algorithm2e 使用宏包的一点心得

打工人
种田大学

2021 年 1 月 5 日

LaTeX 工作室公众号
http://www.latexstudio.net

简 介

本文档主要是作者在平常使用以及解答一些关于 algorithm2e 宏包问题时候的心得, 特写此文档.

相比于 tabbing 制表位的繁杂，algorithms 与 algorithmicx 的简易但不高的可定制性，algorithm2e 相对复杂但是操作性更高，可定制性更强，所以在一定程度上, algorithm2e 更加实用.

algorithm2e 宏包是用于在 LaTeX2e 中编写算法的环境. 算法被定义为像图形这样的浮动对象. 它提供了允许你创建不同种类的关键字的宏，因此提供了一组预定义的关键字. 你还可以更改关键字的版式.

本文档将从实例出发，介绍关于**基本的关键字、外框的设定、一些基本的间距设定等**希望对大家有所帮助.

对于 algorithm2e 这个宏包来说，重要的是找到你需要的关键词以及你需要调整间距的关键词. 本文档只能在解决一些基本的问题，但更多问题需要阅读 *doc* 来解决问题, 更多需要自己探索, 在解决问题中得到更大的进步.

一、从一个实例开始

```
\begin{algorithm}[H]
    \SetAlgoLined
    \KwData{this text}
    \KwResult{how to write algorithm with \LaTeX2e }
    initialization\;
    \While{not at end of this document}{
        read current\;
        \eIf{understand}{
            go to next section\;
            current section becomes this one\;
        }{
            go back to the beginning of current section\;
        }
    }
    \caption{How to write algorithms}
\end{algorithm}
```

Algorithm 1: How to write algorithms

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

algorithm 与 **figure** 和 **table** 一样, 都属于浮动体, 都需要在浮动体的环境下进行排版, 所以大多数与两种常规的环境相似, 但在撰写此文档中也发现, 倘若将上述代码放入一个子文件中再利用 `\include{file}` 后算法块会自动浮动到下一页的顶部, 且这种矛盾无法调和.

在宏包的 doc 里这样解释到: 可选参数 `[Hhtbp]` 的作用类似于图形环境. H 参数强制算法保持不变. 如果使用, 则算法不再是浮动对象. 注意: 算法无法剪切, 因此, 如果在给定位置没有足够的位置放置带有 H 选项的算法, 则 LaTeX 将放置一个空格并将该算法放在下一页.

需要注意的是, 每一行的结尾必须以 `\;` 结束, 只有那些以宏命令开始的不应该以 `\;` 结束, 例如在此实例中的 `\SetAlgoLined`、`\KwData{var}` 属于宏命令, 不需要以 `\;` 结尾, 而例如 `\initialization` 这样的行内的命令需要以 `\;` 结束, 事实上可以理解为需要打出冒号而加入转义字符.

标题的工作方式与图形环境相同, 不同之处在于标题应位于算法的末尾. `\listofalgorithms` 将其用作算法列表的参考名称. 还可以使用与软件包一起提供的 **title** 宏, 但是此宏不会在算法列表中插入一个条目. 关于

标题的设定, 在后续的章节会有讲述.

二、常用的环境以及一些基本的修改

1、常用的环境

algorithm2e 提供 4 种环境:

algorithm: 这是最常用的环境。

algorithm*: 与前者一样, 但它用于两列文本中, 使算法跨两列

procedure: 该环境类似于算法环境, 但是:

- 推荐使用 `ruled` 和 `algoruled`.
- 标题写作 `Procedure name`.
- `\caption` 命令的语法限制如下：您必须输入名称, 后接 2 个大括号, 例如 “Name ()” . 您可以将参数放在大括号中, 然后放在文本中. 如果未提供任何参数, 括号将在标题中删除.
- 现在,label 将 procedure 或 function 的名称（标题中大括号前的文本）作为参考（而不是经典算法环境中的数字）.
- 标题中设置的 procedure 或 function 的名称会自动定义为 KwFunction, 因此可以用作宏. 例如, 如果在过程环境中设置 `\caption{myproc ()}`, 则可以在主文本中使用 `\myproc` 宏. 注意该宏仅在 `\caption` 之后定义!
- `nokwfunc` 无法在 function 和 procedure 环境中使用上述功能. 例如, 如果您将不能使用命令名称的 procedure 或 function 的名称用作数学显示时它则很有用.

procedure*: 与上述的 procedure 一致, 但它用于 two columns 模式中, 使 procedure 跨两列.

function: 与前面的一致, 但标题中使用 Function 而不是 Procedure.

function: 与前面的一致, 但在 two columns 模式中使得文本跨两列.

2、一些基本关键词的修改

如果你不喜欢原来的算法标题或寻找其他方法进行修改, 可以使用以下命令更改算法的名称:

`\SetAlgorithmName{algorithmname}{algorithmautorefname}{list of algorithms name}`

它将重新定义重新定义算法的名称和算法的句子列表. 例如, 我们想定义一个算法的名称为算法 x 我们采用两种方式,`\SetAlgorithmName{算法}{算法}{ }` 及 `\renewcommand{\algorithmcfname}{算法}{}{}`
其中 `<list of algorithms name>` 插入具有标题的所有算法的列表.

`\SetAlgoProcName{aname}{anautorefname}`

设置 procedure 的标题名称, 其中第二个参数是 **hyppref** 宏包的 `\autoref` 将使用的名称.

`\SetAlgoFuncName{aname}{anautorefname}`

与 procedure、algorithm 的修改方法一致, 同样的, 第二个参数也是 **hyppref** 宏包的 `\autoref` 将使用的名称.

算法 2: 如何写一个算法

变量和参数: *this text*

Result: how to write algorithm with LaTeX2e

initialization;

while *not at end of this document* **do**

read current;

if *understand* **then**

go to next section;

current section becomes this one;

else

go back to the beginning of current section;

end

end

输出: this is output

由于在 algorithm2e 中已经预先定义好了一些关键词, 例如 `\KwIn{}`、`\KwOut{}` 以及 `\KwData{}` 等, 我们可以采用如下方法进行重新定义, 即 `\SetKw{KwData}{\textcolor{red}{\textbf{变量和参数:}}}{}`, 同理我们也可以对上述其他参数进行相应的设置. 具体示例及所见效果见 algorithm 2.

在 alogrithm2e 中定义了如下的宏:

输入与输出

- `\KwIn{input}`
- `\KwOut{output}`
- `\KwData{input}`
- `\KwResult{output}`

基本的关键词和块

- `\KwTo`
- `\KwRet{[value]}`
- `\Return{[value]}`
- `\Begin{block inside}`
- `\Begin(begin comment){block inside}`

需要注意的是, 这些关键词是严格区分大小写的, 所以需要确保用户的输入正确. 其他的关键词也可以 `texdoc algorithm2e` 进行查阅.

3、控制算法的布局

由于在引入宏包时, 在 option 选项中加入了 `vlined` 以及 `ruled` 的参数, 所以在伪代码中会有上下尺, 并且在一定的伪代码中会出现垂直尺.

当然, 外部的封装环境不止 ruled 一种选项, 也可以选择完全封闭的选项亦或者完全没有外部边框的选项. 内部提供的封装环境有以下几种选项:

boxed: 将算法封装在框中.

boxruled: 用方框将算法环绕, 将标题放在上方, 并在标题后添加一行.

ruled: 在顶部和底部都有一条线的算法. 请注意, 标题不再位于算法下方, 而是在算法开始时设置.

algoruled: 如上所述, 但在尺后留有多余的空格.

tworuled: tworuled 的行为就 ruled 的一样, 但标题后面没有加一行.

plain: 默认值, 无功能.

`\RestyleAlgo{style}`

由于在可选参数中加入了 ruled 选项, 如果需要临时修改算法的样式的话, 可以采用此命令临时修改某一个算法的外观布局. 例如 *boxed, boxruled, ruled and algoruled.*

`\SetAlgoVlined`

此命令能够让你在每个块的开始和结束之间打印一条垂直线, 然后打印一条水平线.

`\SetInd{before rule space}{after rule space}`

设置垂直标尺之前和之后的空间大小. 在 `\OnLine` 模式下, 缩进空间是这两个值的和, 默认情况下为 0.5em 和 1em.

`\SetAlgoHangIndent {length}`

设置 *hangindent* 用于在 *long* 语句中的第一行之后缩进后继行的缩进长度值.

`\Setvlineskip{length}`

设置小水平线之后的垂直空间的值, 该水平线在 *vlined* 模式下关闭一个 block.

`\SetAlgoSkip{skip command}`

算法在前后放置额外的垂直空间（与文本之间的前后）, 以免出现框状或规则式算法的文本颠簸行. 默认情况下, 这是一个. 您可以使用此宏更改此值. 四种可能性是

- `\SetAlgoSkip{}` 没有额外的垂直跳动.
- `\SetAlgoSkip{smallskip}` 设置为默认的.
- `\SetAlgoSkip{medskip}` 有一个较大的间距.
- `\SetAlgoSkip{bigskip}` 有一个更大的间距.

`\SetAlgoInsideSkip{skip command}`

算法前后没有多余的垂直空间算法的核心. 因此, 文本将以方框或格线样式放在行后. 要放置多余的空间, 需要使用 `\SetAlgoInsideSkip{skip command}` ,

`\IncMargin{length}`
增加算法文本到边界的距离.
`\DecMargin{length}`
减少算法文本到边界的距离.
`\setlength{\interspacetitleruled}{xx em}`
它控制的是 top rule 和 mid rule 和文本间的距离, 默认的距离为 2pt.
`\setlength{\interspacetitleboxruled}{x \lineskip}`
它控制盒式算法中规则和标题之间的垂直空间. 默认的距离为 2 `\lineskip`.
`\SetCustomAlgoRuledWidth{length}`
这个选项控制的是外部封装尺的长度, 如果你的算法不够长可以考虑修改此选项.

4、设置算法的标题和题目

`\SetAlgoCaptionSeparator{sep}`
这将在算法标题(算法 1)和算法名称之间设置分隔符. 默认情况下为 " : ", 标题看起来像 "algorithm3: 名称", 但是现在您可以通过使用示例来更改它, 该名称将为 "algorithm3. 名称" .
`\AlCapSkip{}`
是纯文本和盒装模式下算法主体与 caption 之间距离的尺寸. 可以手动更改, 也可以使用 `\SetAlCapSkip {0ex}` 进行更改, 当然也可以使用 `\SetAlCapSkip{length}` 进行修改.
`\SetAlCapHSkip{length}`
在 ruled 选项中,caption 位于 toprule 和 mid rule 之间, 可以通过设置此选项进行水平缩进.

下面是 title 和 caption 的样式:

`\AlCapSty{<text>}`: sets `<text>` 它与 `\AlCapFnt` 一起使用, 以打印 `Algorithm #:` 更准确的说是 `\AlCapSty{\AlCapFnt Algorithm #:}`.

`\AlCapNameSty{<text>}` 在 caption 名称排版中设置 `<text>`, 该名称与 `\AlCapNameFnt` 一起使用以打印通过调用 `\caption {name}` 设置的标题名称. 更准确地说, 其打印出 `\AlCapNameSty{\AlCapNameFnt 名称}` 给出 "caption" . 默认情况下,`\AlCapNameSty`为 `textnormal`

事实上, 只要了解知道有 *caption*、*title*、*procedre*、*function* 这些 words 后在到文档中查找修改命令即可, 更多样式的修改可以自行参阅文档, 再此不再一一列举.

三、修改 rule 的颜色和厚度

对于希望修改 ruled 环境中的 rule 的颜色 texstackexchange 中提供了一种解决办法. 而想要修改 rule 厚度的尚有待考察, 宏包内没有直接修改的命令, 需要修改原始命令.

```
\makeatletter
\newcommand{\setalgotoprulecolor}[1]{\colorlet{toprulecolor}{#1}}
\let\old@algocf@pre@ruled\@algocf@pre@ruled % Adjust top rule colour
\renewcommand{\@algocf@pre@ruled}{\textcolor{toprulecolor}{\old@algocf@pre@ruled}}

\newcommand{\setalgobotrulecolor}[1]{\colorlet{bottomrulecolor}{#1}}
\let\old@algocf@post@ruled\@algocf@post@ruled % Adjust middle rule colour
\renewcommand{\@algocf@post@ruled}{\textcolor{bottomrulecolor}{\old@algocf@post@ruled}}
```

```
\newcommand{\setalgomidrulecolor}[1]{\colorlet{midrulecolor}{#1}}
\renewcommand{\algocf@caption@ruled}{%
    \box\algocf@capbox{\color{midrulecolor}\kern\interspacetitleruled\hrule
        width\algocf@ruledwidth height\algotitleheightrule depth0pt\kern\interspacealgoruled
            }}
\makeatother

\setalgotoprulecolor{blue!30}% Default
\setalgobotrulecolor{red!30}% Default
\setalgomidrulecolor{green!30}% Default
```

算法 3: 如何写一个彩色边框的算法

变量和参数: *this text*

Result: how to write algorithm with LaTeX2e

```
1 initialization;
2 while not at end of this document do
3     read current;
4     if understand then
5         go to next section;
6         current section becomes this one;
7     else
8         go back to the beginning of current section;
```

输出: this is output