Assignment 1. Sorting Algorithms and Running Times.

Student name: Evgenii Dudkin
e-mail: e.dudkin@stud.uis.no

1 Task 1. Counting the steps

The C programming language was employed to implement four algorithms, namely Insertion sort, merge sort, heapsort, and quicksort, as detailed in Table 1. Apart from conducting in-place sorting of the input array, each algorithm's implementation counts the number of executed steps, representing the operations carried out during the sorting process. A library encapsulating these four algorithm implementations was developed. Additionally, a dedicated project for unit tests was created to thoroughly assess the proper functioning of the library across various scenarios.

Subsequently, in a separate project, the input size was systematically varied, and the corresponding number of steps was graphed for each algorithm against the input size. Figure 1 displays all the plots. The red line in the plots signifies the number of steps relative to the input size, essentially representing the operations executed to sort an array of size n (on the x-axis) for each sorting algorithm. Meanwhile, the blue line represents the average-case or expected asymptotic running time function.

The program is designed to calculate coefficients that enable the determination of the correct asymptotic running time function for each algorithm. Consequently, the plotted functions are observed to align with the asymptotic running times presented in Table 1.

For access to the implementations, code, comments, and other related materials, please refer to the GitHub repository [1].

Algorithm	**Worst-case running time**	**Avarage-case/expected running time**
Insertion sort	$\Theta(n^2)$	$\Theta(n^2)$
Merge sort	$\Theta(n * lg(n))$	$\Theta(n * lg(n))$
Heapsort	$O(n * lg(n))$	-
Quicksort	$\Theta(n^2)$	$\Theta(n * lg(n))$ (expected)

Table 1: Comparison of 4 sorting algorithms

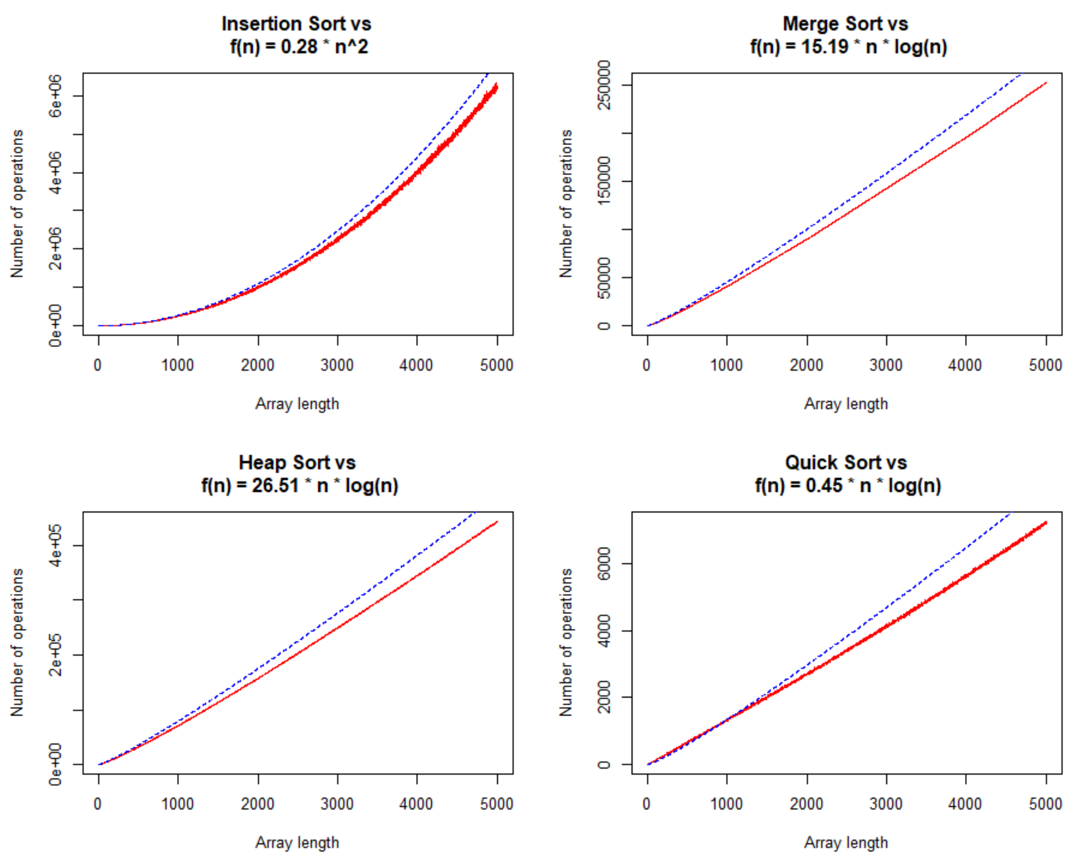


Figure 1: Number of steps vs Asymptotic running time

2 Task 2. Compare true execution time

The Quicksort algorithm was implemented in Go language in addition to the previously discussed C implementation from a prior task. Tests were conducted in two distinct environments, using both Go and C languages. The experiments involved varying the size of the input array, and the corresponding execution times were measured. The outcomes of these tests are presented in Table 2 and Figure 2. Table 2 illustrates the relationship between input size and execution time for each Quicksort implementation, while Figure 2 provides a graphical representation of this relationship.

Although the absolute values of execution time differ when comparing Go and C, it is noteworthy that asymptotically the execution time grows at the same rate, as clearly depicted in the graphical representation. The observation that the same algorithm exhibits varying execution times in different languages and environments is expected. The differences in how programming languages handle certain operations, memory management, and other runtime factors can influence the speed of algorithm execution. Therefore, even with the same algorithmic logic, the underlying language and environment can contribute to differences in observed execution times.

Input size	**C execution time**	**Go execution time**
50000	4 ms	3 ms
100000	9 ms	7 ms
500000	105 ms	82 ms
10000000	26.7 s	23.7 s
30000000	4 m 3 s	3 m 45.3 s

Table 2: Comparison of execution time

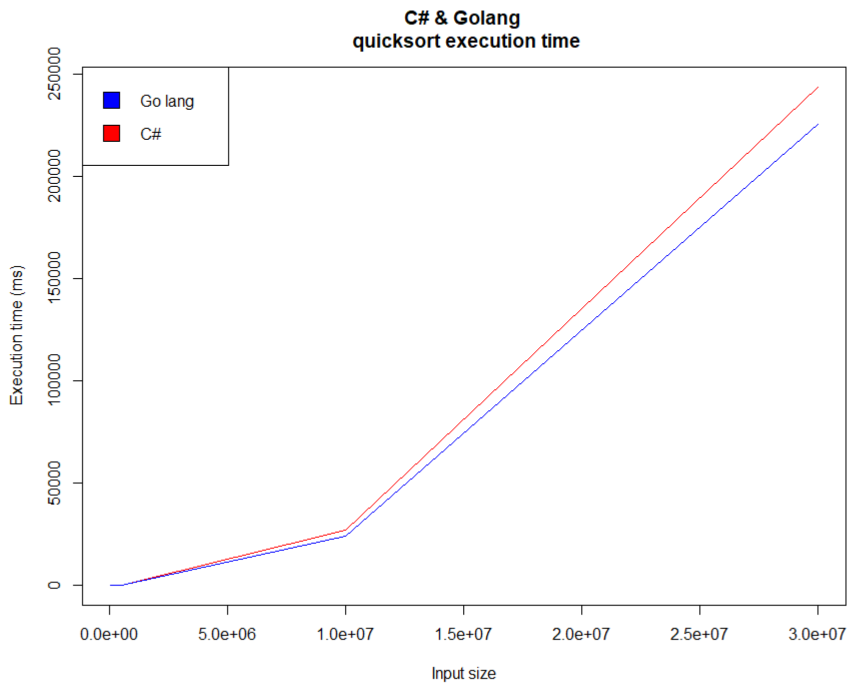


Figure 2: Quicksort execution time comparison

3 Task 3. Basic proofs

3.1

Show that for any real constants a and b, where $b > 0$, $(n + a)^b = \Theta(n^b)$

1) First, we show that $(n+a)^b = O(n^b)$ for any real constants a and b, where $b > 0$ i.e. $\exists C > 0, n_0 \in N : 0 \leq (n + a)^b \leq Cn^b, \forall n > n_0$

$0 \leq (n + a)^b \leq (n + n)^b$ when $n > |a|$ (for any real constants a and b, where $b > 0$)

$\implies 0 \leq (n + a)^b \leq (2n)^b = 2^b n^b = C_1 n^b$ when $n > |a|$ (for any real constants a and b, where $b > 0$)

$\implies$ (n+a)$^b = O(n^b)$

2) Second, we show that $(n + a)^b = \Omega(n^b)$ for any real constants a and b, where $b > 0$ i.e. $\exists C > 0, n_0 \in N : 0 \leq Cn^b \leq (n + a)^b, \forall n > n_0$

$(n+a)^b \geq (n - |a|)^b \geq (n/2)^b$ when $|a| \leq n/2$ (for any real constants a and b, where $b > 0$)

$\implies \ 0 \leq (1/2)^b * n^b = C_2 n^b \leq (n + a)^b$ when $n \geq 2|a|$ (for any real constants a and b, where $b > 0$)

$\implies$ (n+a)$^b = \Omega(n^b)$

From 1) and 2) $\implies$ $(n + a)^b = \Theta(n^b)$, for any real constants a and b, where $b > 0$.

3.2

To show that $n^2/lg(n) = o(n^2)$ we need to show that $\lim_{n\to\infty} \frac{n^2}{lg(n)n^2} = 0$

$\lim_{n\to\infty} \frac{n^2}{lg(n)n^2} = \lim_{n\to\infty} \frac{1}{lg(n)} = 0$

$\implies n^2/lg(n) = o(n^2)$

3.3

To show that $n^2 \neq o(n^2)$ we need to show that $\lim_{n\to\infty} \frac{n^2}{n^2} \neq 0$

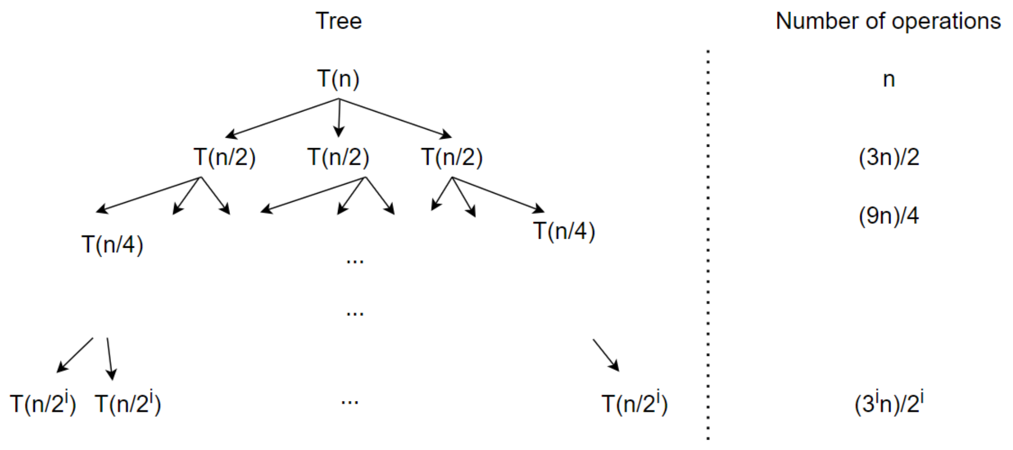
$\lim_{n\to\infty} \frac{n^2}{n^2} = \lim_{n\to\infty} 1 = 1 \neq 0$

$\implies n^2 \neq o(n^2)$

4 Task 4. Divide and Conquer Analysis

$T(n) = 3T(n/2) + \Theta(n)$

Recursion tree method



We divide until input size equals one. It means that $\frac{n}{2^i} = 1$ on the last level. Thus, $i = log_2 n$ is a height of the recursion tree.

To find the running time we need to calculate $S(n) = \sum_{i=0}^{log_2 n} (\frac{3}{2})^i * n$

We can see that it is the sum of a geometric sequence $\implies S(n) = \frac{n((\frac{3}{2})^{log_2 n} - 1)}{\frac{3}{2} - 1}$

$S(n) = 2n(\frac{3^{log_2 n}}{n} - 1) = 2 * 3^{log_2 n} - 2n = 2(2^{log_2 3})^{log_2 n} - 2n = 2 * n^{log_2 3} - 2n$
$S(n) = O(n^{log_2 3})$

Master theorem

Check case 1 of Master theorem:
If $f(n) = O(n^{log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{log_b a})$
$n \leq n^{log_2 3 - \epsilon} \Leftrightarrow 1 \leq log_2 3 - \epsilon \Leftrightarrow \epsilon \leq log_2 3 - 1 \approx 0.58$
$\implies \exists \epsilon > 0$ such that $n = O(n^{log_2 3 - \epsilon})$
$\implies T(n) = \Theta(n^{log_2 3}) = O(n^{log_2 3})$

References

[1] Github Evgenii Dudkin.— URL: `https://github.com/EvgeneDudkin/dat600` (online; accessed: 21.01.2024).