Template für Lösungen und kurze Berichte

Alice Cooper (s0000001@stud.uni-frankfurt.de)
Bob Marley (s0000002@stud.uni-frankfurt.de)
Charlie Chaplin (s0000003@stud.uni-frankfurt.de)

12. August 2024

Diese Vorlage benutzen

Du kannst diese `.tex`-Datei als Vorlage für Lösungen, Projektpläne, Notizen und kurze Berichte benutzen. Falls möglich, solltest du sie mit `lualatex` kompilieren. Du kannst die Datei auf GitHub finden (`https://github.com/goethe-tcs/note-template`) oder auch auf Overleaf (`https://www.overleaf.com/read/bbxtmsfhsfkv`). Wenn du Overleaf benutzen möchtest, musst du ein Overleaf-Konto anlegen und im Menü „Copy Project“ klicken, um eine private Kopie der Datei zu erzeugen, die du dann editieren kannst.

Die Vorlage benutzt die Schriftarten EB Garamond und Source Code Pro. Stelle sicher, dass du sie auch wirklich installiert hast. Normalerweise sind sie bei einer vollen Installation von texlive mit dabei. Du kannst die Schriftarten aber natürlich auch ändern.

Übliche LaTeX-Fehler

Übliche Fehler von LaTeX-Neulingen sind:

- Neulinge benutzen `\\`, um einen neuen Absatz zu beginnen. Stattdessen sollte man `\\` quasi nie benutzen! Man sollte zwischen Absätzen im Quellcode eine leere Zeile lassen. Absätze werden eingerückt, das ist üblich und erwünscht!
- Neulinge schreiben „Sei n eine gerade Zahl“ anstatt „Sei n eine gerade Zahl“. Nur das zweite ist richtig! Wenn man eine mathematische Formel oder ein mathematisches Symbol benutzt, muss man *immer* Mathemodus benutzen, also `$n$` schreiben!
- Neulinge schreiben ”Anführungszeichen” anstatt „Anführungszeichen“: Das erste ist in jeder Sprache falsch! Man sollte also immer `\enquote{..}` benutzen. Dasselbe gilt für ’Apostroph’ anstatt ‚Apostroph‘. Der Befehl passt sich übrigens automatisch auf die ausgewählte Sprache an, da die Anführungszeichen subtil anders sein könnten. Im Englischen wäre das hier korrekt: “Quotation marks”
- Neulinge verwenden nie die Rechtschreibprüfung. Stattdessen sollte man sie immer angeschaltet lassen!
- Neulinge versenden die fertige Datei mit dem Dateinamen `main.pdf`. Wähle stattdessen einen Namen, der *für den Empfänger* sinnvoll ist, zum Beispiel `project-plan-Cooper.pdf`.

Zitieren

Zum Zitieren einer Quelle schreibt man zum Beispiel Karp [1]. Zwischen dem `\cite`-Kommando und dem Namen sollte hierbei unbedingt ein *non-breaking space* `~` stehen, damit die Zeile dort nicht plötzlich endet. Also so: `Karp~\cite{DBLP:conf/coco/Karp72}`. Falls der Name mehrmals hintereinander erwähnt wird, sollte das `\cite` Kommando nur beim ersten Auftauchen benutzt werden.

Bilder

Bilder lassen sich mit `\includegraphics{mein-bild.pdf}` einfügen, welches eine abphotographierte Zeichnung sein kann oder eine mit inkscape produzierte Vektorgraphik. Am Schönsten zeichnet man natürlich mit TikZ.

Programmcode

Programmcode oder Pseudocode kann so eingebunden werden:

```
1  for (int i = 1; i < n; i++) {
2    for (int j = 1; j < i; j++) {
3      std::cout << i; /* Θ(n^2) mal ausgefuehrt */
4    }
5  }
```

Aufgabe 1a)

In Aufgabe 1a) ist zu zeigen, dass Algorithmus A die Laufzeit $O(n^2)$ hat. Oder $\Omega(n \log n)$? Vielleicht auch $\Theta(n)$.

Sei $T(n)$ die Laufzeit von A auf Eingaben der Größe n. Dann gilt $T(n) \leq T(n-1)+10n$, denn A macht für jedes Bit der Eingabe höchstens zehn Rechenschritte und ruft sich dann selbst rekursiv wieder auf, mit einer Eingabe der Größe $n-1$. Wir beweisen nun mithilfe vollständiger Induktion, dass $T(n) \leq 10n^2$ für alle positiven $n \in \mathbb{N}$ gilt:

- Für den Induktionsanfang $n = 1$ stellen wir fest, dass $T(1) = 10 = 10 \cdot 1^2$ gilt.
- Für den Induktionsschluss sei nun $n > 1$. Wir können die Induktionsannahme voraussetzen, dass $T(n-1) \leq 10(n-1)^2$ gilt, und müssen zeigen, dass $T(n) \leq 10n^2$ gilt. Tatsächlich haben wir:

$$\begin{aligned} T(n) &\leq T(n-1) + 10n \leq 10(n-1)^2 + 10n \\ &= 10(n^2 - 2n + 1 + n) = 10(n^2 - n + 1) \leq 10n^2 \,. \end{aligned}$$

  Das war zu zeigen.

Lemma-Satz-Beweis

Lemma 1. *Dies ist ein Lemma.*

Theorem 2. *Dies ist ein Satz.*

Beweis. Theorem 2 folgt direkt aus Lemma 1. □

Literatur

[1] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.