

基于英文txt文档的双端整齐排版问题

信息安全 0810618 余文清

一、“双端整齐排版”问题描述

在文本编辑器中，“整齐排版”的目标是将一个右边距不等的文本，例如下图：

```
Mr. Dursley was the director of a firm called Grunnings, which
made drills. He was a big, beefy man with hardly any neck, although he did have
a very large mustache. Mrs. Dursley was thin and blonde and had nearly
twice the usual amount of neck, which came in very useful as she spent so
much of her time craning over garden fences,
spying on the neighbors. The Dursleys had a small son called Dudley
and in their opinion there was no finer boy anywhere.
```

（以上文本节选自《哈利波特与魔法石》）

转化为一个右边距尽可能相等的文本，如下图：

```
Mr. Dursley was the director of a
firm called Grunnings, which made
drills. He was a big, beefy man with
hardly any neck, although he did have a
very large mustache. Mrs. Dursley was
thin and blonde and had nearly twice the
usual amount of neck, which came in very
useful as she spent so much of her time
craning over garden fences, spying on
the neighbors. The Dursleys had a small
son called Dudley and in their opinion
there was no finer boy anywhere.
```

精确的说，若给定一个文本最大行长度 m ，把第 i 行实际的长度 $l[i]$ 和 m 的差值称为这一行的“松弛”，（例如，在 $m=40$ 的情况下，上图第一行的松弛等于 3，第二行松弛等于 7），那么“整齐排版”的目标是得到一个排版方式，在该方式下：

1. 任意两个单词之间有且仅有一个空格
2. 对于任意的第 i 行，有 $l[i] \leq m$
3. 所有行的松弛之和达到最小。

（还有一点，无论是“整齐排版”还是“双端整齐排版”，都是基于所有字符等宽的前提）

然而，我感觉到“整齐排版”的结果并不令人十分满意。经过“整齐排版”的文本右端仍旧参差不齐。如果能达到下面的效果，那么排版的美观度将大大提升：

```
Mr. Dursley was the director of a
firm called Grunnings, which made
drills. He was a big, beefy man with
hardly any neck, although he did have a
very large mustache. Mrs. Dursley was
thin and blonde and had nearly twice the
usual amount of neck, which came in very
useful as she spent so much of her time
craning over garden fences, spying on
the neighbors. The Dursleys had a small
son called Dudley and in their opinion
there was no finer boy anywhere.
```

我把这样的排版效果，称为“双端整齐排版”。可以看出，“双端整齐排版”是在“整齐排版”的基础上，将“整齐排版”后每一行行末的松弛转移到了单词与单词的间隙中。值得考虑的是，如果单词的间隙过大，也会影响排版的美观程度。通过观察，单词之间的空格数在 1 至 2 个时比较合适。

“双端整齐排版”问题可以如下定义：“双端整齐排版”是对经过“整齐排版”的文本的每一行进行重新的整理。对于任意的第 i 行，用 $w[i]$ 表示第 i 行包含的单词个数，用 $s[i]$ 表示第 i 行的松弛。“双端整齐排版”的目标是将行末的 $s[i]$ 个空格插入到 $w[i]$ 个单词之间。需要注意的是以下两条规则：

1. 经过“整齐排版”的文本中，每两个单词之间已经存在一个空格，因此每两个单词之间至多只能再插入一个空格
2. 根据规则 1，每行至多能有 $w[i]-1$ 个空格被插入到 $w[i]$ 个单词之间。如果 $s[i] \geq w[i]$ ，那么将有 $s[i]-w[i]-1$ 个空格剩余在该行的行末。

二、 英文文本的排版规则

英文文本有众多的写作规范。为了简化问题的规模，本算法仅归纳一下几条主要的英文文本排版规则：

1. 首行缩进 4 个字符。
2. 英文标点可分为前缀标点，后缀标点和其他标点。前缀标点包括 ‘(’，‘[’，‘{’，后缀标点包括 ‘)’，‘]’，‘}’，‘,’，‘.’，‘!’，‘?’，‘:’，‘;’，‘=’。其他标点即除了前缀标点和后缀标点的其他标点。
3. 前缀标点不可以出现在一行的行末，后缀标点不可以出现在一行的行首。
4. 引号不允许嵌套。前引号相当于前缀标点，后引号相当于后缀标点。
5. 引号和它做包含的内容之间不能有空格，如 “hello world” 是合格的文本，而 “ hello world ” 则是不合格的文本。
6. 引号必须成对出现，有了前引号，就必须有后引号。
7. 任意一个单词不可被拆分到不同的两行。

“双端整齐排版”问题默认输入的英文文本文件符合上述 7 条规则。如果输入的文本违反了上述的某一规则，则可能导致结果不是最优，或出错。

三、 算法分析

本程序的目标是接受一个包含英文文本的 txt 文档，经过程序处理，产生一个符合“双端整齐排版”的输出文档 output.txt。

根据前面的介绍，要对一个英文文本完成“双端整齐排版”应当进行 3 个步骤。一是对文本文件进行符合英文排版规则的预处理。二是利用预处理得到的信息进行“整齐排版”。三是对“整齐排版”后的文本进行再处理，将其转化为“双端整齐排版”。下面，从以上三个方面进行分析。

1. 对英文文本的预处理

为了符合英文排版的几条规则。此处引入“词组”的概念。“词组”由单词以及与单词邻接的标点符号构成。在排版后的文本里，词组与词组间至少有一个空格，一个词组不能位于不同的两行。引入词组的好处是：将前缀标点与之后的单词分入同一个词组，可以保证前

缀标点不会位于一行的末尾；将后缀标点与前的单词分入同一个词组，可以保证后缀标点不会位于一行的行首。

预处理的目标是接受一个 txt 文档，对其中的字符进行分组，将结果保存至 words[] 数组。该数组记录每一个词组的长度，例如 words[i] 表示第 i 个词组的长度（以字节为单位）。

预处理前，首先对一个 txt 文档中可能出现的字符进行分类，在本程序中，将字符分为以下几类：

- (1) 前缀标点。在程序中，bool isPrefix(char c) 用于判断字符 c 是否是前缀标点
- (2) 后缀标点。在程序中，bool isSuffix(char c) 判断字符 c 是否是后缀标点
- (3) 引号
- (4) 数字和字母。库函数 int isalnum(char c) 判断字符 c 是否是数字或字母
- (5) 除了 (1) (2) (3) (4) 的非空白可打印字符，不妨称其为符号。
程序中，bool isSymbol(char c) 判断字符 c 是否是一个符号
- (6) 空白符。库函数 int isspace(char c) 判断字符 c 是否是一个空白字符。空白字符的 ascii 码从 0x09 到 0x0D 或 0x20。包括空格以及回车，换行等控制字符

在本程序中，完成预处理的函数是 int getGroup(int *words, ifstream &infile)。参数 1 是保存词组长度的数组，参数 2 是输入流对象。函数返回词组的总数。

初始时，words[i]=0，且为了在首行空出 4 个字符。words[1]=3；

函数定义一下几个变量：

```
int count=1; //记录词组的总数
char tChar;    //tChar 保存当前文件指针指向的字符
char oldChar=' '; //oldChar 记录当前指针的前一个字符
```

具体来说，预处理的算法如下：

从文件头扫描至文件尾

- (1) 取得当前文件指针指向字符，保存至 tChar。
infile.get(tChar)
- (2) tChar 是数字字母或符号。
如果 oldChar 是空白符、后缀标点或后引号，说明这是一个新词组的开始，那么
count++; words[count]++;
否则，words[count]++;
- (3) tChar 是一个前缀标点。
如果 oldChar 为空白符，后缀标点，普通字符，或后引号，则开始一个新词组
count++; words[count]++;
否则，words[count]++;
- (4) tChar 为一个后缀标点。
words[count]++;
- (5) tChar 是前引号。
转入 (2) 处理
- (6) tChar 是后引号。
转入 (3) 处理
- (7) tChar 是空白符，不做任何处理
- (8) 文件指针向下移动，更新 oldChar 为 tChar，转入 (1)

当文件指针到达文件尾后，预处理结束，返回 count

例如，如果文本内容是：

```
if a=1,b=2,c=3
what's the result of (a+b)+c?
```

那么，所得到的词组为

“if” “a=1,” “b=2,” “c=3” “what’s” “the” “result” “of” “(a+b)” “+c?”

words[1...11]中保存的每个词组的长度如下：3,2,4,4,3,6,3,6,2,5,3 （注意第一个 3 表示一个包含 3 个空格的词组）

2. “整齐排版”算法

本程序中用于进行“整齐排版”的函数为：

```
void bestDivide(int *words,int *lineNum,int groupNum,int m)
```

其中，words[]数组提供了每个词组的长度，groupNum 提供词组的总数，m 表示最大的行长度。排版的结果保存至 lineNum[]，该数组记录每个词组所在行的起始词组的下标。

为了方便描述该算法，定义如下几个变量：

设 $spaceLeft[i, j] = m - (j - i) - \sum_{k=i \text{ to } j} words[k]$ ，表示由 words[i...j]组成的一行行尾剩余的空格数，即前面所提的“松弛”。值得注意的是，如果计算出 $spaceLeft[i, j] < 0$ ，说明这一行无法容纳 words[i...j]，应当把 $spaceLeft[i, j]$ 置为 ∞

设 $costLine[j]$ 表示包含 words[1...j] 的文本的最小松弛之和。如果 words[1...j] 的最后一行为 words[i...j]，那么 $costLine[j] = costLine[i-1] + spaceLeft[i, j]$

因此， $costLine[j]$ 的计算方法如下：

当 $j=0$ 时， $costLine[j]=0$ ；

当 $j>0$ 时， $costLine[j] = \min(1 \leq i \leq j) \{costLine[i-1] + spaceLeft[i, j]\}$ 。

lineNum[j]记录包含 words[j]的那一行的起始词组的下标。当 $costLine[j]$ 被计算出之后，如果 $costLine[j] = costLine[k-1] + spaceLeft[k, j]$ ，则 $lineNum[j]=k$ 。

根据上面的定义，可以设计一个动态规划的算法如下：

```
for j=1 to n
  do costLine[j]=  $\infty$ 
  for i=1 to j
    if costLine[i-1]+spaceLeft[i,j]<costLine[j]
      then {
        costLine[j]=costLine[i-1]+spaceLeft[i,j];
        lineNum[j]=i;
      }
```

从时间复杂度上看，计算 $spaceLeft[i, j]$ 可以事先完成。那么该算法的时间复杂度是 $O(n^2)$ 。

从空间复杂度看，存储 $spaceLeft[i, j]$ 需要一个二维数组，该算法的空间复杂度也是 $O(n^2)$ 的。

如此高的空间复杂度，让这个算法几乎没有实际的用途。

例如处理总共有 78493 个词组的《Harry Potter and the Sorcerer's Stone》，如果 $spaceLeft[i][j]$ 是一个 int 型的二维数组， $costLine[j]$ 是 int 型的一维数组， $lineNum[j]$ 是一个 int 型的一维数组，

一个 int 变量相当于 4 个字节。

那么该算法至少需要 $4 \times (78493 \times 78493 + 78493 + 78493) = 24645232140$ bytes 的内存空间，相当于 23GB，以现在的电脑配置，个人 pc 机是无法处理《哈利波特与魔法石》的排版问题的。

实际上，无论从空间复杂度还是时间复杂度，该算法都可以进行优化。

注意到，对于行长度最大为 m 的情况，每一行最多能容纳 $\text{ceil}(m/2)$ 个词组，因为每个词组至少包含一个字符，每两个词组间至少要有个空格。另外，如果某一行由 $\text{words}[i..j]$ 组成，那么该行的词组总数为 $j-i+1$ 个。由此可见，当 $j-i+1 > \text{ceil}(m/2)$ 时， $\text{words}[i..j]$ 必然无法置于同一行，那么计算 $\text{spaceLeft}[i,j]$ 就没有意义，计算 $\text{costLine}[j] = \text{costLine}[i-1] + \text{spaceLeft}[i,j]$ 也没有意义。因此只需要计算 $j-i+1 \leq \text{ceil}(m/2)$ 的情况，计算 $\text{costLine}[j]$ 和 $\text{lineNum}[j]$ 的内层 for 循环可以从 $\max(1, j-\text{ceil}(m/2)+1)$ 迭代到 j 。通过此优化，可以把时间复杂度降低到 $O(n \times m)$ 。

空间复杂度上，没有必要设置记录松弛度的二维数组 $\text{spaceLeft}[i,j]$ 。通过观察可知，存在下面的递推关系： $\text{spaceLeft}[i,j] = \text{spaceLeft}[i+1,j] - \text{words}[i] - 1$ ；可以在计算 $\text{costLine}[j]$ 的内层循环中，利用 i 的递减，计算当前行的松弛度。因此，空间复杂度可以降低至 $O(n)$

经过优化的动态规划算法如下：

```
for j=1 to n
  do costLine[j]=∞
  spaceLeft=m-words[j];
  for i=j downto max(1, j-ceil(m/2)+1)
    if costLine[i-1]+spaceLeft<costLine[j]
      then{
        costLine[j]=costLine[i-1]+spaceLeft;
        lineNum[j]=i;
      }
  spaceLeft=spaceLeft-words[i]-1;
```

经过优化后的动态规划算法，只需要维护 $\text{costLine}[]$, $\text{lineNum}[]$ 两个一维数组。处理 78493 个词组的《Harry Potter and the Sorcerer's Stone》只需要 $4 \times (78493 + 78493) = 627944$ bytes 内存空间，相当于 613KB。约为优化前内存消耗的 $1/39247$ 。有效地节省了 20 多 GB 的内存。

3. “双端整齐排版”算法

经过“整齐排版”后，排版就步入尾声了。“双端整齐排版”实际上是在寻求一种策略，将行尾的空格转移到这一行的单词之间去。

一种简单的策略是将尾部的空格尽量地摆放到一行起始的几个词组之间。这样做方便快捷，但带来的问题是排版出的效果整体上看“左松右紧”，即靠近文档左边的词组间以两个空空间隔，而靠近文档右边的词组间以一个空空间隔。这样的排版效果并不好看。

我选择的策略是“随机策略”。即对于由 $\text{words}[i..j]$ 组成的某一行，设 spaceLeft 记录该行尾部的空格数（即松弛度）。对于 $\text{words}[i..j]$ 组成的一行，最多能将 $j-i-1$ 个空格插入到这些词组中，因此“随机策略”的方法是将个 $\min(\text{spaceLeft}, j-i-1)$ 个空格随机地插入到 $j-i-1$ 个空格当中。

本程序中进行“双端整齐排版”的函数为：

```
void printToFile(ifstream &infile, ofstream &outfile, int *words, int *lineNum, int groupNum, int m)
```

参数 1 为输入流对象

参数 2 为输出流对象

参数 3 为记录每一个词组长度的数组

参数 4 为“整齐排版”函数输出的结果数组，如前面介绍的一样，该数组存放每个词组所在行起始词组的下标

参数 5 为词组总数

参数 6 为最大行长度 m

为方便描述算法，定义如下的变量

$selectNum$ 记录需要插入的空格数，对于由 $words[i...j]$ 组成的一行， $selectNum = \min(spaceLeft, j-i-1)$

$remainNum$ 记录一行中还剩余的可被插入的空格数

则算法可以用下面的伪代码表示：

```
for i=1 to groupNum
    if lineNum[i] != lineNum[i-1]
    then{
        计算第 lineNum[i]行的 spaceLeft;
        j=i;
        while(lineNum[j] != lineNum[j+1])
            j++;
        selectNum= min(spaceLeft, j-i-1);
        remainNum=j-1-i;
    }
    else{
        向流对象 outfile 输出一个空格; //两个词组之间至少一个字符
        if(rand()%remainNum < selectNum) //是否再输出一个空格的概率:select/remain
        then{
            向流对象 outfile 输出一个空格
            selectNum--;
        }
        remainNum--;
    }
    从流对象 infile 中取得 words[i]个非空白可打印字符，
    将它们输出到 outfile
End for
```

四、 算法综合分析

从时间复杂度上看，预处理时间复杂度是 $O(n)$ ；“整齐排版”算法时间复杂度为 $O(n*m)$ ，“两端整齐排版”时间复杂度为 $O(n)$ ，因此总的时间复杂度为 $O(n*m)$

从空间复杂度看，预处理空间复杂度为 $O(n)$ ；“整齐排版”算法空间复杂度为 $O(n)$ ；“两端整齐排版”的空间复杂度为 $O(1)$ ，因此总的空间复杂度为 $O(n)$ 。

五、 程序输入和输出

根据提示，输入需要排版的 txt 文件名和排版后的最大行长度。

需要注意的是：

1. 如果输入的文件名中不包含文件路径，则程序在当前工作目录下寻找该文件
2. txt 文档需要以一个回车符作为结尾
3. 程序假定提供的输入文档符合前面所说的英文排版规则，如果输入的 txt 文档不符合英文排版规范，则本程序可能无法达到最优解或可能出现不可知错误

4. 程序无法处理 unicode，故输入的 txt 文档中尽量不要包含中文字符
5. 排版的结果将保存在当前工作目录的 output.txt 中，如果当前工作目录以存在名为 output.txt，程序会进行覆盖。
6. 请在具有等宽显示字符的文本编辑器内查看 output.txt.如 xp 下的文本编辑器。

六、排版效果演示

a) 原始 txt 文件

```
"You must come and stay this summer," said Ron, "both of you --
I'll send you an owl."

"Thanks," said Harry, "I'll need something to look forward
to." People jostled them as they moved forward toward the gateway
back to the Muggle world. Some of them called:

"Bye, Harry!"

"See you, Potter!"

"Still famous," said Ron, grinning at him.

"Not where I'm going, I promise you," said Harry.

He, Ron, and Hermione passed through the gateway together. "There
he is, Mom, there he is, look!"

It was Ginny Weasley, Ron's younger sister, but she wasn't
pointing at Ron.

"Harry Potter!" she squealed. "Look, Mom! I can see"

"Be quiet, Ginny, and it's rude to point."

Mrs. Weasley smiled down at them.

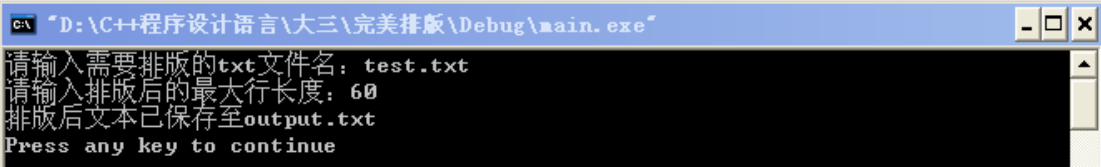
"Busy year?" she said.

"Very," said Harry. "Thanks for the fudge and the sweater,
Mrs. Weasley."

"Oh, it was nothing, dear."

"Ready, are you?"
```

b) 程序输入以及输出



```
C:\ "D:\C++程序设计语言\大三\完美排版\Debug\main.exe"
请输入需要排版的txt文件名: test.txt
请输入排版后的最大行长度: 60
排版后文本已保存至output.txt
Press any key to continue
```

c) 排版后 txt 文件

"You must come and stay this summer," said Ron, "both of you -- I'll send you an owl." "Thanks," said Harry, "I'll need something to look forward to." People jostled them as they moved forward toward the gateway back to the Muggle world. Some of them called: "Bye, Harry!" "See you, Potter!" "Still famous," said Ron, grinning at him. "Not where I'm going, I promise you," said Harry. He, Ron, and Hermione passed through the gateway together. "There he is, Mom, there he is, look!" It was Ginny Weasley, Ron's younger sister, but she wasn't pointing at Ron. "Harry Potter!" she squealed. "Look, Mom! I can see!" "Be quiet, Ginny, and it's rude to point." Mrs. Weasley smiled down at them. "Busy year?" she said. "Very," said Harry. "Thanks for the fudge and the sweater, Mrs. Weasley." "Oh, it was nothing, dear." "Ready, are you?"