图片结构和PNG压缩算法

iOS - 何立东

www.58.com

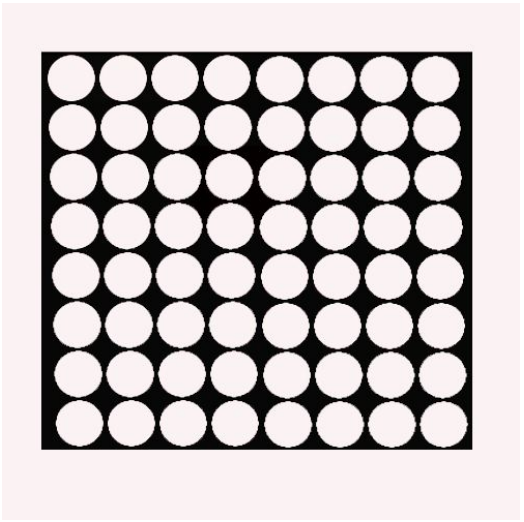
让 生 活 更 简 单

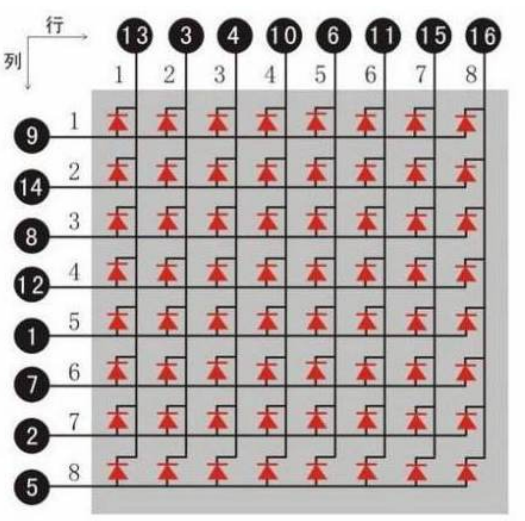
代码点亮一个灯

Int i = 1;

点阵屏的显示

1	2	3
4	5	6
7	8	9



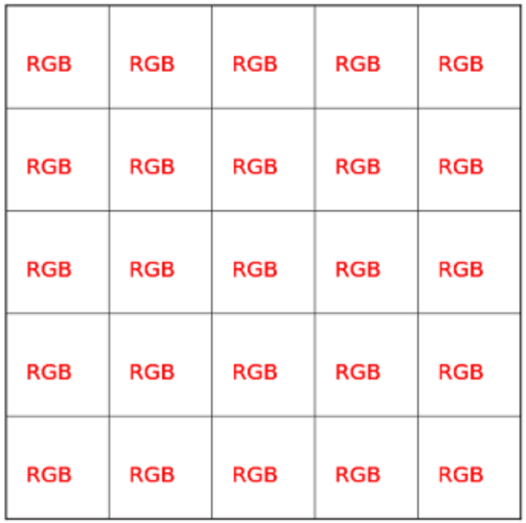


LCD的显示

RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB	RGB

BMP格式

- 内存中一维存储，数据自下往上存储
- BGR / BGRA排列
- bmp文件头(bmp file header)
- 位图信息头(bitmap information)
- 调色板(color palette)
- 位图数据(bitmap data)



BMP 文件头

文件头信息		
字段	大小(字节)	描述
bfType	2	一定为19778，其转化为十六进制为0x4d42，对应的字符串为BM
bfSize	4	文件大小
bfReserved1	2	保留位，为0
bfReserved2	2	保留位，为0
bfOffBits	4	从文件开始处到像素数据的偏移，也就是这两个结构体大小之和

BMP图片结构头

bmp图片结构头		
字段	大小(字节)	描述
biSize	4	此结构体的大小
biWidth	4	图像的宽
biHeight	4	图像的高
biPlanes	2	图像的帧数，一般为1
biBitCount	2	一像素所占的位数，一般是24
biCompression	4	图片压缩类型：一般为0
biSizeImage	4	像素数据所占大小，即上面结构体中文件大小减去偏移(bfSize-bfOffBits)
biXPelsPerMeter	4	水平分辨率：一般为0
biYPelsPerMeter	4	垂直分辨率：一般为0
biClrUsed	4	使用调色板索引数：一般为0
biClrImportant	4	对图像有重要影响的索引数：一般为0

PNG 简介

1. 使用彩色查找表或者叫做调色板可支持256种颜色的彩色图像。
2. 流式读/写性能(streamability)：图像文件格式允许连续读出和写入图像数据，这个特性很适合于在通信过程中生成和显示图像。
3. 逐次逼近显示(progressive display)：这种特性可使在通信链路上传输图像文件的同时就在终端上显示图像，把整个轮廓显示出来之后逐步显示图像的细节，也就是先用低分辨率显示图像，然后逐步提高它的分辨率。
4. 透明性(transparency)：这个性能可使图像中某些部分不显示出来，用来创建一些有特色的图像。
5. 辅助信息(ancillary information)：这个特性可用来在图像文件中存储一些文本注释信息。
6. 独立于计算机软硬件环境。
7. 使用无损压缩。
8. 每个像素为48位的真彩色图像。
9. 每个像素为16位的灰度图像。
10. 可为灰度图和真彩色图添加α通道。
11. 添加图像的γ信息。
12. 使用循环冗余码(**c**yclic **r**edundancy **c**ode，CRC)检测损害的文件。
13. 加快图像显示的逐次逼近显示方式。
14. 标准的读/写工具包。
15. 可在一个文件中存储多幅图像。

PNG文件结构

PNG文件格式中的数据块

数据块符号	数据块名称	多数据块	可选否	位置限制
IHDR	文件头数据块	否	否	第一块
cHRM	基色和白色点数据块	否	是	在PLTE和IDAT之前
gAMA	图像γ数据块	否	是	在PLTE和IDAT之前
sBIT	样本有效位数据块	否	是	在PLTE和IDAT之前
PLTE	调色板数据块	否	是	在IDAT之前
bKGD	背景颜色数据块	否	是	在PLTE之后IDAT之前
hIST	图像直方图数据块	否	是	在PLTE之后IDAT之前
tRNS	图像透明数据块	否	是	在PLTE之后IDAT之前
oFFs	(专用公共数据块)	否	是	在IDAT之前
pHYs	物理像素尺寸数据块	否	是	在IDAT之前
sCAL	(专用公共数据块)	否	是	在IDAT之前
IDAT	图像数据块	是	否	与其他IDAT连续
tIME	图像最后修改时间数据块	否	是	无限制
tEXt	文本信息数据块	是	是	无限制
zTXt	压缩文本数据块	是	是	无限制
fRAc	(专用公共数据块)	是	是	无限制
gIFg	(专用公共数据块)	是	是	无限制
gIFt	(专用公共数据块)	是	是	无限制
gIFx	(专用公共数据块)	是	是	无限制
IEND	图像结束数据	否	否	最后一个数据块

PNG 每个数据块由4部分组成

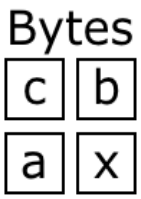
名称	字节数	说明
Length (长度)	4字节	指定数据块中数据域的长度，其长度不超过($2^{31}-1$)字节
Chunk Type Code (数据块类型码)	4字节	数据块类型码由ASCII字母(A-Z和a-z)组成
Chunk Data (数据块数据)	可变长度	存储按照Chunk Type Code指定的数据
CRC (循环冗余检测)	4字节	存储用来检测是否有错误的循环冗余码

PNG的IHDR

域的名称	字节数	说明
Width	4 bytes	图像宽度，以像素为单位
Height	4 bytes	图像高度，以像素为单位
Bit depth	1 byte	图像深度： 索引彩色图像：1，2，4或8 灰度图像：1，2，4，8或16 真彩色图像：8或16
ColorType	1 byte	颜色类型： 0：灰度图像, 1，2，4，8或16 2：真彩色图像，8或16 3：索引彩色图像，1，2，4或8 4：带α通道数据的灰度图像，8或16 6：带α通道数据的真彩色图像，8或16
Compression method	1 byte	压缩方法(LZ77派生算法)
Filter method	1 byte	滤波器方法
Interlace method	1 byte	隔行扫描方法： 0：非隔行扫描 1： Adam7(由Adam M. Costello开发的7遍隔行扫描方法)

过滤算法（5种）

- 不过滤
- X-A
- X-B
- X-(A+B)/2(又称平均值)
- Paeth推断



NONE

52	55	61	66	70	→	52	55	61	66	70
63	59	55	90	109		63	59	55	90	109

X-A

52	55	61	66	70	→	52	55	61	66	70
63	59	55	90	109		63	-4	-4	35	19

X-B

52	55	61	66	70	→	52	55	61	66	70
63	59	55	90	109		63	4	-6	24	39

Average

52	55	61	66	70	→	52	55	61	66	70
63	59	55	90	109		63	0	-5	30	29

Paeth

52	55	61	66	70	→	52	55	61	66	70
63	59	55	90	109		63	-4	-4	35	19

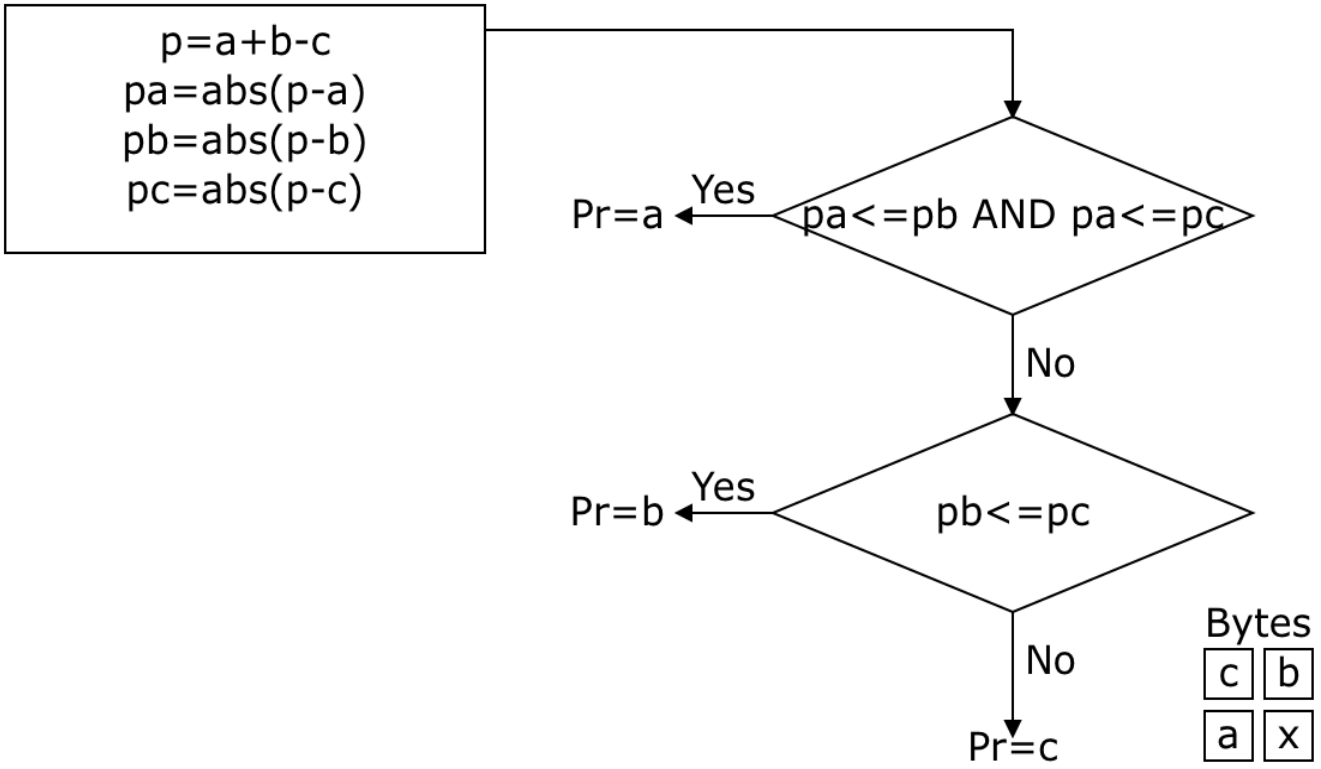
Paeth

p = a + b − c
pa = abs(p - a)
pb = abs(p - b)
pc = abs(p - c)

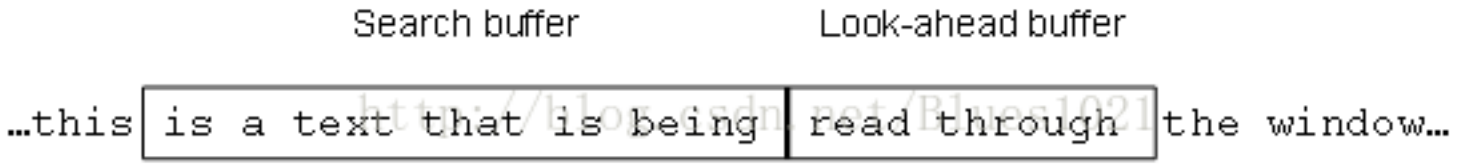
if pa <= pb and pa <= pc
then Pr = a

else if pb <= pc
then Pr = b

else Pr = c return Pr



LZ77算法压缩原理



Encoding of the string:

output tuple: (offset, length, symbol)

abracadabrad

7	6	5	4	3	2	1							output
							a	b	r	a	c	ada…	(0,0,a)
						a	b	r	a	c	a	dab…	(0,0,b)
					a	b	r	a	c	a	d	abr…	(0,0,r)
				a	b	r	a	c	a	d	a	bra…	(3,1,c)
		a	b	r	a	c	a	d	a	b	r	ad	(2,1,d)
a	b	r	a	c	a	d	a	b	r	a	d		(7,4,d)
…ac	a	d	a	b	r	a	d						

Search buffer | Look-ahead buffer

12 characters compressed into 6 tuples

Compression rate: (12*8)/(6*(5+2+3))=96/60=1,6=60%.

LZ77压缩解压原理

input		7	6	5	4	3	2	1
(0,0,a)								a
(0,0,b)							a	b
(0,0,r)						a	b	r
(3,1,c)				a	b	r	a	c
(2,1,d)		a	b	r	a	c	a	d
(7,4,d)	abrac	a	d	a	b	r	a	d


```
for each token (offset, length, symbol)
    if offset = 0 then
        print symbol;
    else
        go reverse in previous output by offset characters and copy
        character wise for length symbols;
        print symbol;
    fi
next
```

Huffman编码的基本原理

- 字符在文件中有定长的编码。
- 根据符号在文件中出现的频率，对这些符号重新编码。
- 出现次数非常多的，用较少的位来表示，
- 出现次数非常少的，我们用较多的位来表示。

10万个字符的文件，文件仅出现abcdef

	a	b	c	d	e	f
频率(千次)	45	13	12	16	9	5
定长编码	000	001	010	011	100	101
变长编码	0	101	100	111	1101	1100

定长编码，文件长度：3 * 100000 = 300000

变长编码，文件长度(45 * 1 + 13 * 3 + 12 * 3 + 16 * 3 + 9 * 4 + 5 * 4) * 1000 = 224000

压缩率：1 – 224000 / 300000 = 0.253

思考

1、PNG图片为什么需要滤波？

2、对PNG图片二次压缩

参考文献

PNG文件格式白皮书 ：
https://www.w3.org/TR/2003/REC-PNG-20031110/

算法导论16章1.63： Huffman编码

Thank you!