第十九讲

Huffman Coding

薛浩

2023 年 6 月 8 日

www.stickmind.com

今日话题

话题 1：编程基础 初学编程的新手，一般应该熟练使用函数和库处理字符串相关的编程任务。

话题 2：抽象数据类型的使用 在尝试实现抽象数据类型之前，应该先熟练使用这些工具解决问题。

话题 3：递归和算法分析 递归是一种强有力的思想，一旦掌握就可以解决很多看起来非常难的问题。

话题 4：类和内存管理 使用 C++ 实现数据抽象之前，应先学习 C++ 的内存机制。

话题 5：常见数据结构 在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

话题 5：常见数据结构

在熟练使用抽象数据类型解决常见问题之后，学习如何实现它们是一件很自然的事情。

- 链表
- 动态数组
- 二叉堆
- 二叉搜索树

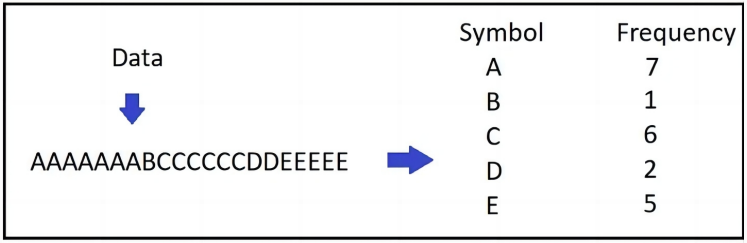


Figure 1: 数据结构和算法

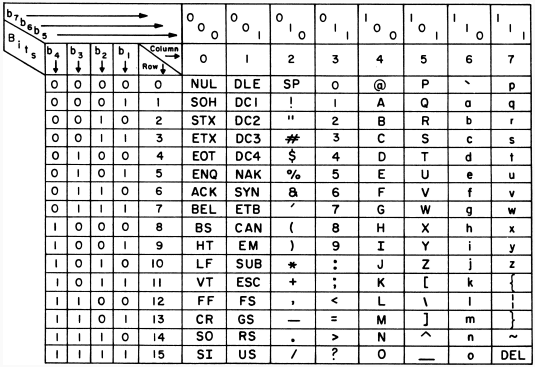
如何压缩 ASCII 文本?

目录

复习：ASCII 编码

ASCII 编码

ASCII（American Standard Code for Information Interchange）是计算机的一种字符编码标准。ASCII 由电报码发展而来，第一版标准发布于 1963 年，至今为止共定义了 128 个字符。在 C++ 中，每个字符由一个字节表示，类型为 char。



Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	´	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	¦
1 1 0 1	13	CR	GS	—	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

练习：文本文件存储

使用文本文件存储 1024 个字符，每个字符将占用 1 个字节，整个文本文件就需要 1kb 的存储空间。而将该文本文件压缩成 zip 格式后，其空间占用仅需 177 个字节。zip 格式是一种基于 Huffman 编码的压缩算法。

```
-rw-r--r-- 1 xuehao 197121    1024 Jun  7 14:43 test.txt
-rw-r--r-- 1 xuehao 197121     177 Jun  7 14:43 test.zip
```

使用 ASCII 编码字符串“HAPPY HIP HOP”将占用 13 个字节，共 104 个位：

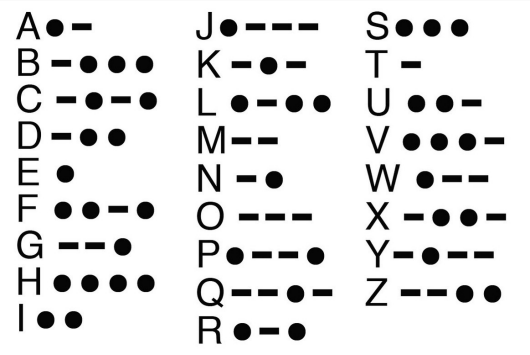
```
01101000 01100001 01110000 01110000 01111001 00100000 01101000
01101001 01110000 00100000 01101000 01101111 01110000
```

那么，如何使用更少的位来编码字符呢？

摩尔斯电码

摩尔斯电码

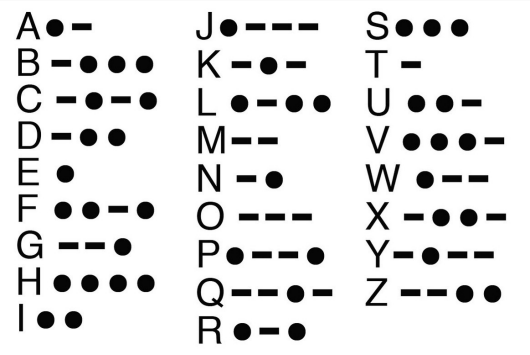
摩尔斯电码（Morse code）是一种时通时断的信号代码，通过不同的排列顺序来表达不同的英文字母、数字和标点符号。一般地，横线的持续时间是点的三倍。



练习：解密摩尔斯电码

根据下图的电码表，解码如下序列：

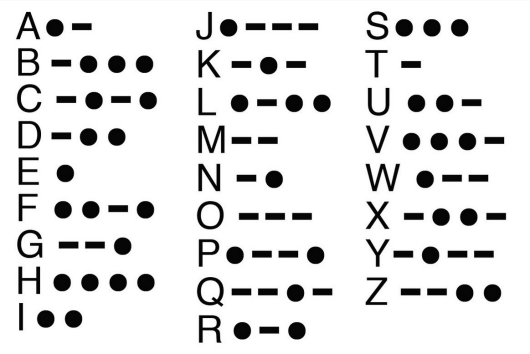
`-•-••--`



练习：加密摩尔斯电码

根据下图的电码表，加密如下字符串（空格为 7 个点的时间）：

```
HAPPY HIP HOP
```



定长与变长编码

定长编码

一种可选的编码方案是为有限的字符开发一种特殊的编码系统，由于每个编码长度相同，所以称为定长编码。

h	000
a	001
p	101
y	011
i	100
o	101
_	110

练习：定长编码

使用定长编码，编码如下字符串：

```
HAPPY HIP HOP
```

h	000
a	001
p	101
y	011
i	100
o	101
_	110

练习：定长编码

使用定长编码，编码如下字符串：

```
HAPPY HIP HOP
```

h	000
a	001
p	101
y	011
i	100
o	101
_	110

```
000 001 010 010 011 110 000 100 010 110 000 101 010
```

此时，只需要 39 个位，压缩率 38%

变长编码

那么，能否打破定长编码的限制，进一步压缩存储空间呢？

h	0
a	1
p	00
y	01
i	10
o	11
_	000

练习：变长编码

使用变长编码，编码如下字符串：

```
HAPPY HIP HOP
```

h	0
a	1
p	00
y	01
i	10
o	11
_	000

练习：变长编码

使用变长编码，编码如下字符串：

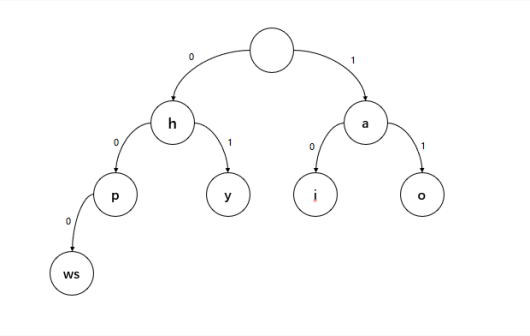
```
HAPPY HIP HOP
```

h	0
a	1
p	00
y	01
i	10
o	11
_	000

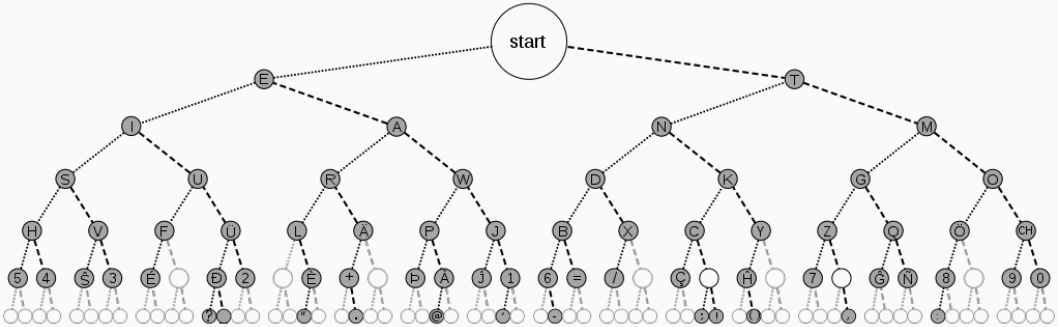
```
0 1 00 00 01 000 0 10 00 000 0 11 00
```

但这样的编码规则，遇到了和摩尔斯电码同样的问题。

变长编码树状表示



摩尔斯电码树状表示


start
E
T
I
A
N
M
S
U
R
W
D
K
G
O
H
V
F
U
L
A
P
J
B
X
C
Y
Z
Q
O
CH
5
4
S
3
É
Ð
2
È
+
Þ
À
Ĵ
1
6
=
/
Ç
;
!
H
7
G
N
8
9
0
?
"
.
@
'
-
(
,
.

Prefix-free 编码

Prefix-free 编码

Prefix-free **编码**是一种字符之间没有相同前缀的编码系统，既符合变长特性又避免了冲突。

h	01
a	1110
p	10
y	1111
i	000
o	001
_	110

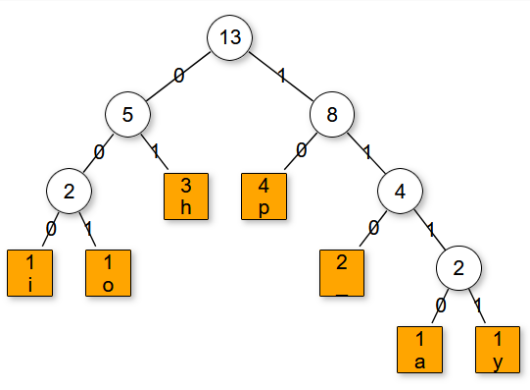
哈夫曼编码

哈夫曼算法是 Prefix-free 无前缀编码的一种实现算法。基于此算法构建出的哈夫曼编码是一种变长编码，其根据字符出现频率构造编码表：

- 概率高的字符使用较短的编码
- 概率低的字符使用较长的编码

哈夫曼编码也可以绘制成树状结构，与变长编码树不同，其所有的字符都存储在叶子节点上，而内部节点只是执行路径，所以这样的树也成为最优二叉树。

哈夫曼编码树



构造哈夫曼编码

构造哈夫曼编码

根据每个字符和其权重构成一个 Node，根据树的递归属性，每个 Node 都是一个独立的 Tree。当两个 Node 组合成一个新的 Tree 时，其根节点权重为叶子节点权重的总和。

- 创建一个优先级队列，用于存储中间生成的 Tree
- 从优先级队列中取出权重最小的两个 Tree
- 把两个 Tree 合并为一个 Tree，权重为其总和
- 将合并后的新 Tree 重新添加到优先级队列
- 重复步骤 2 到 4 的过程，直到生成最后的 Tree

哈夫曼编码

根据以上过程，我们得到了哈夫曼编码：

h	01
a	1110
p	10
y	1111
i	000
o	001
_	110

练习：哈夫曼编码

使用哈夫曼编码，编码如下字符串：

```
HAPPY HIP HOP
```

```
h   01
a   1110
p   10
y   1111
i   000
o   001
_   110
```

练习：哈夫曼编码

使用哈夫曼编码，编码如下字符串：

```
HAPPY HIP HOP
```

```
h   01
a   1110
p   10
y   1111
i   000
o   001
_   110
```

```
01 1110 10 10 1111 110 01 000 10 110 01 001 10
```

One more thing: **如何解码?**

根据上述过程，我们得到了如下的编码结果，大大减少了字符的存储空间。但是如何根据这串序列，解码出我们原有的文本信息呢?

```
01111010101111110010001011001001l0
```

One more thing: **如何解码?**

根据上述过程，我们得到了如下的编码结果，大大减少了字符的存储空间。但是如何根据这串序列，解码出我们原有的文本信息呢?

```
01111010101111110010001011001001l0
```

为了解码最终的结果，我们必须将编码以信息头的形式存储到上述序列的前缀中。

如何压缩 ASCII 文本?