CS799 Ridiculously Advanced Systems Project Report

Peter A. Wiseguy

11th July, 2018

Abstract

This document is a simple template for a typical term or semester paper (lab/course report, "Übungsbericht", etc.) based on the HagenbergThesis LaTeX package.[1] The structure and chapter titles have been formulated to provide a good starting point for a typical *project report*. This document uses the custom class hgbreport which is based on LaTeX's standard report document class with `chapter` as the top structuring element. If you wish to write this report in German you should substitute the line

`\documentclass[english]{hgbreport}`

at the top of this document by

`\documentclass[german]{hgbreport}`.

To omit the default **title page** (as in this document) use the `notitlepage` option, e.g.,

`\documentclass[notitlepage,english]{hgbreport}`.

Also, you may want to place the text of the individual chapters in separate files and include them using `\include{..}`.

Use the abstract to provide a short summary of the contents in the document.

[1] See https://github.com/Digital-Media/HagenbergThesis for the most current version and additional examples. This repository also provides a good introduction and useful hints for authoring academic texts with LaTeX.

Chapter 1

Algorithm test

Algorithm 1.1: Bikubische Interpolation in 2D. Die Funktion $w_{\text{cub}}()$ steht für die eindimensionale kubische Interpolationsfunktion (Zeile 11).

1: **BicubicInterpolation**(I, x, y) ▷ $x, y \in \mathbb{R}$
Input:
Returns the interpolated value of the image I at the continuous position (x, y).
2: $val \leftarrow 0$ Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
3: Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
4: **for** $j \leftarrow 0, \ldots, 3$ **do** ▷ iterate over 4 lines
5: $v \leftarrow \lfloor y \rfloor - 1 + j$
6: $p \leftarrow 0$
7: **for** $i \leftarrow 0, \ldots, 3$ **do** ▷ iterate over 4 columns
8: Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
9: $u \leftarrow \lfloor x \rfloor - 1 + i$
10: Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
11: $p \leftarrow p + I(u, v) \cdot w_{\text{cub}}(x - u)$
12: **end for**
13: $val \leftarrow val + p \cdot w_{\text{cub}}(y - v)$
14: **end for**
15: **return** val
16: **end**

Algorithm 1.2: Finds a minimum makespan role assignment. This function is the MMDR $O(n^5)$ polynomial time implementation, as described by McAlpine et al. It rearranges target positions T so that their index corresponds with the indices of their assigned agents.

1: **RoleAssignment**(A, T)
Input: $A = (\mathbf{a}_0, \ldots, \mathbf{a}_{n-1})$, $T = (\mathbf{t}_0, \ldots, \mathbf{t}_{n-1})$, where $\mathbf{a}_i = (x_i, y_i)$, $\mathbf{t}_j = (x_j, y_j)$.
Returns $T' = (\mathbf{t}'_0, \ldots, \mathbf{t}'_{n-1})$, a permutation of T.
StateNN[1]: Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
2: State Very long texts should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation should have the same indentation.
StateNN (no arg): leftmargin = 18.61494pt, listparindent = 21.9pt, labelwidth = 13.13995pt
StateNN[1]
3: State ▷ another comment
Statex
item[]
4: *edgesSorted* ← **SortAscendingDist**(*Edges*)
5: *lastDistance* ← -1
6: *rank* ← 0
7: *currentIndex* ← 0
Process sorted edges (INDENTATION PROBLEM)
8: **for** $e \in$ *edgesSorted* **do**
no need to explore more. we just want to stop over here.
9: **if** $||e|| >$ *lastDistance* **then**
10: State
StateNN
Statex
11: *rank* ← *currentIndex*
12: **end if**
13: *lastDistance* ← $||e||$
14: $||e|| \leftarrow 2^{rank}$
15: *currentIndex* ← *currentIndex* + 1
16: finally
17: **end for**
18: *perfectMatching* ← **HungarianAlg**(*edgesSorted*) ▷ returns a set of edges
19: **end**