Exercise 1. What is the complexity of the following two code snippets (expressed with Big-O)?

$if\ (condition)\ \{$
$\quad int\ a = 5;$
$\}\ else\ \{$
$\quad int\ b = 10;$
$\}$

Solution: The if-statement is just a declaration of iether a=5 or b=10 os it requires only finite number of iterations. so it has complexity $O(1)$.

Exercise 2.

$for\ (i = 0; i < N; i++)\{$
$\quad for(j = 0;\ j < N; j++)\{$
$\qquad sequence\ of\ statements\ of\ O(1)$

Solution: Each statement inside the inner loop requires a finite number of iterations, let denote these iterations c. Since we perform thense statements $for\ j = 0, 1, ..., N-1$, the statements are performed N times. The outer loop $for\ (i = 0; i < N; i++)$ also performs N times, so the total number of operations is $T(n) = N \cdot cN = cN^{\ 2}$ so the complexity is $O\left(N^2\right)$.

Exercise 3: What is the complexity?

$for\ (i = 0; i < N; i++)\{$
$\quad for(j = i;\ j < N; j++)\{$
$\qquad sequence\ of\ statements\ of\ O(1)$

Solution: Focus on the inner loop first.

$i = 0$ then we would perform N operations

$i = 1$ then we would perform $N - 1$ operations

...

$i = N - 1$ then we would perform 1 operation.

$i = N$ then it would be zero operations.

Thus, the total number of iterations is

$$1 + 2 + ... + (N-1) + N = \frac{(N+1)N}{2} = \frac{N^{2} + N}{2}$$ and this is $O(N^{2})$

Exercise 3: Calculate the complexity of the factorial function below.

The Factorial is define as $x! = x(x-1) \cdot ... \cdot 1$ for all natural numbers x, $\{0, 1, 2, ...,\}$ where

$0! = 1$
$1! = 1$

and
$2! = 2 \cdot 1! = 2 \cdot 1$
$3! = 3 \cdot 2! = 3 \cdot 2 \cdot 1$

Now cosider the code that computes $n!$

$int\ factorial(int\ n)\{$
$\quad if\ (n == 0)\{$
$\quad\quad return\ 1;$
$\quad \}$
$\quad else\{$
$\quad\quad return\ n * factorial(n-1)$

Solution:
Let $T(n)$ be the time complexity of factorial above. The comparison if n==0 cost one unit and in the else statement, the multiplication with cost n cost 1 operation
and calling the $factorial(n-1)$ is 1 operation. In turn, calling $factorial(n-1)$

costs $T(n-1)$.

So we get recursive formula

$T(n) = T(n-1) + 3.$

Futhermore if we replace $T(n-1)$ with $T(n-2)+3$ and so on, then we end up with

$$\begin{aligned} T(n) &= T(n-1)+3 \\ T(n-1) &= T([n-1]-1)+3 \end{aligned}$$

$$= T(n-1-1)+3+3$$
$$= T(n-2) + 3 + 3$$
$$= T(n-3) + 3 + 3 + 3$$

.
.
.

$T(k) = T(n-k) + 3k$

If $k = n$ we get

$$\begin{aligned} T(n) &= T(0)+3n \\ T(n) &= 1+3n \end{aligned}$$

and $T(n) = 1+3n$ is $O(n)$.

Exercise 4: One of the two software packages A or B should be choosen to process data collections, containing each up to 10^9 records. Average processing time of the package A is $T_A(n) = 0.001n$ milliseconds and the average processing time of the package B is $T_B(n) = 500\sqrt{n}$ miliseconds. Which algorithm has better performance in a "Big-O sense"?Work out exact conditions when these packages outperform each other.

Solution: In "Big-O sense" $T_A(n)$ is $O(n)$ and $T_B(n)$ is $O(\sqrt{n})$ so B is better than A. Otherwise we have that B outperforms A when $T_A \geq T_B$ and

$$
\begin{aligned}
&0.001n \ \geq 500\sqrt{n} \\
&\Rightarrow n \geq \ 500\sqrt{n}/0.001 \\
&\Rightarrow n \geq 5 \cdot 10^5\sqrt{n} \\
&\Rightarrow n/\sqrt{n} \geq 5 \cdot 10^5 \\
&\Rightarrow \sqrt{n} \geq 5 \cdot 10^5 \\
&\Rightarrow \left(\sqrt{n}\right)^2 \geq 25 \cdot 10^{10} \\
&\Rightarrow n \geq 25 \cdot 10^{10}
\end{aligned}
$$

So B is only better when $n \geq \ 25 \cdot 10^{10}$ but $10^9 \ < 25 \cdot 10^{10}$ so for processing up $10\ ^9$to data items, software package A is the better choice.

Exercise 5: Assume that the method randomValue requires a constant number of c computational steps to produce each output value, and that the method goodSort takes

$n \log n$ computational steps to sort the array. Determine the Big-Oh complexity for the following fragments of code taking into account only the above computational steps:

$$
\begin{aligned}
&for\ (i = 0; i < n;\ i++)\{ \\
&\qquad for\ (j = 0;\ j < n;\ j++) \\
&\qquad\qquad a[j]\ =\ randomValue(i); \\
&\qquad goodSort(a); \\
&\}
\end{aligned}
$$

Solution: For the inner loop we compute the randomValue n times so the number of iterations for the inner loop is $n \cdot c$. Then after the inner lopp is done, goodSort is used and requires $n \log n$ operations. So to compute the inner loop and then run goodSort it takes a total of $cn + n \log n$ operations. Since the inner loop and goodSort is performed n times we get the total complexity $n(cn + n \log n) = cn^2 + n^2 \log n$.
Because $n^2 \log n$ is the dominant term we have that the complexity of this code is $O\left(n^2 \log n\right)$.

Exercise 6: You have programs A, B, C that have complexities of $O(\log n)$, $O\left(\sqrt{n}\right)$), and

$O\left(n^3\right)$ respectively. Each algorithm spends 15 seconds to process 200 data items. What would the time be for each algorithm spend to process 30000 items

Solution: We can assume, where C, D, E are real constants. Because we have the complexities above we can do the following approximations:

$T_A = C \log n$
$T_B = D\sqrt{n}$
$T_C = En^3$

Find C, D, E

$T(200) = C \log 200 = 15$
$\Rightarrow C = 15 / \log 200 = 1.96$
$T(200) = D\sqrt{200} = 15$
$\Rightarrow D = 15 / \sqrt{200} = 1.06$
$T(200) = E \cdot 200^{3} = 15$
$\Rightarrow E = 15 / 200^{3} = 1.9 \cdot 10^{-6}$

Now calulate $T_A(30000),\ T_B(30000),\ T_C(30000)$

$T_A(30000) = 1.96 \cdot \log 30000 = 8.78$
$T_B(30000) = 1.06 \cdot \sqrt{30000} = 183.60$
$T_C(30000) = 1.9 \cdot 10^{-6} \cdot 30000^{3} = 51300000$

thus, it takes 8.78 seconds for algorithm A, 183.60 seconds for algorithm B and 51300000 seconds for algorithm C to process 30000 items.