**《地理信息系统原理》阶段测验一**

**（2020-2021第1学期）**

**学号 07192329 姓名 武成龙 班级 地理信息科学19-1班**

一、单项选择题（本大题共10小题，每小题2分，共20分）

**1、将矢量数据转换为栅格数据的过程中，通常确定栅格数据结构中的栅格单元尺寸的原则是：（B）。**

**A. 数据量要小 B. 保证最小图斑不丢失**

**C. 时间要快 D. 表示所有地物**

**2、GIS区别于其它信息系统的一个显著标志是：（A）。**

**A. 空间分析 B. 计量分析**

**C. 属性分析 D. 统计分析**

**3、下列栅格结构编码方法中，具有可变分辨率和区域性质的是（D）。**

**A. 直接栅格编码 B. 链码**

**C. 游程编码 D. 四叉树编码**

**4、以线性四叉树表示8\*8的栅格矩阵时，第6行第5列位置处的栅格的十进制MORTON码与四进制MORTON码值分别为：（C）**

**A. 57，302 B. 50，302**

**C. 57，321 D. 50，321**

**5、采用骨架法进行细化时，某一栅格的值为1，该栅格的西、北西、南西、北、东和北东栅格的值分别为0、1、1、0、1、1，经骨架法计算后该栅格的值为（ B ）。**

**A. 1 B. 3 C. 0.75 D. 0.25**

**6、下列只能用于矢量数据压缩的方法是：（D）。**

**A. 压缩软件 B. 游程编码 C. 链码 D. Douglas-Peucker法**

**7、以下哪些不属于矢量数据结构的特点？（D）**

**A. 图形显示质量好 B. 多边形叠置分析较困难**

**C. 数据结构复杂 D. 图形数据量大**

**8、道路作为一种空间实体，是（D）**

**A. 对公路、铁路等现象的共同方面、本质属性与关系的抽象**

**B. 对各类复杂道路现象进行简化抽象得到的结果**

**C. 不可再分割的同类对象**

**D. 以上均正确**

**9、下列关于GIS与MIS的区别，说法正确的是:（C）。**

**A. GIS与MIS没有关系。**

**B. GIS是业务驱动的，MIS是数据驱动的。**

**C. GIS是数据驱动的，MIS是业务驱动的。**

**D. GIS建立在MIS基础之上。**

**10、空间图形的同类元素之间的拓扑关系称为：（ A）。**

**A. 拓扑邻接 B. 拓扑关联**

**C. 拓扑包含 D. 拓扑相交**

二、论述题及编程题（共3小题，每小题20分，共60分）

**1、GIS将现实世界转换到计算机世界，经过了哪些过程？涉及哪些理论、技术、方法？请就以上问题进行详细论述。（20分）**

根据三层次认知模型，从现实世界到计算机世界归纳为三个层次。第一个是从现实世界转换到概念数据模型，即地理空间认知模型，主要有三种方法：基于对象，基于网络和基于域。第二个层次是逻辑数据模型，这一阶段需要建立空间数据模型，如栅格数据模型、矢量数据模型、TIN模型，矢栅一体化模型、面向对象的数据模型等。各种数据模型又需要不同的数据结构，如矢量数据模型除了需要考虑编码存储，还需要考虑建立拓扑关系，由此主要有以下方法：实体式、索引式、双重独立式、链状双重独立式。栅格数据结构除了考虑直接存储外，还需要考虑高效压缩，由此主要有以下编码方式：直接编码、行程编码、块码、链式编码、四叉树编码。第三个层次是物理数据模型，也就是地理实体在计算机内部具体存储形式和操作机制。从物理表示组织的角度：层次模型的物理表示方法有物理邻接法、表结构法、目录法；网络模型的物理表示方法有变长指针法、位图法和目录法等；关系模型的物理表示通常用关系表来完成。从空间数据的存取的角度：主要有文件结构法、索引文件和点索引结构三种。文件结构法包括顺序结构、表结构和随机结构。

**2、如图1所示，V表示点，L表示线，P表示多边形。请给出图1的链状双重独立编码结构（分别写出多边形拓扑文件、弧段拓扑文件、弧段坐标文件）。根据建立的拓扑编码结构，按GIS数据的基本特征组织数据，编写绘制该图的程序（选一种自己熟悉的编程语言），分别存储到文件和数据库。对于点、线、面，创建类，最少包含一个属性、一个方法。（20分）**

V7

V6

V5

V8

V9

V10

V1

V11

V12

V4

V2

V3

V13

V16

V15

V14

L1

L2

L3

L4

L5

L6

L7

P4

P1

P2

P3

图1 原始多边形数据

|  |  |
| --- | --- |
| 多边形号 | 弧段号 |
| P1 | L1,L3,L6,-L7 |
| P2 | L2,L3,L5 |
| P3 | L4,L6,L5 |
| P4 | L7 |

|  |  |  |  |
| --- | --- | --- | --- |
| 弧段号 | 点号 | 左多边形 | 右多边形 |
| L1 | V1,V2,V3,V4 | P1 | 0 |
| L2 | V4,V5,V6 | P2 | 0 |
| L3 | V4,V12,V11,V9 | P2 | P1 |
| L4 | V6,V7,V1 | P3 | 0 |
| L5 | V9,V8,V6 | P3 | P2 |
| L6 | V1,V10,V9 | P1 | P3 |
| L7 | V13,V14,V15,V16,V13 | P4 | P1 |

|  |  |  |
| --- | --- | --- |
| 点号 | x坐标 | y坐标 |
| V1 | 400 | 400 |
| V2 | 1000 | 400 |
| V3 | 1000 | 0 |
| V4 | 600 | 0 |
| V5 | 100 | 0 |
| V6 | 100 | 150 |
| V7 | 100 | 400 |
| V8 | 175 | 150 |
| V9 | 300 | 200 |
| V10 | 375 | 350 |
| V11 | 425 | 185 |
| V12 | 500 | 120 |
| V13 | 800 | 325 |
| V14 | 850 | 225 |
| V15 | 800 | 100 |
| V16 | 750 | 200 |

import turtle

*#点类*

**class** point:

*#属性*

    CX = None

    CY = None

    PID = None

*#构造函数*

**def** \_\_init\_\_(self,list) -> None:

        self.PID = list[0]

        self.CX = int(list[1])

        self.CY = int(list[2])

*#返回名称*

**def** showID(self):

        return self.PID

*#返回坐标*

**def** showX(self):

        return self.CX

**def** showY(self):

        return self.CY

*#线类*

**class** line:

*#属性*

    LID = None

    Node = []

*#构造函数*

**def** \_\_init\_\_(self,list) -> None:

        self.LID = list[0]

        self.Node = list[1:]

**def** showID(self):

        return self.LID

*#读入点坐标*

with open('点坐标.txt','r') as file0:

    points = file0.readlines()

*#创建一个点对象列表*

    Point = []

*#遍历创建点对象*

for index,item in enumerate(points):

    list1 = item.split(',')

    temp = point(list1)

    Point.append(temp)

*# print("点创建成功")*

*#读入弧段坐标*

with open('弧段坐标.txt','r') as flie1:

    lines = flie1.readlines()

*##创建一个线对象列表*

    Line = []

*#遍历创建线对象*

for index,item in enumerate(lines):

    line1 = item.strip().split(',V')

*#创建一个空列表，存放弧段名和点对象*

    pointList = []

    pointList.append(line1[0])*#range(5)*

    for i in range(len(line1)-1):

        pointList.append(Point[int(line1[i+1])-1])

    temp = line(pointList)

    Line.append(temp)

*# print("线创建成功")*

*#画图*

turtle.screensize(3000,1000,"white")

turtle.setup(width=0.75, height=0.75, startx=None, starty=None)

turtle.color("green","yellow")

turtle.penup()

for line in Line:

    for point in line.Node:

        turtle.goto(point.showX()-500,point.showY()-200)*#减去一定数字是为了显示方便*

        turtle.write(str(point.showID()), font=('Arial', 15, 'normal'))

        turtle.pendown()

        turtle.dot(5,'blue')

    turtle.penup()

print()

'''

Python的turtle库画图很方便，但是没有封装好的四点椭圆函数。

优化方向：把一些比较复杂表示的换成临时变量。利用循环用短线段逼近曲线。

'''

**3、对图2编写线性四叉树压缩编码程序。（20分）**

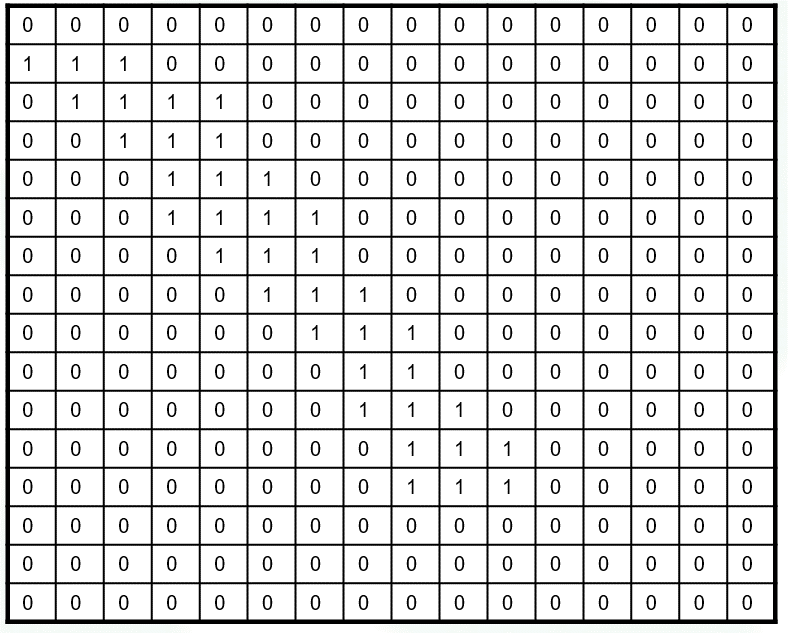


图2 原始栅格数据

import numpy as np

**class** Node:

    '''

    结点类定义

    若四个叶结点非空，val为\*；若空，val为false；若叶结点为0，就为0；若叶结点为1，就为1，

    '''

    val = False

    isLeaf = False

    topLeft = None

    topRight = None

    bottomLeft = None

    bottomRight = None

**def** \_\_init\_\_(self, val, isLeaf, topleft, topRight,

     bottomLeft, bottomRight) -> None:

        self.val = val

        self.isLeaf = isLeaf

        self.topLeft = topleft

        self.topRight = topRight

        self.bottomLeft = bottomLeft

        self.bottomRight = bottomRight

**def** construct(self, grid):

        """

        初始化四叉树

        :type grid: List[List[int]]

        :rtype: Node

        """

        if len(grid) == 1:      *#矩阵为1阶*

            return Node(grid[0][0], True, None, None, None, None)

        real = False

        fatal = False

        for row in grid:    *#判断当前矩阵的真假值*

            if True in row:

                real = True

            if False in row:

                fatal = True

            if real and fatal:

                break

        if real and not fatal:     *#当前矩阵为真*

            return Node(1, True, None, None, None, None)

        if not real and fatal:      *#当前矩阵为假*

            return  Node(0, True, None, None, None, None)

        if real and fatal:          *#当前矩阵既包含真值又包含假值*

            grid = np.array(grid)   *#二维列表转数组，方便下标操作*

            topleft = self.construct(grid[0:len(grid) // 2, 0:len(grid) // 2].tolist())

            topRight = self.construct(grid[0:len(grid) // 2, len(grid) // 2:].tolist())

            bottomLeft = self.construct(grid[len(grid) // 2:, 0:len(grid) // 2].tolist())

            bottomRight = self.construct(grid[len(grid) // 2:, len(grid) // 2:].tolist())

            return Node("\*", False, topleft, topRight, bottomLeft, bottomRight)

**def** show\_node(self,show\_list):

*# 递归读取*

        list = show\_list.copy()

        if self.val != False:

            list.append(self.val)

            if self.isLeaf == False:

                new\_list = []

                list.append(self.topLeft.show\_node(new\_list.copy()))

                list.append(self.topRight.show\_node(new\_list.copy()))

                list.append(self.bottomLeft.show\_node(new\_list.copy()))

                list.append(self.bottomRight.show\_node(new\_list.copy()))

        return list

**def** read\_list():

    with open('四叉树.txt','r') as file:

        lines = file.readlines()

        grid = []

        for item in lines:

            grid1 = []

            item = item.strip()

            for i in item:

                grid1.append(int(i))

            grid.append(grid1)

    return grid

read\_txt = read\_list()

show\_list = []

tree = Node(False,False,None,None,None,None)

tree = tree.construct(read\_txt)

result = tree.show\_node(show\_list)

'''

建立的数组很理想，但是也有些问题。

调试可知，result数组遇到非叶结点先存放\*，然后新建一个数组，存放叶结点。

仍然存在的问题在于，递归输出函数本质上创建了新数组，而其中空结点占用了许多空数组，造成空间浪费。

'''

print(result)