问题描述

问题描述：在计算机中，常用像素点的灰度值序列$\{p_1, p_2, \cdots\cdots p_n\}$表示图像。其中整数$p_i$，$1<=i<=n$，表示像素点i的灰度值。

通常灰度值的范围是0~255。因此最多需要8位（2^8）表示一个像素。若n个灰度序列，则占用空间为8n。

问题分析

分析：并非所有的像素均需要用二进制8位进行存储，因为有的灰度值并没有达到255这么大，所以是否可以采用算法进行改进？

- 23，17，28，23，27，29，30，18，28，27，39，48，129 ，139，178，220，23，9，183，133，19，299....。可以将 23，17，28，23，27，29，30，18，28，27看成一组，这一组都用5个bit位（ 2^5 ）就可以存储；
- {6,5,7,5,245,180,28,28,19, 22, 25,20}这是一组灰度值序列,共12个数字，传统的存储方式：12*8=96位来表示。

问题分析

若分为三组：

第一组4个数，最大是7所以用3位表示；

第二组2个数，最大是245所以用8位表示；

第三组6个数，最大是28所以用5位表示；

得到了最后的位数结果为：4*3+2*8+6*5+11*3=91。

问题分析

灰度值序列 P={10, 12, 15, 255, 1, 2, 1, 1, 2, 2, 1, 1}。

- 分法1：$S1$= {10, 12, 15}，$S2$={255}，$S3$={1, 2, 1, 1, 2, 2, 1, 1}。
- 分法2：$S1$={10, 12, 15, 255, 1, 2, 1, 1, 2, 2, 1, 1}。
- 分法3：分成12组，每组一个数。

存储空间

- 分法1：11×3+4×3+ 8×1+2×8 = 69
- 分法2：11×1+8×12 =107
- 分法3：11×12+4×3+8×1+1×5+2×3=163

问题分析

压缩的原理就是把序列{p_1, p_2, ······p_n}进行设断点，将其分割成一段一段的，同一段的像素所占的位数相同。*分段的过程就是要找出断点，让一段里面的像素的最大灰度值比较小，那么这一段像素(本来需要8位)就可以用较少的位(比如5位)来表示，从而减少存储空间。*

每段像素存储需要的信息：

1. 每段像素包含的像素的个数；（要给个约束，一般设置不超过256）

2. 该段像素内，最大像素所占的位数；

问题分析

图像压缩问题就是要确定像素序列$\{p_1, p_2, \cdots\cdots p_n\}$的最优分段，使得依此分段所需的存储空间最小。

建模：将n个像素的序列$\{p_1, p_2, \cdots\cdots, p_n\}$，分成m段$s_1, s_2, \cdots\cdots, s_m$，同一段的像素所占位数相同。

S1	S2	S3	S4	----------	Sm

问题分析

设第t段，有l[t]个像素，每个占用b[t]位。

段头：记录l[t](8位)和b[t](3位)需要11位。

实际上就是开辟存储空间，记录像素个数和每段最大像素所占位数。

具体像素信息：像素个数*最大像素所占位数；

段头　具体像素信息

段头：像素个数，约束不超过256个，采用最多8位就可存储；每个像素最多8位存储，段头用3位表示。

总位数 $11m+\sum_{i=1}^{m} l[\mathrm{i}]b[\mathrm{i}]$

每一段 $\mathrm{s}_i=11+l[\mathrm{i}]b[\mathrm{i}]$

一、最优子结构性质

设$l[i],b[i],1<=i<=m$是$\{p_1,p_2,\ldots\ldots p_n\}$的一个最优分段，则$l[1],b[1]$是$\{p_1,\ldots\ldots,p_{l[1]}\}$的一个最优分段，且$l[i],b[i],2<=i<=m$是$\{p_{l[1]+1},\ldots\ldots,p_n\}$的一个最优分段。

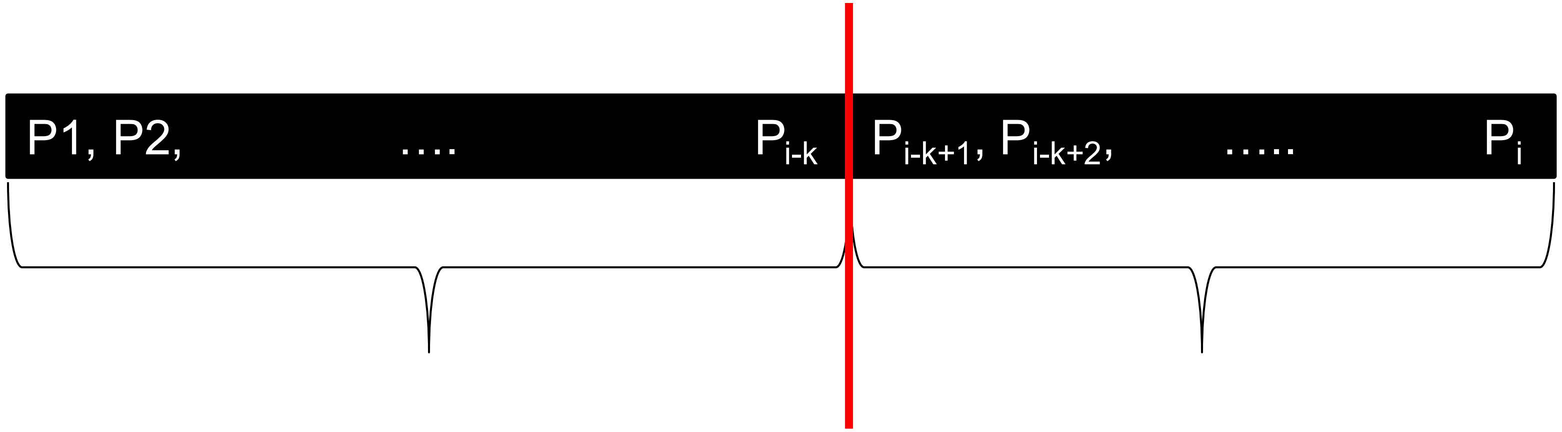
二、递归计算最优值

设s[i], 1<=i<=n是像素序列{p1, p1, ……pi}的最优分段所需的存储位数，则s[i]为前i-k个的存储位数加上后k个的存储空间。由最优子结构性质可得

$$s[i]=\min_{1\le k\le \min\{i,256\}}\{s[i-k]+k\times bmax(i-k+1,i)+11\}$$

$$bmax(i-k+1,i)=\left\lceil \log(\max_{i\le k\le j}\{p_k\}+1)\right\rceil$$

K表示断开位置



伪码

```
Compress
Lmax ←  …  [0]←0
for i ←1 to n do
    b[i] ←length(P[i])
    bmax ←
    S[i]←S[
    l[i] ←1
    for j←2 to min{i, Lmax} do
        if bmax < b[i–j+1]
        then bmax ← b[i–j+1]
        if S[i] > S[i–j] + j *bmax
        then S[i] ← S[i–j] + j *bmax
            l[i]←j
S[i]← S[i] + header
```

每次都将新扫描得元素设为bmax

子问题后

再次扫描，每段里得元素，若最大得小于之前得元素则取之前得元素

最后段长 j

找到更好分段

$P = \langle 10, 12, 15, 255, 1, 2 \rangle$.

$S[1]=15$, $S[2]=19$, $S[3]=23$, $S[4]=42$, $S[5]=50$

$l[1]=1$, $l[2]=2$, $l[3]=3$, $l[4]=1$, $l[5]=2$

10	12	15	255	1	2

$S[5]=50$ 1×2+11 63

10	12	15	255	1	2

$S[4]=42$ 2×2+11 57

10	12	15	255	1	2

$S[3]=23$ 3×8+11 58

10	12	15	255	1	2

$S[2]=19$ 4×8+11 62

10	12	15	255	1	2

$S[1]=15$ 5×8+11 66

10	12	15	255	1	2

6×8+11 59

三、构造最优解

追踪解

算法 Traceback (n , l)

输入：数组 l

输出：数组 C

1. $j \leftarrow 1$ // j 为正在追踪的段数
2. while $n \neq 0$ do
3. $\quad C[j] \leftarrow l[n]$
4. $\quad n \leftarrow n - l[n]$
5. $\quad j \leftarrow j + 1$

第j段长度

$C[j]$：从后向前追踪的第 j段的长度

时间复杂度：$O(n)$