: If $f(n) = O(n^{\log_b a - \epsilon}) \rightarrow T(n) = \Theta(n^{\log_b a})$

Case 2: If $f(n) = \Theta(n^{\log_b a} \log^k n) \rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$

Case 3: If $f(n) = \Omega(n^{\log_b a + \epsilon})$ and regularity holds $\rightarrow T(n) = \Theta(f(n))$