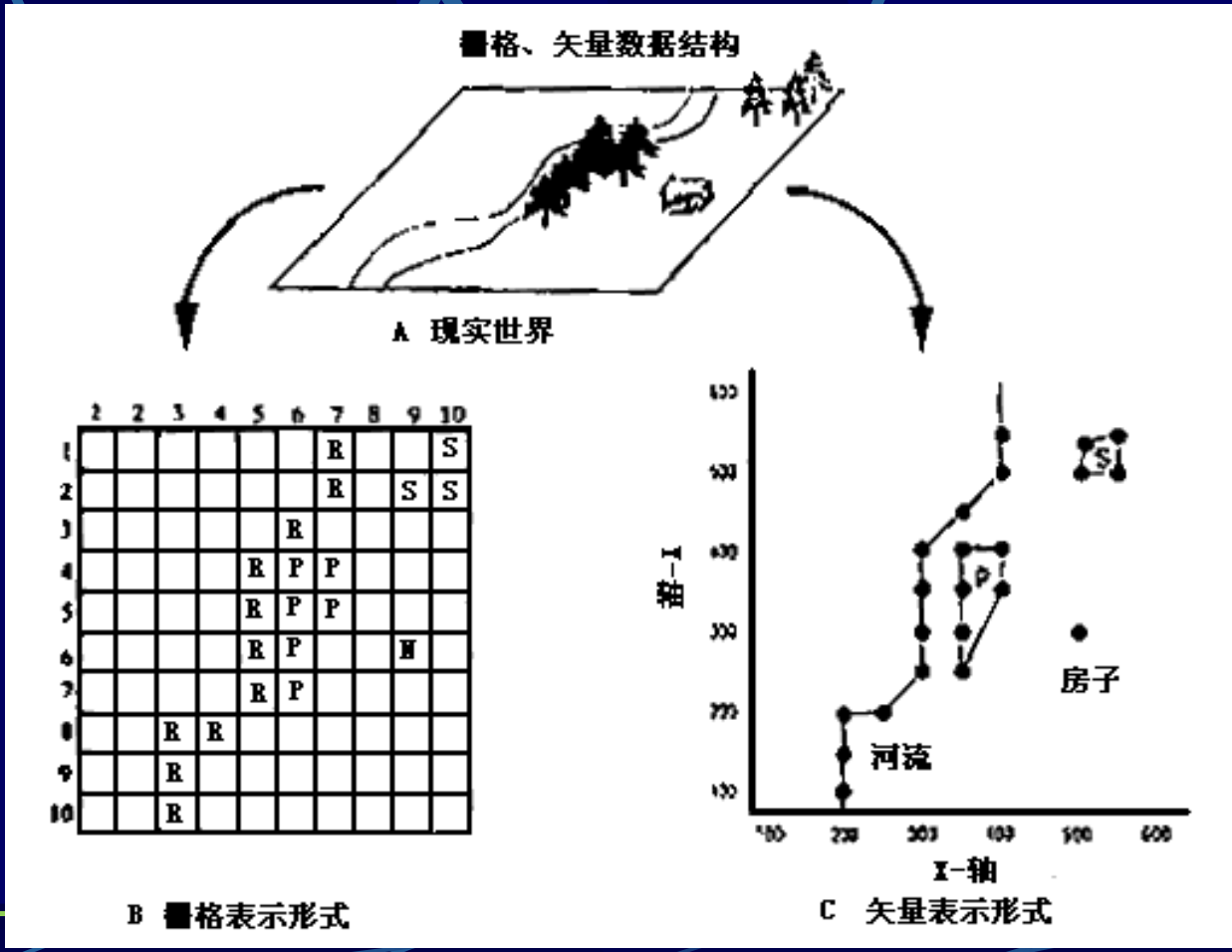
空间数据压缩算法

GIS的内部数据结构—矢量结构和栅格结构

- 内部数据结构：描述地理实体的数据本身的组织方法。
- 内部数据结构基本上可分为两大类：矢量结构和栅格结构（也可以称为矢量模型和栅格模型）（右图所示）。两类结构都可用来描述地理实体的点、线、面三种基本类型。



- 空间数据编码是空间数据结构的实现，即将根据地理信息系统的目的和任务所搜集的、经过审核了的地形图、专题地图和遥感影像等资料按特定的数据结构转换为适合于计算机存储和处理的数据的过程。由于地理信息系统数据量极大，一般采用压缩数据的编码方式以减少数据冗余。

一、栅格数据结构及其压缩

- 栅格结构是最简单最直接的空间数据结构，是指将地球表面划分为大小均匀紧密相邻的网格阵列，每个网格作为一个象元或象素由行、列定义，并包含一个代码表示该象素的属性类型或量值，或仅仅包括指向其属性记录的指针。

在栅格结构中：

- 点用一个栅格单元表示；
- 线状地物沿线走向的一组相邻栅格单元表示，每个栅格单元最多只有两个相邻单元在线上；
- 面或区域用记有区域属性的相邻栅格单元的集合表示，每个栅格单元可有多于两个的相邻单元同属一个区域。

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

（a）点

0	0	0	0	0	0	0	0
0	0	0	6	0	0	0	0
0	6	6	0	6	0	0	0
0	0	0	0	0	6	0	0
0	0	0	0	0	6	0	0
0	0	0	0	0	6	0	0
0	0	0	0	0	0	6	0
0	0	0	0	0	0	0	0

（b）线

0	4	4	7	7	7	7	7
4	4	4	4	4	7	7	7
4	4	4	4	8	8	7	7
0	0	4	8	8	8	7	7
0	0	8	8	8	8	7	8
0	0	0	8	8	8	8	8
0	0	0	0	8	8	8	8
0	0	0	0	0	8	8	8

（c）面

点、线、区域的格网

栅格数据的获取

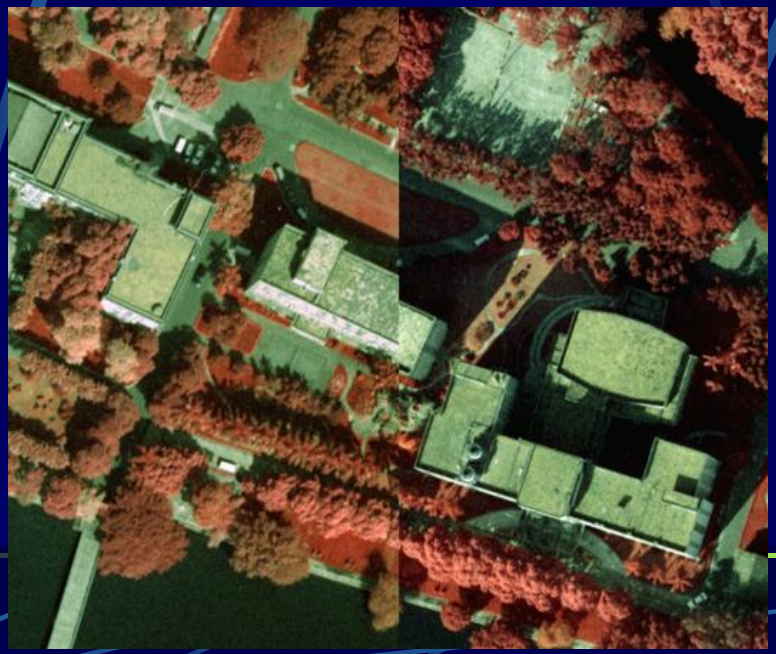
1. 遥感方法获取(航天与航空);
2. 图片扫描获取(纸介质的地图等扫描);
3. 矢量数据转换而来;
4. 由平面上行距,列距固定的点抽样而来

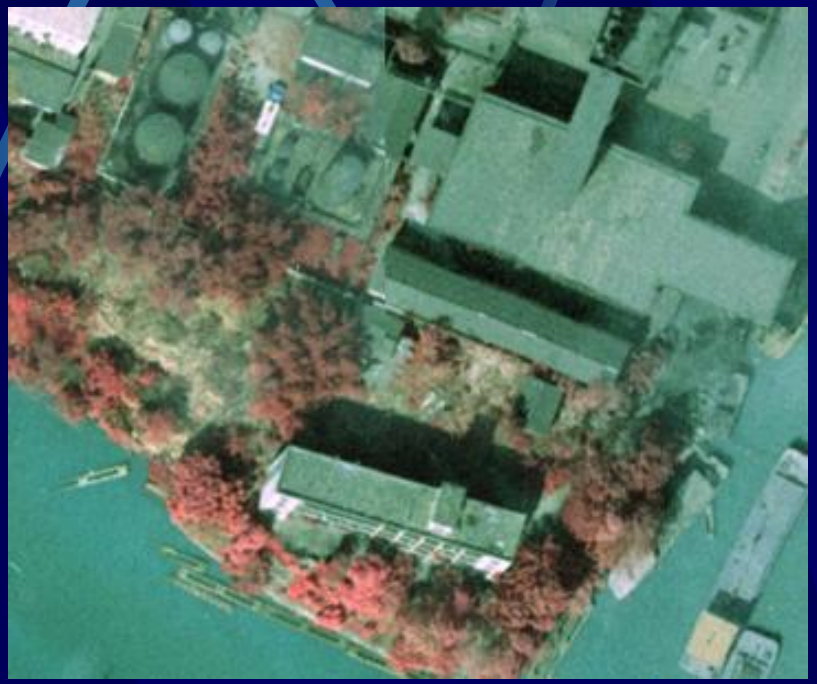
主要包括: (1)中心归属法 (2)长度占优法
(3)面积占优法

栅格数据（1）

栅格数据（2）

栅格数据（3）

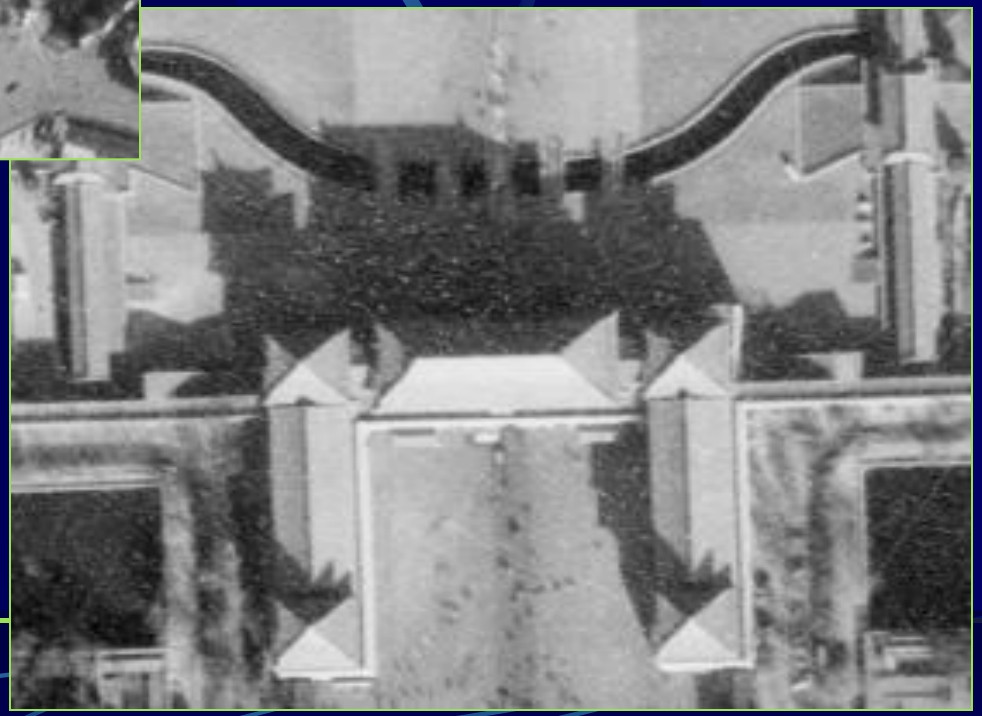




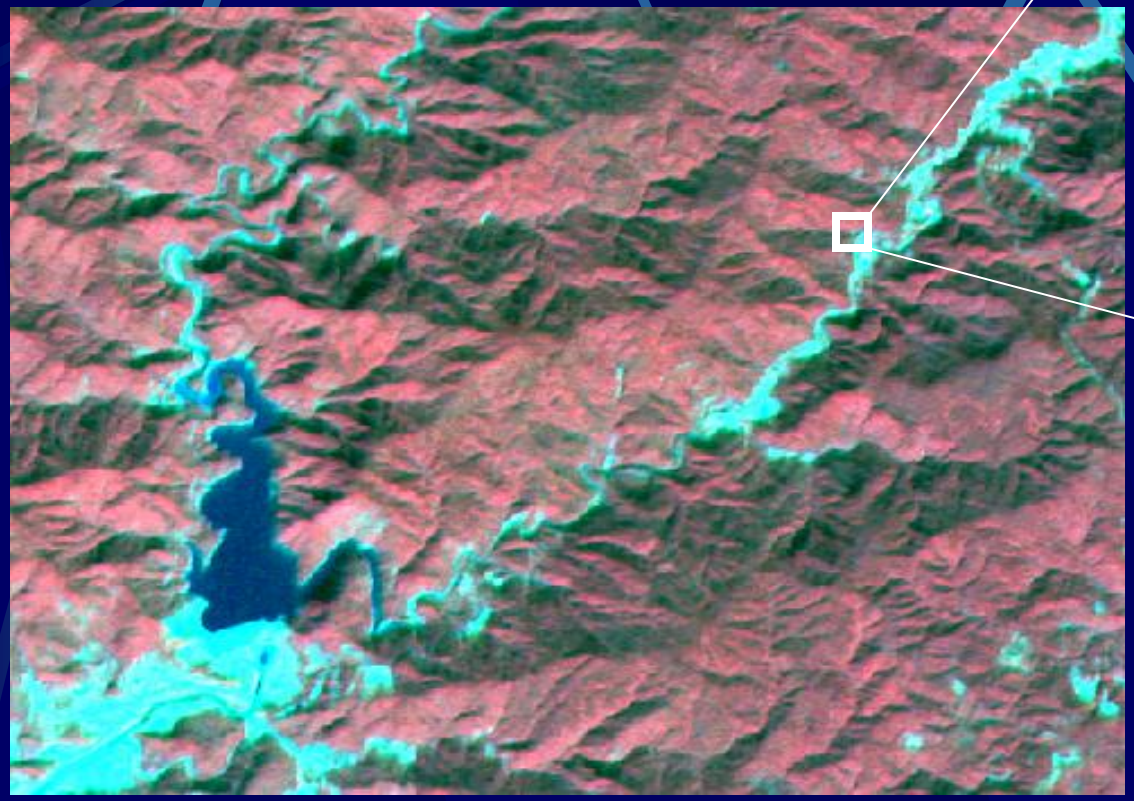
栅格数据（4）

上海东方明珠电视塔

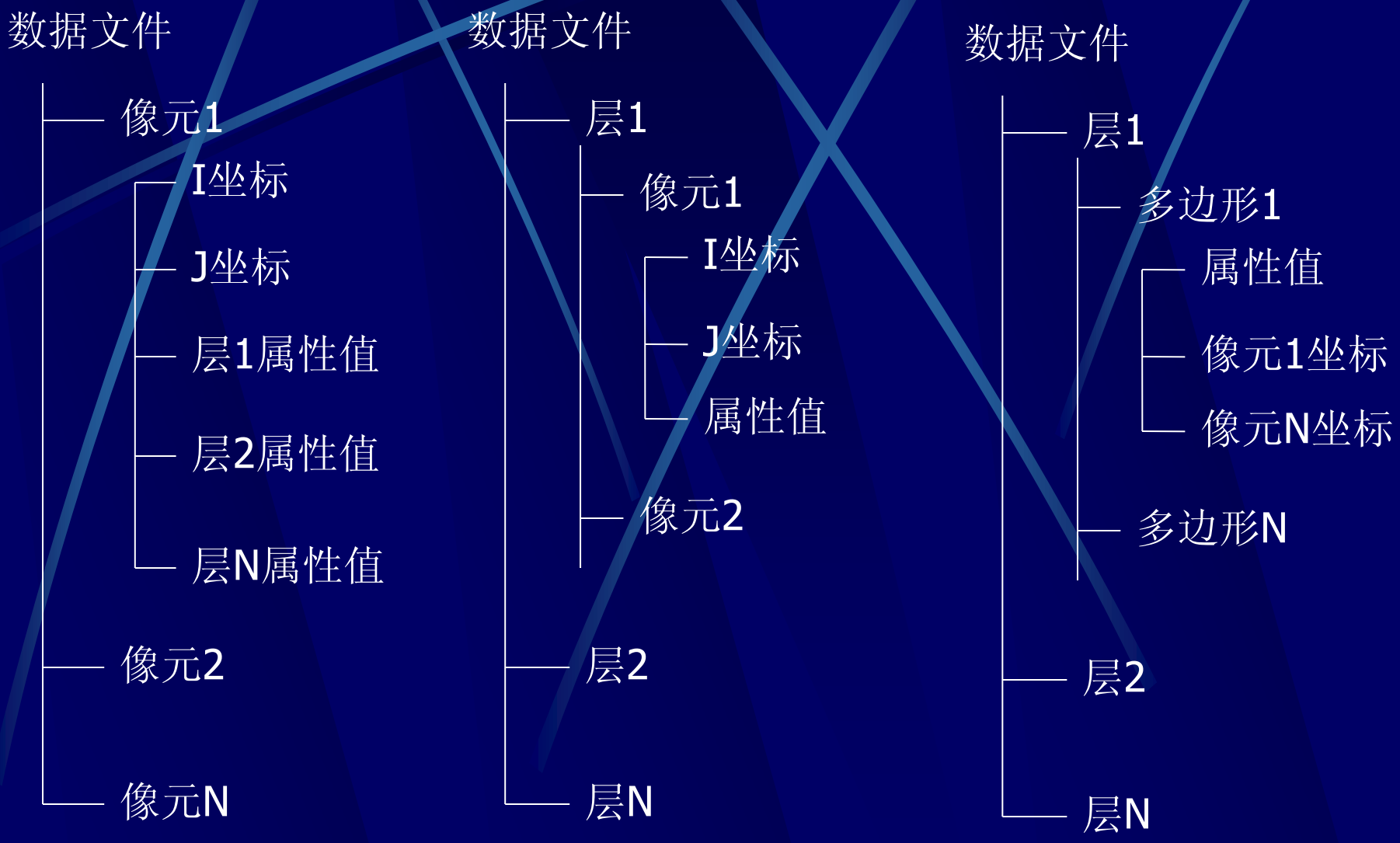
故宫



栅格数据结构



栅格数据的组织



节省空间　　形式简单　　方便制图

栅格数据的压缩算法

- 栅格数据文件记录有3个基本方式：基于像元、基于层和基于面域。这三种方式都离不开对像元坐标和属性的记录。因此基于栅格的空间数据压缩的实质是研究栅格数据的编码，通过编码尽量减少像元数量的存储。

其方法有三大类：（1）从减少记录像元的数量入手；
（2）从减少像元的记录信息量入手；
（3）前两者的结合。

栅格数据的压缩分为无损压缩技术和有损压缩技术。

（1）无损压缩技术是指压缩过程中没有任何信息损失，通过解码操作可以完全恢复原来的信息；

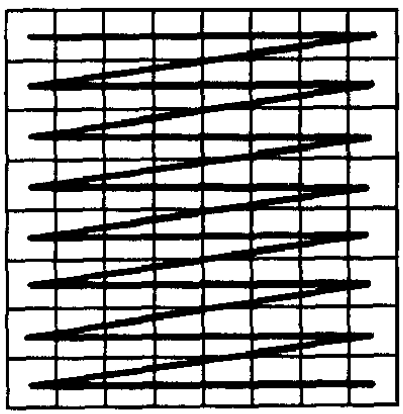
（2）信息有损压缩是指为了提高压缩效率，最大限度地压缩数据，在压缩过程中损失一部分相对不太重要的信息，解码时这部分难以恢复。

栅格数据的压缩

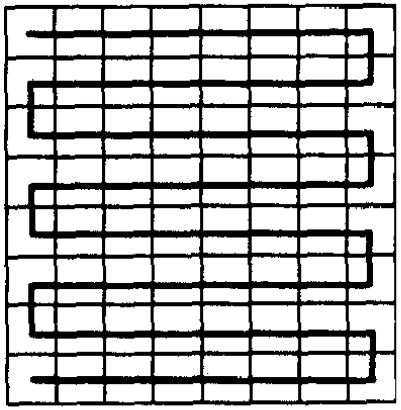
- **1.直接栅格编码**

这是最简单直观而又非常重要的一种栅格结构编码方法，通常称这种编码的图像文件为网格文件或栅格文件，栅格结构不论采用何种压缩编码方法，其逻辑原型都是直接编码网格文件。

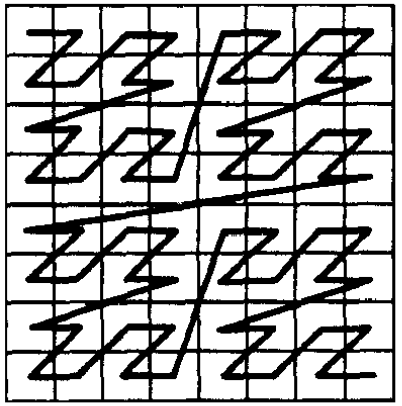
直接编码就是将栅格数据看作一个数据矩阵，逐行（或逐列）逐个记录代码，可以每行都从左到右逐个象元记录，也可以奇数行地从左到右而偶数行地从右向左记录，为了特定目的还可采用其他特殊的顺序。下图为一些常用的栅格排列顺序。



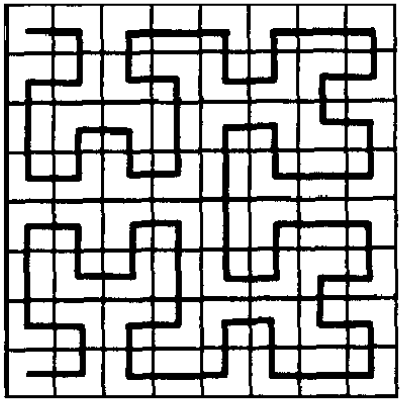
行



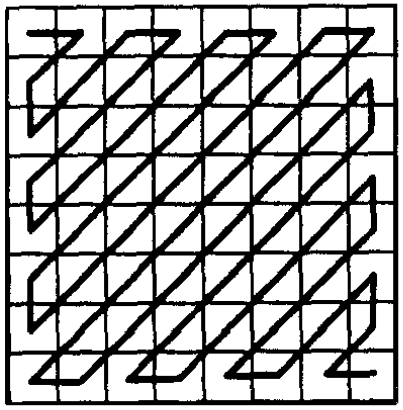
行主序



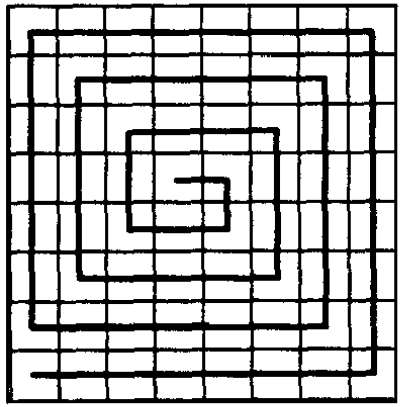
Morton



Peano-Hilbert



对角线

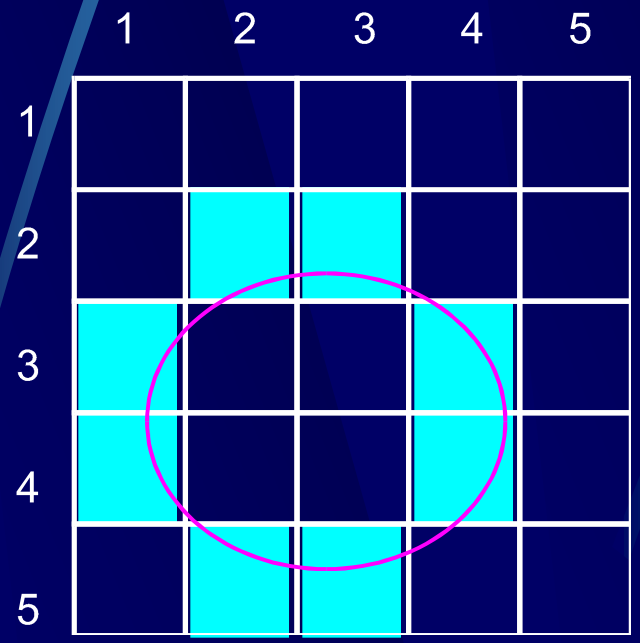


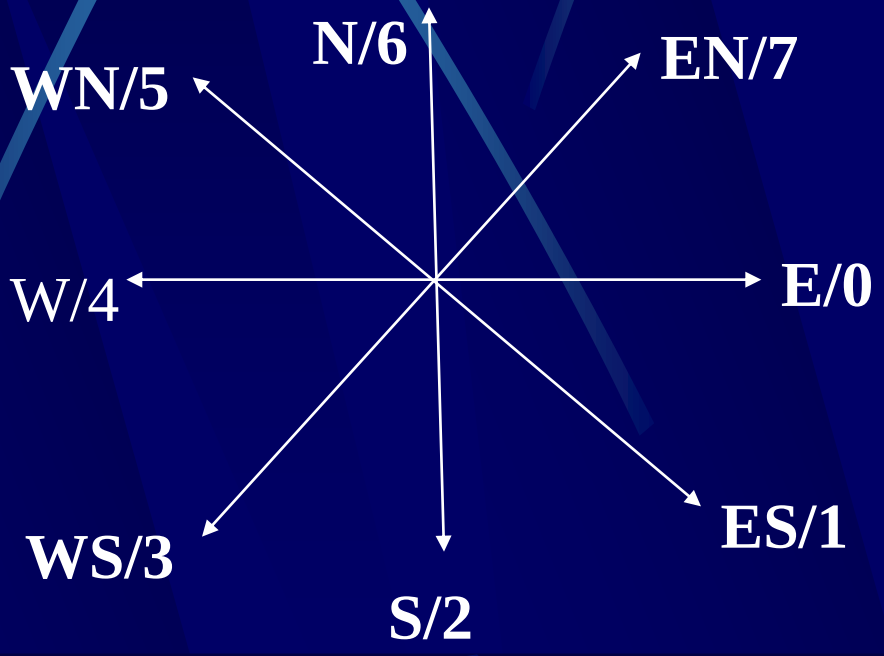
螺旋

一些常用的栅格排列顺序

2. 链式编码（Chain Codes）

链式编码又称为弗里曼链码[Freeman]或边界链码。多边形边界可以表示为：由某一原点开始并按某些基本方向确定的单位矢量链。基本方向可以定为：东=0，东南=1，南=2，西南=3，西=4，西北=5，北=6，东北=7，8个基本方向。



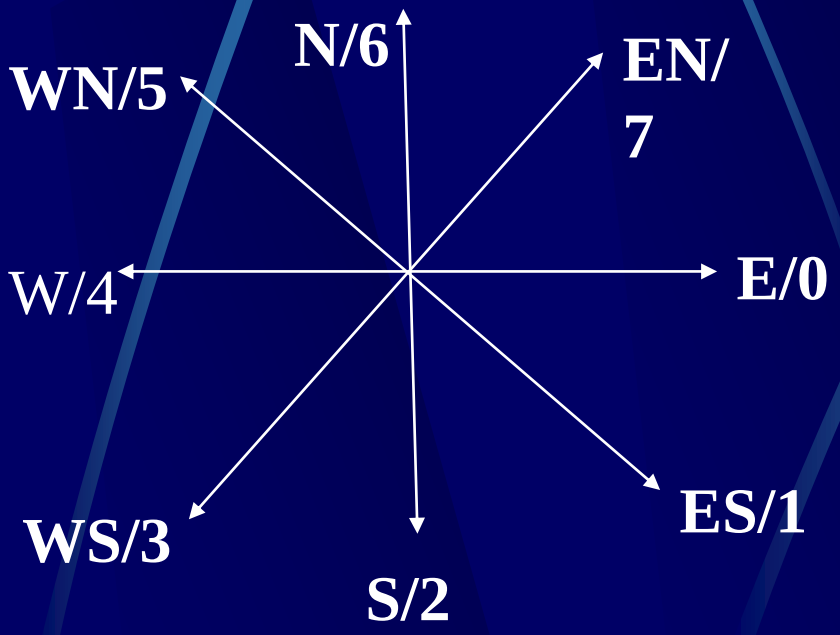


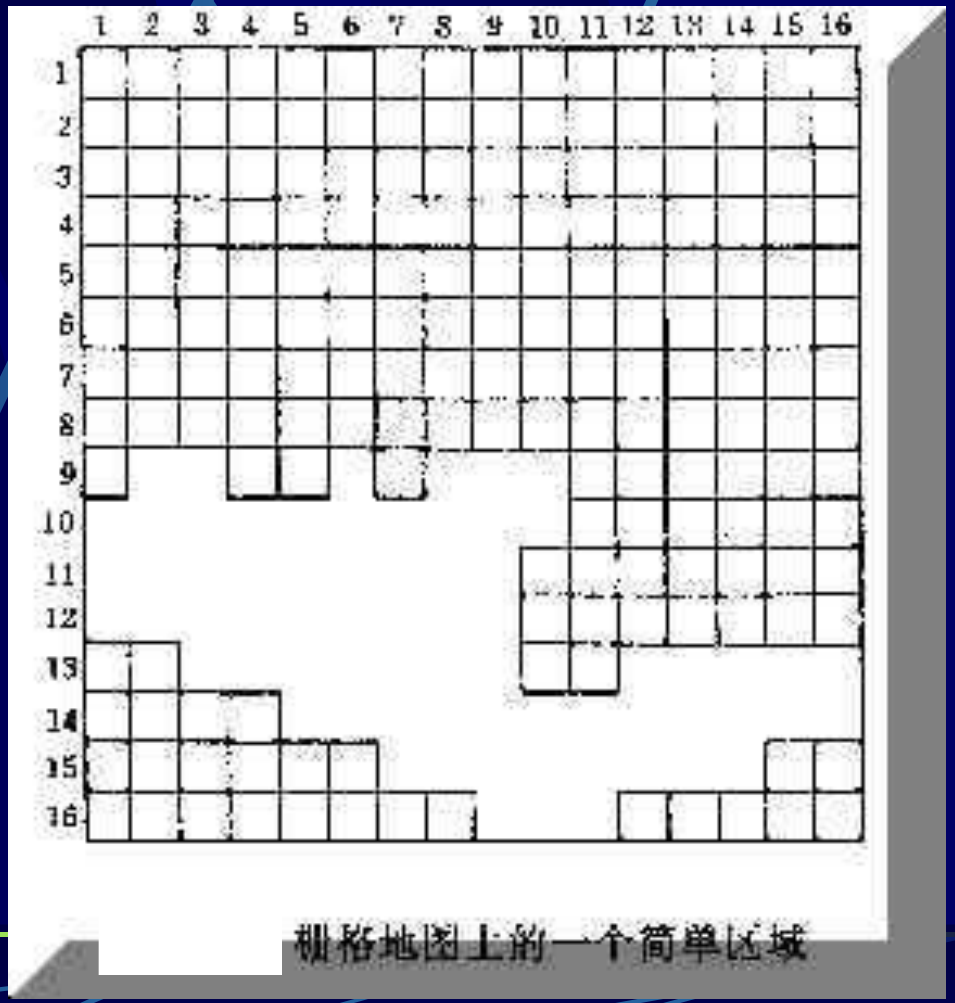
3,1,7,0,1,2,3,4,5,6

4,1,6,7,0,1,2,3,4,5

例子：如果再确定原点为像元(10，1)，则该多边形边界按顺时针方向的链式编码为：

10，l，7，0，1，0，7，1，7，0，0，2，3，2，2，1，0，7，0，0，0，0，2，4，3，4，4，3，4，4，5，4，5，4，5，4，5，4，6，6。




栅格地图上的一个简单区域

优点：链式边码可以有效地压缩栅格数据，而且对于估算面积、长度、转折方向的凹凸度等运算十分方便，比较适合于存储图形数据。

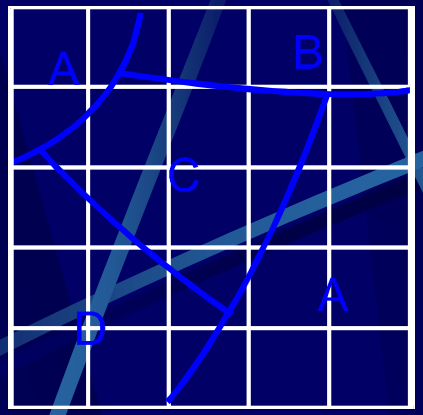
缺点：对边界进行合并和插入等修改编辑工作比较困难，对局部的修改将改变整体结构，效率较低，而且由于链码以每个区域为单位存储边界，相邻区域的边界将被重复存储而产生冗余。

- **3.游程长度编码（Run-Length Codes）**

游程长度编码是栅格数据压缩的重要编码方法。

游程指相邻同值网格的数量，游程长度编码结构是逐行将相邻同值的网格合并记录合并后网格的值及合并网格的长度，其目的是压缩栅格数据量，消除数据间的冗余。

- 游程长度编码结构的建立方法是：将栅格矩阵的数据序列$X_1, X_2, \cdots, X_n$，映射为相应的二元组序列（A_i, P_i），$i=1,2,\cdots,K$，且$K \leqslant n$。其中，A为属性值，P为游程，K为游程序号。



A	A	B	B	B
A	C	C	C	A
D	C	C	A	A
D	D	C	A	A
D	D	A	A	A

面域栅格矩阵结构

A	A	B	B	B
A	C	C	C	A
D	C	C	A	A
D	D	C	A	A
D	D	A	A	A

二元映射

序号	二元组序列
1	（A，2 ）
2	（ B，3）
3	（ A，1 ）
4	（ C，3 ）
5	（ A，1 ）
6	（ D，1 ）
7	…

游程长度编码表示栅格矩阵数据

- 游程编码能否压缩数据量，主要决定于栅格数据的性质，通常可通过事先测试，估算图层的数据冗余度R_e:

$R_e=1-Q/mn$　　式中，Q为图层内相邻属性值变化次数的累加和；m为图层网格的行数；n为图层网格的列数。

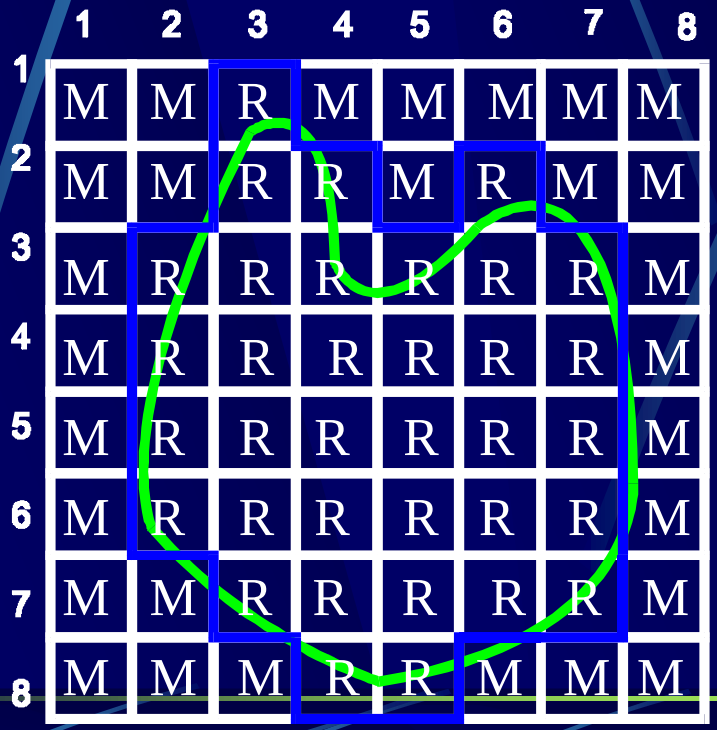
当R_e的值大于1/5的情况下，表明栅格数据的压缩可取得明显的效果。其压缩效果，可由压缩比S=n/K来表征，即压缩比的值愈大，表示压缩效果愈明显。

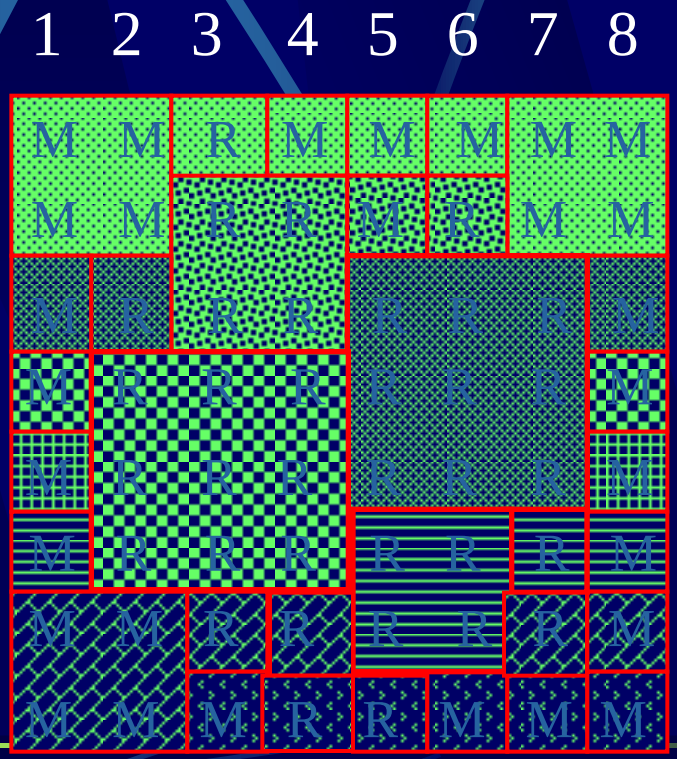
特点：游程长度编码在栅格压缩时，数据量没有明显增加，压缩效率较高，且易于检索，叠加合并等操作，运算简单，适用于机器存储容量小，数据需大量压缩，而又要避免复杂的编码解码运算增加处理和操作时间的情况。

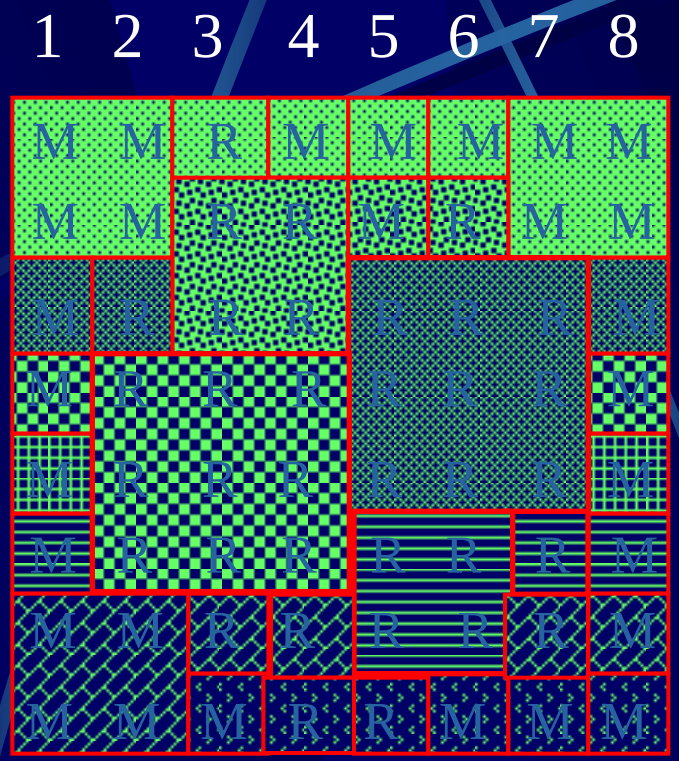
4. 块式编码(Block Codes)

块式编码是将游程扩大到两维情况，把多边形范围划分成若干具有同一属性的正方形，然后对各个正方形进行编码。

块式编码的数据结构由初始位置（行列号）、半径和属性代码组成。







1, 1, 2, M; 1, 3, 1, R; 1, 4, 1, M; 1, 5, 1, M; 1, 6, 1, M; 1, 7, 2, M

2, 3, 2, R; 2, 5, 1, M; 2, 6, 1, R

3, 1, 1, M; 3, 2, 1, R; 3, 5, 3, R; 3, 8, 1, M

4, 1, 1, M; 4, 2, 3, R; 4, 8, 1, M

5, 1, 1, M; 5, 8, 1, M

……

- 特点：块码具有可变的分辨率，即当代码变化小时图块大，就是说在区域图斑内部分辨率低；反之，分辨率高以小块记录区域边界地段，以此达到压缩的目的。因此块码与游程长度编码相似，随着图形复杂程度的提高而降低效率，就是说图斑越大，压缩比越高；图斑越碎，压缩比越低。块码在合并、插入、检查延伸性、计算面积等操作时有明显的优越性。然而在某些操作时，则必须把游程长度编码和块码解码，转换为基本栅格结构进行。

5. 差分映射法

所谓差分映射法，就是选择某一参照值对有关栅格的属性值进行求差运算，根据差值得到一个新的栅格数据层。

参照值的选择有多种方式：

（1）分行选取：则可选为该行首列的属性值，也可选为该行的属性平均值；

（2）全区选取：则可选为首行首列的属性值，也可选为全区的属性平均值。

如下图所示：

120	120	150	150	150	200	200	200
130	130	170	170	170	230	230	230
135	135	135	180	180	180	250	250
140	140	140	200	200	200	200	270
145	145	145	210	210	210	210	210

栅格数据示例

120	0	30	30	30	80	80	80
130	0	40	40	40	100	100	100
135	0	0	45	45	45	115	115
140	0	0	60	60	60	60	130
145	0	0	65	65	65	65	65

数据差分映射结果

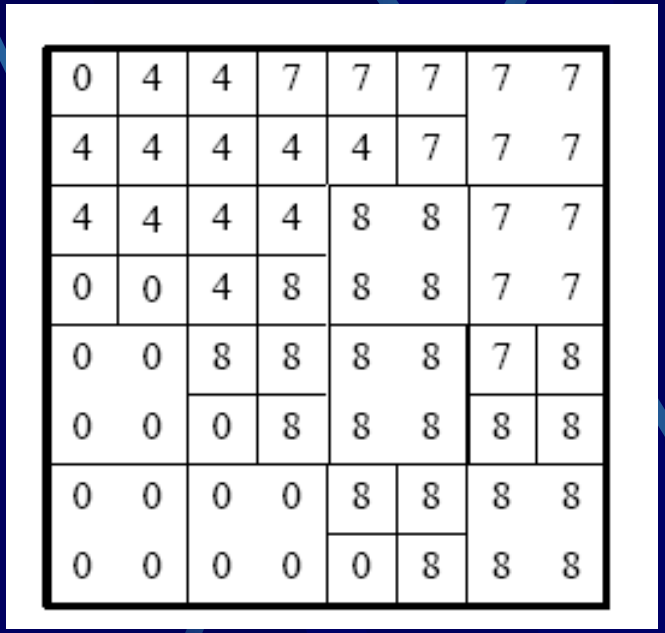
差分映射前后栅格数据记录长度对比

行号	第1列		第2列		第3列		第4列		第5列		第6列		第7列		第8列	
	前	后	前	后	前	后	前	后	前	后	前	后	前	后	前	后
1	1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
2	2	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1
3	2	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1
4	2	2	2	1	2	1	2	1	2	1	2	1	2	1	2	2
5	2	2	2	1	2	1	2	1	2	1	2	1	2	1	2	1
总	9	9	10	5	10	5	10	5	10	5	10	5	10	5	10	6

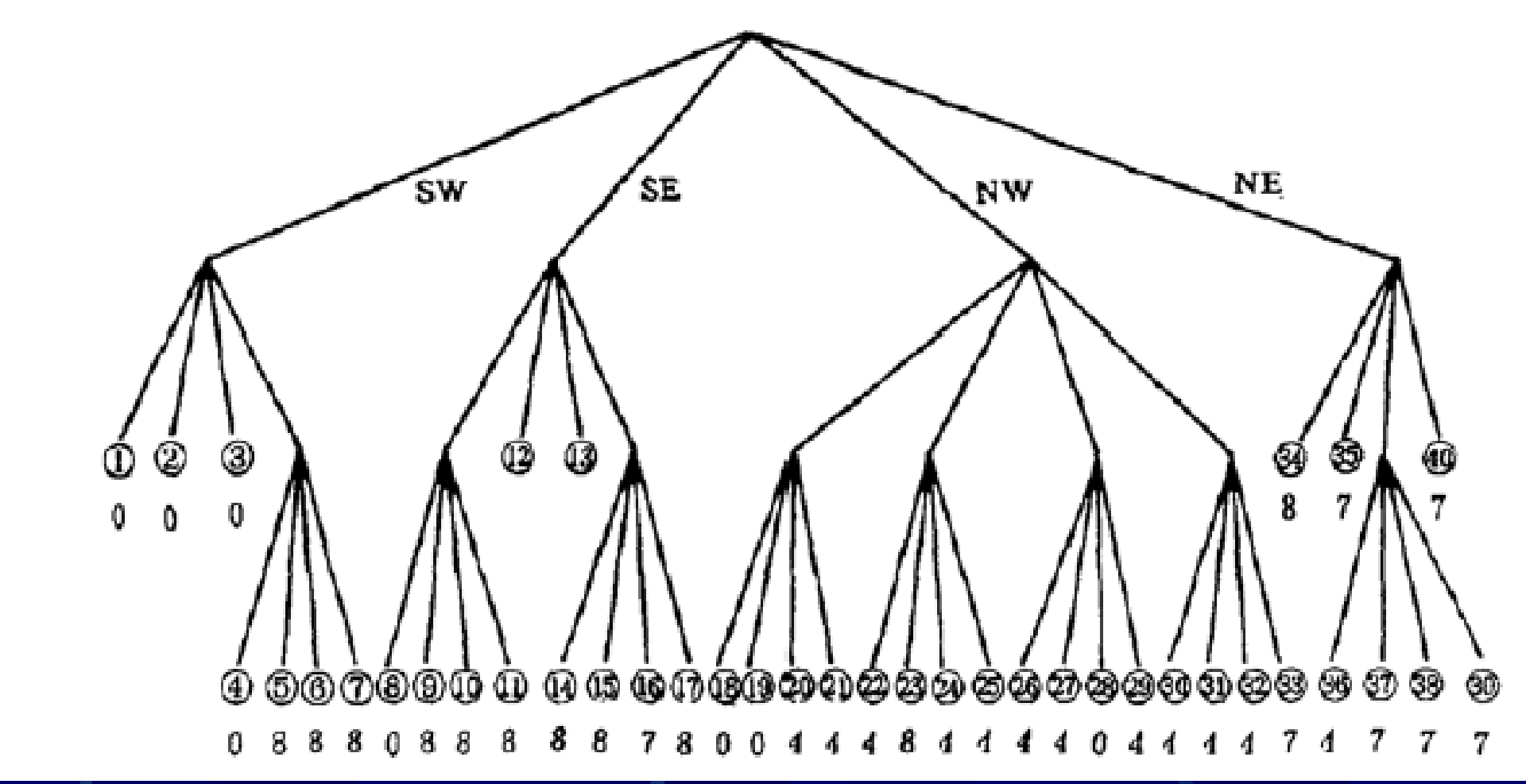
结论：由上表可见，所需字节数由原来的79减少为44，减少44.3%。

- 6. 四叉树编码(Quadtree Encoding)
- 四叉树又称四元树或四分树，是最有效的栅格数据压缩编码方法之一。
- 四分树将整个图像区域逐步分解为一系列方形区域，且每一个方形区域具有单一的属性。最小区域为一个象元。
- 区域分割原则：

  将欲分解区域等分为四个象限，再根据各个象限的象元值是否单一决定要不要再分。如果单一则不再分割，否则同法再分，直到所有象限的象元属性值相同为止。


0 4 4 7 7 7 7 7
4 4 4 4 4 7 7 7
4 4 4 4 8 8 7 7
0 0 4 8 8 8 7 7
0 0 8 8 8 8 7 8
0 0 0 8 8 8 8 8
0 0 0 0 8 8 8 8
0 0 0 0 0 8 8 8

(a) 四叉树分割



0	4	4	7	7	7	7	7
4	4	4	4	4	7	7	7
4	4	4	4	8	8	7	7
0	0	4	8	8	8	7	7
0	0	8	8	8	8	7	8
0	0	0	8	8	8	8	8
0	0	0	0	8	8	8	8
0	0	0	0	0	8	8	8

(a)的四叉树编码

其中最上面的那个结点叫做根结点，它对应整个图形。总共有4层结点，每个结点对应一个象限，如2层4个结点分别对应于整个图形的四个象限，排列次序依次为南西（SW）、南东（SE）、北西（NW）和北东（NE），不能再分的结点称为终止结点（又称叶子结点），可能落在不同的层上，该结点代表的子象限具有单一的代码，所有终止结点所代表的方形区域覆盖了整个图形。从上到下，从左到右为叶子结点编号如图所示，共有40个叶子结点，也就是原图被划分为40个大小不等的方形子区，图的最下面的一排数字表示各子区的代码。

由上面图形的四叉树分解可见，四叉树中象限的尺寸是大小不一的，位于较高层次的象限较大，深度小即分解次数少，而低层次上的象限较小，深度大即分解次数多，这反映了图上某些位置单一地物分布较广而另一些位置上的地物比较复杂，变化较大。正是由于四叉树编码能够自动地依照图形变化而调整象限尺寸，因此它具有极高的压缩效率。

- 采用四叉树编码时，为了保证四叉树分解能不断地进行下去，要求图像必须为$2^n \times 2^n$的栅格阵列，n为极限分割数，n+1为四叉树的最大高度或最大层数，上图为$2^3 \times 2^3$的栅格，因此最多划分三次，最大层数为4，对于非标准尺寸的图像需首先通过增加背景的方法将图像扩充为$2^n \times 2^n$的图像。

为了使计算机既能以最小的冗余存储图像对应的四叉树，又能方便地完成各种图形图像操作，专家们已提出了多种编码方式，下面介绍美国马里兰大学地理信息系统中采用的编码方式，该方法记录了终止结点（或叶子结点）的地址和值，值就是子区的代码，其中地址包括两个部分，共32位（二进制），最右边4位记录该叶子结点的深度，即处于四叉树的第几层上，有了深度可以推知子区的大小；地址由从根结点到该叶子结点的路径表示，0，1，2，3分别表示SW、SE、NW、NE，从右边第5位开始2n字节记录这些方向。

如上图表示的第六个结点深度为3，第一层处于SW象限，记为0；第二层处于NE象限，记为3，第三层处于NW象限，记为2，表示为二进制为：
0000… 000(22位)；001110（6位）；0011（4位）
每层象限位置由两位二进制数表示，共6位，十进制整数为227。这样，记录了各个叶子的地址，再记上相应代码值，就记录了这个图像，并可在此编码基础上进行多种图像操作。

事实上，叶结点的地址可以直接由子区左下角的行列坐标，按二进制按位交错得到。如对于6号叶子结点，在以图像左下角为原点的行列坐标系中，其左下角行、列坐标为（3，2），表示为二进制分别为011和010，按位交错就是001110，正是6号地块。

特点：四叉树编码具有可变的分辨率，并且有区域性质，压缩数据灵活，许多运算可以在编码数据上直接实现，大大地提高了运算效率，是优秀的栅格压缩编码之一。

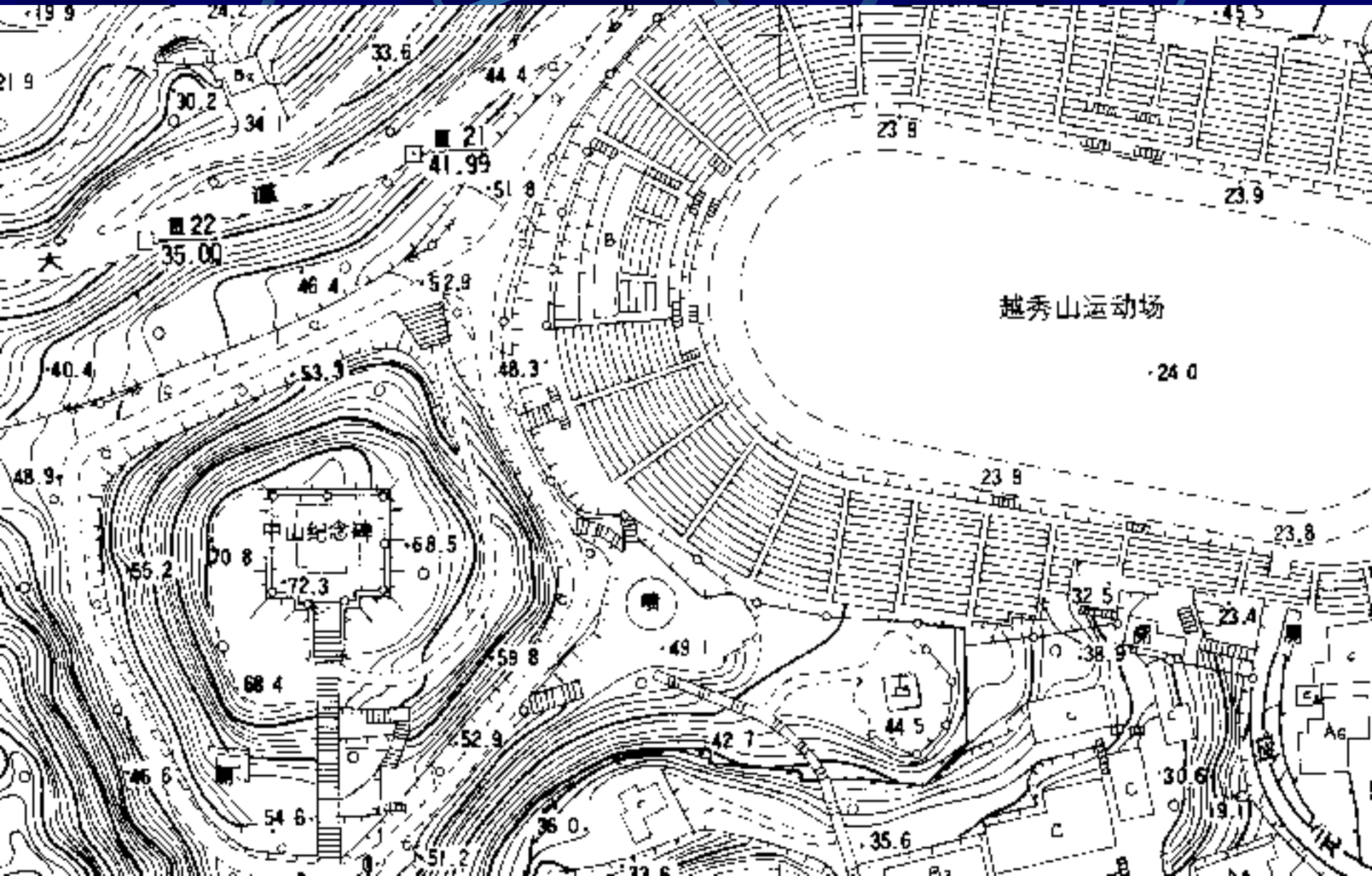
几种编码的比较

- 链式编码的压缩效率较高，已经近矢量结构，对边界的运算比较方便，但不具有区域的性质，区域运算困难；
- 游程长度编码既可以在很大程度上压缩数据，又最大限度地保留了原始栅格结构，编码解码十分容易；
- 块码和四叉树码具有区域性质，又具有可变的分辨率，有较高的压缩效率，四叉树编码可以直接进行大量图形图像运算，效率较高，是很有前途的方法。

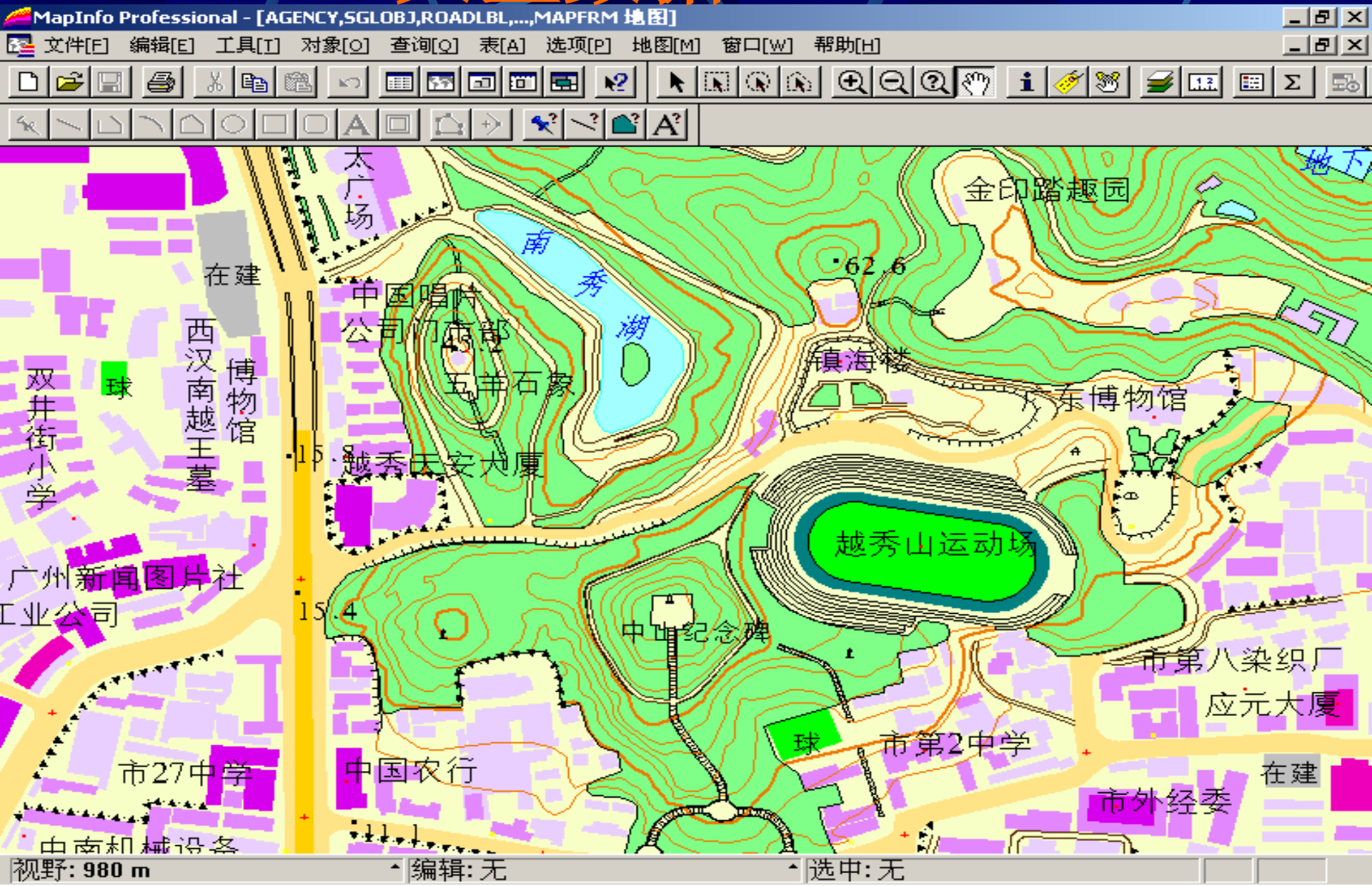
二、矢量数据结构及其压缩

- 定义：通过记录坐标的方式尽可能精确地表示地理实体，即地理实体的形状和位置是由一组所在的坐标参考系中坐标确定的。矢量数据结构是人们较为习惯的一种表示空间数据的方法
- 特点：定位明显、属性隐含，其定位是根据坐标直接存储的，而属性则一般存于文件头或数据结构中某些特定的位置上，这种特点使得其图形运算的算法总体上比栅格数据结构复杂的多，有些甚至难以实现，当然有些地方也有所便利和独到之处，在计算长度、面积、形状和图形编辑、几何变换操作中，矢量结构有很高的效率和精度，而在叠加运算、邻域搜索等操作时则比较困难。
- 在GIS中，地理实体的空间特征首先抽象为点、线、面、体四种基本类型，而这些特征可以用颜色、符号、注记来区分，并由图例、图符和描述性文本来解释。

矢量数据（1）



矢量数据（2）

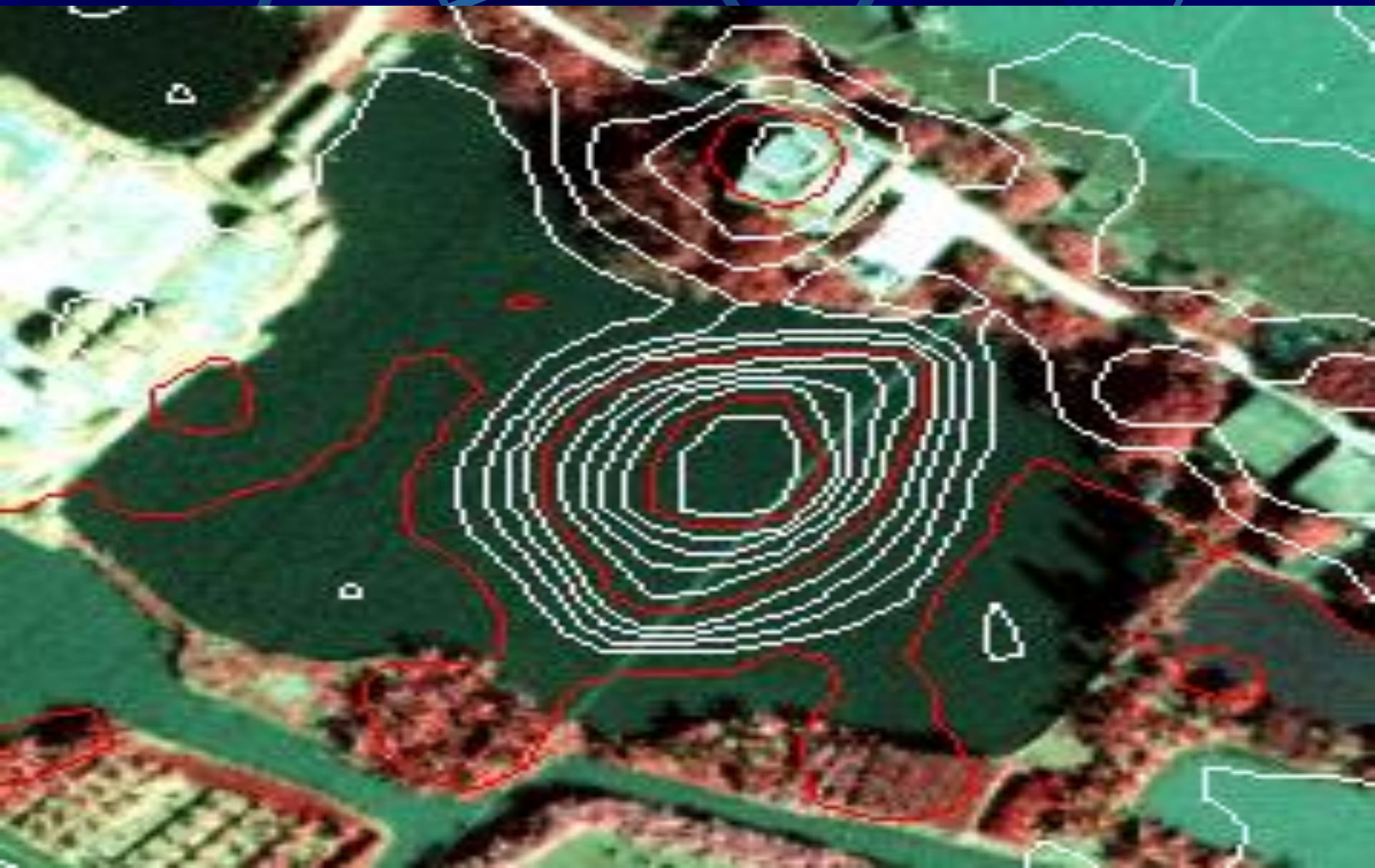

MapInfo Professional - [AGENCY,SGLOBJ,ROADLBL,...,MAPFRM 地图]
文件[F] 编辑[E] 工具[T] 对象[O] 查询[Q] 表[A] 选项[P] 地图[M] 窗口[W] 帮助[H]
太广场
在建
中国唱片公司门市部
南秀湖
金印踏趣园
62.6
镇海楼
广东博物馆
西汉南越王墓博物馆
双井街小学
球
五羊石家
越秀天安大厦
越秀山运动场
广州新闻图片社
工业公司
中山纪念碑
市第八染织厂
应元大厦
球
市第2中学
市27中学
中国农行
在建
市外经委
视野: 980 m
编辑: 无
选中: 无

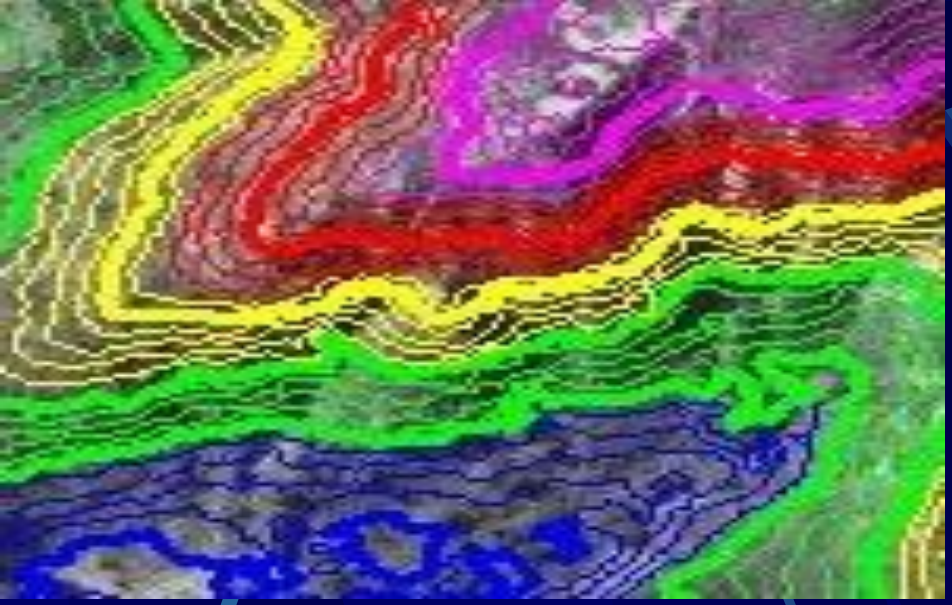
矢量数据（3）

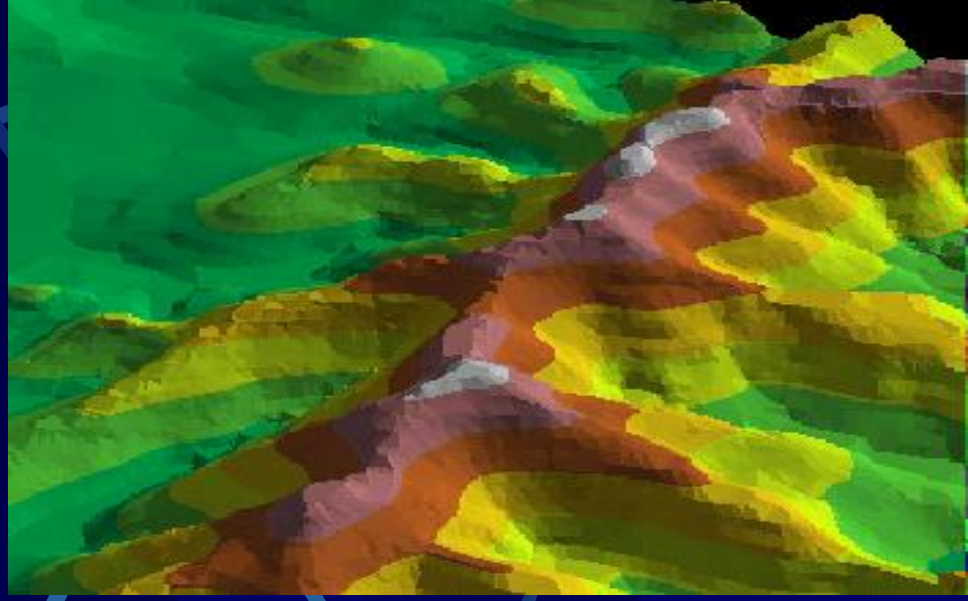
华港花园
广州民族学院
翠湖山庄
华江花园
天河公园
天河区人民政府
岗园新村
棠下
棠东
中山大道
天府路
天河区中医医院
天河高新技术产业开发区
广东省环境保护学校
东郊加油站
广州化学纤维厂
市第六人民医院
员村支线
广东玻璃厂
广州化工厂
广州绢麻纺织厂
员村
程界
员村热电厂
棠下新墟
广东罐头厂
广州南方面粉厂
广州赛马场
华南快速干线
华南大桥
江

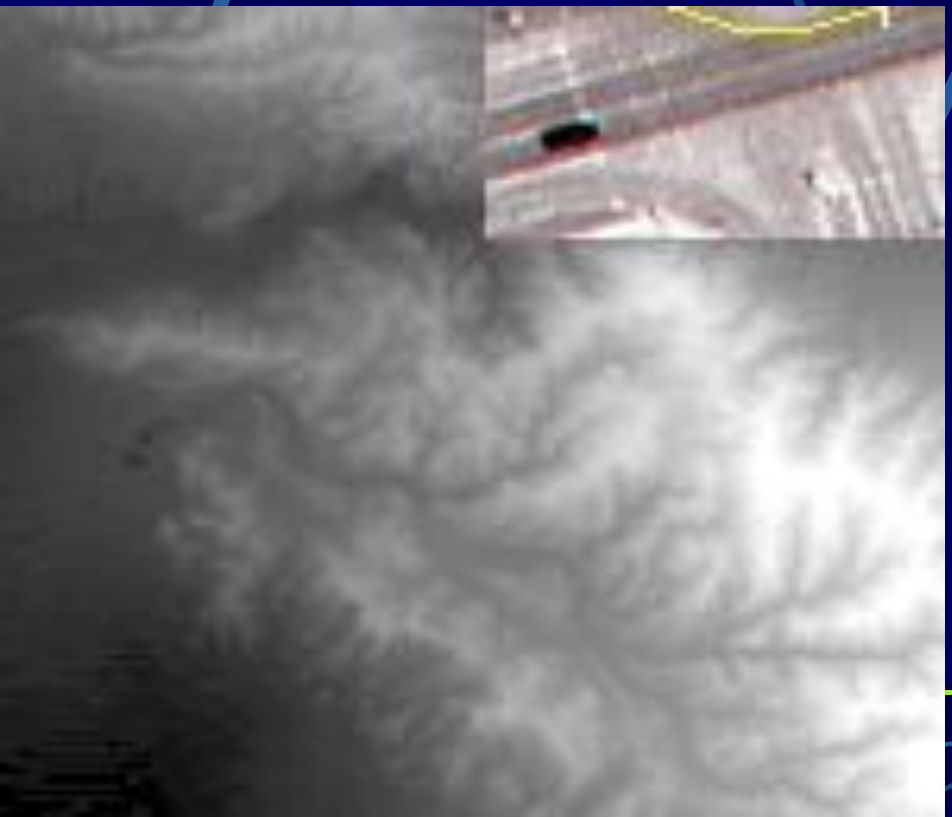
矢量+栅格（1）

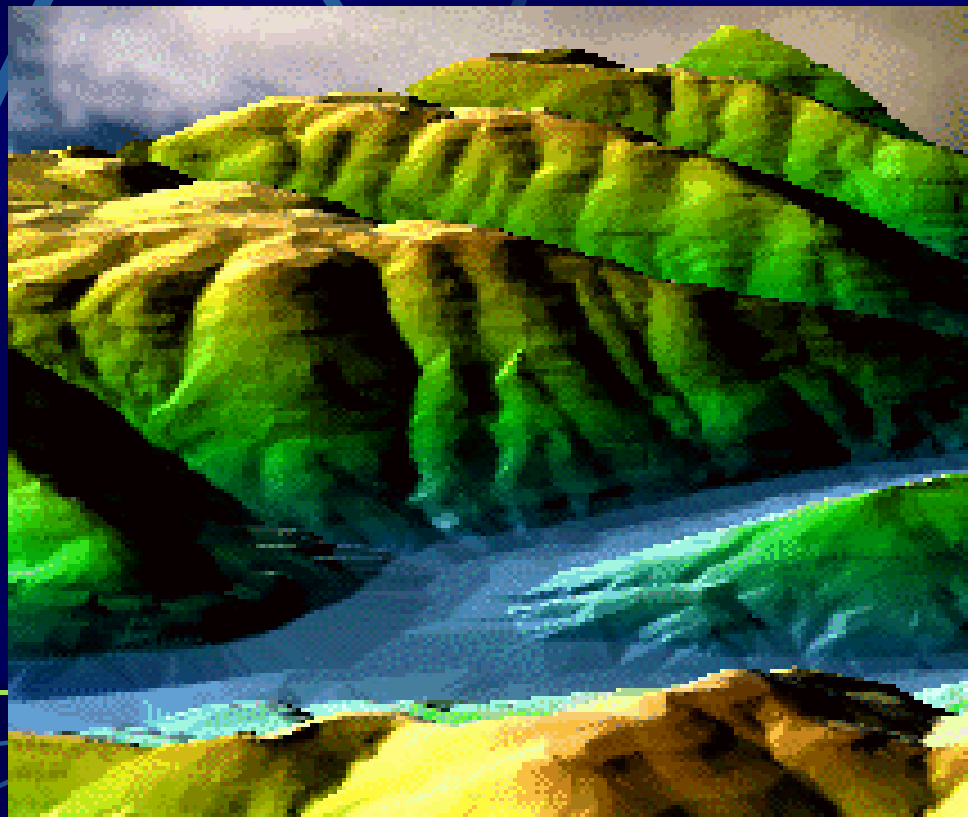
矢量+栅格（2）











矢量数据结构获取方法 {
(1) 手工数字化法；
(2) 手扶跟踪数字化法；
(3) 数据结构转换法。

矢量数据基本类型 ： (1) 点　　(2) 线　　(3) 面

矢量数据基本类型-点

点是点状物或者是可以用点（由单独一对坐标定位）的一切地理或制图实体，有特定的位置。图件的比例尺决定了能否把现实世界的现象表示为点特征。

- 它有可能是点状地物、面状地物的中心点、线状地物的交点、定位点、注记等。

例子：水基准点、建筑物、井、观测点、高程点

矢量数据基本类型-点

在GIS中点有几种类型。线的起点、终点、交点（三条以上坐标链的交汇点）、面的首尾点我们称之为结点（node），而线的中间部分称为中间点（角点vertex）。

- ✓ 实体点(Entity point)：用来代表一个实体;
- ✓ 注记点(Text point)：用于定位注记;
- ✓ 内点(Label point)：用于记录多边形的属性，存在于多边形内;
- ✓ 结点(Node)：表示线的终点和起点、交点;
- ✓ 中间点(角点，Vertex)：表示线段和弧段的内部点。

矢量数据基本类型-线

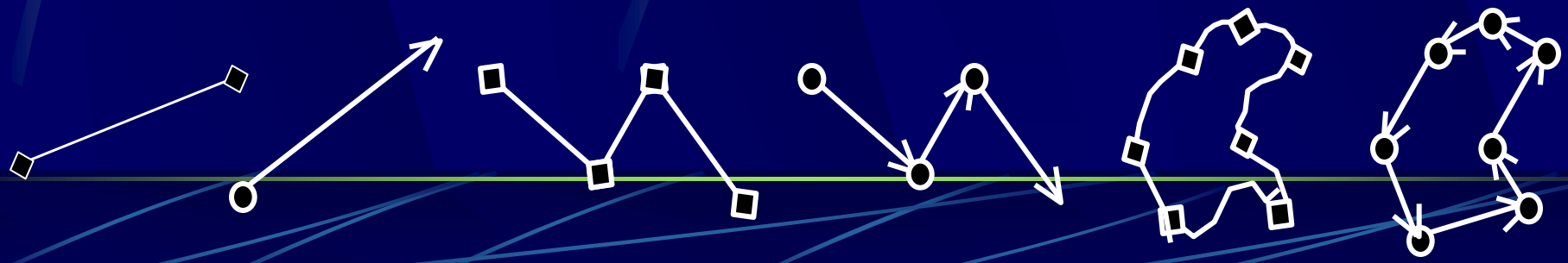
线是对线状地物或地物运动轨迹的全部或部分的描述，可以定义为由直线元素组成的各种线性要素，直线元素由两对以上的坐标定义。最简单的线实体只存储它的起止点坐标、属性、显示符等有关数据。

- 线有方向，两个结点之间的线又叫弧段（arc）。弧段特征可用来定位和描述两点之间连线的地理信息。

矢量数据基本类型-线

- 由一系列坐标点表示，有以下特征：
  - ✓ 实体长度：从起点到终点的总长；
  - ✓ 弯曲度：用于表示象道路拐弯时弯曲的程度；
  - ✓ 方向性：如水流从上游到下游，公路则有单双向之分；

线实体包括：线段、边界、链、网络、多边形线等。



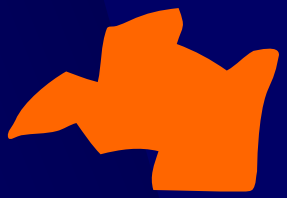
矢量数据基本类型-面（多边形）

面（多边形polygon）是对面状地理实体的表示，由一个封闭的坐标点序列外加内点表示。但多边形矢量编码，不但要标识位置和属性，更重要的是表达拓扑特征,如形状、邻域和层次结构等。多边形由一条或一条以上首尾相连的弧段组成。一个弧段总是被两个而且只被两个多边形所共有。

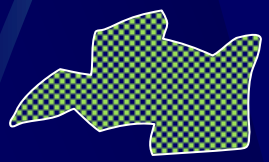
- 复杂的多边形内可以有“岛（洞）”（一种特殊的弧段，这种弧段坐标链头尾相接，独立围成一个封闭的区域。弧段的端点总是结点，而岛弧段端点并非是三条或三条以上坐标链的交汇点，这种端点称为岛结点）。

矢量数据基本类型-面（多边形）

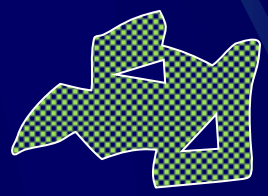
- 多边形有以下特征:
  - ✓ 面积范围;
  - ✓ 周长;
  - ✓ 独立性或与其它地物相邻：如北京及周边省市;



内部区域



简单多边形



复杂多边形

矢量数据的压缩

- 矢量数据的压缩包括两个方面的内容：

(1)在不扰乱拓扑关系的前提下，对采样点数据进行合理的抽稀；

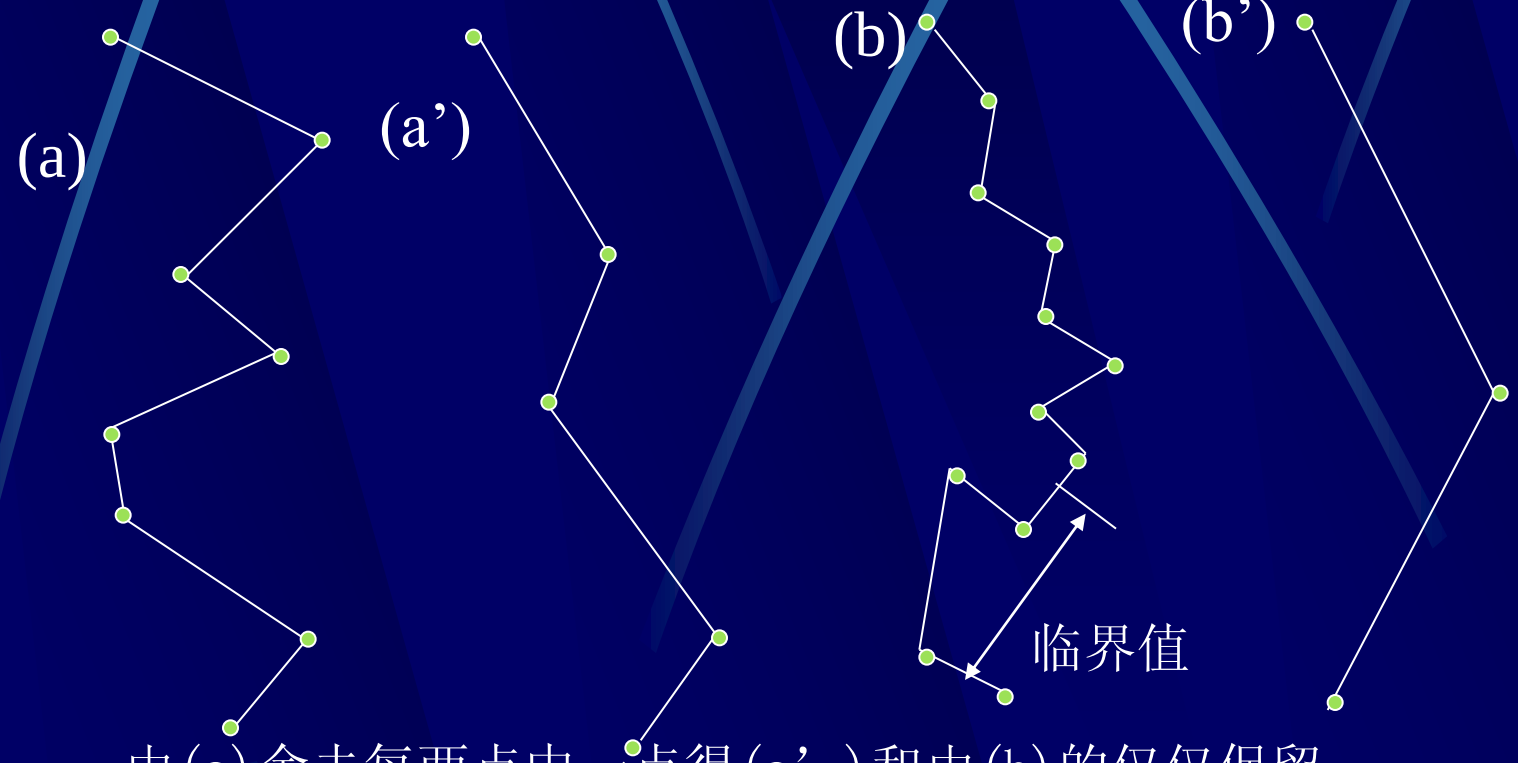
(2)对矢量坐标数据重新进行编码，以减少所需要的存储空间。

注：矢量数据的压缩往往是不可逆的，数据压缩后数据量变小了，但数据的精度降低了。

- 矢量数据压缩的目的是删除冗余数据，减少数据的存贮量，节省存贮空间，加快后继处理的速度。下面介绍几种常用的矢量数据的压缩算法，以及它们之间的异同点。

- **1. 间隔取点法：** 每隔K个点取一点，或舍去那些离已选点比规定距离更近的点，但首、末点一定要保留。如下图所示。

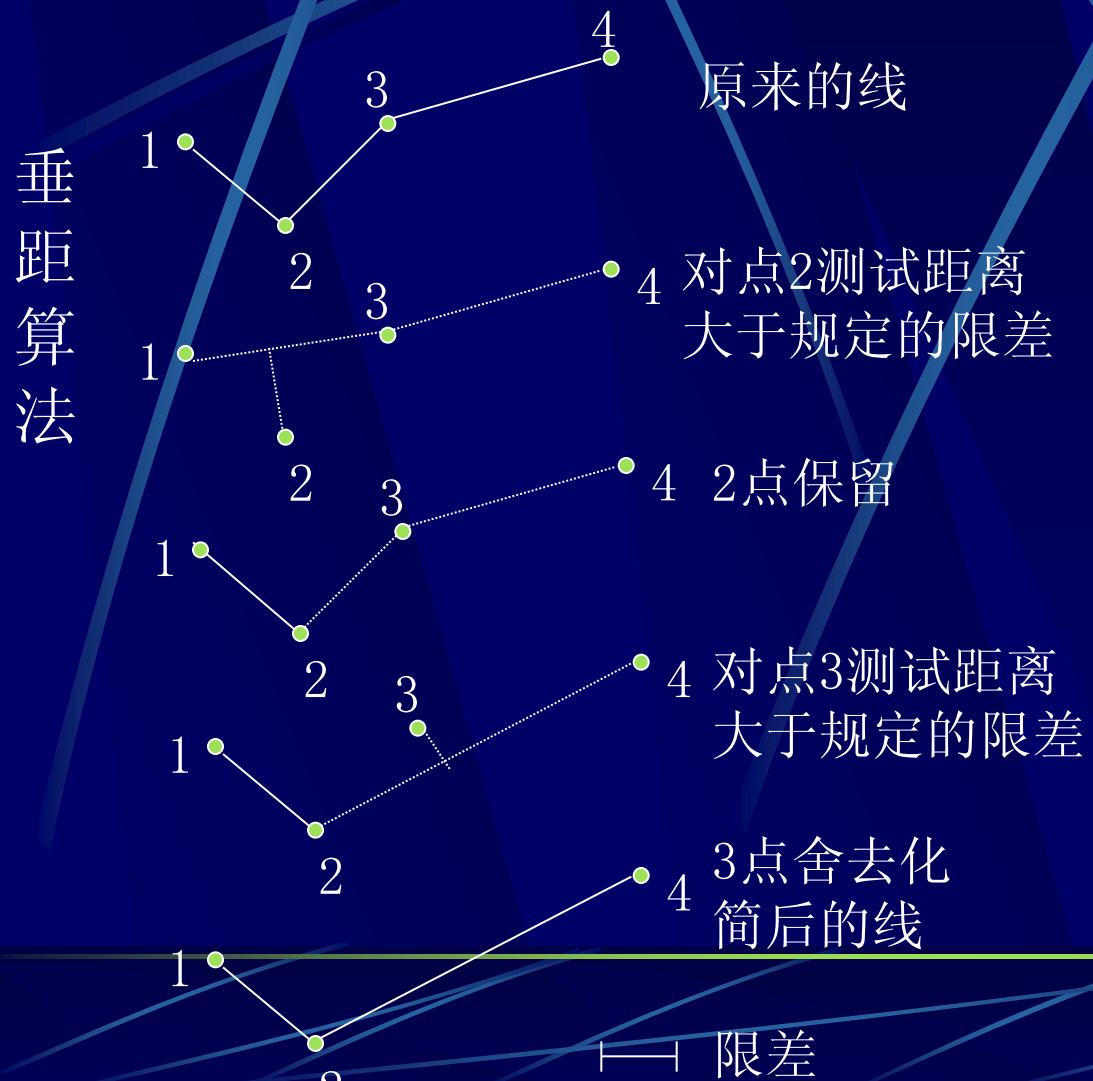
特点：这种方法可大量压缩数字化仪用连续方法获取的点列中的点、曲率显著变化的点，但不一定能恰当地保留方向上曲率显著变化的点。



由(a)舍去每两点中一点得(a’)和由(b)的仅仅保留与已选点距离超过临界值的点得(b’)

2. 垂距法和偏角法

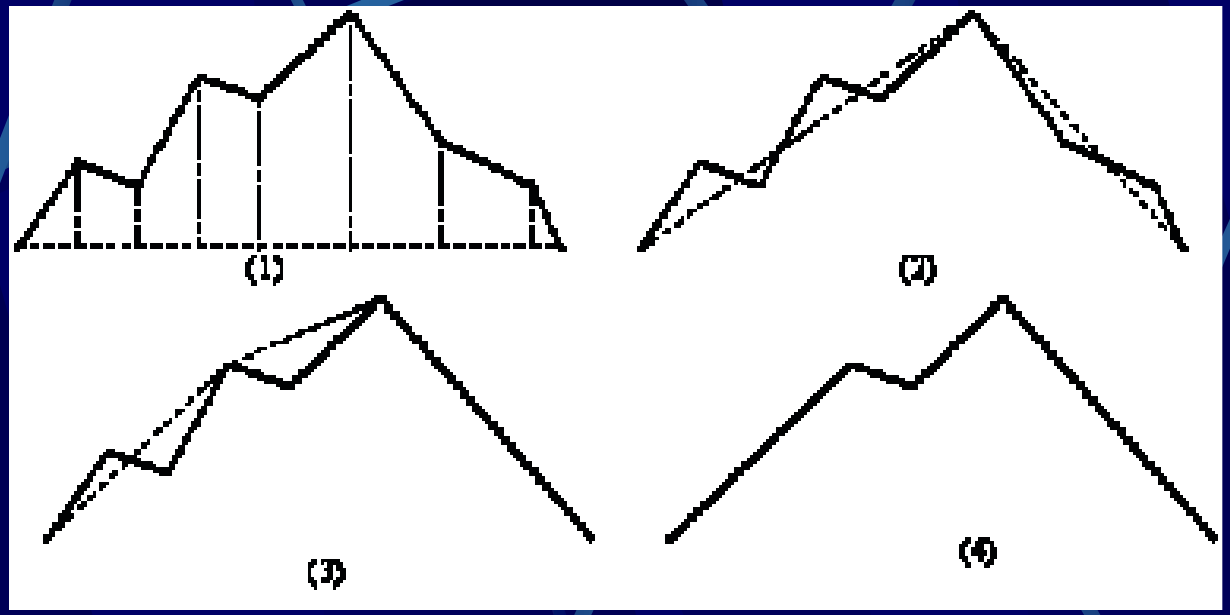
这两种方法都是按垂距或偏角的限差，选取符合或超过限差的点，其过程如图所示。



偏角算法

这两种方法虽然不能同时考虑相邻点间的方向和距离，且有可能舍去不该舍去的点，但较前一种方法有进步。

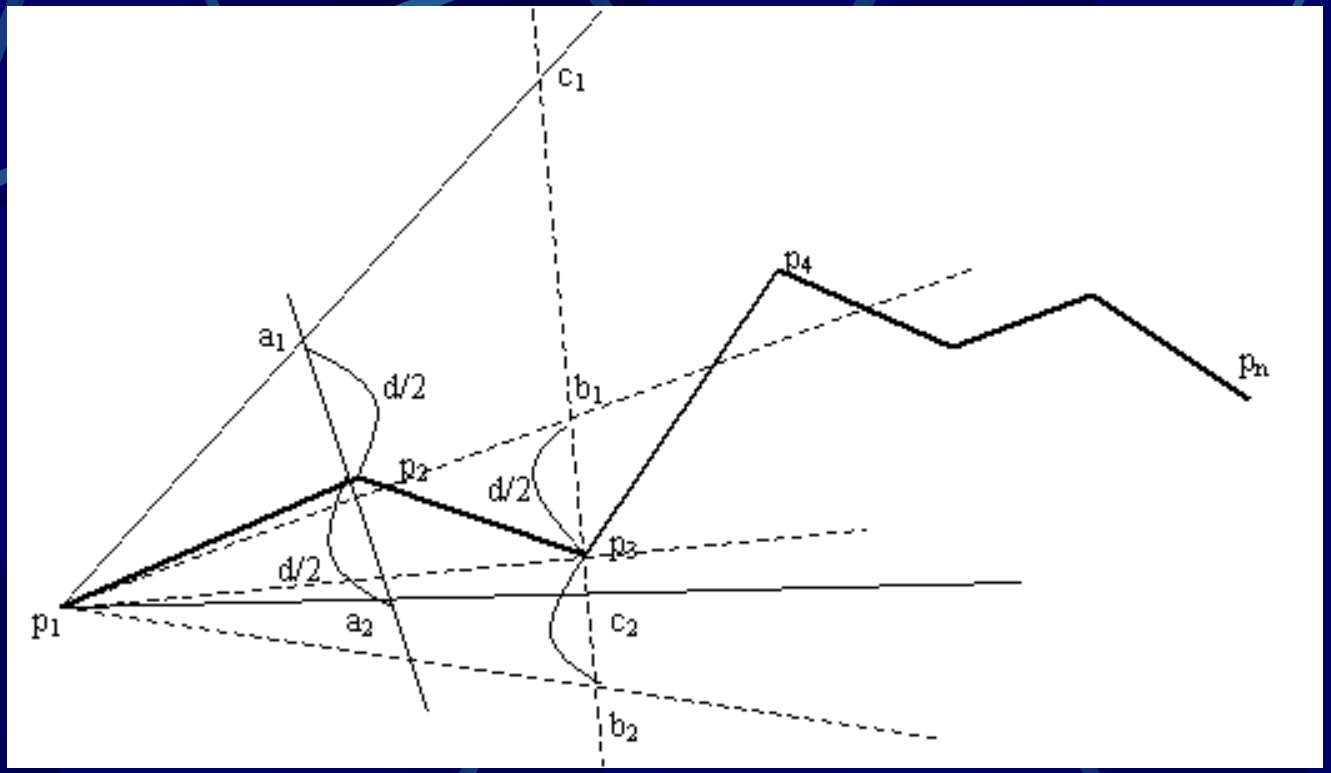
3. 道格拉斯——普克法(Douglas—Peucker)



道格拉斯——普克法示意图

基本思路是：对每一条曲线的首末点虚连一条直线，求所有点与直线的距离，并找出最大距离值dmax，用dmax与限差D相比：
若dmax＜D，这条曲线上的中间点全部舍去；
若dmax≥D，保留dmax对应的坐标点，并以该点为界，把曲线分为两部分，对这两部分重复使用该方法。

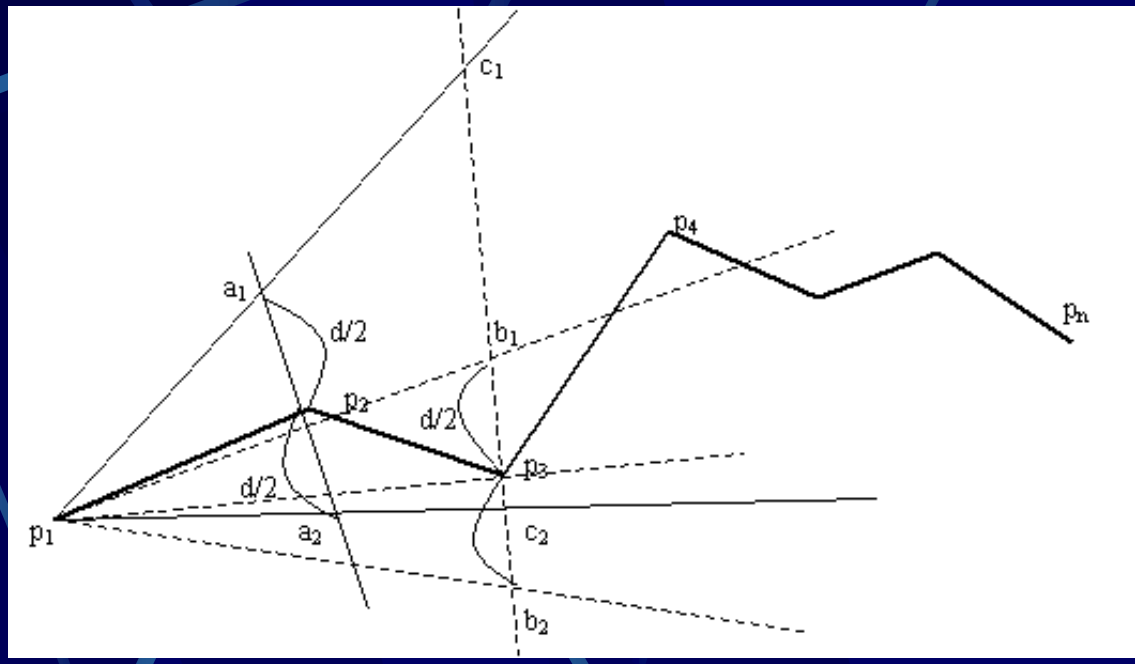
4. 光栏法



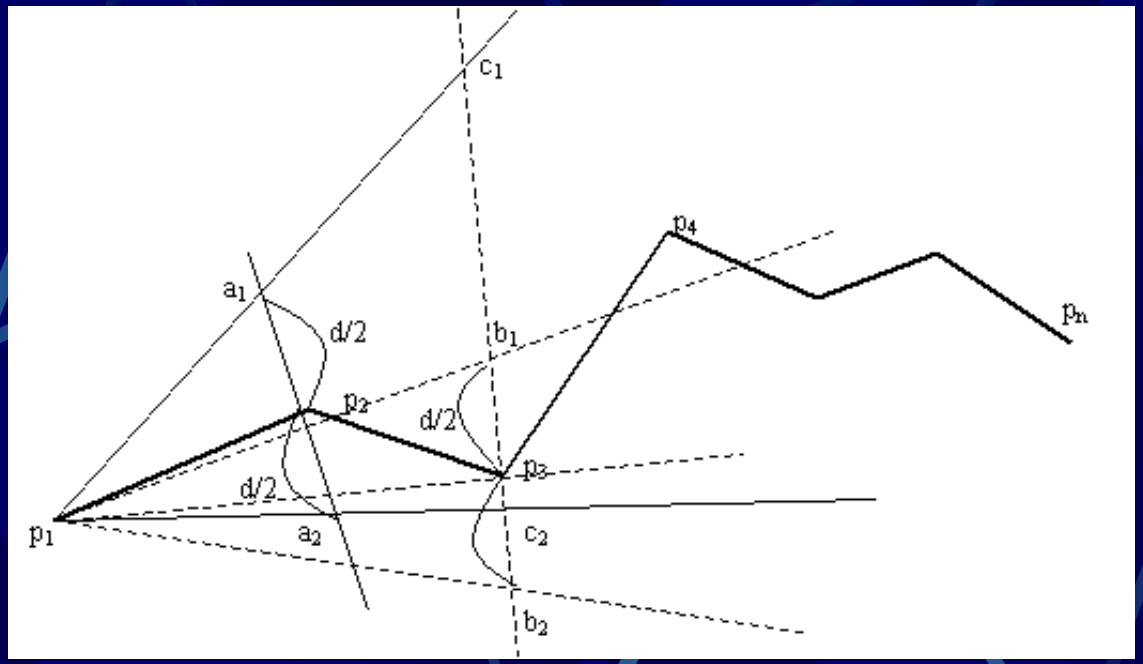
光栏法原理图示

光栏法的基本思想是：定义一个扇形区域，通过判断曲线上的点在扇形外还是在扇形内，确定保留还是舍去。设曲线上的点列为{pi}，i＝1，2，…，n，光栏口经为d，可根据压缩量的大小自己定义，则光栏法的实施步骤可描述为：

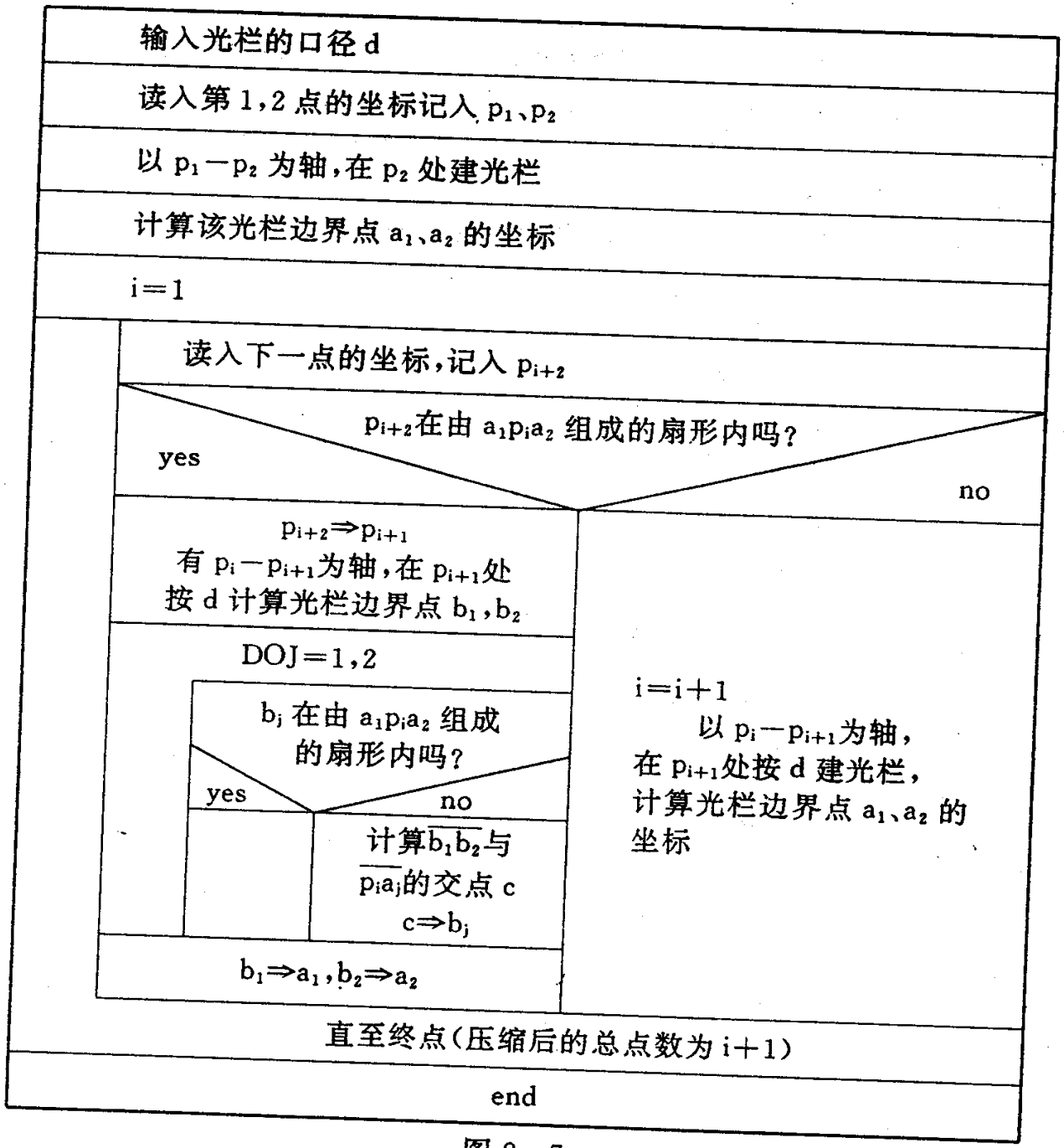
- 1、连接p_1和p_2点，过p_2点作一条垂直于p_1p_2的直线，在该垂线上取两点a_1和a_2，使$a_1p_2=a_2p_2=d/2$，此时a_1和a_2为“光栏”边界点，p_1与a_1、p_1与a_2的连线为以p_1为顶点的扇形的两条边，这就定义了一个扇形(这个扇形的口朝向曲线的前进方向，边长是任意的)。通过p_1并在扇形内的所有直线都具有这种性质，即p_1p_2上各点到这些直线的垂距都不大于d/2。



- 2、若p_3点在扇形内，则舍去p_2点。然后连接p_1和p_3，过p_3作p_1p_1的垂线，该垂线与前面定义的扇形边交于c_1和c_2。在垂线上找到b_1和b_2点，使$p_3b_1=p_3b_2=d/2$，若b_1或b_2点落在原扇形外面，则用c_1或c_2取代。此时用p_1b_1和p_1c_2定义一个新的扇形，这当然是口径(b_1c_2)缩小了的“光栏”。



- 3、检查下一节点，若该点在新扇形内，则重复第(2)步；直到发现有一个节点在最新定义的扇形外为止。
- 4、当发现在扇形外的节点，如p_4，此时保留p_3点，以p_3作为新起点，重复1°～3°。如此继续下去，直到整个点列检测完为止。所有被保留的节点(含首、末点)，顺序地构成了简化后的新点列。


输入光栏的口径 d
读入第 1，2 点的坐标记入 p_1、p_2
以 p_1-p_2 为轴，在 p_2 处建光栏
计算该光栏边界点 a_1、a_2 的坐标
i=1
读入下一点的坐标，记入 p_{i+2}
p_{i+2}在由 $a_1p_ia_2$ 组成的扇形内吗？
yes
no
$p_{i+2}\Rightarrow p_{i+1}$
有 p_i-p_{i+1}为轴，在 p_{i+1}处
按 d 计算光栏边界点 b_1，b_2
DOJ=1,2
b_j 在由 $a_1p_ia_2$ 组成
的扇形内吗？
yes
no
计算$\overline{b_1b_2}$与
$\overline{p_ia_j}$的交点 c
$c\Rightarrow b_j$
$b_1\Rightarrow a_1$，$b_2\Rightarrow a_2$
i=i+1
以 p_i-p_{i+1}为轴，
在 p_{i+1}处按 d 建光栏，
计算光栏边界点 a_1、a_2 的
坐标
直至终点(压缩后的总点数为 i+1)
end

压缩算法的比较

- 如果某种矢量数据的压缩算法既能精确地表示数据，又能最大限度地淘汰不必要的点，那就是一种好的算法。具体可以依据简化后曲线的总长度、总面积、坐标平均值等与原始曲线的相应数据的对比来判别。
- 通过分析可以发现，大多数情况下道格拉斯——普克法的压缩算法较好，但必须在对整条曲线数字化完成后才能进行，且计算量较大；光栏法的压缩算法也很好，并且可在数字化时实时处理，每次判断下一个数字化的点，且计算量较小；垂距法算法简单，速度快，但有时会将曲线的弯曲极值点p值去掉而失真。

面域的数据压缩算法

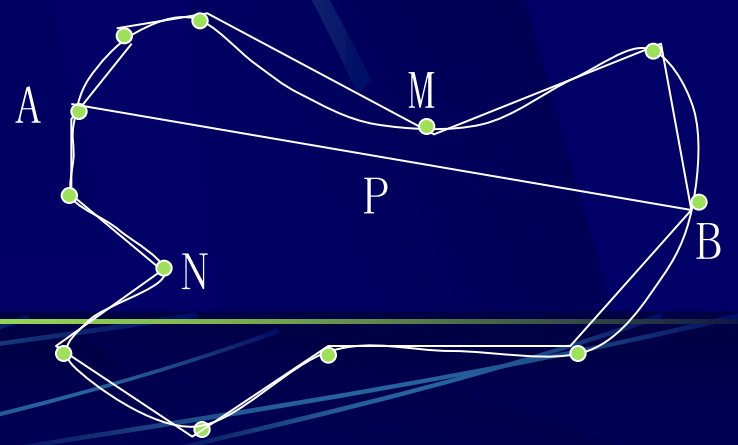
- 面域的空间数据压缩过程可以看成是组成其边界的曲线段的分别压缩，每段边界曲线的压缩过程如前所述。但有两个问题需要注意。
- **1. 封闭曲线的数据压缩**

面域由首尾相连的封闭曲线组成。可以人为地将该封闭线分割为首尾相连的两段曲线，然后可以按前面的方法进行压缩。曲线的分割的原则是：

(1) 原节点是分割点之一；

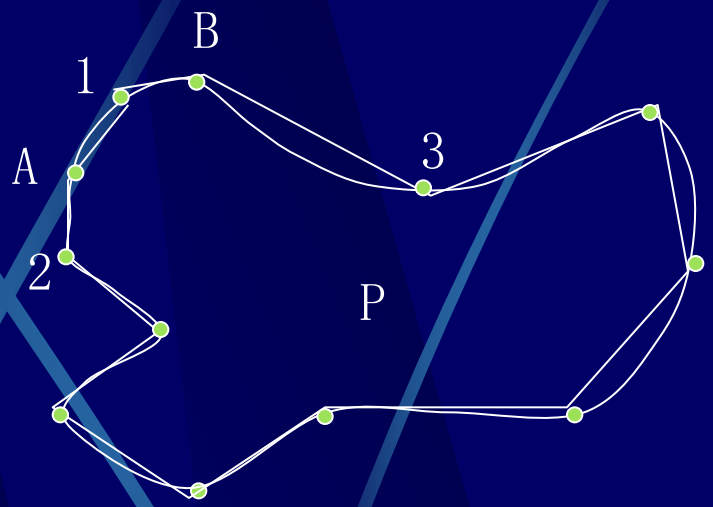
(2) 离原节点最远的下一节点是分割点之二。

如右图所示，多边形P的边界曲线可以分割为AMP和BNA两段，进而对曲线段AMB、BNA分别进行压缩。



2. 公共节点的取舍问题

为了防止产生数据冗余，当前后曲线段过渡时很平缓，曲线段公共节点可以不成为特征点，即该点前后的两段曲线可以直接用该点前后的两个特征点的连线来代替。



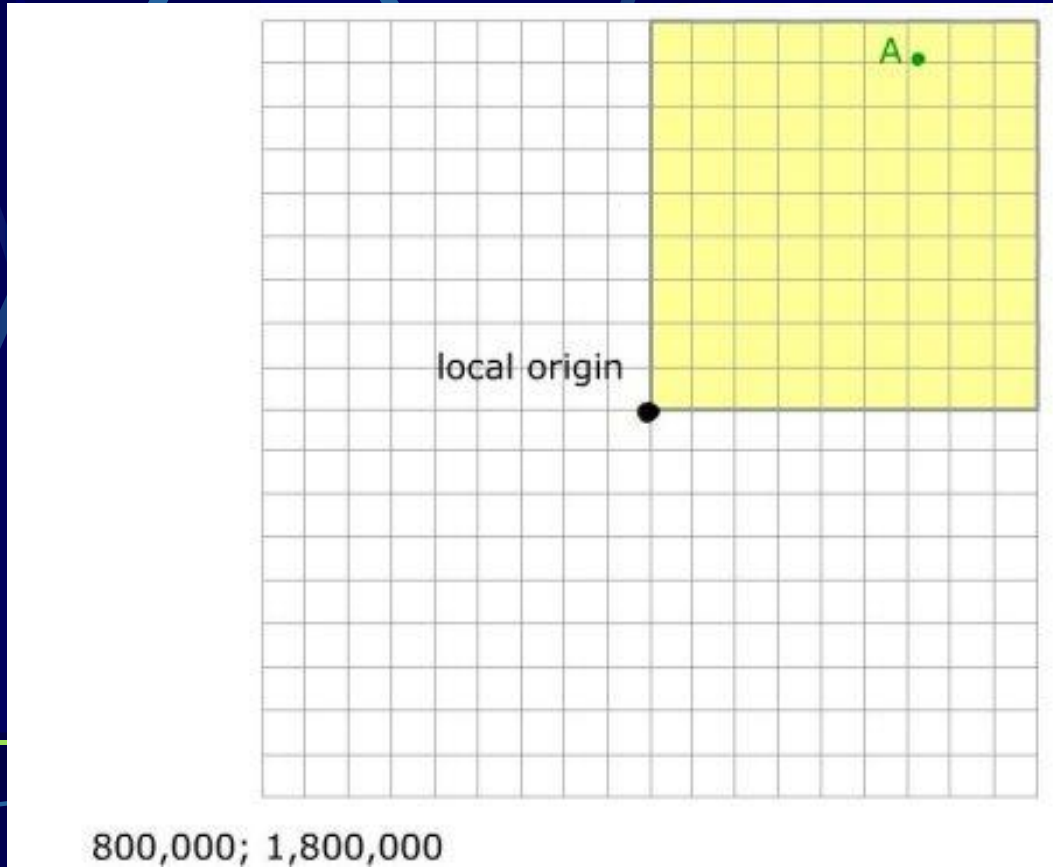
因此，在处理面域空间数据压缩时，可以在边界曲线分段压缩的基础上，增加一个步骤，即对边界曲线的端点进行可删性检验：如果前一曲线最后提取的中间特征点与后一曲线最先提取的中间特征点之间的曲线满足级差控制条件，则两条曲线的连接节点可以删减；否则，不可删减。

注意：在处理公共节点的取舍时要慎重！

传统方法小结

- 栅格数据的压缩
  - 直接栅格编码
  - 链式编码（Chain Codes）
  - 游程长度编码（Run-Length Codes）
  - 块式编码(Block Codes)
  - 差分映射法
  - 四叉树编码(Quadtree Encoding)
- 矢量数据结构及其压缩
  - 间隔取点法
  - 垂距法和偏角法
  - 道格拉斯——普克法(Douglas—Peucker)
  - 光栏法

- For example:
  Point A (897‘345.32; 1’898‘765.98)
- 如果存在大量点
  - 平移坐标中心点



- Float -> short(4byte->2byte)

78.00	387.00
177.00	370.00
291.00	404.00
407.00	399.00
473.00	338.00
576.00	378.00
616.00	331.00
591.00	219.00
593.00	126.00
512.00	63.00
353.00	19.00
264.00	69.00
170.00	41.00
74.00	66.00
28.00	146.00
59.00	227.00
25.00	265.00
89.00	307.00

7800	38700
17700	37000
29100	40400
40700	39900
47300	33800
57600	37800
61600	33100
59100	21900
59300	12600
51200	6300
35300	1900
26400	6900
17000	4100
7400	6600
2800	14600
5900	22700
2500	26500
8900	30700

文本->二进制

7800	38700
17700	37000
29100	40400
40700	39900
47300	33800
57600	37800
61600	33100
59100	21900
59300	12600
51200	6300
35300	1900
26400	6900
17000	4100
7400	6600
2800	14600
5900	22700
2500	26500
8900	30700

位置: C:\
大小: 358 字节 (358 字节)
占用空间: 4.00 KB (4,096 字节)

原始大小

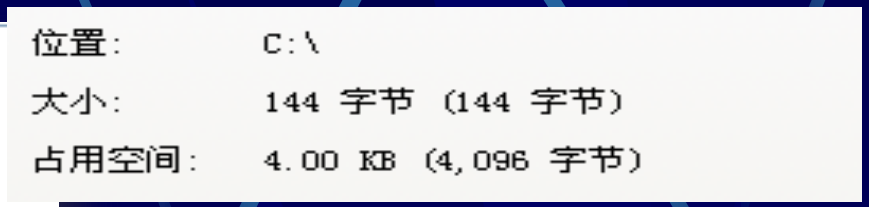
位置: C:\
大小: 144 字节 (144 字节)
占用空间: 4.00 KB (4,096 字节)

int 类型

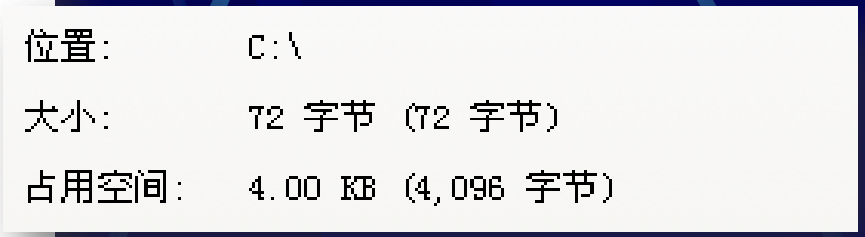
数据采用不同长度记录

7800	38700
17700	37000
29100	40400
40700	39900
47300	33800
57600	37800
61600	33100
59100	21900
59300	12600
51200	6300
35300	1900
26400	6900
17000	4100
7400	6600
2800	14600
5900	22700
2500	26500
8900	30700

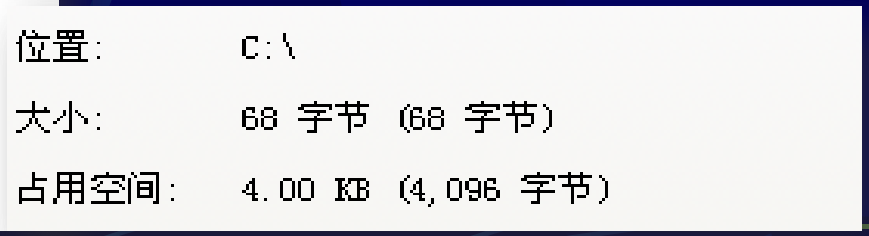
78	387
177	370
291	404
407	399
473	338
576	378
616	331
591	219
593	126
512	63
353	19
264	69
170	41
74	66
28	146
59	227
25	265
89	307



int 类型



short类型



不定长存储

78	387
177	370
291	404
407	399
473	338
576	378
616	331
591	219
593	126
512	63
353	19
264	69
170	41
74	66
28	146
59	227
25	265
89	307

-222	87
-123	70
-9	104
107	99
173	38
276	78
316	31
291	-81
293	-174
212	-237
53	-281
-36	-231
-130	-259
-226	-234
-272	-154
-241	-73
-275	-35
-211	7

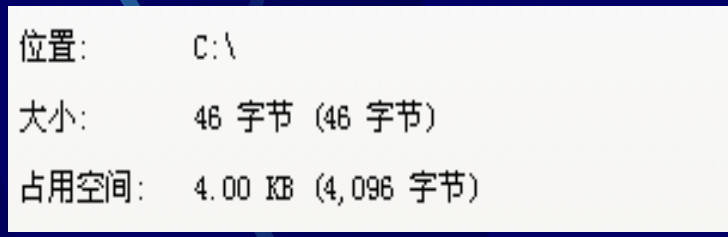
300为中心，不定长存储

位置:	C:\
大小:	64 字节 (64 字节)
占用空间:	4.00 KB (4,096 字节)

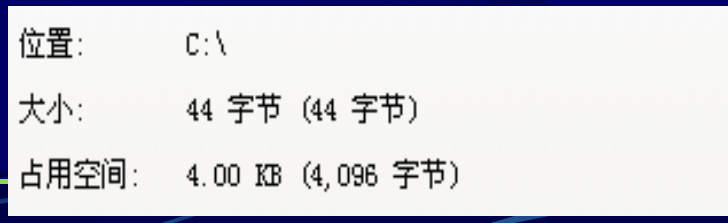
78	387
177	370
291	404
407	399
473	338
576	378
616	331
591	219
593	126
512	63
353	19
264	69
170	41
74	66
28	146
59	227
25	265
89	307

78	387
99	-17
114	34
116	-5
66	-61
103	40
40	-47
-25	-112
2	-93
-81	-63
-159	-44
-89	50
-94	-28
-96	25
-46	80
31	81
-34	38
64	42

标志位4字节长，大小变为46字节



标志位减为2字节长，大小变为44字节



78	387
177	370
291	404
407	399
473	338
576	378
616	331
591	219
593	126
512	63
353	19
264	69
170	41
74	66
28	146
59	227
25	265
89	307

78	387
99	-17
114	34
116	-5
66	-61
103	40
40	-47
-25	-112
2	-93
-81	-63
-159	-44
-89	50
-94	-28
-96	25
-46	80
31	81
-34	38
64	42

78	307
99	63
15	-29
2	-39
-50	-56
37	101
-63	-87
-65	-65
27	19
-83	30
-78	19
70	94
-5	-78
-2	53
50	55
77	1
-65	-43
98	4

前面一组坐标不变，占2*2共4个字节，后面所有占1*2*17共34个字节，共38字节