This document shows the ingredients of a good assignment report for this course. The \LaTeX source code of this document exemplifies almost everything you need to know about \LaTeX in order to typeset a professional-looking assignment report (for this course). Use it as a starting point for imitation and delete everything irrelevant. The usage of \LaTeX is optional, but highly recommended, for reasons that will soon become clear to those who have never used it before; any learning time is outside the budget of this course, but will hugely pay off, if not in this course then in the next course(s) you take and when writing a thesis or other scientific report.

Part 1

Insertion Sort

\textsc{Insertion-Sort}\footnote{\textsc{Insertion-Sort} is an efficient algorithm for sorting a small number of elements. Insertion sort works the way many people sort a hand of playing cards. We start with an empty left hand and the cards face down on the table. We then remove one card at a time from the table and insert it into the correct position in the left hand. To find the correct position for a card, we compare it with each of the cards already in the hand, from right to left, as illustrated in Figure\footnote{At all times, the cards held in the left hand are sorted, and these cards were originally the top cards of the pile on the table." \textit{(Quoted from page 17 of CLRS3 \footnote{CLRS3.})}].}

In the sequel of this demo report, assume that the problem tasks in the assignment were as follows (actual assignment statements in this course may have other tasks):

A. Implement \textsc{Insertion-Sort} as a Python function \texttt{insertion_sort(A)}, where the elements to be sorted are provided in an integer array \texttt{A} indexed from 0 to \texttt{n−1}. The sorting is to be done in place, returning \texttt{A} in non-decreasing order, that is: \texttt{A[0] ≤ A[1] ≤ ··· ≤ A[n−1]}.

For brevity, you can refer to a sequence in non-decreasing order as a sorted sequence.

B. Compute the best, average, and worst-case time complexities of \textsc{Insertion-Sort}.

A Specification and Program

A specification of sorting and our Python implementation of \textsc{Insertion-Sort} are given in Listing\footnote{Your report need not contain an explanation, like in this paragraph, of the problem to be solved: you can start with the task answers, assuming the reader has read the problem statement in the assignment.} where \texttt{n} is referred to as \texttt{len(A)}.
B Complexity Analysis

The program in Listing 1 has two nested loops, so we analyse it starting from the inner loop, in lines 13 to 17 whose purpose is to insert $A[j]$ into the sorted sequence $A[0..j-1]$, assuming $j > 0$, yielding the sorted sequence $A[0..j]$. Let $T_{ins}(j)$ denote the running time of this inner loop:

$$T_{ins}(j) = \begin{cases} 
\Theta(1) & \text{if } A[j-1] \leq A[j] \\
\Theta(j) & \text{if } A\left\lceil \frac{j-1}{2} \right\rceil < A[j] \leq A\left\lfloor \frac{j+1}{2} \right\rfloor \quad \text{(if } j > 1) \\
\Theta(j) & \text{if } A[j] < A[0] 
\end{cases}$$

assuming that every comparison takes constant time and every assignment takes constant time.

We can now analyse the outer loop, and hence the whole algorithm. Let $n$ denote $\text{len}(A)$ and let $T(n)$ denote the running time of insertion_sort($A$). We get the following recurrence:

$$T(n) = \begin{cases} 
\Theta(1) & \text{if } n < 2 \\
T(n-1) + T_{ins}(n) & \text{if } n \geq 2 
\end{cases}$$

Using recurrence (B), we get the following time complexity results:

- $T(n) = \Theta(n)$ in the best case, where the array is already non-decreasingly ordered before the sorting, so that $T_{ins}(n) = \Theta(1)$ at every iteration of the outer loop, because $A[j]$ is always kept by the inner loop behind the sorted sequence $A[0..j-1]$. This result follows from Theorem 1 below, for the constants $a = 1$ and $b = 2$.

- $T(n) = \Theta(n^2)$ in the average case, defined here as follows: on average over the iterations of the outer loop, the inner loop inserts $A[j]$ into the middle of the sorted sequence $A[0..j-1]$, so that $T_{ins}(n) = \Theta(n)$ on average at every iteration of the outer loop. This can be proven by induction: (insert your proof here).

- $T(n) = \Theta(n^2)$ in the worst case, where the array is non-increasingly ordered before the execution of the algorithm, so that $T_{ins}(n) = \Theta(n)$ at every iteration of the outer loop, because $A[j]$ is always inserted by the inner loop at the beginning of the sorted sequence $A[0..j-1]$. This result has the same proof as in the average case above.

In conclusion, **INSERTION-SORT** takes $O(n^2)$ time for an array of $n$ elements.
def insertion_sort(A):
    """
    Sig: int[0..n-1] → int[0..n-1] # the indices (not the values) run from 0 to n-1
    Pre: (none)
    Post: A[0..n-1] is a non-decreasingly ordered permutation of its original elements
    Ex: insertion_sort([5,7,3,12,1,7,2,8,13]) = [1,2,3,5,7,7,8,12,13]
    """
    for j in range(1, len(A)):
        # Invariant: A[0..j-1] is a sorted permutation of its original elements
        # Variant: len(A) - j
        key = A[j]
        i = j - 1
        while i >= 0 and A[i] > key:
            # Invariant: A[i+2..j] has the original elements of A[i+1..j-1]
            # Variant: i
            A[i+1] = A[i]
            i = i - 1
        A[i + 1] = key

Listing 1: Python implementation of the Insertion-Sort algorithm on page 18 of CLRS3 [1].

Compare for example line 13 with line 5 of that algorithm: arrays are indexed from 1 in CLRS3 but from 0 in Python. Note also that a closed interval ℓ..u, as used in the mathematical notation of the comments, is denoted by the right-open interval ℓ : u + 1 in Python; you can use the Python notation in comments and the running text, as long as you comply with the Python semantics.

Theorem 1. The following recurrence, for some constants a and b:

\[ T(n) = \begin{cases} 
\Theta(1) & \text{if } n < b \\
 a \cdot T(n-1) + \Theta(1) & \text{if } n \geq b 
\end{cases} \]

has \( \Theta(n) \) as closed form for \( a = 1 \), and \( \Theta(a^n) \) as closed form for \( a > 1 \).

Proof. By induction (left as an exercise to the reader in the AD1 course). □

References

Checklist before Submitting

In order to protect yourself against an unnecessary loss of points, use the following checklist before submitting:

- Crosscheck your report against the assignment instructions.
- Remember that when submitting you implicitly certify (a) that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, (b) that each teammate can individually explain any part starting from the moment of submitting your report, and (c) that your report and attachments are not freely accessible on a public repository.
- Spellcheck all documents, including the comments in the source code.
- Proofread, if not grammar-check, your report at least once per teammate.

More LaTeX and Technical Writing Advice

Unnumbered itemisation (only to be used when the order of the items does not matter):

- Unnumbered displayed formula:
  \[ E = m \cdot c^2 \]

- Numbered displayed formula, which is cross-referenced somewhere:
  \[ E = m \cdot c^2 \]

- Formula — the same as formula (B) — spanning more than one line:
  \[
  E = m \cdot c^2
  \]

Numbered itemisation (only to be used when the order of the items does matter):

1. First do this.
2. Then do that.
3. If we are not finished, then go back to Step 2 else stop.
Figure 2: A binary search tree (on the left), a binary min-heap (in the middle), and a binomial tree of rank 3 (on the right).


Use \textit{...} in mathematical mode for each multiple-letter identifier in order to avoid typesetting the identifier like the product of single-letter ones. For example, note the typographic difference between the identifier $WL$, obtained through $\textit{WL}$, and the product $WL$, where there is a small space between the $W$ and the $L$, obtained through $WL$.

Do not use programming-language-style lower-ASCII notation (such as ! for negation, && for conjunction, || for disjunction, and the equality sign = for assignment) in algorithms or formulas (but rather use $\neg$, $\land$ or & or and, $\lor$ or or, and $\leftarrow$ or :=, respectively), as this testifies to a very strong confusion of concepts.

Figures can be imported with \includegraphics or drawn inside the \LaTeX{} source code using the highly declarative notation of the tikz package: see Figure 2 for sample drawings. It is perfectly acceptable in this course to include scans or photos of drawings that were carefully done by hand.

If you are not sure whether you will stick to your current choice of notation or terminology, then introduce a new (possibly parametric) command. For example, upon
\begin{verbatim}
\newcommand{\Cardinality}[1]{\left\lvert#1\right\rvert}
\end{verbatim}
the formula $\Cardinality{S}$ typesets the cardinality of set $S$ as $|S|$ with autosized vertical bars and proper spacing, but upon changing the definition of that parametric command to
\begin{verbatim}
\newcommand{\Cardinality}[1]{\# #1}
\end{verbatim}
and recompiling, the formula $\Cardinality{S}$ typesets the cardinality of set $S$ as #$S$. You can thus obtain an arbitrary number of changes in the document with a constant-time change in its source code, rather than having to perform a linear-time find-and-replace operation within the source code, which is painstaking and error-prone. The source code of this document has some useful predefined commands about mathematics and algorithms.

Use commands on positioning (such as \hspace, \vspace, and \noindent) and appearance (such as \small for reducing the font size, and \textit for italics) very sparingly, and

---

2 Use footnotes very sparingly, and note that footnote pointers are never preceded by a space and always glued immediately behind the punctuation, if there is any.
ideally only in (parametric) commands, as the very idea of mark-up languages such as \LaTeX\ is to let the class designer (usually a trained professional typesetter) decide on where things appear and how they look. For example, \texttt{\textbackslash{emph}} (for emphasis) compiles (outside italicised environments, such as \texttt{\texttt{\textbackslash{theorem}}}) into \textit{italics} under the \texttt{article} class used for this document, but it may compile into \textbf{boldface} under some other class.

If you do not (need to) worry about how things look, then you can fully focus on what you are trying to express!

Note that \textit{no} absolute numbers are used in the \LaTeX\ source code for any of the references inside this document. For ease of maintenance, \texttt{\textbackslash{label}} is used for giving a label to something that is automatically numbered (such as an algorithm, equation, figure, footnote, item, line, part, section, subsection, or table), and \texttt{\textbackslash{ref}} is used for referring to a label. An item in the bibliography file is referred to by \texttt{\textbackslash{cite}} instead. Upon changing the text, it suffices to recompile, once or twice, and possibly to run BibTeX again, in order to update all references consistently.

Always write \texttt{Table~\textbackslash{ref}(tab:maths)} instead of \texttt{Table \textbackslash{ref}(tab:maths)}, by using the non-breaking space (which is typeset as the tilde $\sim$) instead of the normal space, because this avoids that a cross-reference is spread across a line break, as for example in “Table \texttt{[1]}”, which is considered poor typesetting.

The rules of English for how many spaces to use before and after various symbols are given in Table \texttt{2}. Beware that they may be very different from the rules in your native language.

* Feel free to report to the head teacher any other features that you would have liked to see discussed and exemplified in this template document.
<table>
<thead>
<tr>
<th>Topic</th>
<th>\LaTeX{} code</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greek letter</td>
<td>$\Theta, \Omega, \epsilon$</td>
<td>$\Theta, \Omega, \epsilon$</td>
</tr>
<tr>
<td>multiplication</td>
<td>$m \cdot n$</td>
<td>$m \cdot n$</td>
</tr>
<tr>
<td>division</td>
<td>$\frac{m}{n}, m \div n$</td>
<td>$\frac{m}{n}, m \div n$</td>
</tr>
<tr>
<td>rounding down</td>
<td>$\lfloor n \rfloor$</td>
<td>$\lfloor n \rfloor$</td>
</tr>
<tr>
<td>rounding up</td>
<td>$\lceil n \rceil$</td>
<td>$\lceil n \rceil$</td>
</tr>
<tr>
<td>binary modulus</td>
<td>$m \mod n$</td>
<td>$m \mod n$</td>
</tr>
<tr>
<td>unary modulus</td>
<td>$m \equiv n \mod \ell$</td>
<td>$m \equiv n \mod \ell$</td>
</tr>
<tr>
<td>root</td>
<td>$\sqrt{n}, \sqrt[3]{n}$</td>
<td>$\sqrt{n}, \sqrt[3]{n}$</td>
</tr>
<tr>
<td>exponentiation, superscript</td>
<td>$n^i$</td>
<td>$n^i$</td>
</tr>
<tr>
<td>subscript</td>
<td>$n_i$</td>
<td>$n_i$</td>
</tr>
<tr>
<td>overline</td>
<td>$\overline{n}$</td>
<td>$\overline{n}$</td>
</tr>
<tr>
<td>base 2 logarithm</td>
<td>$\lg n$</td>
<td>$\lg n$</td>
</tr>
<tr>
<td>base $b$ logarithm</td>
<td>$\log_b n$</td>
<td>$\log_b n$</td>
</tr>
<tr>
<td>binomial</td>
<td>$\binom{n}{k}$</td>
<td>$\binom{n}{k}$</td>
</tr>
<tr>
<td>sum</td>
<td>$\sum_{i=1}^n i$</td>
<td>$\sum_{i=1}^n i$</td>
</tr>
<tr>
<td>numeric comparison</td>
<td>$\leq, &lt;, =, \neq, &gt;, \geq$</td>
<td>$\leq, &lt;, =, \neq, &gt;, \geq$</td>
</tr>
<tr>
<td>non-numeric comparison</td>
<td>$\prec, \nprec, \preceq, \succeq$</td>
<td>$\prec, \nprec, \preceq, \succeq$</td>
</tr>
<tr>
<td>extremum</td>
<td>$\min, \max, +\infty, \bot, \top$</td>
<td>$\min, \max, +\infty, \bot, \top$</td>
</tr>
<tr>
<td>function</td>
<td>$f : A \to B, \circ, \mapsto$</td>
<td>$f : A \to B, \circ, \mapsto$</td>
</tr>
<tr>
<td>sequence, tuple</td>
<td>$\langle a, b, c \rangle$</td>
<td>$\langle a, b, c \rangle$</td>
</tr>
<tr>
<td>set</td>
<td>${a, b, c}, \emptyset, \mathbb{N}$</td>
<td>${a, b, c}, \emptyset, \mathbb{N}$</td>
</tr>
<tr>
<td>set membership</td>
<td>$\in, \not\in$</td>
<td>$\in, \not\in$</td>
</tr>
<tr>
<td>set comprehension</td>
<td>${i \mid 1 \leq i \leq n}$</td>
<td>${i \mid 1 \leq i \leq n}$</td>
</tr>
<tr>
<td>set operation</td>
<td>$\cup, \cap, \setminus, \times$</td>
<td>$\cup, \cap, \setminus, \times$</td>
</tr>
<tr>
<td>set comparison</td>
<td>$\subseteq, \supseteq$</td>
<td>$\subseteq, \supseteq$</td>
</tr>
<tr>
<td>logic quantifier</td>
<td>$\forall, \exists, \nexists$</td>
<td>$\forall, \exists, \nexists$</td>
</tr>
<tr>
<td>logic connective</td>
<td>$\land, \lor, \neg, \implies$</td>
<td>$\land, \lor, \neg, \implies$</td>
</tr>
<tr>
<td>logic</td>
<td>$\models, \equiv, \vdash$</td>
<td>$\models, \equiv, \vdash$</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>$&amp;, # \approx, \sim, \ell$</td>
<td>$&amp;, # \approx, \sim, \ell$</td>
</tr>
<tr>
<td>dots (context-sensitive)</td>
<td>$\ldots, \cdots, \vdots, \ddots$</td>
<td>$\ldots, \cdots, \vdots, \ddots$</td>
</tr>
<tr>
<td>parentheses (autosizing)</td>
<td>$\langle m^n \rangle$</td>
<td>$(m^n)$</td>
</tr>
<tr>
<td>identifier of &gt; 1 character</td>
<td>$\text{mathit}(\text{identifier})$</td>
<td>$\text{identifier}$</td>
</tr>
</tbody>
</table>

Table 1: The typesetting of elementary mathematics. Note very carefully when italics are used by \LaTeX{} and when not, as well as all the horizontal and vertical spacing performed by \LaTeX{}.

<table>
<thead>
<tr>
<th>number of spaces after</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of spaces before</td>
<td>0</td>
<td>/ -</td>
</tr>
<tr>
<td>1</td>
<td>( (</td>
<td>,</td>
</tr>
</tbody>
</table>

Table 2: Spacing rules of English