

## Miscellanies

西二闲画生  
Department of Life,  
Zhonghua Uni.,  
WH, China  
*[zhu@riseup.net](mailto:zhu@riseup.net)*

November 24, 2018



# Preface

$\text{\LaTeX}$  typesets text as *What You See Is What You Type* compared with *What You See Is What You Get*. It gives finer control over text layout with a host of commands and add-ons. This book firstly demonstrates how  $\text{\LaTeX}$  supports Chinese typesetting and then moves on to syntax. A bit of  $\text{\AcTeX}$  with Emacs configuration and key bindings are dicussed as well. Gradually, topics from different disciplines (i.e. math and language) are introduced in separate parts.

Priorly, I just planned to learn  $\text{\LaTeX}$  through trial and error. As it progressed, I found  $\text{\LaTeX}$  an awesome typesetting tool, creating beautiful text output. Hence, I decided to make it a book containing different parts, covering formal subjects like math, physics, etc. Ultimately, I want to maintain a centralized offline jottings which can be testified by the book title *Miscellanies*.



# Contents

<b>Preface</b>	<b>iii</b>
<b>I L<sup>A</sup>T<sub>E</sub>X</b>	<b>1</b>
<b>1 L<sup>A</sup>T<sub>E</sub>X Resources</b>	<b>3</b>
1.1 Find References . . . . .	3
1.2 AU <sub>C</sub> TeX . . . . .	3
1.2.1 Inserting . . . . .	3
1.2.2 L <sup>A</sup> T <sub>E</sub> X Sources Formatting . . . . .	3
1.2.3 Font Specifiers . . . . .	3
<b>2 汉语</b>	<b>5</b>
2.1 CJK 包 . . . . .	5
2.1.1 CJK 和 CJK* 环境 . . . . .	5
2.1.2 字体 . . . . .	5
2.2 xeCJK 更好 . . . . .	6
2.3 fontspec 和 ctex . . . . .	6
2.4 中英文空格 . . . . .	6
2.5 行首缩进 . . . . .	6
2.6 局部字体命令 . . . . .	7
<b>3 L<sup>A</sup>T<sub>E</sub>X Tutorial</b>	<b>9</b>
3.1 T <sub>E</sub> X Units . . . . .	9
3.2 Special characters . . . . .	9
3.3 verbatim and verb . . . . .	10
3.4 item list . . . . .	10
3.5 Code listings . . . . .	11
3.6 Straight Quotes . . . . .	12
3.7 floating . . . . .	12
3.7.1 Figure . . . . .	12
3.7.2 Table . . . . .	14
3.7.3 Plot . . . . .	16

3.8	center and centering . . . . .	16
3.9	Math Equations . . . . .	18
3.10	Document Structure . . . . .	20
3.11	Boxes . . . . .	20
3.11.1	input and include . . . . .	24
3.11.2	biblatex biber . . . . .	24
3.11.3	RefTeX . . . . .	25
<b>II</b>	<b>Mathematics</b>	<b>27</b>
<b>4</b>	<b>Combinatorics</b>	<b>29</b>
4.1	Set and Multiset . . . . .	29
4.2	Counting Principles . . . . .	30
4.3	Permutation of Sets . . . . .	31
4.4	Combination of Sets . . . . .	32
4.4.1	Combination Formulas . . . . .	33
4.4.2	Application of Combination . . . . .	34
4.5	Permutation of Multiset . . . . .	36
4.6	Combination of Multiset . . . . .	37
4.7	Polynomial . . . . .	39
4.8	Grouping and Distribution . . . . .	40
4.8.1	Distribution as Injection . . . . .	40
4.8.2	Grouping . . . . .	40
4.8.3	Distribution without Injection . . . . .	42
4.8.4	Distribution of Multiset . . . . .	42
4.9	Summary . . . . .	43
<b>5</b>	<b>卡特兰数</b>	<b>45</b>
5.1	卡特兰公式 . . . . .	45
5.2	卡特兰数应用 . . . . .	49
<b>III</b>	<b>数列</b>	<b>51</b>
5.3	等差数列 . . . . .	53
5.4	等差数列幂和 . . . . .	53
5.5	等差数列的积和 . . . . .	55
5.6	幂和与积和的联系 . . . . .	57
<b>IV</b>	<b>Algorithm</b>	<b>59</b>
<b>6</b>	<b>Tree</b>	<b>61</b>
6.1	前序中序求后序 . . . . .	61

<b>7 栈</b>	<b>63</b>
7.1 进栈出栈 . . . . .	63
7.2 先进后出 . . . . .	64
7.3 栈深度最小值 . . . . .	64
7.4 出栈序列数 . . . . .	65
7.5 进出栈递推方法 . . . . .	66
7.5.1 最先出栈元素 . . . . .	66
7.5.2 $a_1$ 出栈位置 . . . . .	67
7.5.3 递归编程 . . . . .	69
7.6 进出栈总结 . . . . .	69
 <b>V Programming</b>	 <b>71</b>
<b>8 C Language</b>	<b>73</b>
8.1 申明和定义 . . . . .	73
8.2 作用域和生存周期 . . . . .	73
8.3 编译内存布局 . . . . .	74
8.4 数组和链表 . . . . .	77
 <b>9 CPP</b>	 <b>79</b>
 <b>VI Brainteaser</b>	 <b>81</b>
<b>10 Brain Teasers</b>	<b>83</b>
补全数列空缺 . . . . .	83
牛吃草 . . . . .	83
相向而行 . . . . .	84
背向而行 . . . . .	84
交换变量 . . . . .	84
 <b>11 行测图行推理</b>	 <b>87</b>
11.1 规律总结 . . . . .	87
 <b>VII Chemistry</b>	 <b>89</b>
<b>12 走进化学世界</b>	<b>91</b>
12.1 物质及其变化 . . . . .	91
 <b>13 物质列表</b>	 <b>93</b>
 <b>Reference</b>	 <b>95</b>
 <b>Appendices</b>	 <b>97</b>

<b>A Long Code Line</b>	<b>99</b>
A.1 Appendix Notes . . . . .	99
A.2 Embedded fonts in PDF . . . . .	99
A.3 pgfplotstable template . . . . .	100
<b>Postscript</b>	<b>107</b>



# **Part I**

# **L<sup>A</sup>T<sub>E</sub>X**



# Chapter 1

## L<sup>A</sup>T<sub>E</sub>X Resources

### 1.1 Find References

In part 2 on page 5, we have a simple clarification on how to insert Chinese characters in English articles. We will focus on L<sup>A</sup>T<sub>E</sub>X itself in this part.

Please find more appropriate at L<sup>A</sup>T<sub>E</sub>X ctan. Also, please find answers at [1]. You'd better go through the tutorial series [2]. My personal website [3] might give some ideas.

### 1.2 AUCTeX

#### 1.2.1 Inserting

Have a look at L<sup>A</sup>T<sub>E</sub>X

#### 1.2.2 L<sup>A</sup>T<sub>E</sub>X Sources Formatting

Use C-c C-q C-s to format L<sup>A</sup>T<sub>E</sub>X *section* code. C-c C-q C-p is to format *paragraph* text.

#### 1.2.3 Font Specifiers

Use C-c C-f C-i to insert *italic* text. Similarly, Use C-c C-f C-b to insert *bold* text. Read more at Insert Font Specifiers.

Binding	Function
C-c C-e	insert environment
C-c C-s	insert section
C-c C-m	macro (command)
M-return	another item
C-c C-c	compile $\text{\LaTeX}$
C-c C-v	preview output
C-c =	show document layout by RefTeX
C-c C-o C-b	fold the buffer
C-c C-o b	unfold the buffer
C-c C-r	compile selected region
C-c C-b	compile current buffer

Table 1.1: AUCTeX Bindings

C-c C-f C-i	italic
C-c C-f C-b	bold
C-c C-f C-e	emphasize
C-c C-f C-s	slight italic
C-c C-f C-r	roman
C-c C-f C-f	sans
C-c C-f C-d	delete font formatting

## Chapter 2

# 汉语

对字体的引用要注意，如果一个字体族 `family` 有多个变种<sup>1</sup>，请引用字体的 `postscriptname`。

默认情况下，`LATEX` 会把文档里用到的字体子集内嵌到生成的 PDF 里，可以通过命令行 `pdffonts filename.pdf` 查询，结果如 A 所示。

此例说明，对应文档内嵌了七个体子集。`LATEX` 文档通常只会用到字体文件的部分 `code point`，所以只需要内嵌使用到的部分即可。每个字体的前面都有一串无规则字母，代表对应字体文件被内嵌的子集。还有一列叫 `sub` 也说明是否是子集。

## 2.1 CJK 包

在英文文档里插入少量中文

### 2.1.1 CJK 和 CJK\* 环境

在英文文档里用 CJK 包的 CJK 或 CJK\* 环境插入少量汉字。后者会忽略汉字后面的空格，推荐使用。

### 2.1.2 字体

`bsmi` 字体可能没有，尝试换一个如 `gkai`, `gbsn` 等。

---

<sup>1</sup>如 `regular`, `bold`, `italic` 等。

## 2.2 xeCJK 更好

xeCJK 在处理细节上更好，关键是可方便设置中英文字体。用 xeCJK 则不需要特殊环境包裹汉字，只需在“导言区”设置好中英文字体即可。

xeCJK 的 indentfirst 选项：

```
\usepackage[indentfirst]{xeCJK}
```

已过时，推荐直接用 indentfirst 宏包：

```
\usepackage{indentfirst}
```

## 2.3 fontspec 和 ctex

还有 *fontspec* 和 *ctex* 包可以实现中文输入，更多详情请看 *L<sup>A</sup>T<sub>E</sub>X* post.

- *fontspec* sets western or Chinese fonts (i.e. main, sans, serif, bold, italic, and bolditalic).
- CJK sets Chinese, Japanese, and Korean fonts, punctuation etc. for TeX engine.
- xeCJK similar to CJK but for XeTeX engine.
- CTeX sets CJK/xeCJK document layout.

## 2.4 中英文空格

我们想要的效果是英文字符前后自动加上空格。xeCJK 则会自动处理，但是效果不是非常好。

CJK 会忽略 *T<sub>E</sub>X* 源码里的空格。用 CJK 宏集里的 CJKspace 包，源码里英文后空格会保留，中文后空格依然被 CJK\* 压缩。还有一个方法是在需要空格的地方用 tilde ~.

## 2.5 行首缩进

中文习惯每段行首缩进两个汉字。英文有两种缩进格式。默认是每小节的第一段缩进，后面的段不缩进，段间没有空行。另一种是不缩进，每段间留有空行。

本文档只插入少量中文作例子，固保留英文默认缩进方式。如要换成中文缩进，则用 indentfirst 包，使每节第一段也缩进，如下：

```
\usepackage[indentfirst]{\setlength{\parindent}{2em}
```

如若换成英文第二种缩进，则用`parskip`包，`skip`表示相临两段之间的间隙大小：

```
\usepackage[parfill]{parskip}
```

也可用相对应的命令实现：

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex plus 0.5ex minus 0.2ex}
```

## 2.6 局部字体命令

前面提到的方法都是全局生效，有没有类似`\emph{}`和`\textit{}`这种临时改中文字体的方法呢？

在`xecjk.tex`中有如下代码：

```
1 % set new font family
  \setCJKfamilyfont{zhyawei}{msyh}[
3   Path = fonts/,
   Extension = .ttf,
5   BoldFont = {*bd}
  ]
7
  % create font family command alias
9 \NewDocumentCommand{\yahei}{*}{\CJKfamily{zhyawei}}
```

Listing 2.1: New font family

根据需求定义一个新的字体族 `family` 名 `zhyawei`，新字体族的引用方法是：

```
1 {\CJKfamily{zhyawei}{这里输中文}}
  % -or-
3 {\CJKfamily{zhyawei}这里输中文}
```

Listing 2.2: Inline Chinese fonts

为了方便，我们还定义一个新 `NewDocumentCommand` 命令，便于快速引用。`{\yahei{这是新定义的雅黑字体样例。}}` 的效果如：这是新定义的雅黑字体样例。

特别注总，命令要放在一个 `group` 里，即用 `{}` 围起来，否则从当前位置起，所有中文字体都改了。实际，`fontspec` 提供类似功能，具体请参考 `fontspec` 文档。如：

```
\fontspec{<font name>}[<fontfeatures>] 和 \newfontfamily
```





## Chapter 3

# L<sup>A</sup>T<sub>E</sub>X Tutorial

### 3.1 T<sub>E</sub>X Units

---

mm	millimetre $\approx 1/25$ inch	□
cm	centimetre = 10 mm	□
in	inch = 25.4 mm	□
pt	point $\approx 1/72$ inch $\approx \frac{1}{3}$ mm	□
em	approx width of an ‘M’ in the current font	□
ex	approx height of an ‘x’ in the current font	□

---

Figure 3.1: T<sub>E</sub>X Units

### 3.2 Special characters

The following 10 characters has special meanings in L<sup>A</sup>T<sub>E</sub>X.

& % \$ # \_ { } ~ ^ \

Outside *verb*, the first 7 should escaped by *backslash*. Of the last three, we use L<sup>A</sup>T<sub>E</sub>X macros, namely `\textasciitilde`, `\textasciicircum` and `\textbackslash`.

```
1 \& \% \$ \# \_ \{ \} \textasciitilde{} \textasciicircum{} \textbackslash{}
```

Listing 3.1: Special Characters

Specially, L<sup>A</sup>T<sub>E</sub>X uses tilde ~ as a non-breaking space. You would usually use non-breaking spaces for punctuation marks in some languages, for units and currencies, for initials, etc.

### 3.3 verbatim and verb

When to show raw text without  $\text{\LaTeX}$  command being executed, use *verbatim* environment or *verb* command. Both have an asterisk alternative like `\begin{verbatim}`\*} and `\verb*`. With extra \*,  $\text{\LaTeX}$  will typewrite space as `\_`.

You can't use `\verb` in other command's parameters. For example, this is not allowed:

```
1 \href{https://tex.stackexchange.com/a/23653}{When should we use
  \verb|\begin{center}| instead of \verb|\centering|?}
```

Listing 3.2: Illegal verb

Instead, we use `\texttt` and escape special characters <sup>1</sup> like:

```
2 \href{https://tex.stackexchange.com/a/23653}{When should we use
  \texttt{\textbackslash\begin\{center\}} instead of \texttt{\textbackslash\centering}?}
```

Listing 3.3: texttt

### 3.4 item list

Using lists is quite straightforward and does not require you to add any additional packages. For unordered list,  $\text{\LaTeX}$  provides the *itemize* environment and for ordered list there is the *enumerate* environment. With both environments, elements should be declared beginning with `\item` command. We can use M-Enter binding to insert this command automatically.

Take a look at unordered list 3.4:

```
2 \begin{itemize}
  \item One
  \item HKUST
  \item HUST
4 \end{itemize}
```

- One
- HUST
- HKUST

Here is an ordered list 3.4:

---

<sup>1</sup>For instance, backslash and {.

1. Jim
2. Gray
  - This is a
  - nested unordered
  - list within ordered list
3. Who are you?

```

1 \begin{enumerate}
  \item Jim
3 \item Gray
  \begin{itemize}
5 \item This is a
  \item nested unordered
7 \item list within ordered
    list
  \end{itemize}
9 \item Who are you?
\end{enumerate}

```

### 3.5 Code listings

Although *verb* and *verbatim* enclose raw text, they don't highlight source code like C, Java etc. This is where *listings* package come into usage.

To insert a code block, use *lstlisting* environment like

```
\begin{lstlisting}[language=bash,caption={Bash},frame=single]
```

Here is an example:

```

2 #!/bin/bash
  echo 'Hello, world!'

```

Listing 3.4: Bash

Furthermore,

```
\lstinputlisting[language=C,frame=tb,basicstyle=\scriptsize\ttfamily]{
  helloworld.c}
```

Listing 3.5: Include code file

command includes code source file:

```

1 # include<stdio.h>
3 int main(void *) {
  printf("Hello, world!\n");
5 return 0;
}

```

Listing 3.6: C98

This is useful when the code needs updating frequently.

To prevent *lstlisting* from breaking pages, we should wrap it within *minipage* environment as 3.9.

Sometimes, code line is too long to fit page width, leaving tailing part out of page. We could modify *basicstyle* to one of *footnotesize*, *scriptsize* and *tiny*. Check long text line example A. For more options, read Font Size and font size and point (pt) relation.

This is LARGE text while this is Huge.

### 3.6 Straight Quotes

By default, *listings* typesets both single and double as *back curve ones*.

We should import *textcomp* package and *fontenc* package with T1 option. Then, enable *upquote=true* for code block. Read more at How can I get straight double quotes in listings?

This method, however, is discouraged as *fontenc* would override the whole document  $\text{\LaTeX}$  font settings. The underlying cause is *serif(roman)* font. Default roman fonts does not provide straight quotes. We should tell *listings* to use *monospace* font instead by

```
\lstset{basicstyle=\ttfamily}
```

Read more at Consolas: Straight Quotes.

### 3.7 floating

#### 3.7.1 Figure

First, we import *graphicx* package and then set *graphicspath* in preamble part:

```
1 \usepackage{graphicx}
2 \graphicspath{{figs/}{./}}
```

Listing 3.7: *graphicx*

*graphicx* accepts EPS, PDF, PNG or JPEG formats. *graphicspath* defines a list of directories relative to that of *master.tex*. By this means, we can access images anywhere without restriction like `\graphicspath{{../..}/img/}}`. The trailing slash cannot be omitted.

Then, we use command *includegraphics* like

```
\includegraphics[width=.5\textwidth,scale=0.1]{myimg}
```

Notice that we do not write figure file extension. Here is the output:



With bare *includegraphics*, if remaining space within current page does not hold the figure,  $\LaTeX$  would start a new page instead, leaving current page partially empty. The method is to wrap it with *figure* environment:

```
\begin{figure}[tbp]
2  \centering
  \includegraphics[width=\textwidth]{Boobs}
4  \caption{Boobs}
  \label{fig:Boobs}
6  \end{figure}
```

Listing 3.8: Figure Floating

We call this technique as *floating*. Apart from *placement specifier* benefits, we can define *label* and *caption* within *floating* environment. Note that `\label{}` command must come **after** `\caption{}` command since the reference number is generated by `\caption{}`. For details, refer to Why does an environment's label have to appear after the caption?.

The `tbp`<sup>2</sup> specifies position of floating environment.

Spec	Permission to place the float ...
<code>h</code>	<i>here</i> at the very place in the text where it occurred. This is useful mainly for small floats.
<code>t</code>	at the <i>top</i> of a page
<code>b</code>	at the <i>bottom</i> of a page
<code>p</code>	on a special <i>page</i> containing only floats.
<code>!</code>	without considering most of the internal parameters <sup>a</sup> , which could otherwise stop this float from being placed.

Figure 3.2: Floating Placement Specifiers

Here is a nice boobs example 3.3:



Figure 3.3: boobs

Sometimes, it's necessary to put figures side by side for clear comparasion. We use *subfigure* within *figure* like 3.4. Please be noted that we add non-breakable symbol `~` between *subfigure* to leave some space between figures.

### 3.7.2 Table

Similarly, we can wrap *tabular* within *table* environment to *float* table 1.1 like 3.9.

---

<sup>2</sup>*tbp* is the default *placement specifier*. We could set it to *!htbp*.

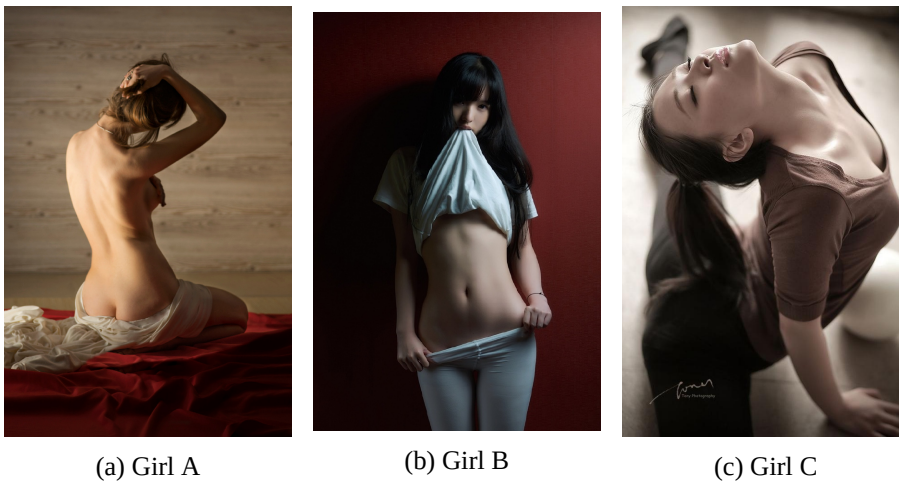


Figure 3.4: Three Girls

```

\begin{table}[!htbp]
2  \centering
  \begin{tabular}{|l|r|}
4    \hline
      Binding & Function \\ \hline \hline
6    \verb|C-c C-e| & insert environment \\ \hline
      \verb|C-c C-s| & insert section \\ \hline
8    \verb|C-c C-m| & macro (command) \\ \hline
      \verb|M-return| & another item \\ \hline
10   \verb|C-c C-c| & compile \LaTeX{} \\ \hline
      \verb|C-c C-v| & preview output \\ \hline
12   \verb|C-c =| & show document layout by RefTeX \\ \hline
      \verb|C-c C-o C-b| & fold the buffer \\ \hline
14   \verb|C-c C-o b| & unfold the buffer \\ \hline
      \end{tabular}
16   \caption{AUCTeX Bindings}
      \label{tab:auctex-bindings}
18 \end{table}

```

Listing 3.9: Table Floating

In *tabular* environment, we align column texts as *left*, *center* and *right*. What about decimal number cell? Ideally, we want to align numbers at dot, where package *siunitx* plays a role by S 3.1 alignment specifier.

```
\usepackage[round-mode=places,round-precision=3]{siunitx}
```

Listing 3.10: Decimal Alignment

To make prettier table, we use package *booktabs* which provides *toprule*, *midrule* and *bottomrule* to override <sup>3</sup> dull *hline*.

<sup>3</sup>It is not replace!

<b>List 1</b>	<b>Decimal 2</b>	<b>Letter 3</b>
$\alpha$	$\beta$	$\gamma$
1	1385.100	a
2	3.067	b
3	87.369	c

Table 3.1: Number Alignment at Dot

<i>Ampere</i>	<i>Voltage</i>	<i>Energy</i>
A	V	J
3	5	45
5.303	4.871	136.955

Table 3.2: Table automation from .csv file.

To make more complex tables, we can use package *multirow* and *multicolumn* environment respectively.

When typesetting large tables, it is tedious and error-prone which could be avoided by *pgfplotstable* package that generate tables from external .csv file. Programs such as Excel, OpenOffice Calc or even emacs org-mode can export data sheets as .csv files. The *pgfplotstable* template A.3 generate table 3.2.

### 3.7.3 Plot

The *pgfplots* package is a powerful tool, based on *tikz* package, dedicated to create scientific graphs. Similar to *pgfplotstable*, *pgfplots* can read data from .csv file.

Since *pgfplots* is based on *tikz*, the *axis* environment should be enclosed within *tikzpicture* environment. Template A.3 generates plot 3.5.

*addplot* plots 2D figure, for 3D, we use *addplot3* 3.6.

Besides *pgfplotstable* and *pgfplots*, we could also use *tgiz* package or *circuitkiz* to draw pictures from within your LaTeX document.

## 3.8 center and centering

The main difference between *center* environment and *centering* command is the former leave vertical space before and after it while the later would not.



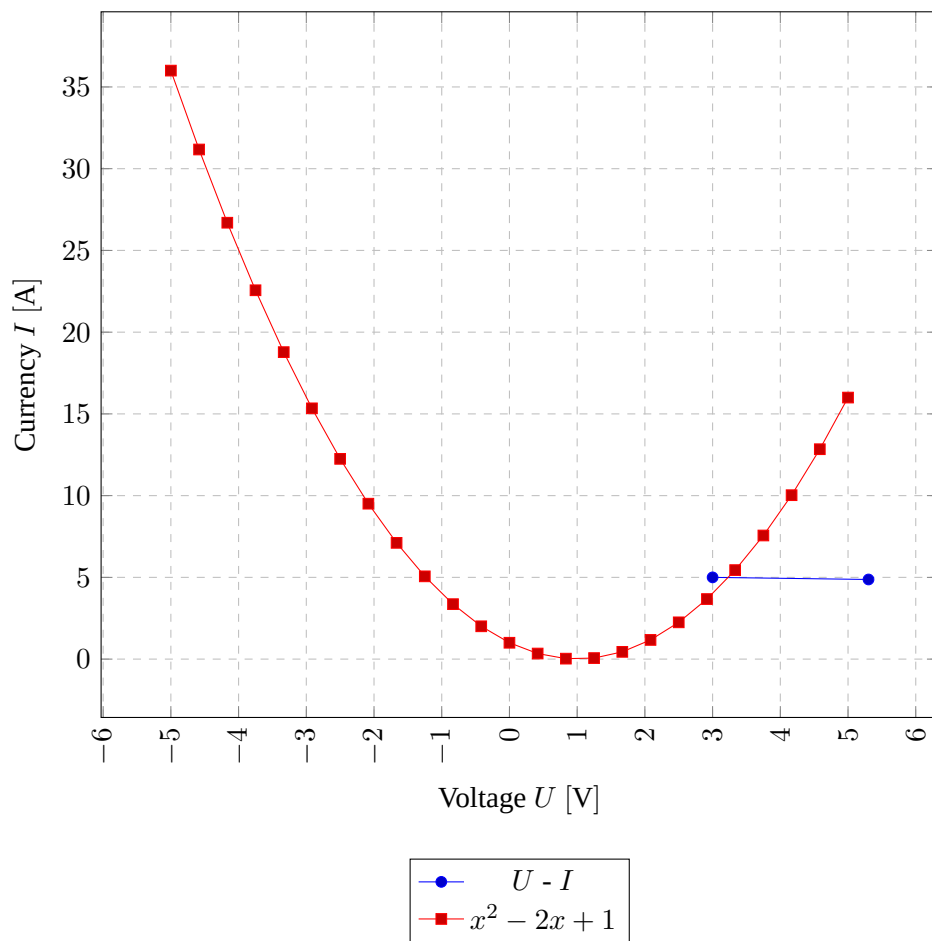


Figure 3.5: pgfplots by table csv file

We usually use `\centering{}` command within a group by curly braces, figure, table, or `\begin{group} \centering ... \end{group}` 3.11.

```

1 {\centering{centering require line break by \par, empty line or \\
3 before closing the group, otherwise text following the group would
   be centered either.}\\
   }

```

Listing 3.11: centering within braces

centering require line break by `\par`, empty line or `\\` before closing the group, otherwise text following the group would be centered either.

This line is not centered! Check When should we use `\begin{center}` instead of `\centering`? and center vs. centering

Exmple using the mesh parameter

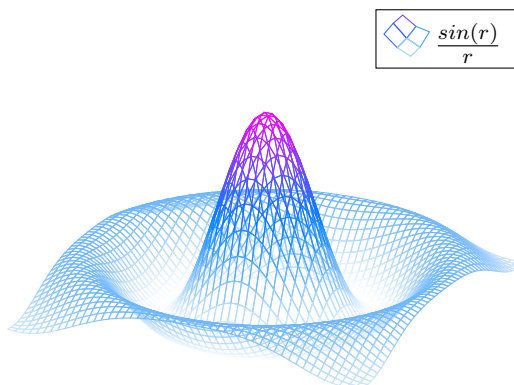


Figure 3.6: pgfplots 3D

### 3.9 Math Equations

To insert simple inline equation, we wrap it by  $\$$  like  $\$c^2=a^2+b^2\$$ :  $c^2 = a^2 + b^2$ .

For tall or deep inline math expressions or sub expressions, we can enclose them by  $\backslashsmash$  command. This makes  $\text{\LaTeX}$  ignore the height of these expressions. This keeps the line spacing even like:  $d_{e_p}$  followed by  $h^{i^g_h}$ .

This is an example without  $\backslashsmash$ . You will find line space is much bigger. A  $d_{e_p}$  expression followed by a  $h^{i^g_h}$  one.

If we want equaton occupy a whole line, then wrap it by double  $\$$  or single bracket  $\backslash[ \backslash]$ . The equation will be centered automatically.

```
$(1+x)^n=\sum_{k=0}^n\binom{n}{k}x^k$
```

Listing 3.12: Equation in new line

This is the output:

$$(1+x)^n = \sum_{k=0}^n \binom{n}{k} x^k$$

Let's have another example:

$$A_1 + A_{100}$$

After examing the example above, you are recommended to tune AUCTeX 1.2 configuration a little bit. Enable math mode manually by  $\text{\C-c ~}$  or globally into Emacs startup:

```
1 (add-hook 'LaTeX-mode-hook 'LaTeX-math-mode)
```

Listing 3.13: L<sup>A</sup>T<sub>E</sub>X Math Mode

We just prepend `\` and type your desired math symbol. AUCTeX automatically completes it. If given a prefix argument C-u, the symbol will be surrounded by dollar signs. For example, if you type C-u `\beta`, AUCTeX will typeset  $\beta$ . Read AUCTeX manual Entering Mathematics for more details.

The `$` method by T<sub>E</sub>X does provide some basic mathematics features, but it is limited. we should `\usepackage{amsmath}`. What is more, we can `\usepackage{amssymb}` to make math symbols look shiny.

A bit further, we usually want to label and number a equation so that we can refer to it somewhere else. Furthermore, it is better for a long equation to occupy a new line. To do this, enclose equation with *equation* environment like (3.1). Einstein says

$$E = mc^2 \tag{3.1}$$

In order that multiple equations align properly at *ampersand* `&`, we use *align* or *align\** instead. (3.9). Each single equation must be separated by *linebreak*

. The version with asterisk just removes equation numbers.

$$E = mc^2 \tag{3.2}$$

$$F = ma \tag{3.3}$$

We use *aligned* environment to align induction or substitution lines of a single equation. Also, *aligned* should be enclosed within an equation environment.

$$\begin{aligned} f(x) &= x^2 \\ \frac{1}{x} &= g(x) \\ F(x) &= \int_b^a \frac{1}{\sqrt[3]{x}} \end{aligned}$$

Compared to *aligned*, *align* and *align\** introduce extra space between lines to make the output cleaner. Therefore, irrespective of single equation or multiple equations, we'd better use *align* and/or *align\**.

*matrix* environment must be enclosed by equation marker `&` or *equation* environment.

```

1 $
  \begin{matrix}
3   1 & 0 \\
   0 & 1
5 \end{matrix}
$

```

Listing 3.14: Matrix

Similar, `matrix` breaks line by `\\` and `&` separates columns. Here is a *bmatrix* example:

$$A_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \quad (3.4)$$

Recall that `$$` or `\[ \]` let an equation placed on a new line without label. They are equivalent to the `\begin{equation*}` environment.

### 3.10 Document Structure

### 3.11 Boxes

Everything in  $\text{\LaTeX}$  is embedded in boxes, from letter to words, *tabular* to *includegraphics*.  $\text{\TeX}$  builds pages by gluing<sup>4</sup> boxes together according to the default  $\text{\TeX}$  rules, default  $\text{\LaTeX}$  rules, or document commands.

Box is of paramount importance to get the base of  $\text{\LaTeX}$  typesetting. Boxes are placed relative to other boxes, while visible elements are placed relative to the boxes which contain them.

Let's have a look at letter box illustration 3.7.



Figure 3.7: Letter Box

<sup>4</sup>Squeeze and/or Stretch

`\parbox` is a box to wrap text into lines and lines are broken into pages.

```
\parbox[pos][height][contentpos]{width}{text}
```

Listing 3.15: `parbox` box

*width* is a forced parameter to define box width<sup>5</sup> This argument can be relative value like `.8\textwidth`, absolute value like `5ex`, or  $\LaTeX$  and  $\TeX$  macros like `\width`. Similarly, we set *height* in the same way.

*pos* selects which *baseline* of text to align with neighbouring box. It can be `center`, `top`, and `bottom`. For details, read the link above. Check 3.8 for real effect.

This is a text line before *parbox* This a text within *parbox*. Please This a tex line  
check text box and text align-  
ment carefully.

after *parbox*.

*contentpos* positions the contents of the box within the box which is straightforward. It only takes effect when box is larger than texts it encases.

However, if the *contentpos* is present and not the same as *pos* and *pos* is not `center`, the `\parbox` will align at its borders instead of text baseline 3.9.

This is a text line before *parbox* This a tex line

This a text within *parbox*. Please  
check text box and text align-  
ment carefully.

after *parbox*.

We also have *minipage* environment box which is almost identical to `\parbox`. The difference between a *minipage* and a `\parbox` is that you cannot use all commands and environments inside a `\parbox`, while almost anything is possible in a *minipage*. Hence, without special requirement, we use the later one.

```
1 \begin{minipage}[pos][height][contentpos]{width} text \end{minipage}
```

Listing 3.16: *minipage* box

Additionally, there are *mbox*, *makebox* and *framebox*.

In order to let multiple boxes stand side by side, we should make sure their *width* in total is less than 1, like 3.4.

---

<sup>5</sup>It is **not** text width.

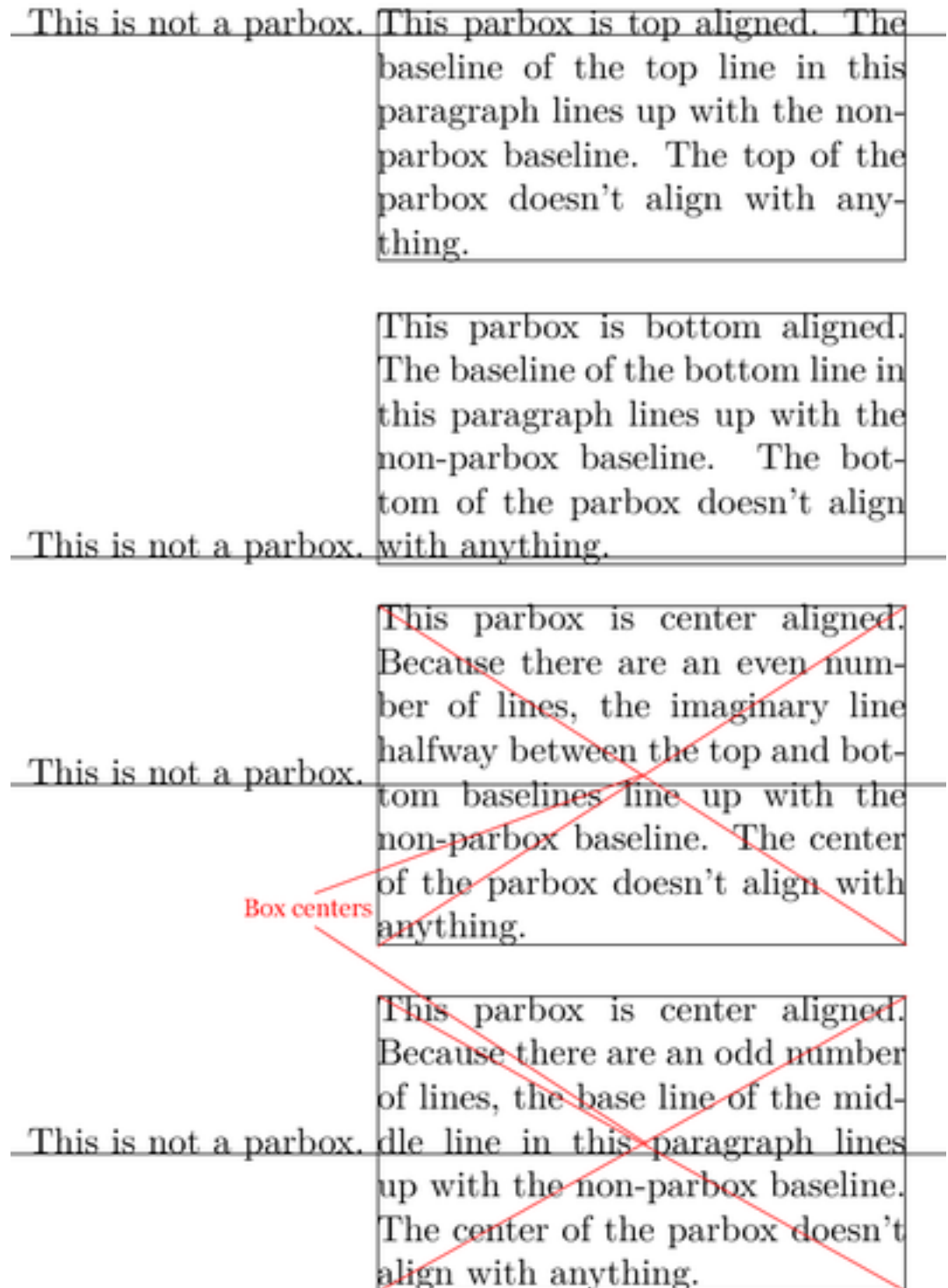


Figure 3.8: parbox baseline alignment

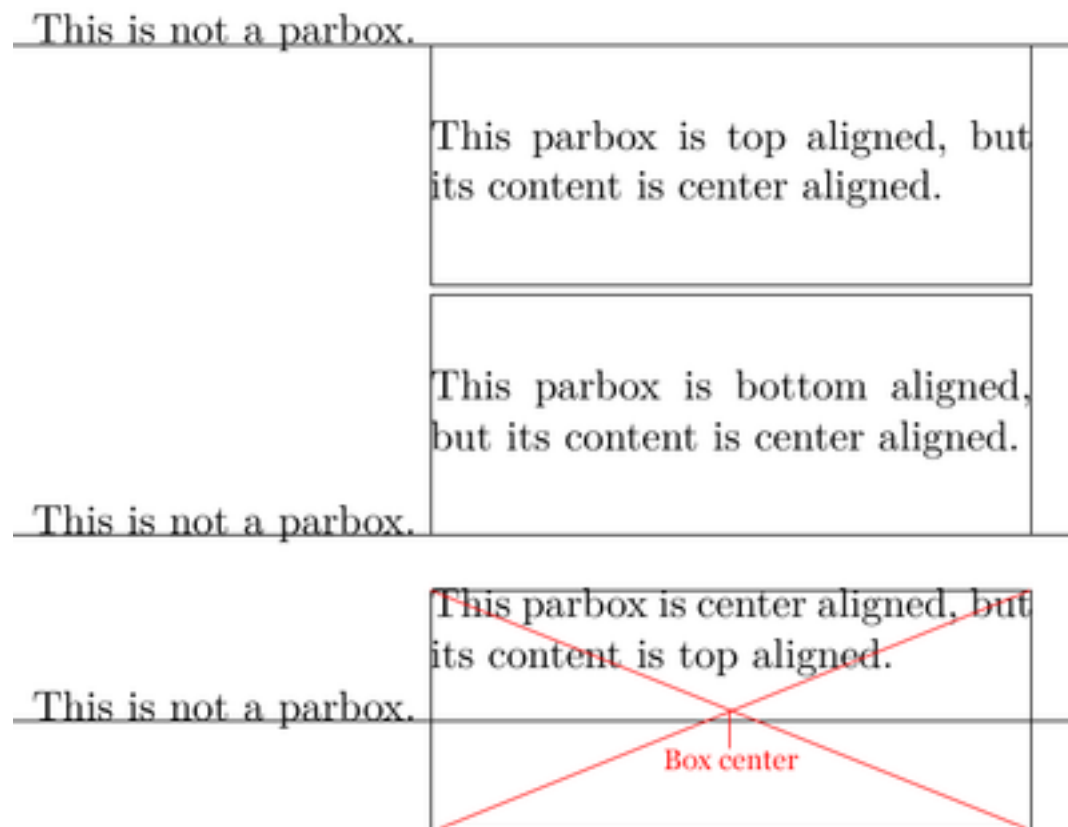


Figure 3.9: parbox border alignment

### 3.11.1 input and include

Like other programming language,  $\text{\LaTeX}$  supports splitting large document into different parts and importing them individually.

Use `\include{filename}` in the *document body* to insert the contents of another file named *filename.tex*. Note that  $\text{\LaTeX}$  will start a new page before processing the material input from *filename.tex*. *include* is commonly used for *chapter* section.

*include*'s sibling command `\includeonly{filename1,filename2,...}` is used in *preamble* part to insert only a subset of *included* files.

Alternatively, `\input{filename}` is allowed in both *preamble* and *body* parts. It does not start a new page like *include* but allow recursive *include* which is impossible for *include*.

No matter which you choose, please omit  $\text{\LaTeX}$  file extension *.tex*.

### 3.11.2 biblatex biber

Generally speaking, *biblatex* and *natbib* are packages that handle  $\text{\LaTeX}$  citations in *.tex* source file. However, reference entries are stored separately in external *.bib* file. Hence, we require an intermediate tool to format *.bib* into *.tex* source. That's when we meet *biber* and *bibtex* which are named as *biblatex backend*.

Firstly, we should use *biblatex* package: `\usepackage[backend=biber]{biblatex}` and point out the external *.bib* file to import: `\addbibresource{myref.bib}` in *preamble* part. Do not omit *.bib* extension.

Secondly, in the end of  $\text{\LaTeX}$  source (but before `\end{document}`), we print all reference entries: `\printbibliography[heading=bibintotoc]`.

Then, cite as you go. Before that, we should have a look at *.tex* format. The bibliography files must have the standard *bibtex* syntax. Here is a reference entry:

```

1 @article{einstein,
   author = "Albert Einstein",
3   title = "{Zur Elektrodynamik bewegter K{\\"o}rper}. ({German})
   [{0n} the electrodynamics of moving bodies]",
5   journal = "Annalen der Physik",
   volume = "322",
7   number = "10",
   pages = "891--921",
9   year = "1905",
   DOI = "http://dx.doi.org/10.1002/andp.19053221004",
11  keywords = "physics"
}
```

Listing 3.17: BibTeX entry sample



`@article` tells the reference is an article. We also have `@book`, `online` etc. *einstein* is the reference label that we will *cite* with: `\cite{einstein}`.

### 3.11.3 RefTeX

RefTeX has been bundled and pre-installed with Emacs since version 20.2. We just add to Emacs:

```

2 ; with AUCTeX LaTeX mode
  (add-hook 'LaTeX-mode-hook 'turn-on-reftex)
4 ; with Emacs latex mode
  (add-hook 'latex-mode-hook 'turn-on-reftex)

```

Listing 3.18: Enable RefTeX

To make cross-reference clickable (i.e. table of contents), use package *hyper-ref*.

Although `C-c C-m` can prompt for `\cite` command, we'd better use RefTeX instead. RefTeX wraps itself round four LaTeX macros: `\label`, `\ref`, `\cite`, and `\index`, making the process more intelligent.

As mentioned earlier, RefTeX binding `C-c =` displays document layout in a newly created buffer.

To *cite*, we use `C-c [` binding and press <sup>6</sup>`^M`. Afterwards, type a *regular expression* to search reference entries in *.bib* file. Please be noted, there is no completion prompt for *regular expression*.

For reference to objects (i.e. figures) within the document, we use `C-c )`. For details, please visit RefTeX in a Nutshell. RefTeX seems not to support *href* to URLs.

Sometimes, RefTeX cannot find newly created labels, references etc. We should tell it to *reftex-parse-all* or *reftex-parse-one* to parse  $\LaTeX$  source files.

---

<sup>6</sup>It is Enter key or Ctrl and letter 'm', not Emacs Meta Alt key.



**Part II**

**Mathematics**



## Chapter 4

# Combinatorics

Combinatorics, namely *Combinatorial Mathematics*, mainly studies *permutation* and *combination* of a *set* or *multiset*.

组合数学里经常要用到**映射**的概念，在后文的描述里，会多次出现**对应**这样的字眼，表示两个集合间元素的映射。通常为了方便分析，把元素、对象的不同用**编号**、**颜色**等标签来表示。

### 4.1 Set and Multiset

组合数学问题总要对某个集合的元素进行操作，虽然我们通常不需要特别说明。如 10 个人进行全排列，那么对应的集合由这 10 个人组成。

普通集合如：

$$S = \{ a_1, a_2, a_3, \dots, a_n \}, i \in \mathbb{Z} : i \in [1, n]$$

表示有  $n$  种**不同**元素，并且每种元素只有一个，称  $S$  为**单重集（合）**。

更一般的，**多重集（合）**：

$$M = \{ n_1 \cdot a_1, n_2 \cdot a_2, \dots, n_k \cdot a_k \}$$

突出元素**种类**，即  $a_i$  表示第  $i$  种元素，而  $n_i$  则表示第  $i$  种元素的个数。通常元素总个数用  $n$  表示：

$$n = n_1 + n_2 + \dots + n_k$$

不难发现，单重集是多重集的特例，此时

$$k = n, n_i = 1, i \in \mathbb{Z} : i \in [1, k]$$

如果多重集里每种元素有无限个，则记为：

$$M = \{ \infty \cdot a_1, \infty \cdot a_2, \dots, \infty \cdot a_k \}$$

普通集合和普通排列组合对应，多重集合和多重排列组合对应，这也是在正式介绍排列组合前先说下集合的定义。

排列组合里，集合元素通常都具体化为**不同**颜色小球，组合表示从集合里取小球出来，而排列表示进一步把取出的小球排队，或放进**不同**的盒子里，此时盒子的编号代表队列从头至尾的位置号。

注意符号表示的不同。单重集里  $n$  既表示元素种类和个数。而多重集用  $k$  表示元素种数，用  $n$  和  $n_i$  表示个数。注意符号表示的习惯：

$$a_i, n_i, k, n, S, M$$

## 4.2 Counting Principles

加法原理：设集合  $S$  可以划分成若干不相交的子集  $S_1, S_2, \dots, S_m$ ，则：

$$|S| = |S_1| + |S_2| + \dots + |S_m|$$

乘法原理：设集合  $S$  是序偶  $(a, b)$  的集合，其中  $a$  来自于集合  $A$ ,  $b$  来自于集合  $B$ .  $A$  中每个元素  $a$  都要与  $B$  中每个元素  $b$  配对。则：

$$|S| = |A| \times |B|$$

减法原理：设集合  $A$  是全集  $U$  的子集，补集<sup>1</sup>  $\bar{A} = U \setminus A = \{x \mid x \in U, x \notin A\}$ ，则：

$$|A| = |U| - |\bar{A}|$$

除法原理：设  $S$  是有限集，被划分为  $k$  个两两不相交的部分，每部分皆有  $m$  个元素，则：

$$k = \frac{|S|}{m}$$

后面会发现，很多应用都以单重集的排列、组合为原子操作，再结合计数四原则完成。

---

<sup>1</sup>用 `overline`, `bar`, `complement`, 或 `smallsetminus` 命令表示。后者的好处是同时指明了全集和子集。

### 4.3 Permutation of Sets

定义：从  $n$  个元素的单重集合  $S$  中，取出  $r$  个元素按次序排成一列，称为  $S$  的一个  $r$ -排列，记为：

$$P(n, r), P_n^r, A_n^r$$

注意，定义里用  $r$  表示所取元素个数。

当  $r = n$  时， $S$  的  $n$ -排列简称为  $S$  的排列或  $n$  个元素的全排列。

从  $n$  个元素中取  $r$  个元素的排列的典型例子是从  $n$  个**不同**颜色的球中，取出  $r$  个，放入  $r$  个**不同**的盒子里，每盒 1 个，盒子编号就映射成队列位置。显然第 1 个盒子有  $n$  种球可选，第 2 个盒子有  $n-1$  种球可选，……，第  $i$  个盒子有  $n-(i-1)$  种球可选，……，第  $r$  个盒子有  $n-(r-1)$  个球可选。

可以得出

$$A_n^r = n(n-1) \cdots [n-(i-1)] \cdots [n-(r-1)], \quad r \leq n, \quad n, r \in \mathbb{Z}^+$$

定义阶乘 factorial：

$$\begin{aligned} n! &= n \times (n-1) \times (n-2) \cdots 2 \times 1 \\ 0! &= 1 \end{aligned}$$

排列组合是不会有 0 的情况的，但为了数学定义和计算的完备性，要考虑 0 等情况。其实负数的阶乘也有定义，不过在组合数学里没意义，在此不考虑。后面还会遇到类似形式的定义。那么：

$$\begin{aligned} A_n^r &= \frac{n!}{(n-r)!}, \quad 0 \leq r \leq n \\ A_n^0 &= 1 \\ A_n^n &= n! \end{aligned}$$

排列还分为**直线排列**和**圆排列**。直线排列就是我们常说的排列，圆排列是指把取出的元素排成一个圆形，

上面定义的是**直排列**，**圆排列**定义为取出的元素排成一个圆圈，等于是让直线排列的首尾相连接。从  $n$  个元素中取出  $r$  个构成的圆  $r$ -排列数为：

$$A_n^r / r, \quad (1 \leq r \leq n)$$

因为 1 个圆排列从任一位置断开都是一个不同的直排列，也就是说  $r$  个直排列对应 1 个圆排列，所以要在直排列基础上除以  $r$ 。注意，是除不是乘。特殊的， $n$  个元素的全圆排列数为  $(n-1)!$ 。

圆排列还有一种情况是，可以翻转圆排列，如  $n$  个不同颜色的珠子串成一条项链。一条项链的一面是一个普通圆排列，但翻转这个项链，它的另一面是一个新圆排列，所以每个可翻转圆排列对应 2 个普通圆排列。因此，**可翻转圆排列**数为

$$\frac{A_n^r}{2 \cdot n}$$

## 4.4 Combination of Sets

上节讨论了单重集的排列，这节讲单重集的组合。排列和组合的主要区别是**不同**元素是否有**次序**。组合取出元素**堆**在一起，而排列在组合的基础上对取出的元素堆进行排队或入盒。

一旦取出，便不再区分组合堆内球的不同。如果取出多个堆，堆间区别是球的个数：堆内无序，堆间也无序。后面多次用到这个原则。

从  $S$  中取  $r$  个元素而不进行排序，称为  $S$  的一个  $r$ -组合，实际就是生成一个  $S$  的  $r$  元素子集，可以看成是取出元素堆在一起，没有顺序：组合即组堆，也即生成子集。

组合的对应彩色球问题是从  $n$  个色彩**不同**的小球中取出  $r$  个（堆在一起），此时没有放入盒子的操作：只取不入。如果一定要有入盒操作，那么所有的盒子相同，没有编号区分，此时入盒与否没有意义。 $r$ -组合数记为

$$C(n, r), C_n^r, \binom{n}{r}$$

很显然，一个  $r$ -组合可以生成  $r!$  个排列，所以：

$$\begin{aligned} C_n^r &= \frac{A_n^r}{r!} = \frac{n!}{(n-r)!r!}, \quad r, n \in \mathbb{Z}_0^+ : r \in [0, n] \\ C_n^r &= 0, \quad r > n \\ C_n^0 &= 1 \\ C_n^n &= 1 \\ C_0^0 &= 1 \end{aligned}$$

从上面方程中的特例可以看出，完备性定义对数学计算的意义。阶乘，排列数，组合数的完备性定义主要考虑  $r > n$  和  $n, r = 0$  的情况。一般地，我们只考虑  $0 \leq r \leq n$  的情况。对于  $r, n < 0$  的情况不在排列组合的讨论范围。在其它计算领域即便碰到，也不难，只要严格按照阶乘的定义计算即可。

由公式：

$$A_n^r = C_n^r r!, \quad r, n \in \mathbb{Z}_0^+ : r \in [0, n]$$



可以得出，除原始定义外，排列操作可看作先取组合，得到一堆元素，再对此堆列队。可以说**排列操作暗含了组合子问题**。简而言之，1 个组合对应  $r!$  个排列，排列数是组合数的  $r!$  倍。后面更复杂的分配分组问题也遵循此规律。

#### 4.4.1 Combination Formulas

组合数的计算非常重要，因为组合公式是一个很重要的数学工具，如多项式的系数和组合数紧密相联。本小节着重讲组合数的几个公式。

组合数和排列数计算都化成阶乘的计算。不过，当参数很大时，计算阶乘不是件容易的事。但从组合数的原始定义可知：取  $r$  个元素得到一个子集，余下的就是  $n - r$  元素的补集，一一对应。也就是说，每取一个  $(n - r)$ -组合就得到一个  $r$ -组合：

$$C_n^r = C_n^{n-r}, \quad r, n \in \mathbb{Z}_0^+ : r \in [0, n]$$

当  $r$  很大时， $n - r$  就很小，便于计算。我们还可以想办法降级阶数，让其变得更小：

$$C_n^r = C_{n-1}^r + C_{n-1}^{r-1}$$

由此公式可得出 杨辉三角形，也称 *Pascal* 三角形。

有趣的是让  $r$  遍历 0 到  $n$ ，可以看出所有的组合数之和就是集合  $S$  的幂集的元素个数：

$$\sum_{r=0}^n \binom{n}{r} = 2^n$$

此公式可以用二项式证明：

$$(x + y)^n = \sum_{r=0}^n C_n^r x^r y^{n-r}$$

令  $x, y = 1$  即可。

更一般地，组合公式的证明**双重计数 double counting** 法：从不同（一般 2 种）角度对集合计数。更多关于双重计数，看 Double Counting 和 Mod-03 Lec-17 Double counting - Part (2)。

如上面的组合数求和公式，左边表示利用加法原则，以子集元素个数  $r$  来对  $S$  的幂集计数。要证明，我们以另外一个角度来计数：利用乘法原则，针对一个元素是否加入子集来计数。 $\forall a_i, 1 \leq i \leq n$  要么在某个子集里，要么不在，只有两种可能。对  $a_i$  遍历， $a_1$  有两种可能， $a_2$  有两种可能，……， $a_n$  有两种可能，所以  $S$  有  $2 \times 2 \times \cdots \times 2 = 2^n$  个子集。

### 4.4.2 Application of Combination

现有  $n$  个管理员管理某保密装置 (Minimum number of locks and keys)。要求任何  $\leq r$  个管理员都打不开该装置，至少需  $r+1$  个。假设该装置有  $s$  把钥匙，给每个管理员分配其中的  $t$  把。问题是已知  $n, r, r \leq n$ , 求  $s, t, t \leq s$ . 并进一步给出该装置的钥匙分配方案。

设管理员集合为：

$$M = \{m_1, m_2, \dots, m_n\}, r \leq n$$

钥匙集合为：

$$K = \{k_1, k_2, \dots, k_s\}, t \leq s$$

先求钥匙数  $s$ . 由描述知，管理员集合  $M$  的任一  $r$ -组合 (管理员子集) 都不能凑齐  $s$  把钥匙，可知任一  $r$ -组合至少还缺 1 把钥匙。显然，任两个  $r$ -组合的钥匙不能完全相同，也即不能缺同一把钥匙，否则  $2r$  个管理员还因缺一个把钥匙而打不开保密装置，而且这样的分配方案没有意义。 $M$  的  $r$ -组合数是  $C_n^r$ . 所以：

$$s \geq C_n^r$$

从上分析可看出， $M$  的所有  $r$  元素子集到  $K$  的一个单射：每个  $r$  元素子集的像是它所缺失的那把钥匙。但它不一定是满射，因为  $s \geq C_n^r$  时，额外的钥匙会被所有的  $r$ -组合所覆盖，这样的钥匙没的原像。其中等号成立的条件是每个  $r$ -组合刚好缺一把钥匙，此时构成一个满射。

再求每个管理员分得的钥匙数  $t$ .  $M$  的任一  $(r+1)$  元素子集都能打开装置，说明其中任一管理员可补齐剩下  $r$  个管理员所缺的那把钥匙。一个管理员  $m_i$  所参与的  $(r+1)$ -组合有  $C_{n-1}^r$  个，所以：

$$t \geq C_{n-1}^r$$

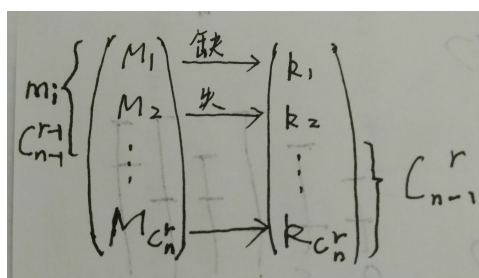
在分析钥匙的分发方案之前，让我们回顾下上面的杨辉三角形，本题已出现公式里两个分项  $C_n^r$  和  $C_{n-1}^r$ ，只不过后者的意义稍变了。

$s$  取最小值  $C_n^r$  时，我们列出  $M$  的所有  $r$  元素子集到  $K$  的满射：

$$f(M_i) = k_j$$

$$\{M_i \mid M_i \text{ is a set of size } r, i = 1, \dots, C_n^r\} \xrightarrow{\text{缺失}} K$$

显然这样的映射有很多种，我们只需选其一，如下图所示：



对于某个管理员  $m_i$ , 他所在  $r$  元素子集有  $C_{n-1}^{r-1}$  个, 到此, 杨辉三角形公式里三个分项全部出现。很显然这管理员也缺失了这些  $r$  元素子集对应的钥匙像, 所以该管理员应分得剩下的  $C_{n-1}^r$  个钥匙像。针对所有管理员进行类似分配即可。

前面提过, 当  $s, t$  的不等式不取等号时, 此映射不是满射, 有钥匙没有原像, 此时要求所有的  $r$ -组合都包含这样的钥匙。

现假设  $s = C_n^r + 1$ , 此时的分配方案只需在原基础上稍加修改即可: 多出的这枚新钥匙再分配给每个管理员。这样  $t = C_{n-1}^r + 1$ . 此设计仍满足保密要求。我们会发现这样的额外钥匙是没有必要的。每个管理员都多了 1 枚同样的钥匙, 起不到额外的安全作用。

分配方案的关键是根据  $s$  值列出  $r$  元素子集到钥匙的映射。现给出一个实际的例子。如果  $n = 5, r = 3$ , 则  $s = C_5^3 = 10, t = C_4^3 = 4$ . 下面给出一个映射。多出的一个钥匙 (编号 11) 是为了说明映射不一定是满射。

123	缺	→ 1
124		→ 2
125		→ 3
134		→ 4
135		→ 5
145		→ 6
234		→ 7
235		→ 8
245		→ 9
345		→ 10
		11

每个管理员参与了  $C_{5-1}^{3-1} = 6$  个 3 元素子集, 对应到映射图上, 有 6 行。不失一般性, 管理员  $m_1$ , 参与了前 6 行, 缺失前 6 枚钥匙, 所以分得后 4 枚钥匙 7, 8, 9, 10. 依此类推, 我们可以得出如下分配方案。注意, 该方案考虑到了第 11 枚钥匙。

Managers \ Keys	Keys										
	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	0	1	1	1	1	1
2	0	0	0	1	1	1	0	0	0	1	1
3	0	1	1	0	0	1	0	0	1	0	1
4	1	0	1	0	1	0	0	1	0	0	1
5	1	1	0	1	0	0	1	0	0	0	1

Table 4.1: Safe Device

## 4.5 Permutation of Multiset

排列组合还会考虑所取元素元素是否**重复**，从而生成（多）重排列和（多）重组合。

前面两节介绍了单重集的排列组合，对应地，多重集也有排列组合，定义和单重集一样，唯一的区别是取出的元素可能有重复。

设多重集  $M$  为：

$$M = \{ n_1 \cdot a_1, n_2 \cdot a_2, \dots, n_k \cdot a_k \}, \quad \forall i = 1, 2, \dots, k, r \leq n_i$$

或：

$$M = \{ \infty \cdot a_1, \infty \cdot a_2, \dots, \infty \cdot a_k \}$$

由于每种元素的个数超过所取数，所以排列里  $r$  个元素可能全部来自某一种。

多重集  $M$  的  $r$ -排列数是：

$$k^r$$

这是很容易理解的，因为队中每个位置都有  $k$  种可能。

多重集  $r$ -排列对应的彩球模型可以看成**可放回**（可重复）地对单重集取球、排队或入盒。单重集里每个元素可无限重复使用，标记好队列位置后又放回球堆里。当然按照多重集的定义，可以看成是对多重集取球、排队或入盒。如不多于四位的三进制数的个数为  $3^4$ 。对应的多重集是  $M = \{ \infty \cdot 0, \infty \cdot 1, \infty \cdot 2 \}$ 。

如果  $r = n = \sum_{i=1}^k n_i$ ，则称为  $M$  的全排列。多重集全排列（也即  $n$ -排列）是对**有限集**  $M$  的**所有**元素（如彩球）进行操作。具体来说，就是对全部  $n$  个  $k$  种颜色的球进行排队。

显然此时  $r$  大于所有的  $n_i$ 。  $M$  的全排列数为：

$$\frac{n!}{n_1! n_2! \cdots n_k!}$$

全排列的证明可以先从直觉来分析。 $n!$  表示所有元素的全排列, 但是  $a_i$  在队列重复了  $n_i$  次, 这此重复实际只表示一种队列, 所以除以  $n_i!$ . 如果某个  $n_i$  等于 1, 那么在被除式中是  $1!$ .

有趣的是, 实际证明中, 用的是组合思路。第 1 种元素  $a_1$  要占据队列里的  $n_1$  个位置, 所以有  $C_n^{n_1}$  种可能, 第二种元素要占据剩下  $n - n_1$  个位置中的  $n_2$  个, 所以有  $C_{n-n_1}^{n_2}$  种可能, ……依此类推, 最后一种元素有  $C_{n_k}^{n_k} = 1$  种可能:

$$C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}$$

从证明过程可知, 当  $k = 2$  时, 多得集的全排列数在数值上等于单重集的组合数  $C_n^{n_1} = C_n^{n_2} = n!/(n_1! n_2!)$ . 如用两面红旗, 三面黄旗依次悬挂在一根旗杆上, 问可以组成多少种不同的标志? 答案是  $5!/(2! 3!) = 10$ .

下面看一个棋盘的例子。在一个  $k \times k$  的棋盘上放  $k$  个车, 使得任意两个车之间不能互吃, 有多少种方法? 棋盘上车全是相同的, 没有区别。要想车不互吃, 任意两个车的行列值都不同。每行一个车  $a_{1j_1}, a_{2j_2} \cdots a_{kj_k}$ , 只需给不同行的车选列即可, 所以方法数是  $k!$ .

假设是  $k$  个不同 (色) 的车呢? 保持上面排列不变, 现对车的颜色进行调换, 有  $k!$  种, 所以方法数是  $k! k!$ . 实际是对车所在的行进行全排列, 也即行和列都要全排列。行的全排列负责车的不同, 而列的全排列负责车不互吃。

假设是  $n_1$  个红车,  $n_2$  个蓝车, …… $n_k$  个黄车, 总共  $n$  个呢? 先解决车在行上的全排列:  $\frac{n!}{n_1! n_2! \cdots n_k!}$ . 再针对不同的列全排列  $n!$ . 方法数是:

$$\frac{n!}{n_1! n_2! \cdots n_k!} \cdot n!$$

多重集  $r$ -排列要求每种元素个数不少于  $r$ , 若  $\exists t, n_t < r$ , 那么情问变复杂了。这时没有公式计算, 要具体针对  $n_t$  列举分析。如第  $t$  种元素出  $1, 2, \dots, t$  个。

例: 9 个元素的多重集  $S = 3 \cdot a, 2 \cdot b, 4 \cdot c$  的 8-排列数为多少? 此例中,  $r = 8$  大于  $n_i$ , 所以不能套用公式。  $n = 9$  只比  $r$  大 1, 所以列队中,  $a, b, c$  之一少出一个元素。如少一个  $a$  排列数是  $\frac{8!}{2! 2! 4!}$ , 同理可算另两项, 再用加法原则:

$$\frac{8!}{2! 2! 4!} + \frac{8!}{3! 1! 4!} + \frac{8!}{3! 2! 3!}$$

## 4.6 Combination of Multiset

多重集  $S$  的含有  $r$  个元素的子多重集就叫做  $S$  的  $r$ -组合, 或表述为取出  $r$  个元素的堆。和多重集的排列比少了列队操作。

关于多重集的排列问题可以小结如下:

设  $S = \{n_1 \cdot a_1, n_2 \cdot a_2, \dots, n_k \cdot a_k\}$ , 且  $n = n_1 + n_2 + \dots + n_k$ , 则  $S$  的  $r$ -排列数  $N$  满足:

- (1) 若  $r > n$ , 则  $N = 0$ ;
- (2) 若  $r = n$ , 则  $N = \frac{n!}{n_1! n_2! \dots n_k!}$ ;
- (3) 若  $r < n$ , 且对一切  $i = 1, 2, \dots, k$  有  $n_i \geq r$ , 则  $S$  的  $r$ -排列数是  $k^r$ ;
- (4) 若  $r < n$ , 且存在某个  $n_i < r$ , 则对  $N$  没有一般的求解公式, 解法将在后面讨论。

Figure 4.1: Summary on Multiset Permutation

同多重集的组样, 假设每种元素个数都不小于  $r$ . 方法数即  $k$  元线性不定方程:

$$r = x_1 + x_2 + \dots + x_k, \quad x_i \in \mathbb{Z}_0^+, i = 1, 2, \dots, k$$

的非负整数的解数。非负是表明可能某  $x_t$  为 0, 此时元素  $a_t$  没有出现在组合堆里。

计算用 隔板法或 插空法。

可以看成这样, 有  $r$  个 1 在一行上, 占有  $r$  位置, 包含首尾共有  $r + 1$  个空位, 插上  $k - 1$  个板子。第  $i$  板子前面的 1 个数是  $x_i$  的解, 第  $k - 1$  个板子后面的 1 个数是  $x_k$  的解。如果某个板  $k_t$  前没有 1 而是另一个板 (两个板子处在同一空位), 则  $x_t$  解为 0. 如从

$$| 1 | | 1 1 \dots 1 1 |$$

看出:

$$x_1 = 0, x_2 = 1, x_3 = 0, x_k = 0$$

非负正整数解的思路是:  $r$  个 1 的位置和  $k - 1$  的板位置共  $r + (k - 1)$  个位置, 从中取  $r$  个作为 1 的位置或取  $k - 1$  个作为板的位置:

$$C_{r+k-1}^r \quad \text{or} \quad C_{r+k-1}^{k-1}$$

不考虑多重集组合, 单就方程本身来说, 如果是求正整数解呢?

$$r = x_1 + x_2 + \dots + x_k, \quad x_i \in \mathbb{Z}^+, i = 1, 2, \dots, k$$

正整数解暗含了  $r \geq k$ , 非负解则无此限制。  $r$  个 1 形中间有  $r - 1$  个空档, 插入  $k - 1$  个板, 且每个空档最多一个板, 则解数:

$$C_{r-1}^{k-1}$$

这比非负情况的简单。

实际非负整数解和正整数解之间可以互相转换，形成一个满射，而解个数不变。下面是把非负解转成正整数解的方法，即变量加 1:

$$r + k = (x_1 + 1) + (x_2 + 1) + \cdots + (x_k + 1), \quad x_i \in \mathbb{Z}_0^+, i = 1, 2, \dots, k$$

新方程的解和原方程的解一一对应（满射：减一），这种变换思路很重要。新方程的解是正整数解：

$$C_{r+k-1}^{k-1}$$

把正整数解转成非负解，是把变量减 1. 利用转换原理，我们把方程：

$$x_1 + x_2 + x_3 + x_4 = 20, \quad x_1 \geq 3, x_2 \geq 1, x_3 \geq 0, x_4 \geq 5$$

转成非负解形式：

$$(x_1 - 3) + (x_2 - 1) + (x_3 - 0) + (x_4 - 5) = 11, \quad x_1 \geq 3, x_2 \geq 1, x_3 \geq 0, x_4 \geq 5$$

或正整数解形式：

$$(x_1 - 2) + (x_2 - 0) + [x_3 - (-1)] + (x_4 - 4) = 15, \quad x_1 \geq 3, x_2 \geq 1, x_3 \geq 0, x_4 \geq 5$$

后用插板法。

总结：

- 正整数：插板不相临；从中间空槽取  $k - 1$ .
- 非负整数：插板可相临；从位数取  $k - 1$ .

## 4.7 Polynomial

这节谈下多项式和多重集排列、组合之间的联系。多项式：

$$(x_1 + x_2 + \cdots + x_k)^n$$

的展开式可写成：

$$\sum_{n_1 + n_2 + \cdots + n_k = n} \frac{n!}{n_1! n_2! \cdots n_k!} x_1^{n_1} x_2^{n_2} \cdots x_k^{n_k}, \quad n_i \in \mathbb{Z}_0^+ : n_i \in [0, n], i = 0, 1, \dots, k$$

由此可看出，多项式的每项的系数是一个多重集的全排列数  $\frac{n!}{n_1! n_2! \cdots n_k!}$ . 还可算出展开式的项数是多重集组合数  $\binom{n+k-1}{n}$ .

如果原多项式里  $x_i$  前还带有系数如  $a_i$ :

$$(a_1 x_1 + a_2 x_2 + \cdots + a_k x_k)^n$$

则情展开式的系数在多重集全排列数基础上还有乘以对应的原始系数。

如果问多项式的系数之和是多少，只需给所有  $x_i$  赋值 1 即可：

$$(a_1 + a_2 + \cdots + a_k)^n$$

对于所有  $a_i = 1$  的情况，则系数和是  $k^n$ 。

## 4.8 Grouping and Distribution

下面是先说分组和分配问题，方法数和多重集全排列数有关。

### 4.8.1 Distribution as Injection

将  $n = n_1 + n_2 + \cdots + n_k$  个 **不同的**球  $a_i$  (单重集) 放到  $k$  个 **不同的**对应盒子  $b_i$  里 (单重集)。  $b_1$  放  $n_1$  个，  $b_2$  放  $n_2$  个，  $\cdots$ ，  $b_k$  放  $n_k$  个。如果  $k = n$ ，它就变成单重集的全排列，此时  $n_i = 1$ 。简单描述：把  $n$  个不同色球分成  $k$  堆  $n_i$ ，依次分配给不同盒子  $b_i$ 。

此模型称为**单重集的定向分配**问题。分配强调是分给**不同的**盒子，而定向是说  $n_i$  映射到  $b_i$ ，每个盒子所分得的**数量固定**，即每堆的球数已经限定。入盒操作相当于组球堆贴标签。

计算过程和多重集全排列的类似，选取  $n_1$  个  $C_n^{n_1}$ ，再在剩下球里取  $n_2$  个  $C_{n-n_1}^{n_2}$ ， $\cdots$

这个取堆顺序是  $n_1, n_2, \cdots, n_k$ ，实际按  $i$  的任何一种排列顺序来取都可以，只要所取之堆放入对应的盒子即可。如第一次取  $n_5$  个，相当于给盒子  $b_5$  取球，那么取出后就放入盒子  $b_5$ 。

计算如下：

$$C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k} = \frac{n!}{n_1! n_2! \cdots n_k!}$$

此计算利用了**乘法原则**。

最后，多重集全排列计算过程是给每种色球找队列位置 (共  $n$  个)，而上面的模型是给每个盒子 (共  $k$  个) 选色球。此模型里彩球集合和空盒集合是单重集，但放球后，盒子集合是多重集。每个盒子代表一种元素，里面的球数代表这种元素的个数。

### 4.8.2 Grouping

定向分配把球堆分给不同但固定的盒子，如取球放入  $k$  个的相同的盒子里 (盒子没有编号)，此模型称为**单重集的分组**问题。



由于盒子相同，所以有没有入盒操作对结果不影响，所以入盒是一个空操作。所以有的题里，只提到分组，没有入盒操作（如分派给谁）。单重集分组可以看成把取出球堆也堆在一起，形成一个大堆，大堆内元素是小球堆：分组即分堆。

不失一般性，假设按  $n_1, n_2, \dots, n_k$  的顺序取堆，如果就此打住，那就成了定向分配。到底区别哪？没有**去重**！

球堆的区别是球个数  $n_i$  值的大小（球已取完，不再考虑其不同），可能某些堆的球数相等，这些堆算作是相同的堆。

不失一般性，假设  $n_1 = n_2 = n_3$  这 3 个堆球数相等，它们之间先取谁后取谁没有变化，同理  $n_7 = n_9$  这 2 个堆的先后也不改变分组操作（没有特殊说明，后面都据此假设）。所以要除掉重复：

$$\frac{C_n^{n_1} C_{n-n_1}^{n_2} \dots C_{n_k}^{n_k}}{3! 2!}$$

对所有的相同数值做类似去重除法即可。做题时，尽量把相同的  $n_i$  写在一起，好去识别重复。

我们有理由把

$$\{n_1, n_2, \dots, n_k\}$$

看成多重集：这  $k$  个值分成  $l$  种，每种里有  $t_j$ ,  $j = 1, 2, \dots, l$  个元素，有  $\sum_{j=1}^l t_j = k$ . 拿上面的例子来说：

$$\{3 \cdot n_1, 1 \cdot n_4, 1 \cdot n_5, 1 \cdot n_6, 2 \cdot n_7, \dots, 1 \cdot n_k\}$$

所以去重就是除以多重集每种元素个数的阶乘  $t_1! t_2! \dots t_j!$ ，所以标准表达是：

$$\frac{C_n^{n_1} C_{n-n_1}^{n_2} \dots C_{n_k}^{n_k}}{t_1! t_2! \dots t_j!}$$

一个特例是，所有  $n_i$  全相等，表示是平均分组，所以：

$$\frac{C_n^{n_1} C_{n-n_1}^{n_2} \dots C_{n_k}^{n_k}}{k!} = \frac{n!}{k! \left[\left(\frac{n}{k}\right)!\right]^k}$$

再回过头看定向分配，发现可以分解成分组、定向分配 2 步：

$$\frac{C_n^{n_1} C_{n-n_1}^{n_2} \dots C_{n_k}^{n_k}}{3! 2!} \cdot 3! 2!$$

先把不同球数的堆放入对应的盒子。后面乘以  $3!$  表示  $n_1 = n_2 = n_3$  这 3 个堆和  $b_1, b_2, b_3$  间可任意分派。乘以  $2!$  也是同样的道理。

一句话：**分配问题暗含了分组子问题**。

### 4.8.3 Distribution without Injection

有定向就有不定向，不定向分配也把球堆分派给不同的盒子，但是没有固定分配关系：球堆  $n_i$  和盒子  $b_j$  间没有固定映射关系。把  $n$  个不同色球分成  $k$  堆  $n_i$ ，分配给不同盒子  $b_j$ 。此模型称为 **单重集的不定向分配** 问题。

始终记住，盒子的不同、盒子编号代表队列位置。不定向分配分解成分组、不定向分派 2 步：

$$\frac{C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k}}{3! 2!} \cdot k!$$

由于分组里已考虑到重复问题，后面的排列就不再考虑  $3! 2!$  重复，而应把  $n_1 = n_2 = n_3$  看作是不同的堆，否则就会多次去重。

不定向分配还可以看成在定向分配基础上对  $k$  个球堆（数值  $n_i$ ）进行全排列操作，方法数是：

$$C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k} \cdot \text{球堆全排队数}$$

在前节分组问题说到，堆的球数  $n_i$  是个多重集，所以我们乘以多重集的全排列数：

$$\begin{aligned} C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k} \cdot \frac{k!}{3! 2!} &= \frac{C_n^{n_1} C_{n-n_1}^{n_2} \cdots C_{n_k}^{n_k}}{3! 2!} \cdot k! \\ &= \frac{n!}{3! 2! \cdot n_1! n_2! \cdots n_k!} \cdot k! \end{aligned}$$

公式的推导过程也说明了不同的解题思路。

计算过程总结：

1. 球一旦取出，组合计数完毕，就不再考虑球的不同。
2. 取出的球堆区别于堆内球的个数。
3. 可能某些堆球数相等，这些球堆看作是相同的堆。其实是一个多重集。

### 4.8.4 Distribution of Multiset

上面的问题里，单重集每个球不同，如果所有球全相同呢？若像单重集分组分配样规定  $n_i$ ，则堆数确定，分组问题固定，只有 1 种方法。定向分配问题： $n$  个相同的球分成  $k$  组  $n_i$  定向派给不同盒子  $b_i$ ，也只有 1 种方法。不定向分配数是多重集的全排列如  $\frac{k!}{3! 2!}$ 。

所以一般不规定堆内球数，如把  $n$  个同色球放入  $k$  个不同的盒子里，盒子不为空。其实这个问题可化成求  $k$  元线性不定方程的正整数解。方法数是：

$$C_{n-1}^{k-1}$$

此时小球可看作是只有 1 种元素的多重集  $M = \{n \cdot a_1\}$ . 此模型可称为含 1 种元素的**多重集定向分配**问题。

如果盒子可以为空，则化成求方程的非负整数解。

含一种元素的**多重集分组和不定向分配**问题没有统一解法，因为分组和不定向要考虑去重问题，但是去重的前担是要知道  $n_i$  数值。所以只能一一列举出方程解，再考虑每种解的去重，无法直接写出计数公式。

## 4.9 Summary

分组分配总结：

1. 单重集分组：固序取堆、去  $n_i$  重。
2. 单重集定向分配：固序取堆。
3. 单重集不定向分配：固序取堆、去  $n_i$  重、排列。
4. 多重集定向分配：插板法、解方程。
5. 多重集分组和不定向分配：没有公式，只可一一列举。



## Chapter 5

# 卡特兰数

### 5.1 卡特兰公式

卡特兰数 Catalan Number 在许多算法计数问题中都有应用，如出栈计数，二叉树形态计数等。卡特兰数也叫明安图数。因为最提出这数的是中国明代数学家明安图。但因为近代以来，科学技术主要由西方科学家发起，所以很多发现用西方人物命名。

明安图数其实是一个组合数：

$$C_n = \frac{C_{2n}^n}{n+1} = \frac{1}{n+1} \binom{2n}{n} = \frac{(2n)!}{n!(n+1)!}$$

卡特兰数  $C_n$  表示对  $n$  个元素的某种顺序计数。注意这里符号  $C_n$  不是前面的  $n$  个元素的组合数里的  $C_n^r$ 。

卡特兰数可以写成另外一种形式：

$$\begin{aligned} C_n &= \frac{C_{2n}^n}{n+1} \\ &= \frac{1}{n+1} \frac{(2n)!}{n!n!} \\ &= \frac{1}{n(n+1)} \frac{(2n)!}{(n-1)!n!} = \left(\frac{1}{n} - \frac{1}{n+1}\right) \frac{(2n)!}{(n-1)!n!} \\ &= \frac{1}{n} \frac{(2n)!}{(n-1)!n!} - \frac{1}{n+1} \frac{(2n)!}{(n-1)!n!} = \frac{(2n)!}{n!n!} - \frac{(2n)!}{(n-1)!(n+1)!} \\ &= C_{2n}^n - C_{2n}^{n+1} \end{aligned}$$

请注意此化简过程用到

$$\frac{1}{n(n+1)} = \frac{1}{n} - \frac{1}{n+1}$$

在处理组合数问题时,  $\frac{1}{n-1}, \frac{1}{n}, \frac{1}{n+1}$  经常被用到来化简。这种倒数可以和阶成结合, 组成新的组合数。

上面说的是卡特兰数结果, 卡特兰数的递推关系式是:

$$\begin{aligned} C_n &= \frac{C_{2n}^n}{n+1} \\ &= \sum_{k=1}^n C_{k-1} C_{n-k} \\ &= C_0 C_{n-1} + C_1 C_{n-2} + \cdots + C_{n-1} C_0 \\ &= \sum_{k=0}^{n-1} C_k C_{n-1-k} \end{aligned}$$

如何由这个递推关系得到结果式, 要用到产生式或叫母函数 **Generating Function**:

$$g(x) = C_0 x^0 + C_1 x^1 + \cdots + C_{n-1} x^{n-1} + C_n x^n + \cdots$$

关于如何用母函数求系数, 细节请参考组合学课件, 这里给出计算过程。上面公式里, 我们发现每项是子规模的乘积, 所以对母函数平方:

$$\begin{aligned} g^2(x) &= (C_0 x^0 + C_1 x^1 + \cdots + C_n x^n + \cdots)(C_0 x^0 + C_1 x^1 + \cdots + C_n x^n + \cdots) \\ &= C_0^2 x^0 + (C_0 C_1 + C_1 C_0) x^1 + \cdots + (C_0 C_n + C_1 C_{n-1} + \cdots + C_n C_0) x^n + \cdots \\ &= C_1 x^0 + C_2 x^1 + \cdots + C_{n+1} x^n + \cdots \end{aligned}$$

对上式乘以  $x$  得:

$$x \cdot g^2(x) = C_1 x^1 + C_2 x^2 + \cdots + C_n x^n + C_{n+1} x^{n+1} + \cdots$$

对上式加  $C_0 x^0 = 1$  得:

$$\begin{aligned}
 1 + x \cdot g^2(x) &= C_0x^0 + C_1x^1 + C_2x^2 + \cdots + C_nx^n + \cdots \\
 &= g(x) \\
 x \cdot g^2(x) - g(x) + 1 &= 0
 \end{aligned}$$

解上式得：

$$g(x) = \frac{1 \pm \sqrt{1 - 4x}}{2x}$$

通过变换，得出母函数通项关系，去除系数  $C_k$ ，进而解出母函数关于  $x$  的表达式。 $g(x)$  是最开始的系数是由  $C_k$  定义的，通过转换后计算出系数，就得到  $C_k$ 。

根据二项式的推广：

$$\begin{aligned}
 (x + y)^\alpha &= \sum_{k=0}^{\infty} \binom{\alpha}{k} x^k y^{\alpha-k}, \quad a \in \mathbb{R} \\
 \binom{\alpha}{k} &= \frac{\alpha(\alpha-1) \cdots (\alpha-k+1)}{k!}
 \end{aligned}$$

由此得：

$$\begin{aligned}
\sqrt[2]{1-4x} &= [1 + (-4x)]^{\frac{1}{2}} \\
&= \sum_{k=0}^{\infty} \binom{1/2}{k} (-4x)^k \\
&= 1 + \sum_{k=1}^{\infty} \binom{1/2}{k} (-4x)^k \\
&= 1 + \sum_{k=1}^{\infty} \frac{\frac{1}{2} \cdot \frac{-1}{2} \cdot \frac{-3}{2} \cdots \frac{3-2k}{2}}{k!} (-1)^k 2^{2k} x^k \\
&= 1 - \sum_{k=1}^{\infty} \frac{1 \cdot 3 \cdots (2k-3)}{k!} 2^k x^k \\
&= 1 - \sum_{k=1}^{\infty} \frac{1(2 \cdot 1) 3(2 \cdot 2) \cdots (2k-3)[2 \cdot (k-1)]}{k!(k-1)!} 2x^k \\
&= 1 - \sum_{k=1}^{\infty} \frac{1 \cdot 2 \cdot 3 \cdots (2k-3)(2k-2)}{k!(k-1)!} 2x^k \\
&= 1 - \sum_{k=1}^{\infty} \frac{[2(k-1)]!}{k!(k-1)!} 2x^k
\end{aligned}$$

结合上式得：

$$\begin{aligned}
g(x) &= \frac{1 \pm \sqrt[2]{1-4x}}{2x} \\
&= \frac{1 \pm [1 - \sum_{k=1}^{\infty} \frac{[2(k-1)]!}{k!(k-1)!} 2x^k]}{2x}
\end{aligned}$$

考虑到定义时，系数  $C_k$  是正数，所以正负号里选负号：



$$\begin{aligned}
g(x) &= \frac{1 \pm \sqrt{1-4x}}{2x} \\
&= \frac{1 \pm [1 - \sum_{k=1}^{\infty} \frac{[2(k-1)]!}{k!(k-1)!} 2x^k]}{2x} \\
&= \frac{1 - [1 - \sum_{k=1}^{\infty} \frac{[2(k-1)]!}{k!(k-1)!} 2x^k]}{2x} \\
&= \sum_{k=1}^{\infty} \frac{[2(k-1)]!}{k!(k-1)!} x^{k-1} \\
&= \sum_{k=0}^{\infty} \frac{(2k)!}{k!(k+1)!} x^k
\end{aligned}$$

通过系数对比，得出  $C_k = \frac{(2k)!}{k!(k+1)!} = \frac{1}{k+1} C_{2k}^k$ 。

## 5.2 卡特兰数应用

卡特兰数的来自于应用问题，7 通过元素进出栈顺序数问题，来说明公式推导过程。

长度为  $2n$  的 Dyck Word ( $n$  个  $x$  和  $n$  个  $y$  组成字符串) 数;  $n$  对括号匹配组成合法运算式数;  $n$  个节点组成二叉树方案数;  $2n+1$  个节点的满二叉树数都是  $C_n$ . 特别此  $n$  个节点的二叉树, 添加  $n+1$  个叶子节点, 形成满二叉树, 个数都是  $C_n$ .

二叉树前序序列和中序序列的关系。给定某二叉树前序序列或后序序列, 求中序序列的可能数, 结果也是卡特兰数。相当于以前序序列或后序序列入栈, 中序序列出栈。

有  $n \times n$  的小方格组成的正方形, 要求从一个对角走到别一对角, 只能横向或竖向行走, 不能回头且不能穿过对角线, 问有多少种单调路径? 如从左下角往右上角行走, 单调路径表示每一步只能向右或向上, 即向最终方向走。如果不考虑对角线问题, 则竖向应走  $n$  步, 横向应走  $n$  步, 共  $2n$  步, 方法数是  $C_{2n}^n$ . 若考虑对角线限制, 假假沿在对角线下方走, 则说明任何时刻, 向右步数不少于向上步数。肯体参考上面提到的进出栈分析。

这些不同问题可以互相转化。如进栈、出栈, Dyck Word 里的  $x$  和  $y$ , 左、右括号都互相对应。全部用数学描述是  $2n$  个  $\pm 1$  串。

注意卡特兰数  $C_n$  里有  $n$  表示规模, 但是实际问题里,  $n$  可能表示的是  $n$  对元素, 而不是  $n$  个元素。如括号匹配里,  $n$  对括号有  $n$  个左括号和  $n$  个右括号。



## **Part III**

# **数列**



## 5.3 等差数列

我们知道等差数列的前  $n$  项和是首项加尾项乘以项数除以 2.

$$\begin{aligned}
 a_1 &= a_1 \\
 &= a_0 + d \\
 a_2 &= a_1 + d = a_1 + d \\
 a_3 &= a_2 + d = a_1 + 2 \cdot d \\
 &\dots \\
 a_n &= a_{n-1} + d \\
 &= a_1 + (n-1) \cdot d \\
 &= a_0 + n \cdot d
 \end{aligned}$$

这里特别提到  $a_0$ . 严格来说, 数列下标从 1 开始 (表示第一个项), 0 不属于数列下标, 但有时为了计算方便, 要借用  $a_0 = a_1 - d$ .

等差数列, 任意连续三项中, 前后两项之和是中间项的 2 倍, 因中间项减等差  $d$  是前一项, 加等差  $d$  是后一项。

等差数列的前  $n$  项和:

$$S_n = \frac{(a_1 + a_n) \cdot n}{2}$$

## 5.4 等差数列幂和

等差数列的幂和是指等差数列前  $n$  项的  $t$  ( $t = 1, 2, \dots$ ) 次幂之和:

$$S_t(n) = \sum_{k=1}^n a_k^t = a_1^t + a_2^t + \dots + a_n^t$$

基本思路是升维: 借用  $t+1$  次二项展开式, 得到数列相邻两项的  $t+1$  次幂差。一般来说, 我们关注的是平方、立方, 即  $t = 1, 2$ . 后面不加说明, 假设  $t$  为 2, 则  $t+1 = 3$ :

$$\begin{aligned}
 (a+b)^3 &= a^3 + 3a^2b + 3ab^2 + b^3 \\
 a_{n+1}^3 - a_n^3 &= (a_n + d)^3 - a_n^3 \\
 &= 3d \cdot a_n^2 + 3d^2 \cdot a_n + d^3
 \end{aligned}$$

对  $n$  遍历:

$$\begin{aligned}
a_2^3 - a_1^3 &= 3d \cdot a_1^2 + 3d^2 \cdot a_1 + d^3 \\
a_3^3 - a_2^3 &= 3d \cdot a_2^2 + 3d^2 \cdot a_2 + d^3 \\
&\dots \\
a_{n+1}^3 - a_n^3 &= 3d \cdot a_n^2 + 3d^2 \cdot a_n + d^3
\end{aligned}$$

对  $n$  个式子求和，得出公式：

$$a_{n+1}^3 - a_1^3 = 3d \cdot \sum_{k=1}^n a_k^2 + 3d^2 \cdot \sum_{k=1}^n a_k + n \cdot d^3$$

此式很明显可以直接算出平方和。但此式受限于  $a_0, d$  没法直接化简，所以我们关键是住思路。其实等差数列前  $n$  项和可以看成  $t = 1$ ，其前  $n$  项和除了首尾相加外，还可以借助  $t + 1 = 2$  次展开式。

下面以前  $n$  个自然数的平方和为例介绍求解思路 ( $a_1 = 1, d = 1$ )。首先，我们有：

$$\begin{aligned}
a_{n+1}^3 - a_n^3 &= (n+1)^3 - n^3 \\
&= 3n^2 + 3n + 1
\end{aligned}$$

对  $n$  遍历：

$$\begin{aligned}
2^3 - 1^3 &= 3 \times 1^2 + 3 \times 1 + 1 \\
3^3 - 2^3 &= 3 \times 2^2 + 3 \times 2 + 1 \\
&\dots \\
n^3 - (n-1)^3 &= 3(n-1)^2 + 3(n-1) + 1 \\
(n+1)^3 - n^3 &= 3n^2 + 3n + 1
\end{aligned}$$

把这  $n$  个式子加起来：

$$\begin{aligned}
(n+1)^3 - 1 &= 3 \cdot \sum_{k=1}^n k^2 + 3 \cdot \sum_{k=1}^n k + n \\
3 \cdot \sum_{k=1}^n k^2 &= (n+1)^3 - 3 \cdot \frac{(1+n) \cdot n}{2} - (n+1) \\
\sum_{k=1}^n k^2 &= \frac{n(n+1)(2n+1)}{6}
\end{aligned}$$

通过公式，我们发现平方和 ( $t = 2$ ) 与线性和 ( $t = 1$ ) 的关系：

$$\frac{\sum_{k=1}^n k}{\sum_{k=1}^n k^2} = \frac{2n+1}{3}$$

对于其它等差数列，我们用同样方法。如计算  $1^2, 4^2, 7^2, \dots, (3n-2)^2$ 。

## 5.5 等差数列的积和

上面讲的是等差数列每项的  $t$  次幂和，如果求和时，每项不是  $t$  次幂，而改成相临  $t$  项积呢？为简便起见，这里假设  $t = 2$ ，更高次幂依此类推即可。

$$S(n) = \sum_{k=1}^n a_k \cdot a_{k+1} = a_1 \cdot a_2 + a_2 \cdot a_3 + \dots + a_n \cdot a_{n+1}$$

类似上面方法，思路是对乘积变换，前后项可能消除子项。具体是这样的，升维：把二项积变换三项积：

$$a_n \cdot a_{n+1} = x \cdot a_{n-1}a_n a_{n+1} + y \cdot a_n a_{n+1} a_{n+2}$$

其中  $x$  和  $y$  是特定系数，实际上  $-x = y = \frac{1}{3d}$ ：

$$\begin{aligned} x \cdot a_{n-1}a_n a_{n+1} + y \cdot a_n a_{n+1} a_{n+2} &= x \cdot (a_n - d)a_n a_{n+1} + y \cdot a_n a_{n+1}(a_n + 2d) \\ &= x a_n \cdot a_n a_{n+1} - x d \cdot a_n a_{n+1} + y a_n \cdot a_n a_{n+1} + 2y d \cdot a_n a_{n+1} \\ &= [(x+y)a_n + (2y-x)d] \cdot a_n a_{n+1} \end{aligned}$$

$x$  和  $y$  的值应独立于  $a_0, d$  成立，则：

$$\begin{aligned} (x+y)a_n + 2yd - xd &= 1 \\ x+y &= 0 \\ 2y-x &= 1 \end{aligned}$$

可以算出  $-x = y = \frac{1}{3d}$ ：

$$a_n a_{n+1} = \frac{1}{3d} (-a_{n-1}a_n a_{n+1} + a_n a_{n+1} a_{n+2})$$

对  $n$  遍历得：

$$\begin{aligned}
a_1 a_2 &= \frac{1}{3d}(-a_0 a_1 a_2 + a_1 a_2 a_3) \\
a_2 a_3 &= \frac{1}{3d}(-a_1 a_2 a_3 + a_2 a_3 a_4) \\
&\dots \\
a_n \cdot a_{n+1} &= \frac{1}{3d}(-a_{n-1} a_n a_{n+1} + a_n a_{n+1} a_{n+2})
\end{aligned}$$

得出前等差数列的前  $n$  项积和：

$$S(n) = \sum_{k=1}^n a_k \cdot a_{k+1} = \frac{1}{3d}(a_n a_{n+1} a_{n+2} - a_0 a_1 a_2)$$

不难发现，此式关键是首尾两个三项积之差。注意这里借用了  $a_0$ 。下面以

$$S(n) = 1 \times 4 + 4 \times 7 + 7 \times 10 + \dots + (3n-2)(3n-2+3)$$

为例。可推出：

$$\begin{aligned}
(3n-2)(3n-2+3) &= \frac{1}{9}[-(3n-2-3)(3n-2)(3n-2+3) + (3n-2)(3n-2+3)(3n-2+6)] \\
&= \frac{1}{9}[-(3n-5)(3n-2)(3n+1) + (3n-2)(3n+1)(3n+4)]
\end{aligned}$$

每项被变换成加减法，可以通部分抵消，达到求和目的：

$$\begin{aligned}
1 \times 4 &= \frac{1}{9}[-(-2) \times 1 \times 4 + 1 \times 4 \times 7] \\
4 \times 7 &= \frac{1}{9}[-1 \times 4 \times 7 + 4 \times 7 \times 10] \\
7 \times 10 &= \frac{1}{9}[-4 \times 7 \times 10 + 7 \times 10 \times 13] \\
&\dots \\
(3n-2)(3n+1) &= \frac{1}{9}[-(3n-5)(3n-2)(3n+1) + (3n-2)(3n+1)(3n+4)]
\end{aligned}$$

这里  $a_0 = 1 - 3 = -2$ 。所有等式相加，得：

$$S(n) = \frac{1}{9}[(3n-2)(3n+1)(3n+4) + 8]$$



## 5.6 幂和与积和的联系

至此等差数列的幂和与积和都可得出。我们发现求幂和与积和的通用思路是**升维**。幂和借用  $t+1$  次二展开式，而积和借用  $t+1$  项积。

其实幂和可以用积和的方式计算：

$$\begin{aligned}a_n^2 &= a_n \cdot [(a_n + d) - d] \\&= a_n \cdot [a_{n+1} - d] \\&= a_n a_{n+1} - d \cdot a_n\end{aligned}$$



## **Part IV**

# **Algorithm**



## Chapter 6

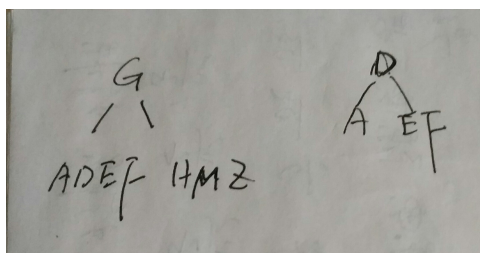
# Tree

### 6.1 前序中序求后序

给出一个二叉树的前序遍历 **GDAFEMHZ** 和中序遍历 **ADEFGHMZ**, 求后序遍历。首先, 二叉树的前序遍历, 中序遍历, 后序遍历的区别是根节点是先、中、后访问。问题的关键是:

1. 前序遍历的第一个节点是根节点;
2. 在中序遍历里找到根结点, 其左边是左子树节点, 右边是右子树节点;
3. 对左子树重复上面两步;
4. 对右子树重复上面两步。
5. 二叉树被求出, 从而得到对应的后序遍历。

上面的例子里, 从前序遍历知 **G** 是根节点, 从后序遍历知, 左子树是 **ADEF**, 右子树是 **HMZ**, 此顺序就是左子树的中序遍历顺序。



但还要找到对应的前序顺序。对左子树 **ADEF** 来说, 在前序遍中的顺序是 **DAFE**, 说明左子树的根节点是 **D**, 从其中序顺序可知, 左子树是 **A**, 右子树是 **EF**. 重复此过程即可得出二叉树, 再得出后序遍历 **AEFDHZMG**.

同理, 已知后序遍历和中序遍历, 也可求出前序遍历。唯一的区别是后序遍

历的最后一个节点是根节点，每次找根节点时从最后找。不管是哪种题，中序遍历一定要有！

## Chapter 7

# 栈

### 7.1 进栈出栈

栈是一种连续结构，数据在栈内连续存储。栈有栈顶和栈底类似链表的首部和尾部，区别是栈在垂直上下方向生长，链表在水平左右向生长。如果需要，我们也可以用水平示意图表示栈，左边是栈底，往右生长，如数组模拟栈就是这种思路。C/C++ 内存中的栈区即是一种栈结构。

给定一有  $n$  个元素的有序序列  $a_1, a_2, \dots, a_n$ ，对栈有两种基本操作：进栈 push 和出栈 out。无论是进栈还是出栈，都是对栈顶进行操作。进栈也称压栈，表示保持元素间相对顺序把元素存到栈顶，而出栈则表示取出栈顶的元素。

进栈、出栈组成一个动态操作序列，在  $n$  个元素依次进栈的过程中，伴随着元素随机出栈。动态操作序列有  $2n$  个步骤，其中  $n$  个是进栈操作，另  $n$  个是出栈操作。进出栈操作结束，栈为空，进栈的  $n$  个元素亦全部出栈。

进栈顺序事先给定，用下标  $i$  表示，表示为  $a_i, i \in \{1, 2, \dots, n\}$ 。出栈序列是进栈序列的某种排列，可记为  $a_{j_l}, j_l \in \{1, 2, \dots, n\}, l \in \{1, 2, \dots, n\}$ 。进栈操作和  $a_{j_l}$  出栈操作相间出现。 $2n$  个动态操作序列可记成：

$$c_k, k \in \{1, 2, \dots, 2n\}, c_k \text{ 是 } a_i \text{ 进栈或 } a_{j_l} \text{ 出栈}$$

可以肯定  $c_1$  肯定代表  $a_1$  进栈， $c_{2n}$  表示某元素出栈。

实际上给定一个完整的出栈序列  $a_{j_l}$  我们可以还原进出栈操作序列  $c_k$ ，所以通常只需出栈序列  $a_{j_l}$  即可。给出进栈序列  $a_i$  在  $c_k$  中的位置，我们可以推出剩下  $n$  个位置的出栈序列  $a_{j_l}$ 。

## 7.2 先进后出

进栈出栈遵循**先进后出，后进先出**的原则。对于当前栈内元素，后进栈的压在先进栈的上边，栈顶永远是当前最后一个进栈元素，所以先进栈的肯定比后进栈的后出栈，或说后进栈的肯定比先进栈的先出栈。换句话说，任一时刻，栈的状态决定了当前栈内元素的相对出栈顺序。

总结起来，进出栈序列  $c_k$  满足：

- $a_i$  间的相对位置固定，因为进栈顺序事先给定。
- $a_i$  和  $a_{j_l}$  符合先进后出。

可以看出，上面说的进栈出栈，不是指所有  $n$  个元素全部一次性入栈，再一次性出栈。实际过程是进栈、出栈两种操作是交替进行，只要栈不为空，就有可能进行出栈操作。如  $a_1, a_2$  依次入栈（栈顶到底依次是  $a_2, a_1$ ，所以  $a_2$  肯定比  $a_1$  先出栈），接着  $a_2$  立即出栈， $a_3$  入栈（栈内依次是  $a_3, a_1$ ，所以  $a_3$  肯定比  $a_1$  先出栈）。

前对对进出栈过程分析的比较清楚，任一时刻的栈内元素符合先进后出原则。那么此原则是如何体现在完整操作序列  $c_k$  里的呢？

**出栈序列里任一元素  $a_{j_l}$ ，比其后出且先进的元素倒序。这些元素排在  $a_{j_l}$  后面（后出栈），而且下标比  $j_l$  小（先进栈）。它们（包括  $a_{j_l}$  在内）一定是按下标降序排列！**

我们可以据此判断一个出栈序列是否正确。如 3, 4, 5, 1, 2, 9, 8, 7, 6 是出栈序列的下标  $j_l$ ，显然这个出栈序列不正确，因为出现了 3, 1, 2。存在某个栈状态，3 压在 1 和 2 上面，栈内元素由上到下依次是 3, 2, 1。

为了便于运算，我们一般用下标直接代替元素本身！在验证时，我们遍历  $j_l$ ，从其开始，后面比其小的是否全部降序。由上例，4, 1, 2 和 5, 1, 2 都不合法。

## 7.3 栈深度最小值

给出出栈序列  $a_{j_l}$ ，问栈深度最小值是多少。

在进出栈动态操作过程中，元素或进栈或出栈，栈内元素个数不定。栈需容纳未出栈元素。如果每进栈一次，再出栈一次，则栈深度只需 1 即可。如果所有元素全部入栈再出栈，则栈深度最少为  $n$ 。

对于一般情况，该如何算呢？其实很简单，只需利用上面的降序原则。遍历出栈序列，对  $a_{j_l}$ ，数出从其开始往后，降序元素个数（包含  $a_{j_l}$ ），最大值即为栈的最小深度。



## 7.4 出栈序列数

出栈序列数是指, 给出进栈序列  $a_i$ , 有多少种出栈序列  $a_{ji}$ ?

操作序列  $c_k$  有  $2n$  个位置, 一旦确定了进栈序列  $a_i$  的  $n$  个位置, 则立马得到出栈序列。也可以选出  $n$  个位置作为出栈序列, 但是出栈序列内部还有先后问题, 所以用前一种思路更好。

从  $2n$  个位置中选取  $n$  个的方法数是组合数  $C_{2n}^n$ , 但进栈序列的选位方法数比这小。如  $a_1$  进栈肯定在第 1 位, 而且  $a_i$  肯定不能排在最后  $n$  个位置。所以最终的方法数肯定是  $C_{2n}^n$  除以或减去某个数。

如果对进出栈计数, 用  $+1$  表示一次进栈,  $-1$  表示一次出栈, 则任何时候栈内元素个数就是这些  $\pm 1$  的和。显然, 任一时刻, 栈内元素不能是负数个, 入栈次数肯定不少于出栈次数,  $+1$  个不小于  $-1$  个数,  $\pm 1$  的和  $\geq 0$ 。

对应到  $c_k$ ,  $k \in \{1, 2, \dots, 2n\}$  序列, 从头至尾扫描 (水平示意图), 对  $k$  遍历, 对任意  $k$ , 前  $k$  个位置里,  $+1$  个数不少于  $-1$  个数。进出栈操作完毕,  $n$  个  $+1$  和  $n$  个  $-1$  总和为 0。有时为了方便, 直接用  $\pm$  表示即可。有的地方用  $+1$  和 0 分别表示进栈、出栈, 最后的  $C_k$  是一个二进制数。

总结起来是, **进出栈序列  $C_k$  的所有前缀子串皆满足  $+1$  个数不小于 (大于等于)  $-1$  个数。**

特别地, 当  $k = 1$  时, 前 1 位只能是  $a_1$  进栈, 和加 1 为 1。当  $k = 2n$  时, 最后 1 位是某元素出栈, 和减 1 为 0。

所以我们要在  $C_{2n}^n$  的基础上排除和为负的情况, 也即在栈为空时或为负时, 进行出栈操作, 也即在遍历  $k$  的过程中遇到和为负数情况。

假设在遍历  $k$  的过程中, 第 1 次遇到和为  $-1$  时, 有  $m$  个  $-1$ , 则  $+1$  为  $m-1$  次, 此时  $k = 2m-1$ ,  $m \in \{1, 2, \dots, n\}$ 。显然  $k$  是奇数,  $m$  最大值是  $n$ 。特别地, 第  $k$  位是  $-1$ , 前面  $2m-2$  位中, 分别有  $m-1$  位  $+1$  和  $-1$ 。后面  $2n-k$  位里面有  $n-m+1$  个  $+1$ , 剩下  $n-m$  位是  $-1$ 。

对于奇数  $k$ , 非法方法数是不是

$$C_{2m-2}^{m-1} C_{2n-(2m-1)}^{n-(m-1)} = C_{2m-2}^{m-1} C_{2n-(2m-1)}^{n-m}$$

呢? 此式表示前  $2m-2$  个位置里选  $m-1$  个出来放  $+1$  表示进栈, 剩下的  $2n-(2m-1)$  个位置里选  $n-(m-1)$  个放剩下的  $+1$ 。从  $-1$  的角度分析, 结果是  $C_{2m-2}^{m-1} C_{2n-(2m-1)}^{n-m}$ 。答案是否定得! 因为这种选取方法不能保证前面  $k-1$  位里不出现和为负的情况。

我们可以换个思路。第 1 次遇到和为  $-1$  时, 后面  $2n-k$  位的和是  $+1$ , 即进栈次数比出栈多 1。如果交换后面的  $\pm 1$ , 则可以得到一个满射, 组合计数不变。交换后面的  $\pm 1$  后,  $2n$  位里面有  $n+1$  个  $-1$  和  $n-1$  个  $+1$ , 总和为  $-2$ ,

出栈比进栈多 2 次。反过来, 总和为  $-2$  时总能找到第 1 次和为  $-1$  的情况。所以总的非法进出栈方法数是  $C_{2n}^{n+1} = C_{2n}^{n-1}$ , 那么合法进出栈总数为:

$$C_{2n}^n - C_{2n}^{n+1} = \frac{C_{2n}^n}{n+1}$$

这其实是一个卡特兰数 5.

至此, 我们从组合数的角度直接得出进栈数序列数为卡特兰数。进出栈序操作序列, 对应组合数学问题, 是在  $2n$  个位置上, 选取  $n$  个放  $+1$  或  $-1$ , 要求是对位置  $k$  遍历, 和为非负整数。

## 7.5 进出栈递推方法

下面我们从递推方法分析进出栈问题。分析算法问题时, 我们可以找出问题不同规模之间的联系, 具体点就是递推关系。在算法领域, 初始条件非常重要。同一公递推公式的因为不同的初始条件而产生不同的数列。

用  $f(n)$  表示出栈序列数。先看进出栈的小规模问题。明显  $f(1) = 1$ , 只有 1 个元素  $a_1$  时, 其进栈再出栈, 只有一种可能。对于  $f(2) = 2$  时:

1. 第一种是  $a_1$  进,  $a_1$  出,  $a_2$  进,  $a_2$  出:  $+ - + -$
2. 第二种是  $a_1$  进,  $a_2$  进,  $a_2$  出,  $a_1$  出:  $+ + - -$

当  $n > 2$  时, 我们就要抽象分析了。分析有两个思路。第一种是考虑  $n$  个元素里, 哪个最先出栈, 也即出栈序列里, 谁是第 1 个元素。第二种是考虑第 1 个元素  $a_1$  出栈时所在位置, 也即  $a_1$  在出栈序列排第几位。我们先看第一种思路。

### 7.5.1 最先出栈元素

如果是  $a_1$  最先出栈, 则出栈序列是  $a_1, a_{j_1}, j_1 \in \{2, 3, \dots, n\}$ . 很明显,  $a_1$  前两位是  $+ -$ , 表示  $a_1$  进栈后立马出栈。 $a_1$  进栈、出栈后不影响其后元素的进出栈, 因为后面  $n - 1$  个元素根本还没入栈, 这是一个  $n - 1$  规模的子问题, 有  $f(n - 1)$  种可能。

如果是  $a_2$  最先出栈。此时其前元素  $a_1$  已入栈并且没出栈。同理  $a_2$  最先出栈不影响后面元素  $a_i, i \in \{3, 4, \dots, n\}$  的出栈序列, 因为后面元素根本还没入栈。其后元素有  $f(n - 2)$  种可能出栈序列。其前面的  $a_1$  穿插于其后元素出栈序列之间。

对于一般情况,  $a_k, k \in \{1, 2, \dots, n\}$  最先出栈。此时  $a_i, i \in \{1, 2, \dots, k - 1\}$  已入栈并且没出栈。由前面分析可知, 这  $k - 1$  个元素元素下标比  $k$  小 (先

进栈), 但比  $a_k$  后出栈, 在出栈序列里必须是 (相对) 降序排列 (包括  $a_k$  在内), 即  $a_k, a_{k-1}, \dots, a_1$ . 因为在  $a_k$  先出栈的条件下, 从栈顶到栈底依次是  $a_{k-1}, a_{k-2}, \dots, a_1$ . 特殊的, 如果  $k = n$ , 则  $a_n$  最先出栈, 说是所有  $n$  个元素依次全部进栈, 然后依次倒序出栈。

对于下标比  $k$  大的元素而言,  $a_k$  最先出栈不影响  $a_i, i \in \{k+1, k+2, \dots, n\}$  出栈。后面元素的出栈是规模为  $n-k$  的子问题, 有  $f(n-k)$  种可能。

这个一般情况有多少种可能呢? 考虑出栈序列  $a_k, a_{j_l}, j_l \neq k, j_l \in \{1, 2, \dots, n\}$ . 除了第 1 位  $a_k$ , 余下  $n-1$  位里先选  $k-1$  个出来按序放  $a_{k-1}, a_{k-2}, \dots, a_1$ . 余下的  $n-k$  位是  $a_i, i \in \{k+1, k+2, \dots, n\}$  的出栈序列, 对应序列数是  $f(n-k)$ . 所以最终是是:

$$C_{n-1}^{k-1} f(n-k) = C_{n-1}^{n-k} f(n-k), k \in \{1, 2, \dots, n\}$$

为保证定义完备性, 当  $k = n$  时, 我们定义  $f(n-n) = f(0) = 1$ . 所以  $a_i, i \in \{1, 2, \dots, n\}$  的出栈序列数是:

$$f(n) = \sum_{k=1}^n C_{n-1}^{n-k} f(n-k) = \sum_{k=1}^n C_{n-1}^{k-1} f(n-k)$$

$$f(1) = f(0) = 1$$

**很可惜这个结果是错的!** 乘法原理要求相乘的两个部分没有交集。上面的分析里, 保证了后面  $n-k$  个元素的出栈相对序列, 也保证了前面  $k-1$  个元素倒序出栈, 但当两个部分合在一起时, 用乘法原理时, 有部分不符合要求。如  $a_k, a_{n-1}, a_2, a_1, a_{n-3}, \dots$  这个出栈序列里,  $a_k$  前面的  $a_2, a_1$  保持了降序,  $a_k$  后面的  $a_{n-1}, a_{n-3}$  是子问题, 但是合在一起时,  $a_{n-1}, a_2, a_1, a_{n-3}$  这 4 个元素不是严格降序。

虽然这个分析错误, 但能加强我们对进出栈问题的理解。

### 7.5.2 $a_1$ 出栈位置

下面从最先入栈的元素  $a_1$ , 也即第 1 个元素出栈序号入手分析问题。设  $a_1$  第  $k, k \in \{1, 2, \dots, n\}$  位出栈, 则在  $a_1$  前有  $k-1$  个元素出栈,  $a_1$  后有  $n-k$  个元素出栈。

可以肯定,  $a_1$  第  $k$  位出栈时, 栈刚好为空, 表明有  $k$  个元素入栈并完全出栈。那么这  $k-1$  个元素是哪些呢?  $a_1$  第 1 位入栈后, 有  $k-1$  个元素先入进出栈,  $a_1$  等这些元素出栈后才最后出栈。明显这  $k-1$  个元素就是  $a_i, i \in \{2, 3, \dots, k-1\}$ .

假设  $a_l$ ,  $k \leq l \leq n$  在前  $k$  个出栈元素中。若  $a_l$  在第 1 位出栈, 据上一小节思路, 这  $k$  个位置显然放不下  $a_l, a_{l-1}, \dots, a_1$  这  $l$  个元素。若  $a_l$  在第 2 位, 则  $a_l$  和  $a_1$  间的距离更短了, 而且此时第 1 位是比  $a_l$  更靠后的元素了, 同样容不下那么多元素。

以  $k = 2$  为例,  $a_1$  出栈前要等 1 个元素先出栈。这个元素只能是  $a_2$ 。假若是  $a_3$ , 则出栈序列是  $a_3, a_1, \dots$ , 显然中间的  $a_2$  没有正确出栈。从另一个角度看,  $a_3$  第 1 个出栈, 这个序列没有保证  $a_3, a_2, a_1$  间相对降序出栈。因此  $a_1$  等的不可能是  $a_2$  后的任何元素。

这  $k$  个元素的进出栈是一个完整的子问题, 最先入栈的最后出栈。由于第  $k$  位确定为  $a_1$ , 则规模为  $k - 1$ 。方法数是  $f(k - 1)$ 。

在  $a_1$  后面进出栈的元素是  $a_i$ ,  $i \in \{k + 1, k + 2, \dots, n\}$  这  $n - k$  个元素。通过上面分析得,  $a_1$  在第  $k$  位出栈时, 其前  $k - 1$  个元素和后  $n - k$  个元素的进出栈序列完全独立, 互不影响。后面  $n - k$  个元素的进出栈也是一个完整的子问题, 规模为  $n - k$ , 方法数是  $f(n - k)$ 。

根据乘法原理,  $a_1$  第  $k$  位出栈时, 方法数为  $f(k - 1)f(n - k)$ 。由此  $n$  个元素的进出栈方法数是:

$$f(n) = \sum_{k=1}^n f(k-1)f(n-k)$$

$$f(0) = 1$$

此公式就是卡特兰数 5 的递推公式。不像上节, 这节分析没有错误。两个子问题的进出栈序列集合交集为空, 符合乘法原理。

不难发现, 进出栈动态操作过程中, 一旦栈为空, 就得到一个规模更小的完整子问题。进出栈问题, 可以分成很多小的子问题, 这此子问题互不影响。特别地, 当出栈序列为  $a_1, a_2, \dots, a_n$  时, 每一步出栈操作都定义了一个子问题, 规模为 1, 一共有  $n$  个子问题。任意  $k$ ,  $a_k$  进栈再出栈时栈为空, 构成一个进出栈子问题。

前面分析中, 在给出  $a_1$  出栈得到两个子问题, 刚好互逆, 并集是进出栈序列的全集。对这两个子问题, 我们还可以分成规模更小的子问题。这是从已知结果前提下, 一个递归分解过程。

另外, 在出栈序列里, 若  $a_1$  所在位置为  $k$ , 可以肯定  $a_1$  前的元素必须是  $a_i$ ,  $i \in \{2, 3, \dots, k-1\}$ 。同理,  $a_1$  后的元素肯定是  $a_i$ ,  $i \in \{k+1, k+2, \dots, n\}$ 。如果出现超出此范围的元素, 则出栈序列非法。由此也可以快速排除一个非法的出栈序列。不过, 这只是进出栈序列的一个必要非充分条件, 不能用来找出正确出栈序列。

### 7.5.3 递归编程

进出栈问题里，栈的状态可以由两个参数表示，特入栈元素（栈外）个数  $n$  和栈内元素个数  $k$ ， $n$  和  $k$  组成一个有序队  $(n, k)$ ，我们用符号  $f(n, k)$  表示进出栈操作序列。下一步可以进栈  $f(n-1, k+1)$ ，也可以是出栈  $f(n, k-1)$ 。显然  $k=0$  时不能出栈。

$f(n, 0)$  表示初始状态， $f(0, 0)$  表示操作结束。用此思路，我们可以用递归函数  $f(n, k)$  列出所有进出栈序列。

## 7.6 进出栈总结

1. 进出栈序列串，任意前缀子串满足  $+1$  个数不小于  $-1$  个数。
2. 出栈序列串，任意  $k$ ，下标比  $k$  小的并排  $a_k$  后的元素，其下标降序。
3. 出栈序列串， $a_1$  出栈时，左边是  $a_i, i \in \{1, 2, \dots, k-1\}$  子问题，右边是  $a_i, i \in \{k+1, k+2, \dots, n\}$  子问题。
4. 进出栈动态过程中，每次栈为空时，得到一个子问题，所有子问题合并是总问题。



# **Part V**

## **Programming**





## Chapter 8

# C Language

### 8.1 申明和定义

1. `extern` 申明 `declare` 一个变量，只说明存在某个数据类型的变量符号存在（插入到符号表里），而不分配空间。一般来说，在一个文件定义，在另一个要访问该变量的文件里申明（如头文件）。
2. 没有 `extern` 关键字时，一般就是变量定义。变量定义隐含了申明。定义时要分配内存空间。
3. 函数的声明不需要给出 `extern` 关键字。

### 8.2 作用域和生存周期

变量的属性有作用域和生存周期（`storage duration`）之分。从作用域来说有全局变量和局部变量。从生存周期来说有自动变量（`automatic variable`）和静态变量（`static`）。

全局变量在花括号外定义的，默认从定义处开始，到文件结束可见。要想真正实现“全局”的作用域，得在引用前申明变量（函数同理）。局部变量在花括号内部定义，只在花括号内可以被访问，也即局部作用域。花括号可以包裹函数体，也可以是包裹一小段块代码。同名局部变量覆盖 `override` 全局变量，此时要访问全局变量应用花括号和 `extern` 关键字。

生存周期和作用域紧密相联。自动变量是指进入作用域时分配内存，而离开时释放内存，变量被销毁。静态变量在进程运行期间一直存在，不受作用域影响。注意静态变量可以在花括号内（代码块或函数）定义！

自动变量和局部变量是同一个概念，只是强调的属性不同罢了，可以称为“局部自动变量”。

静态变量其实是全局变量的缩小版，作用域缩小，不能被 `extern` 申明引用，所以静态变量在花括号外或文件之外是不可见的。静态变量只在所定义花括号内（局部作用域）或花括号外所定义处至当前文件结尾（文件作用域）范围内有效。同时，静态变量又可以看成是局部自动变量的升级版，主要是生存周期变长，作用域（文件作用域）可能增大。静态变量是非常特殊的可以按作用域分成局部静态变量和文件静态变量。

全局变量  $\xleftrightarrow[\text{作用域增大}]{\text{作用域减小}}$  静态变量  $\xleftrightarrow[\text{生存周期变长}]{\text{生存周期变短}}$  局部自动变量

下面总结几个要点：

1. 鉴于作用域的重要性，我们一般先定义变量、函数，再引用。否则要先申明，才能引用。
2. 全局、静态变量没有显式初始化时，程序会默认初使化为 0。初始化（显式或隐式）只会进行一次，即使定义语句多次被进程触及。
3. 局部自动变量没有显式初始化时，初始值不确定，依赖编译器，因为 C 语言对此没有规定。局部自动变量定义时应初使化。
4. 提倡定义全局常量，尽量避免定义全局变量。如果实在需要，用 `static` 静态变量减小作用域。
5. `extern` 申明、引用其它文件的全局变量。最好放在头文件里。
6. C++ 对所有变量、对象全部要初始化！

## 8.3 编译内存布局

C 语言编译后虚拟内存分很多区域。我们这里说的是虚拟内存，假设机器上只有一个程序在运行，占有全部内存。实际中程序执行时所用的内存由操作系统分配在物理内存上：虚拟内存到物理内存的映射！

图 8.1 给出了虚拟内存的安排，简洁地说，就两部分，代码区和数据区。数据区按作用域、生命周期可分堆、栈、常量、全局、静态。对于全局和静态变量，按是否初始化，分为初始化和未初始化。常量肯定算作初始化的。其中的 `uninitialized data`，也即 BSS 表示 “`block started by symbol`”。总结来看是：堆，栈，常，代，全，静（初始化或未初使化）。

特别的，图中的最高地址处的命令行参数和环境变量也算作是栈，只不过是程序在加载时分配，相当于最早入栈的。其中命令行参数部分是传给 `main` 函数的 `argc` 和 `argv`。程序运行时还可以指定环境变量，而不是用系统默认的，如语言编码等。如 `LC=zh_CN.GB18030 ./a.out arg1`。

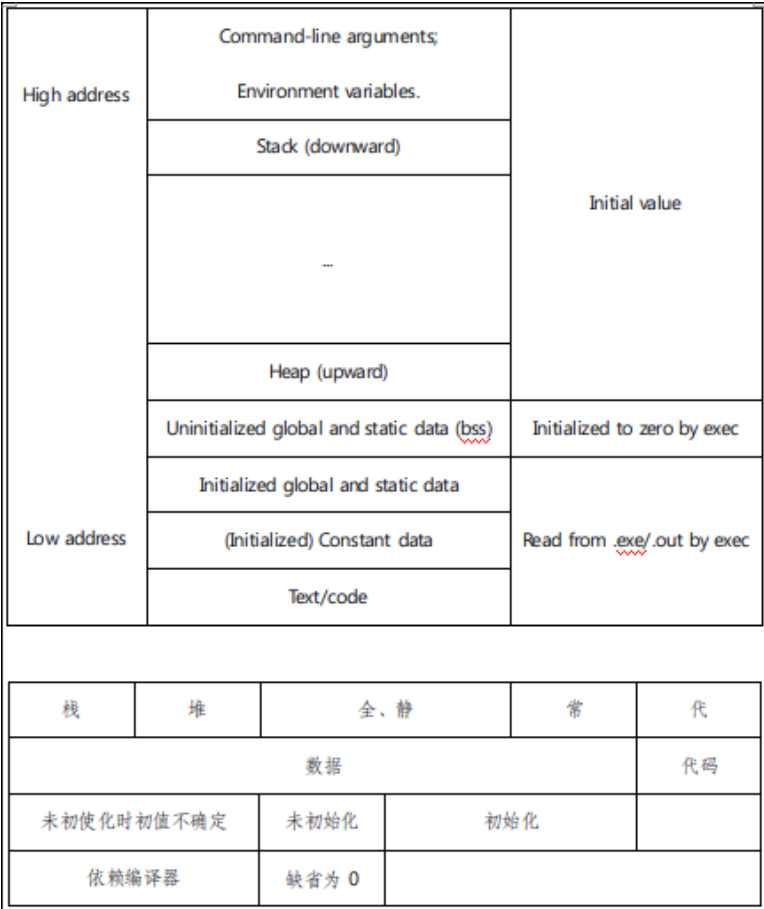


Figure 8.1: C Memory Layout

那么编译后的可执行文件如 `a.out` (Linux 的 ELF 格式) 或 `.exe` 是什么样的结构呢? 可执行文件里没有 `BSS` 段, 因为还没有值, 只有真正执行时才会有对应的内存。同理栈、堆也不存在, 这三部分只对进程有意义。只有常量, 初始化过的全局和静态变量。不过可执行文件还包含符号表, 段表, 库链接表, 调试等内容。

堆需手动分配、释放, 常用 `new/delete` 和 `malloc/free` 操作, 由低地址往高地址增长。堆的特点是可根据需求动态分配大小 `dynamic allocation on demand`。 `new/malloc` 在堆上分配的内存空间是全局有效的。如果不手动 `delete/free`, 则这些分配到的内存在程序执行过程中一直被其占据, 直到程序结束, 造成常说的“内存泄露”, 导致程序被卡死。所以通常在相同作用域内申请和释放(如函数内)。

栈是机器自动分配、释放。主要放函数参数, 局部变量(自动变量)等。栈上内存主只在函数局部才有效, 函数结束会立即被释放。相反, 栈的增长方向是由高往低, 这样是合理的, 如果往同一个方向增长, 不便于两个区间的管理。

常量区是指存放常量的地方, 如:

```
1 char * p = "hello, world";  
2 printf("%d", 3);
```

里 `p` 是自动变量, 分配在栈上。但是字符串“`hello, world`”和数字 3 这样的属于量放在常量区。常量区也算作初始化区的一部分。

代码区不用说就是存放放程序代码的地方。

全局变量区是指存放全局变量的地方。静态区是放静态变量的地方。通常这两个区是合并在一起的。而且这个“全静区”是可以分成未初始化区和已初始化区两部分。

程序执行时, 先把可执行文件(代码, 常量, 已初始化全、静变量)加载进内存低地址, 再分配未初始化的全、静变量空间。进一步环境变量和程序参数入栈。后面就是在运行时栈和堆的操作了。

说了这么多, 我们来看看 Linux ELF 可执行文件的结构:

```
1 zack@tux ~/workspace/c $ size a.out  
2  text  data  bss   dec   hex filename  
   2391   616   16   3023   bcf a.out
```

Listing 8.1: EFL Layout

这里 `text` 指代码段大小, `data` 是已初使化全、静变量和常量段大小, `bss` 段内数据全零。后面的 `dec` 和 `hex` 是前面列的和。

8.4 数组和链表

数组和链表都是线性存储结构，但实现方法不同，通常前者在栈上由机器自动分配，后者在堆上由程序动态分配。实现方法的不同表现出对应的优缺点。

结构 优缺点	数组	链表
访问 增删	通过下标随机访问 慢：要先扩容再移动元素	要遍历链表顺序查找 快：只需修改指针指向

Table 8.1: 数组和链表

如果是在中间插入删除（如有序结构），则数组更麻烦，移动的元素更多。



## **Chapter 9**

# **CPP**





## **Part VI**

# **Brainteaser**



## Chapter 10

# Brain Teasers

### 补全数列空缺

例 3 5 10 25 75 () 875. 观察此数列发现, 相邻两项的数间没有什么特别关系, 相乘相加都不是。这时应考虑相邻三、四项项的关系, 会发现  $3 \times 5 + 10 = 25$ , 类似地有  $5 \times 10 + 25 = 75$ . 因此空处的值应是  $10 \times 25 + 75 = 325$ . 再来一个例子, 5 6 16 28 60 (). 分析得出  $5 \times 2 + 6 = 16$ , 还有  $6 \times 2 + 16 = 28$ . 所以空处的值应是  $28 \times 2 + 60 = 116$ . 此两例考虑连续多个数之间的递推关系, 类似于 Fibonacci Sequence.

### 牛吃草

一块均匀生长的草地, 可供 27 头牛吃 6 周, 或 23 牛吃 9 周, 问多少牛可吃 18 周?

我们先设一头牛吃草的速度是  $a$ , 单位可以是  $\text{kg/w}$ , 则 6 周吃  $27 \times 6 \cdot a \text{ kg}$  的草。设草的生长速度是  $b$ , 单位可以是  $\text{kg/w}$ , 那么 6 周长成了  $6 \cdot b \text{ kg}$  的草。那么是不是  $27 \times 6 \cdot a = 6b$  呢?

不是! 因为牛吃草前, 草地上本来就有部分草, 设为  $c \text{ kg}$ , 则  $27 \times 6 \cdot a = 6b + c$ , 同理我们得出:

$$27 \times 6 \cdot a = 6 \cdot b + c$$

$$23 \times 9 \cdot a = 9 \cdot b + c$$

$$x \times 18 \cdot a = 18 \cdot b + c$$

通过前两个方程, 可得出  $b = 15 \cdot a$  和  $c = 72 \cdot a$ , 进而求出  $x = 19$ .

### 相向而行

甲乙两地相距 90 米, A, B 二人分别从甲乙同时出发, 在两地来回跑动。甲的速度是 3 m/s, 乙的是 2 m/s. 问 10 分钟内甲乙相遇几次?

第一次相遇时二人行使 90 米, 时间是  $90 \div (3 + 2) = 18\text{s}$ . 相遇后二人背向而行, 直至达到对端。此时二人又行使了 90 米, 用时 18s.

二人再次相向而行, 准备下一次相遇。此后一直重复此过程。可以发现相遇一次二人行使了 180 米, 花时 36s, 所以 10 分钟相遇次数是  $10 \times 60 \div 36 = 16\frac{2}{3}$ , 结果是分数, 那么是 16 次还是 17 次呢?

应是 17 次, 因为第次相遇时, 在 36s 的前半程, 所以分数部分应四舍五入!

上面这种方法不够明了。第一次相遇需 18s, 下一次相遇需 36s, 再下一次相遇需 36s, 此后每一次相遇都需 36s. 我们找相遇的时间点更合理, 得到一个等差数列:

$$t_1 = 18$$

$$t_2 = 18 + 36$$

...

$$t_n = 18 + (n - 1) \cdot 36$$

根据  $t_n = 10 \times 60$  可以算出  $n = 17\frac{1}{6}$ . 由此可知, 相遇 17 次, 因为剩下的  $1/6$  还没到第 17 次相遇时间点。

### 背向而行

有一 300 米圆跑道, 甲乙在同一地点背向而行, 速度分别是 3.5m/s 和 4m/s. 问第 10 次相遇时, 甲离出发地还有多远?

每相遇一次, 两人共行走了 300 米, 其中甲行走了  $\frac{3.5}{(3.5+4)} \times 300 = 140$  米, 而乙行走了 160 米。

那么相遇 10 次时, 甲累计走了  $10 \times 140 = 1400$  米, 而  $1400 \div 300 = 4 \cdots 200$ , 也就是说走了 4 圈后, 第 5 圈走了 200 米, 所以离出发点还剩 100 米。

### 交换变量

有两个整数  $a = 5$ ,  $b = 7$ , 不通过临时变量, 如何交换二者的值?

方法一: 异或 XOR. 异或的意思是, 让两个数的二进制位互相比对, 同为 0 或 1 时结果为 0, 有一个 1 一个 0 (相异) 时结果为 1. 所以任何数异或自己的结果是 0. 对应的 C 语言结果是:

```
1 a = a^b;  
  b = a^b;  
3 a = a^b;
```

Listing 10.1: Swap by XOR

方法二：加减法。

```
1 a = a + b;  
  b = a - b;  
3 a = a - b;  
  // Or  
5 a = (a + b) - (b = a)
```

Listing 10.2: Swap by Arithmetic Operations



## Chapter 11

# 行测图行推理

### 11.1 规律总结

首先分析图形外观，考察图形画法，得出图形里的分部要素，也可称为元素。

1. 边。三角形、四边形、五边形……
2. 曲线。
3. 圆。
4. 区。图形被分割成不同的封闭区域。
5. 交点。边上的交点。

找出各图形元素后，再分析其规律。

1. 行列。从行和列两个维度来分析规律。
2. 数量。每种元素的数量可能呈某种关系，如相等，差数列。还可以是行、列上数量关系。如边数，交点数，区域数等。
3. 笔画。是否是一笔画完。
4. 对称。如轴对称（竖轴、横轴），旋转对称（中心对称，**180度**），不对称。如英文字母的对称性。还可能数对称轴（相同的、不同的）的数量。
5. 旋转。图形整体依次顺时针、逆时针旋转一个角度。
6. 翻转。图形整体以某轴翻转 **180度**。如水平翻转。
7. 平移。某元素在图内平移一定距离。如小方格每次移顺时针移 **3个位置**。

8. 相对。不同元素在图内相对位置发生变化。相对位置可以是内部、外部，相交、相接、分离，也可以是直角处、圆弧处，还可以是在最长边、最短边处。
9. 加减。相邻图形或元素之间加减组合出新图。如第 1 个图形和第 2 个图形组合起来是第 3 个图形。常见的是九宫格里，行、列间图形是加减关系。有时甚至是先旋转再加减。
10. 类别。图形可以按其属性分类。如同为生活类用品，同属用腿、用手、手脚并用的运动项目。



**Part VII**

**Chemistry**



## Chapter 12

# 走进化学世界

化学就是研究物质及其变化，它不仅研究已经存在的物质，还要研究和创造自然界原本不存在的新物质。例如，半导体材料，电阻几乎为零的超导体，有记忆能力的新材料，等等。

化学在保证人类生存和提高生活质量上有很大帮助。例如：

- 化肥和农药，增加粮食产量。
- 化学合成药物，抑制细菌和病毒。
- 化学新能源、新材料。

近代化学突破源于两点：

1. 物质是由分子原子构成的，分子中原子的重新组合是化学变化的基础。
2. 元素周期表。

这两个突破使得化学研究有迹可寻。化学的科学定义：**化学是在分子、原子层上研究物质性质、组成、结构与变化规律的科学**。研究物质是指研究其性质、组成和结构等静态特征，而变化是化学反应，也即原子的重新组合，这是一个动态过程。

### 12.1 性质及变化

物质变化分两两种：

- 物理变化：没有生成新物质的变化，如物质形态发生变化。气态、液态、固态三态之间的相互转换的过程就是物理变化。
- 化学变化或叫化学反应：生成新物质的变化（原子重新组合），常表现为颜色变化，放出气体，生成沉淀等。化学变化还伴随物质能量变化，

如吸热、放热、发光等。

虽然物理变化也通常伴随能量转换，但主要借用外部能量。而化学变化和能量可以只来自物质本身，不需要外部能量辅助。

依据物质变化过程中表现出的性质，可有：

- 化学性质：物质在化学变化中表现出的性质。例如，铜在潮湿的空气中生成铜绿。
- 物理性质：物质不需要化学反应就表现出来的性质。物质颜色、状态、气味、硬度、熔点、沸点、密度等都是物理性质。例如，常态下，氧气是种无色、无味的气体。注意，物理性质可能是物理变化过程表现出的性质，如物质的三态。

外界条件改变时，物质性质也会随着变化，因此，描述物质性质时往往要注明条件。如，当温度升高时，固态冰变成液态水，再加温，水会沸腾。液体的沸点是物理性质，但受大气压强的影响。大气稀薄的地方，大气压强变小，这时水的沸点会降低，容易烧开。

### 12.1.1 大气压

由于大气压强是变化的，人们把  $1\text{atm} = 101\text{kPa} = 76\text{cm}$  水银柱重量 当作标准大气压强。大气压强是指大气对浸在它里面的物体产生的压强，也叫大气压或气压，可以用空气的重力或分子热运动来解释其产生机理，而且这两种解释是等价的。

那么如何用重力和分子热运动解释呢？在密封空间内，气压的产生应从微观上来解释。气体分子热运动时，撞击空间内壁，产生作用力（内壁也同时产生反作用力），这个作用力就是气体压力。没有内壁就不会产生气体压力！压强是单位面积上的压力，表示强度，不用考虑内壁的存在。

由于分子作用力各向同性，所以任一点的压强在不同方向上相同。

对于空气呢？空气分子同样作热运动，但是没有内壁，怎么产生压力呢？有地球引力！地球引力相当于上述密闭空间内壁的反作用力。所以空气中任意一点也有空气压力，进而也有压强。同理，空气压力也是各向同性的。

至于压强的数值计算，两种情况有所不同，关键是算出分子热运动的作用力。对于密闭空间，它是  $P = nTR/\text{volume}$ 。对于空气，因为重力和空气分子热运动产生的撞击力平衡，大气压强是大气施加于单位面积上的重力，即该地单位面积垂直向上延伸到大气层顶的空气柱的总重力。简单的数学公式是  $P = m \cdot g/\text{area}$ 。实际中要考虑不同高度处重力加速度  $g$  的不同。

## 12.2 化学以实验为基础

本节主要说下蜡烛燃烧问题。蜡烛燃烧时，蜡固体先熔化、气化，再与空气中氧气发生化学反应。

外焰是红白色，温度高；而内焰为红色且边缘是蓝色，温度低。焰心没有发生燃烧，所以不烧手。温度的高低主要由与氧气接触面积决定。外焰处氧气最多，燃烧最充分，温度最高。焰心处氧气已消耗殆尽，所以没有燃烧。

色温和温度是两个不同的概念。色温反应的是单个分子的能量，能量越高，颜色越偏蓝。温度是分子热运动的宏观测量，不仅要考虑单个分子能量，还应考虑分子总个数。在相同分子个数情况下，蓝焰温度肯定比黄色或红色高。

但在蜡烛燃烧时，外焰温度除了氧气多外，和热气体上升也有关系。另外，既然外焰燃烧更充分，为何不见蓝色。实际是有蓝色的，只是被遮住了。外焰处蜡蒸气非常多，被外焰加热后，变成白炽色，把燃烧的蓝色吸收了。所以外焰的高温不是因为黄色，而是蓝色（充分燃烧）和上升热气。

## 12.3 化学实验

先看几条注意事项：

- 手不接触药品，不尝药品，鼻孔不可太靠近瓶口（特别是气体）。
- 实验剩余药品不能放回原瓶，不能随意丢弃，不能带出实验室。要放入指定容器。
- 节约药品。未说明剂量时，液体一般取  $1 - 2\text{ mL}$ ，固体只需盖满试管底部即可。

