学　　号　16337259(谢江钊)
密　　级　2016 级

武汉大学本科毕业论文

基于流水线 CPU 的分支预测技术性能研究

院 (系) 名 称：　数据科学与计算机学院

专 业 名 称：　计算机类

学 生 姓 名：　谢江钊　徐正金

指 导 教 师：　李国桢　教授

二〇一八年一月

Computer Organization Theory
Course Essay

Research on the Performance of Branch Prediction Technology Based on Pipeline CPU

School (Department): School of Data and Computer Science
Major: Computer Class
Candidate: Xie Jiangzhao Xv Zhengjin
Supervisor: Prof. Li Guozhen

Sun Yat-sun University

January, 2018

郑 重 声 明

本人呈交的论文, 是独立进行研究工作所取得的成果, 所有数据、图片资料真实可靠. 尽我所知, 除文中已经注明引用的内容外, 本论文的研究成果不包含他人享有著作权的内容. 对本论文所涉及的研究工作做出贡献的其他个人和集体, 均已在文中以明确的方式标明. 本论文的知识产权归属于培养单位.

本人签名: _____ 日期: _____

摘　　要

在文章中我对 CPU 发展历史以来所采取的预测策略进行了软件上的模拟，对流水线 CPU 动态预测的性能作出了分析统计。关于此次的 CPU 设计，我实现了 32 位 MIPS 指令集。在学习 CPU 设计过程中，流水线 CPU 的出现极大地提高了 CPU 的执行效率，但同时也需要合理解决三大冲突。对于控制冲突而言，需要做到尽可能准确预测。近代 CPU 在这一方面做分支预测已经非常准确，但是流水线深度的逐渐增加使得哪怕微小的提升也将极大影响 CPU 的性能。因此文中对过往和近年来热门的预测方法作出评价。

为了减少工作量，我采用 Python 作为工具来解决这个问题。软件层面上我实现了对 MIPS 汇编程序的模拟执行，并对预测效果作出了统计。

分析结果表现混合分支预测器的性能非常优越，相比于传统的局部分支预测和全局分支预测，能够在更短时间内提高预测率并且拥有更加稳定准确的预测正确率。文中的分析结果提供了一种可行性高的办法来评价 CPU 预测采取策略，可以作为探究分支策略效果的一种手段。

关键词: 计算机组成原理; 流水线 CPU; 动态分支预测;

ABSTRACT

In the article, I have carried on the software simulation to the forecast tactics that has taken since the CPU development history, has carried on the analysis statistics to the pipeline CPU dynamic forecast performance. About this CPU design, I have realized 32 MIPS instruction set. In learning CPU design process, The advent of pipelined CPUs has greatly increased the efficiency of CPU implementation, but at the same time it also requires a reasonable solution to the three major conflicts. For control conflicts, forecasts need to be as accurate as possible. Modern CPU in Branch prediction is already very accurate on the one hand, but the gradual increase in pipeline depth makes even a small increase will greatly affect the CPU performance. Therefore, the text in the past and in recent years the popular prediction method to make an assessment.

In order to reduce the workload, I use Python as a tool to solve this problem. At the software level, I implemented the simulation of the MIPS assembler and made statistics on the predicted results.

The results of the analysis show that the performance of the hybrid branch predictor is superior. Compared with the traditional local branch prediction and global branch prediction, the hybrid branch predictor can improve the prediction rate in a shorter time and has a more stable and accurate prediction accuracy.

The analysis results in this paper provide a highly feasible way to evaluate the CPU pre-

diction strategy, which can be used as a means to explore the effect of the branch strategy.

Key words: Computer composition principle; Pipeline CPU; Dynamic branch prediction

目　　录

1 绪论

1.1 分支预测算法的意义

分支预测器的重要性在课本中已经得到相当详尽的论述，在这里就不再从原理上去解释它了。回到工业生产中去，近年来 CPU 的发展是巨大的，技术的更新集中在更高的制程，频率，深流水线，指令集并行，超线程技术等等上。其中的流水线加深，更是使得条件分支带来的控制冲突变得日趋严重。分支预测进行得不准确，意味着预测的指令全部都要抛弃不再进行，这一点会导致极大的性能浪费。其中，奔腾 4 便是一个失败的例子。奔腾 4 采用的 31 级超长流水线，把主频提高到了 3Ghz 以上，但是性能却比不过同时期低主频 14 级流水线的 AMD 处理器，其中一个重要的原因便是没有采用高效的分支预测器，导致流水线中断问题极其严重，导致了性能的严重下降。今天的 CPU 普遍采用 14-16 级流水线原因之一，便是分支预测技术瓶颈。

业界引入了动态分支预测技术，开发出多种预测方案，使得控制冲突逐渐得到解决。其中较为贴近现代的便是局部分支预测器，全局分支预测器，循环分支预测器以及间接分支预测器等等，在这里我将对一部分方案作出解释。另外，由 S. McFarling 提出的混合（融合）分支预测器相比之下会有更加好的预测结果，同样非常值得考虑作为 CPU 预测器发展的方向。

更前沿地，当前高端 CPU 已经迈进了神经网络预测以及多种预测器相结合的道路，在这里由于技术过于复杂便不再进行赘述。预测器的发展需要考虑到速度和效果，这一点也是需要企业来进行权衡的。如何平衡两者的关系，也是未来发展的一个重点。做这个项目的时候感触良多，对指令执行特点和 CPU 自身设计有了很多的体会。而这一个项目实现的内容将来也能够不断改进，作为未来的教学预测性能预测手段。

1.2 技术发展现状与趋势

1.2.1 静态分支预测技术

静态分支技术是最简单的技术，分为两种，强制跳转或强制不跳，而不依赖于过去的历史记录。后来的技术变得更加复杂，假定了向后分支必然发生，而向前的分支不会发生，对应着程序设计中经常使用到的 for 循环和 if 判断特性，能够更好地进行预测。在早期的时候一些处理器甚至允许分支预测提示出现在代码中，但是后来便不再采用。对于这种做法，理论上是不应该存在的，对于程序员来说，底层的 CPU 实现应该做到透明，而不是反过来去负责做分支预测。

静态分支预测技术的优点在于对硬件实现基本没有要求，但是预测的效果却见不得理想，尤其对于日渐复杂的程序而言，其准确率是不能够被接受的。

1.2.2 1bit 分支预测器

1bit 分支预测器引入了一位分支预测缓存, 记录分支最后一次的选择，然后预测下一次也做出同样的决定。但是这种预测方法有一个问题在于，如果遇到 TFTFTFT... 的情况，会造成大量的空翻，从而使得性能被严重浪费。并且由于其总是预测分支跳转/不跳转一定发生，一旦出现例外情况就会造成两次错误。为了解决这个问题，引入 2bit 分支预测器，来给预测引入一个缓冲区域。

1.2.3 2bit 分支预测器

2bit 分支预测器是基于饱和计数器进行的，饱和计数器是一个拥有 4 个状态的状态机: 强跳转，弱跳转，弱不跳转，他们之间的相互转换关系如下:

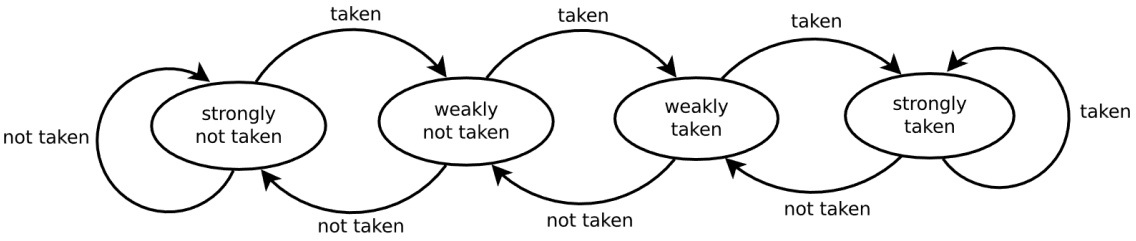

图 1.1 饱和计数器[?]

2bit 分支预测器以跳转指令的 PC 地址作为索引，然后查找到对应的饱和计数器，根据高位输出结果:

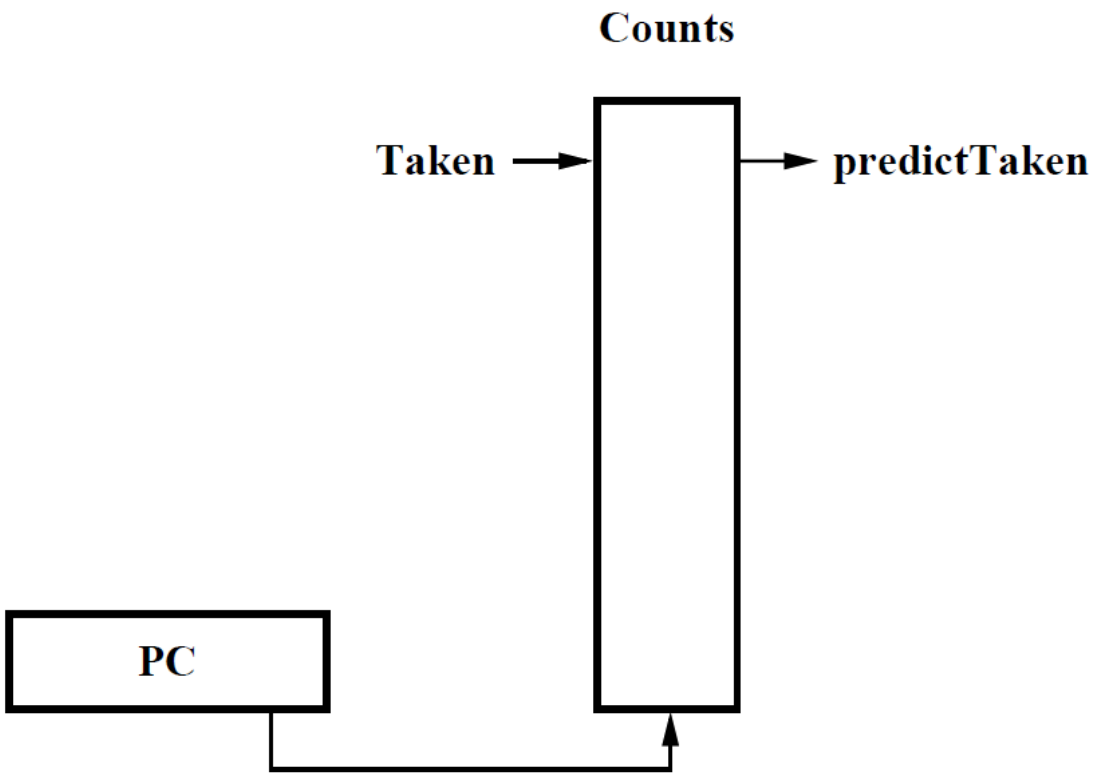

图 1.2　2bit 分支预测器[?]

这种方法的优点在于指令必须选择某个分支连续两次，才能够实现状态的翻转，进而改变预测的分支。回到刚才的情况，一开始的时候饱和计数器处于强跳转的状态，那么一次 F 的影响只能使计数器回到弱跳转状态，避免了接连的空翻，相比之下错误率下降。同样的道理，我们还可以提出 nbit 分支预测器，来给予更多的缓冲空间，但是这种做法也导致了跳转不能够灵活地得到改变，在测试中反而导致预测率下降。

1.2.4　局部分支预测器

前面所描述的 2bit 分支预测器，用状态机的办法把过去若干条指令改变的结果保存了下来，但是这样有一个问题，认为本次的跳转与之前的若干条指令跳转有关，但是处理起来却把本次预测结果应该基于过去的情况。举个例子，假如之前

的几次跳转结果是 TTF，那么这一次的结果如果仅仅看作前几次的衍生的话，便有可能出错。例如对于一个循环而言，循环 3 次跳转最后一次不跳转 (跳出循环)，如果依靠 2bit 分支预测器就会出现预测错误的结果，因为前三次的跳转不能够表明这一次也一定跳转。相对应的是，我们可以把前三次的历史记录看作是一种模式，然后在此基础上构建饱和计数器，经过若干次训练，便可以得到一个结果: 三次的跳转必然伴随着一次结果，这样我们往后的预测便会变得准确，提高了我们的正确率。

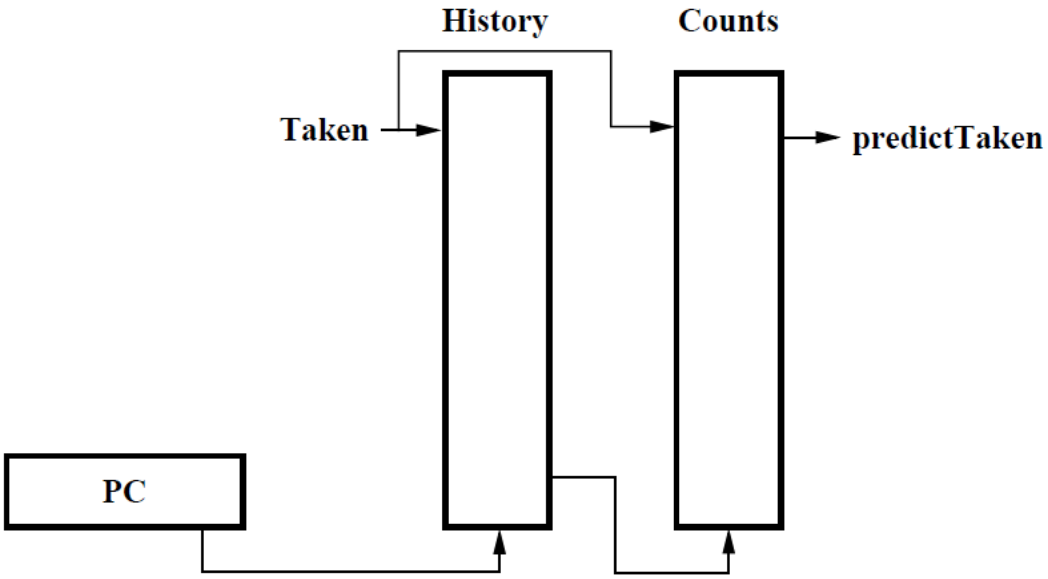

图 1.3　局部分支预测器[?]

局部分支预测器构建了两个表: 一个被称为 BHT(Branch History Table)，另一个被称为 BPT(Branch Pattern Table)。BHT 以跳转指令的地址作为索引，记录改分支的历史记录，然后根据分支的历史记录去查询对应的饱和计数器，根据计数器的高位进行相对应预测。预测的结果同样会不断更新历史和计数器。在经过大量训练后便可以得到相当优秀的正确率。这种预测方式被现代 CPU 普遍地采纳。

1.2.5 全局分支预测器

局部分支预测器仅仅通过指令自身的预测历史预测了指令的跳转，但是却没有考虑到指令之间的相关性。对于某些指令，例如

if(a>0) todo;

if(a<=0) todo;

第一条指令和第二条指令必然只会发生一条，通过观察第一条指令的跳转情况便可以预测第二条指令的跳转结果。对于一个不断变化的 a，这个时候局部分支预测器便不能仅仅通过自身的历史记录来推测出下一次跳转情况。因此，全局分支预测器采取的做法是，记录下前几条指令的跳转情况，然后通过这个历史记录来查找对应的饱和计数器，从而解决了跳转指令的相关问题。

全局分支预测器通过 GR(Global History) 来索引饱和计数器，获得对应的预测结果。

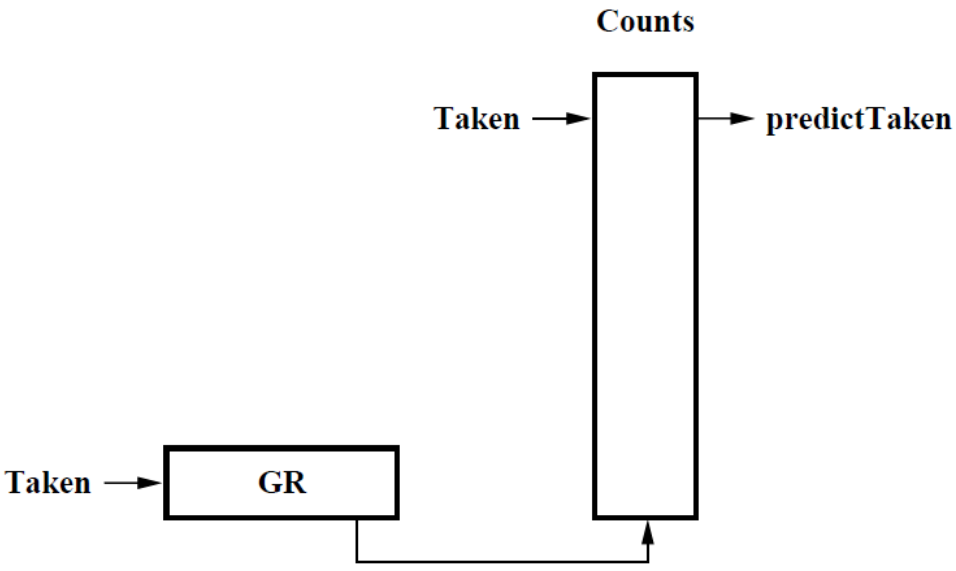

图 1.4　全局分支预测器[?]

1.2.6 混合分支预测器

相比以上两种分支预测器，混合分支预测器仅仅是简单地将他们组合起来，从而在面对复杂的分支情况时能够更有效地采取不同的预测器，获得更加优秀的表

现。混合分支预测器将预测的任务分别交给全局分支预测器和局部分支预测器去预测，然后根据它们的预测结果以及预测的指令地址，来更新自己的索引表，这个表记录着若干的饱和计数器[?]，饱和计数器的状态决定了全局分支预测器决定要采纳哪一种预测结果。

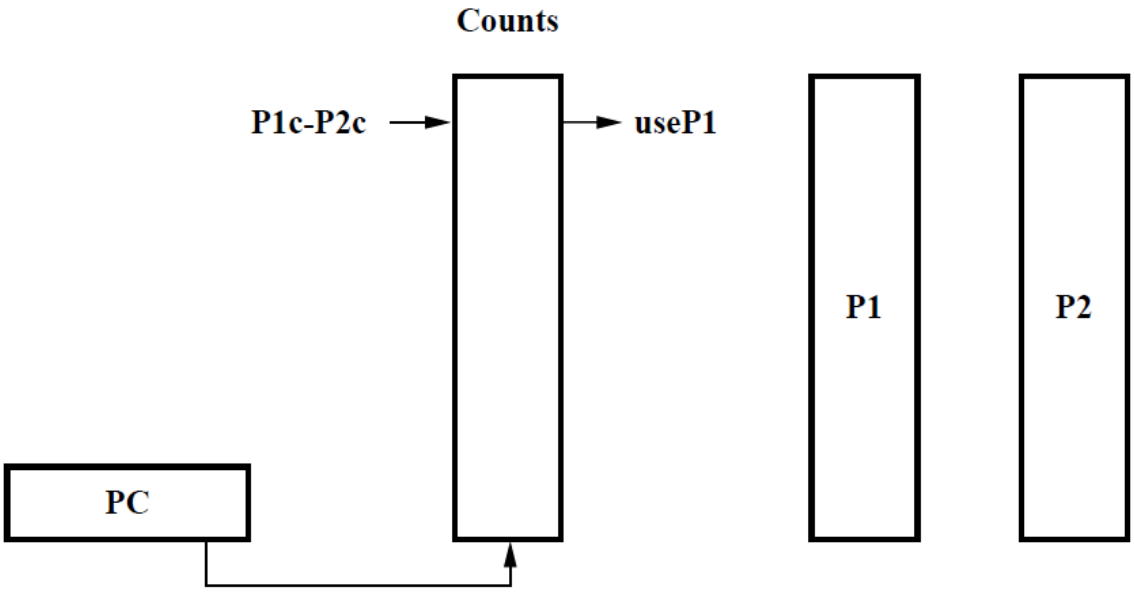

图 1.5　混合分支预测器[?]

1.2.7　基于上述分支预测器的优化

由于硬件方面的限制，上述分支预测器的索引不可能直接取到整个 PC 的大小，相对地，我们会采用记录 PC 低位的方法来作为表的索引。对于全局历史分支预测器，还发展出了 PC 地址和 GR 记录做异或的方法来作为索引，这种做法能更好地避免冲突。但是受制于所写汇编程序的大小，相关的优化也就没有进行。对于 PC 地址我做了对 32 取模，用来作为索引。

2　分支预测算法的软件实现与验证

2.1　32 位 MIPS 指令集实现

指令集方面我实现了所有 32 位 MIPS 指令:

- R 型指令:add,addu,sub,subu,and,or,xor,nor,slt,sltu,sll,srl,sra,sllv,srlv,srav,jr
- I 型指令:addi,addiu,andi,ori,xori,lui,lw,sw,beq,bne,slti,sltiu
- J 型指令:j,jal

至于更具体的细节可以参考http://blog.csdn.net/yixilee/article/details/4316617 在实现文件中，对读入的每条指令分解为 opcode,rs,rt,rd,reserved 等部分，从而实现对指令的译码，这一点与硬件上保持一致。关于软件的更多细节在此不再赘述，项目的详细文档可以在 https://github.com/xiejiangzhao/MIPS_Predict获取，在此基础上还可以加入对 x86 指令集，新增其他预测方法的支持。

2.2 不同预测方法的效果测试

我在文档中写了一份简短的冒泡排序，然后对若干随机数进行排序，得到以下结果: 可见，在这种情况下，全局分支预测器和两位预测器取得了非常不错的效

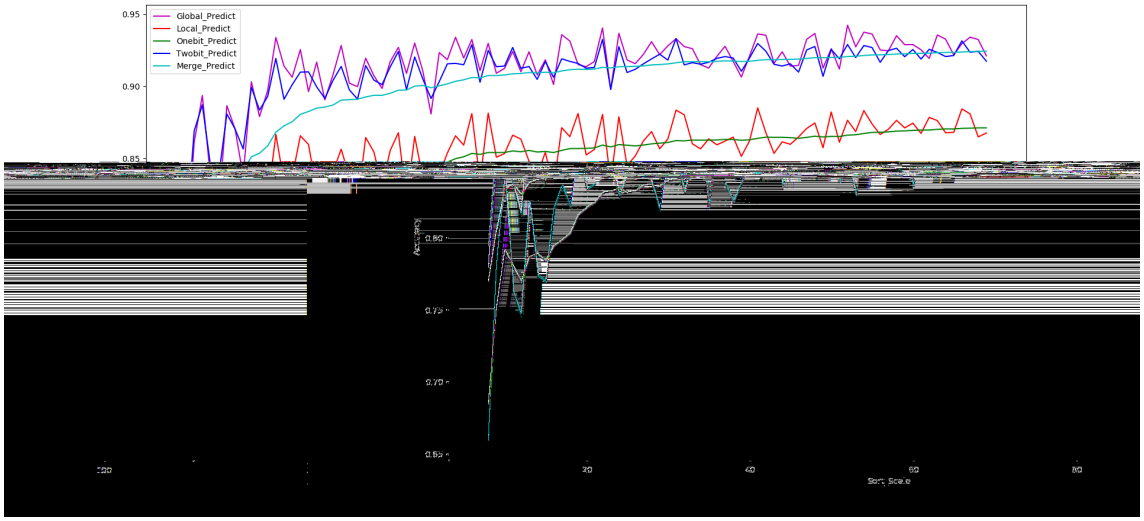

图 2.1 冒泡排序预测结果[?]

果，融合分支预测器的结果更加平稳。

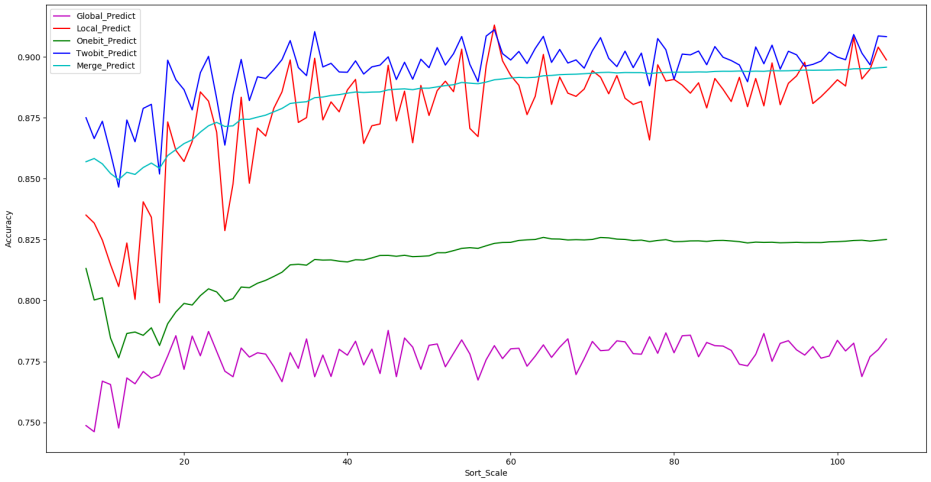

图 2.2　冒泡排序预测结果[?]

2.3　字体调节

\songti	宋体
\heiti	**黑体**
\fangsong	仿宋
\kaishu	楷书

2.4　字号调节

字号命令: \zihao

\zihao{0} 初号字 English

\zihao{-0} 小初号 English

\zihao{1} 一号字 English

\zihao{-1} 小一号 English

\zihao{2} 二号字 English

\zihao{-2} 小二号 English

\zihao{3} 三号字 English

\zihao{-3} 小三号 English

\zihao{4} 四号字 English

\zihao{-4} 小四号 English

\zihao{5} 五号字 English

\zihao{-5} 小五号 English

\zihao{6} 六号字 English

\zihao{-6} 小六号 English

\zihao{7} 七号字 English

\zihao{8} 八号字 English

2.5 已加入的常用宏包

cite 参考文献引用, 得到形如 [3-7] 的样式.

color,xcolor 支持彩色.

enumerate 方便自由选择 enumerate 环境的编号方式. 比如

`\begin{enumerate}[(a)]` 得到形如 (a), (b), (c) 的编号.

`\begin{enumerate}[i)]` 得到形如 i), ii), iii) 的编号.

另外要说明的是, itemize, enumerate, description 这三种 list 环境, 已经调节了其间距和缩进, 以符合中文书写的习惯.

2.6 标点符号的问题

建议使用半角的标点符号, 后边再键入一个空格. 特别是在英文书写中要注意此问题!

双引号是由两个左单引号、两个右单引号构成的: `` ''. 左单引号在键盘上数字 1 的左边.

但是, 无论您偏向于全角或半角, 强烈建议您使用实心的句号, 只要您书写的是自然科学的文章. 原因可能是因为, 比如使用全角句号的句子结尾处的 "x。" 容易误为数学式 x_0(`$x_0$`) 吧.

2.7 引用的问题

2.7.1 参考文献的引用

参考文献的引用, 用命令 `\cite{ }`. 大括号内要填入的字串, 是自命名的文献条目名.

比如, 通常我们会说:

关于此问题, 请参见文献 [**?**]. 作者某某还提到了某某概念[?].

上文使用的源文件为:

```
关于此问题, 请参见文献 \cite{r2}. 作者某某还提到了某某概念 \upcite{r1}.
```

其中 `\upcite` 是自定义命令, 使文献引用呈现为上标形式.

(**注意**: 这里文献的引用, 有时需要以上标形式出现, 有时需要作为正文文字出现, 为什么?)

另外, 要得到形如 [**?**, **?**, **?**, **?**] 的参考文献连续引用, 需要用到 cite 宏包 (模板已经加入), 在正文中使用 `\cite{r1,r3,r4,r5}` 的引用形式即可. 或者, 连续引用的上标形式: 使用 `\upcite{r1,r2,r3}`, 得到[?, ?, ?].

2.7.2 定理和公式的引用

定理 2.7.1 (**谁发现的**) 最大的正整数是 1.

证明　要找到这个最大的正整数, 我们设最大的正整数为 x, 则 $x \geqslant 1$, 两边同时乘以 x, 得到

$$x^2 \geqslant x. \tag{2.1}$$

而 x 是最大的正整数, 由 (**??**) 式得到

$$x^2 = x.$$

所以

$$x = 1.$$

定理 **??** 是一个重大的发现.

定义 2.7.1 (**整数**)　正整数 (例如 1, 2, 3)、负整数 (例如 −1, −2, −3) 与零 (0) 合起来统称为**整数**.

注 2.7.1　整数集合在数学上通常表示为 **Z** 或 $\mathbb{Z}$, 该记号源于德语单词 Zahlen(意为 “数”) 的首字母.

性质 2.7.1　任意两个整数相加、相减、相乘的结果, 仍然是整数.

例 2.7.1　$1 + 2 = 3$.

推论 2.7.1　在整数集合内, 相加、相减、相乘运算是封闭的.

2.8　图形与表格

支持对 eps, pdf, jpg 等等常见图形格式.

再次澄清一个误会: LaTeX 支持的图形格式绝非 eps 这一种. 无需特意把图片转化为 eps.

用形如 `\includegraphics[width=12cm]{Daisy.jpg}` 的命令可以纳入图片.

如图 **??** 是一个纳入 jpg 图片的例子.

表格问题, 建议使用 “三线表”, 如表 **??**.

图 2.3　一个彩色 jpg 图片的例子

表 2.1　一般三线表

123	4	5	123	4	5123	4	5	123	4	5
67	890	13	123	4	5123	4	5	123	4	5
67	890	13	123	4	5123	4	5	123	4	5
67	890	13	123	4	5123	4	5	123	4	5

3 其他事项

以下是广告时间, 插播一段广告:

- 插图的制作, 建议用 pgf, 也叫 tikz. pgf 的长处是源文件直接植入 TEX 文档, 管理起来非常方便. 这里有我写的一个关于初次使用 pgf 的帖子:
  http://bbs.ctex.org/forum.php?mod=viewthread&tid=30480.
- 生成参考文献, 建议使用 BibTeX. 这里有我写的一个文档:
  http://bbs.ctex.org/forum.php?mod=viewthread&tid=26056.

  使用 BibTeX 做参考文献时, 借助 EndNote 或者 NoteExpress, 可以非常漂亮简单地解决 bib 文件的录入问题. NoteExpress 在校图书馆网站有正版软件提供下载. 当然 EndNote 本身就是 Thomson Corporation 推出的 (和 SCI 搜索引擎是同一家公司), 和多个重要文献搜索引擎有良好的功能配合.

  Google 学术搜索也提供了文献的 bib 格式. 录入参考文献时, 偶尔用一用 Google 学术搜索, 还可以核查或减少录入的错误, 并减少录入的工作量.
- 幻灯片的制作, 建议使用 Beamer. 这里有我写的一个模板, 谨供参考:
  http://bbs.ctex.org/forum.php?mod=viewthread&tid=27695.

参考文献

[1] Scott McFarling,Combining Branch Predictors,WRL Technical Note TN-36, 1993.

[2] Wikipedia,https://en.wikipedia.org/wiki/Branch_predictor.

[3] 邓建松等,《LaTeX 2$_\varepsilon$ 科技排版指南》, 科学出版社.

[4] 吴凌云,《CTeX FAQ (常见问题集)》, *Version 0.4*, June 21, 2004.

[5] Herbert Voß, Mathmode, http://www.tex.ac.uk/ctan/info/math/voss/mathmode/Mathmode.pdf.

致　　谢

感谢你, 感谢他和她, 感谢大家.

附录 A　测试

A.1　第一个测试

测试公式编号

$$1+1=2. \tag{A.1}$$

表格编号测试

表 A.1　测试表格

11	13	13	13	13
12	14	13	13	13

附录 B　附录测试

测试

附录 C　附录测试

测试