1. 选择题

B B C D A

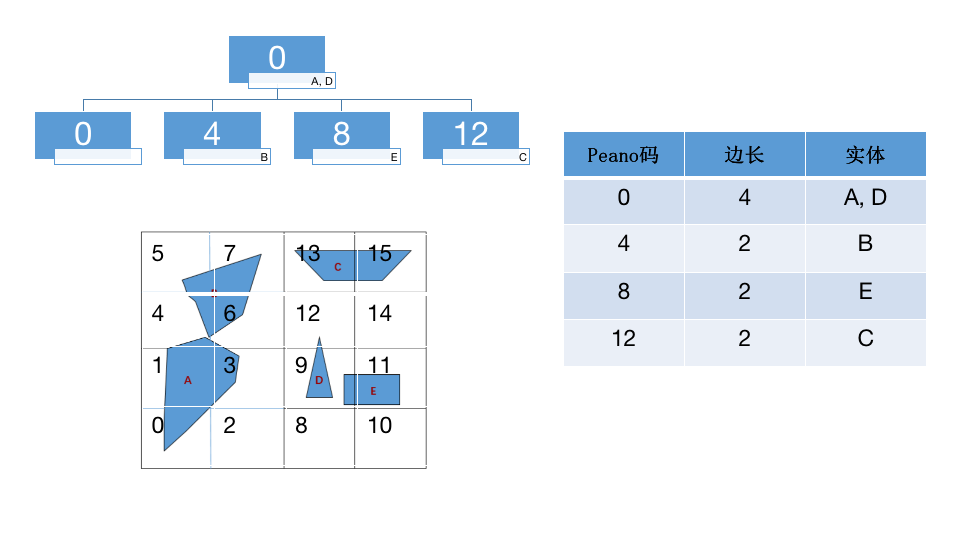
D D D D A

A A D C A

C A A C C

1. 分析题

1、



①邻接关系：

面：

P1/P2,P3;P2/P1,P3;P3/P1,P2,P4;P4/P3

点：

N1/N2,N6;N2/N1,N3,N7;N3/N2,N4,N7;N4/N3,N5;N5/N4,N6;N6/N1,N5,N7;N7/N2,N3,N6;

N8/N9,N10;N9/N8,N10;N10/N8,N9

线：

A1/A2,A6,A8;A2/A1,A8,A9,A3;A3/A2,A9,A4;A4/A3,A5;A5/A6,A7,A4;A6/A1,A7,A5;

A7/A8,A9,A6,A5;A8/A1,A2,A7,A9;A9/A2,A3,A8,A7;A10/A11,A12;A11/A10,A12;

A12/A10,A11

②关联关系：

点/线：

N1/A1,A6;N2/A1,A8,A2;N3/A2,A9,A3;N4/A3,A4;N5/A5,A4;N6/A6,A7,A5;

N7/A7,A8,A9;N8/A10,A11;N9/A11,A12;N10/A10,A12

线/面：

A1/P1;A2/P2;A3/P3;A4/P3;A5/P3;A6/P1;A7/P1,P3;A8/P1,P2;A9/P2,P3;A10/P3,P4;

A11/P3,P4;A12/P3,P4

线/点：

A1/N1,N2;A2/N2,N3;A3/N3,N4;A4/N4,N5;A5/N5,N6;A6/N1,N6;A7/N6,N7;A8/N2,N7;

A9/N7,A13;A10/N8,N10;A11/N8,N9;A12/N10,N9

面/线：

P1/A1,A8,A7,A6;P2/A2,A8,A9;P3/A3,A4,A5,A7,A9,-A10,-A11,-A12;P4/A10,A11,A12

③包含关系：

P4包含于P3

④联通关系：

A1与A2连通；A2与A3连通；A3与A4连通；A4与A5连通；A5与A6连通；A10与A11连通；A11与A12连通

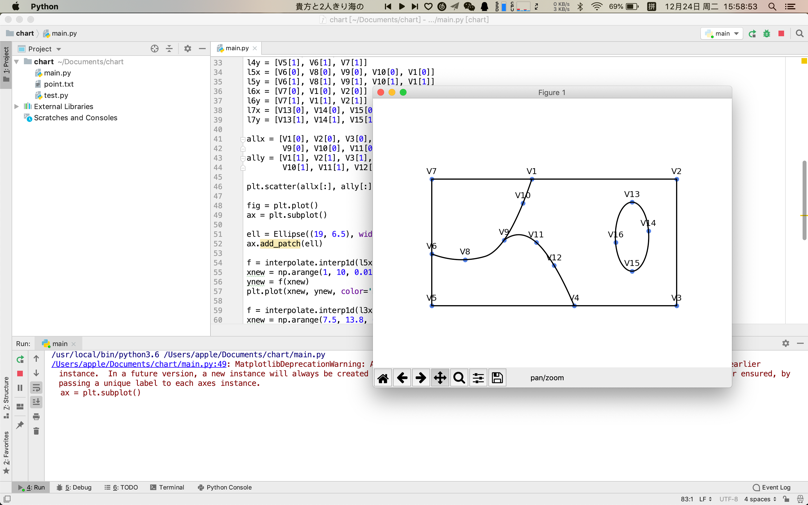
3、

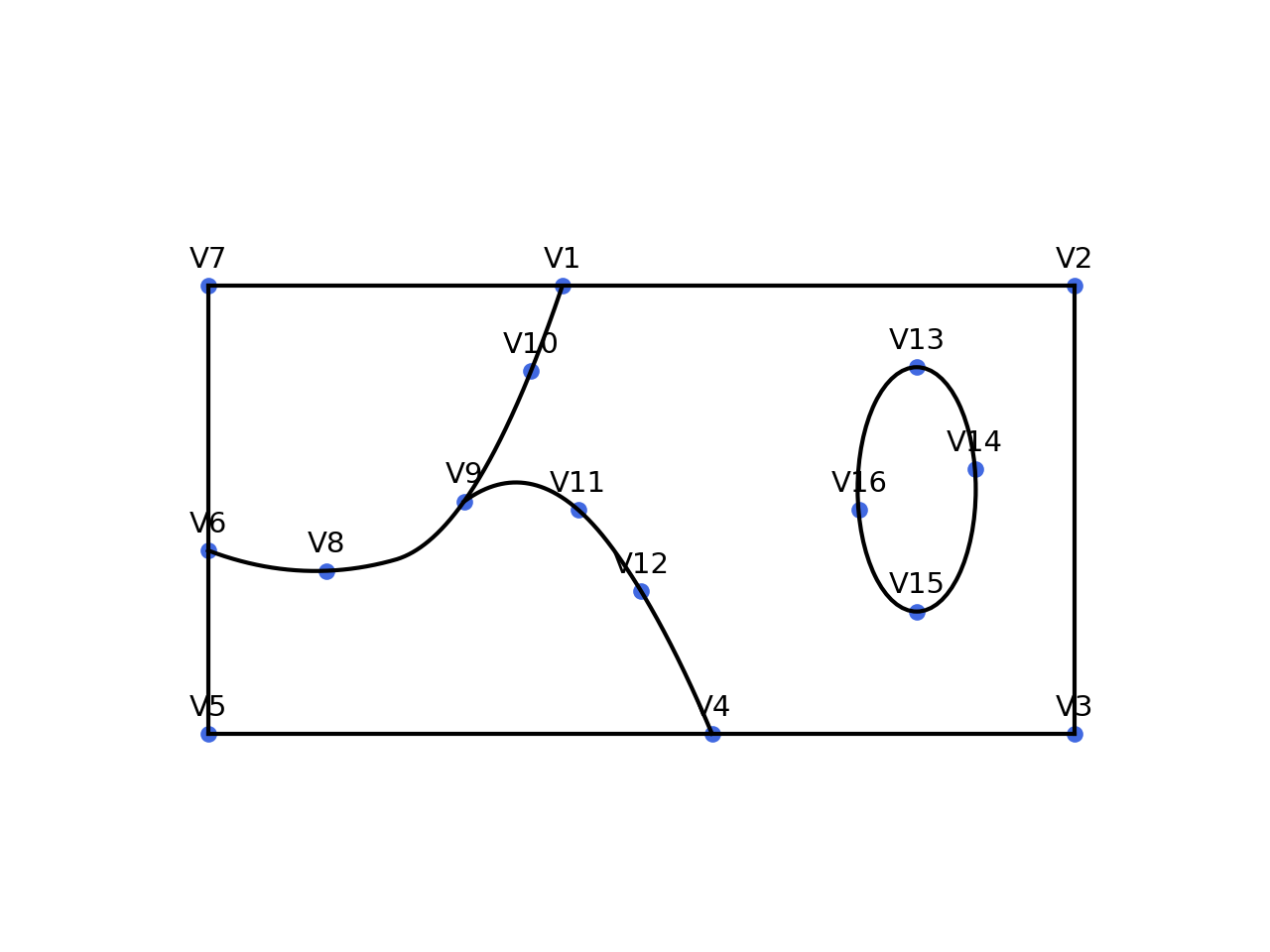
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 多边形拓扑文件 | | | | | | |
| 多边形ID | 弧段号 | | | | 属性 | |
| P1 | L1,L3,L6,-L7 | | | |  | |
| P2 | L2,L5,L3 | | | |  | |
| P3 | L4,L6,L5 | | | |  | |
| P4 | L7 | | | |  | |
| 弧段拓扑文件 | | | | | | | | |
| 弧段ID | | | 起始点 | 终结点 | | 左多边形 | | 右多边形 |
| L1 | | | V1 | V4 | | O | | P1 |
| L2 | | | V4 | V6 | | O | | P2 |
| L3 | | | V4 | V9 | | P2 | | P1 |
| L4 | | | V6 | V1 | | O | | P3 |
| L5 | | | V6 | V9 | | P3 | | P2 |
| L6 | | | V1 | V9 | | P1 | | P3 |
| L7 | | | V13 | V13 | | P1 | | P4 |
| 弧段坐标文件 | | | | | | | | | |
| 弧段ID | | 坐标串 | | | | | | | |
| L1 | | (x1,y1),(x2,y2),(x3,y3),(x4,y4) | | | | | | | |
| L2 | | (x4,y4),(x5,y5),(x6,y6) | | | | | | | |
| L3 | | (x4,y4),(x12,y12),(x11,y11),(x9,y9) | | | | | | | |
| L4 | | (x6,y6),(x7,y7),(x1,y1) | | | | | | | |
| L5 | | (x6,y6),(x8,y8),(x9,y9) | | | | | | | |
| L6 | | (x1,y1),(x10,y10),(x9,y9) | | | | | | | |
| L7 | | (x13,y13),(x14,y14),(x15,y15),(x16,y16),(x13,y13) | | | | | | | |

程序如下：

*# -\*- coding:utf-8 -\*-***import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** matplotlib.patches **import** Ellipse  
**from** scipy **import** interpolate  
  
plt.figure()  
  
V1 = [10, 11.5]  
V2 = [23, 11.5]  
V3 = [23, 0.5]  
V4 = [13.8, 0.5]  
V5 = [1, 0.5]  
V6 = [1, 5]  
V7 = [1, 11.5]  
V8 = [4, 4.5]  
V9 = [7.5, 6.2]  
V10 = [9.2, 9.4]  
V11 = [10.4, 6]  
V12 = [12, 4]  
V13 = [19, 9.5]  
V14 = [20.48, 7]  
V15 = [19, 3.5]  
V16 = [17.54, 6]  
  
l1x = [V2[0], V3[0]]  
l1y = [V2[1], V3[1]]  
l2x = [V3[0], V4[0], V5[0]]  
l2y = [V3[1], V4[1], V5[1]]  
l3x = [V9[0], V11[0], V12[0], V4[0]]  
l3y = [V9[1], V11[1], V12[1], V4[1]]  
l4x = [V5[0], V6[0], V7[0]]  
l4y = [V5[1], V6[1], V7[1]]  
l5x = [V6[0], V8[0], V9[0], V10[0], V1[0]]  
l5y = [V6[1], V8[1], V9[1], V10[1], V1[1]]  
l6x = [V7[0], V1[0], V2[0]]  
l6y = [V7[1], V1[1], V2[1]]  
l7x = [V13[0], V14[0], V15[0], V16[0]]  
l7y = [V13[1], V14[1], V15[1], V16[1]]  
  
allx = [V1[0], V2[0], V3[0], V4[0], V5[0], V6[0], V7[0], V8[0],  
 V9[0], V10[0], V11[0], V12[0], V13[0], V14[0], V15[0], V16[0]]  
ally = [V1[1], V2[1], V3[1], V4[1], V5[1], V6[1], V7[1], V8[1], V9[1],  
 V10[1], V11[1], V12[1], V13[1], V14[1], V15[1], V16[1]]  
  
plt.scatter(allx[:], ally[:], 25, **"royalblue"**)  
  
fig = plt.plot()  
ax = plt.subplot()  
  
ell = Ellipse((19, 6.5), width=6, height=3, angle=90.0, linewidth=1.5, fill=**False**)  
ax.add\_patch(ell)  
  
f = interpolate.interp1d(l5x, l5y, kind=**'quadratic'**)  
xnew = np.arange(1, 10, 0.01)  
ynew = f(xnew)  
plt.plot(xnew, ynew, color=**'black'**)  
  
f = interpolate.interp1d(l3x, l3y, kind=**'quadratic'**)  
xnew = np.arange(7.5, 13.8, 0.01)  
ynew = f(xnew)  
plt.plot(xnew, ynew, color=**'black'**)  
  
plt.plot(l6x, l6y, **"black"**)  
plt.plot(l1x, l1y, **"black"**)  
plt.plot(l2x, l2y, **"black"**)  
plt.plot(l4x, l4y, **"black"**)  
  
plt.xlim((0, 25))  
plt.ylim((0, 18))  
my\_x\_ticks = np.arange(0, 25, 5)  
my\_y\_ticks = np.arange(0, 18, 5)  
plt.xticks(my\_x\_ticks)  
plt.yticks(my\_y\_ticks)  
ID = 0  
**for** x, y **in** zip(allx, ally):  
 ID += 1  
 plt.text(x, y+0.3, **"V{}"**.format(ID), ha=**'center'**, va=**'bottom'**, fontsize=10.5)  
plt.xticks([])  
plt.yticks([])  
plt.axis(**'off'**)  
plt.show()

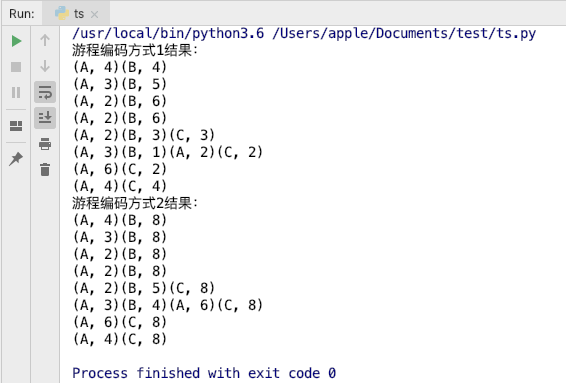
输出结果如下：





4、

*# -\*- coding:utf-8 -\*-*L1 = [**'A'**, **'A'**, **'A'**, **'A'**, **'B'**, **'B'**, **'B'**, **'B'**]  
L2 = [**'A'**, **'A'**, **'A'**, **'B'**, **'B'**, **'B'**, **'B'**, **'B'**]  
L3 = [**'A'**, **'A'**, **'B'**, **'B'**, **'B'**, **'B'**, **'B'**, **'B'**]  
L4 = [**'A'**, **'A'**, **'B'**, **'B'**, **'B'**, **'B'**, **'B'**, **'B'**]  
L5 = [**'A'**, **'A'**, **'B'**, **'B'**, **'B'**, **'C'**, **'C'**, **'C'**]  
L6 = [**'A'**, **'A'**, **'A'**, **'B'**, **'A'**, **'A'**, **'C'**, **'C'**]  
L7 = [**'A'**, **'A'**, **'A'**, **'A'**, **'A'**, **'A'**, **'C'**, **'C'**]  
L8 = [**'A'**, **'A'**, **'A'**, **'A'**, **'C'**, **'C'**, **'C'**, **'C'**]  
I = [L1, L2, L3, L4, L5, L6, L7, L8]  
  
  
**def** ycbm1(L):  
 code = []  
 last = L[0]  
 c = 0  
 **for** i **in** range(len(L)):  
 **if** L[i] == last:  
 c += 1  
 **else**:  
 code.append(**'('**)  
 code.append(last)  
 code.append(**', '**)  
 code.append(str(c))  
 code.append(**')'**)  
 c = 1  
 last = L[i]  
  
 code.append(**'('**)  
 code.append(last)  
 code.append(**', '**)  
 code.append(str(c))  
 code.append(**')'**)  
 **return ''**.join(code)  
  
  
**def** ycbm2(L):  
 code = []  
 last = L[0]  
 c = 0  
 **for** i **in** range(len(L)):  
 **if** L[i] == last:  
 c += 1  
 **else**:  
 code.append(**'('**)  
 code.append(last)  
 code.append(**', '**)  
 code.append(str(c))  
 code.append(**')'**)  
 c += 1  
 last = L[i]  
  
 code.append(**'('**)  
 code.append(last)  
 code.append(**', '**)  
 code.append(str(c))  
 code.append(**')'**)  
 c += 1  
 **return ''**.join(code)  
  
  
print(**"游程编码方式1结果："**)  
**for** i **in** range(0, len(I)):  
 print(ycbm1(I[i]))  
print(**"游程编码方式2结果："**)  
**for** i **in** range(0, len(I)):  
 print(ycbm2(I[i]))



线性四叉树压缩结果：

|  |  |  |  |
| --- | --- | --- | --- |
| MD码 | 属性值 | MD码 | 属性值 |
| 0 | A | 40 | A |
| 7 | B | 48 | B |
| 8 | A | 49 | C |
| 12 | B | 50 | A |
| 32 | A | 52 | C |
| 36 | B | 56 | A |
| 38 | A | 58 | C |
| 39 | B |  |  |

5、

points = []  
tol = 1 *# 设置阈值*

**def** Douglas(self, p1, p2):  
 A = (p1.y - p2.y)  
 B = (p2.x - p1.x)  
 C = (p1.x \* p2.y - p2.x \* p1.y) *# 直线的一般式: A \* x + B \* y + C = 0*

m = points.index(p1)  
 n = points.index(p2) *# 存储待计算点的位置信息*

distance = [] *# 存储所有中间点到直线的距离信息* middle = **None** *# 初始化后来需要使用的中间点  
 # 设置基线条件，即计算点相邻时停止递归* **if** n == m + 1:  
 **return** *# 计算各中间点到直线的距离且依次存储在distance列表里* **for** i **in** range(m + 1, n):  
 d = abs(A \* points[i].x + B \* points[i].y + C) /

math.sqrt(A \* A + B \* B)  
 distance.append(d)  
  
 dmax = max(distance)  
  
 **if** dmax < tol:  
 **for** i **in** range(m + 1, n):  
 **del** points[i] *# 若距离小于阈值则在内存中删除该点* **else**:  
 **for** i **in** range(m + 1, n):  
 **if** abs(A \* points[i].x + B \* points[i].y + C) /

math.sqrt(A \* A + B \* B) == dmax:  
 middle = points[i]  
 self.Douglas(p1, middle)  
 self.Douglas(middle, p2) *# 若距离大于阈值则重置首尾点，对新的两段线递归*

6、

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 21 | 23  A | 29 | 31 | 53 | 55 | 61 | 63 |
| 20 | 22 | 28 | 30 | 52 | 54 | 60 | 62 |
| 17 | 19  B | 25 | 27 | 49 | 51 | 57 | 59 |
| 16 | 18 | 24  C | 26 | 48  D | 50 | 56 | 58 |
| 5 | 7 | 13 | 15 | 37 | 39 | 45 | 47 |
| 4 | 6 | 12 | 14 | 36 | 38 | 44 | 46 |
| 1 | 3 | 9 | 11 | 33 | 35 | 41 | 43 |
| 0 | 2 | 8 | 10 | 32 | 34 | 40 | 42 |

|  |  |  |  |
| --- | --- | --- | --- |
| Peano码 | 空间对象 | Peano码 | 空间对象 |
| 5 | B | 25 | A |
| 7 | B, C | 26 | D |
| 13 | C | 27 | A |
| 15 | C | 28 | A |
| 18 | B | 37 | D |
| 19 | B | 39 | D |
| 22 | A | 48 | D |

第一阶段：去掉B

第二阶段：去掉A

第三阶段：没有去掉

第四阶段：去掉C

7、

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *V0* | *V1* | *V2* | *V3* | *V4* | *V5* | *V6* | *V7* | *V8* |
| *V0* | *0* | *5* | *∞* | *4* | *8* | *∞* | *∞* | *∞* | *∞* |
| *V1* | *5* | *0* | *2* | *∞* | *∞* | *∞* | *∞* | *∞* | *∞* |
| *V2* | *∞* | *2* | *0* | *7* | *∞* | *∞* | *∞* | *3* | *∞* |
| *V3* | *4* | *∞* | *7* | *0* | *4* | *8* | *6* | *∞* | *∞* |
| *V4* | *8* | *∞* | *∞* | *4* | *0* | *4* | *∞* | *∞* | *∞* |
| *V5* | *∞* | *∞* | *∞* | *8* | *4* | *0* | *5* | *∞* | *15* |
| *V6* | *∞* | *∞* | *∞* | *6* | *∞* | *5* | *0* | *10* | *∞* |
| *V7* | *∞* | *∞* | *3* | *∞* | *∞* | *∞* | *10* | *0* | *12* |
| *V8* | *∞* | *∞* | *∞* | *∞* | *∞* | *15* | *∞* | *12* | *0* |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *V0* | *V1* | *V2* | *V3* | *V4* | *V5* | *V6* | *V7* | *V8* |
| *标定V0* | *0* | *(5)* | *(∞)* | *(4)* | *(8)* | *(∞)* | *(∞)* | *(∞)* | *(∞)* |
| *标定V3* | *0* | *(5)* | *(11)* | *4* | *(8)* | *(12)* | *(10)* | *(∞)* | *(∞)* |
| *标定V4* | *0* | *(5)* | *(11)* | *4* | *8* | *(12)* | *(10)* | *(∞)* | *(27)* |
| *标定V5* | *0* | *(5)* | *(11)* | *4* | *8* | *12* | *10* | *(∞)* | *(27)* |
| *标定V6* | *0* | *(5)* | *(11)* | *4* | *8* | *12* | *10* | *(20)* | *(27)* |
| *标定V1* | *0* | *5* | *(7)* | *4* | *8* | *12* | *10* | *(20)* | *(27)* |
| *标定V2* | *0* | *5* | *7* | *4* | *8* | *12* | *10* | *(10)* | *(27)* |
| *标定V7* | *0* | *5* | *7* | *4* | *8* | *12* | *10* | *10* | *(22)* |
| *标定V8* | *0* | *5* | *7* | *4* | *8* | *12* | *10* | *10* | *22* |

8、

面状符号在地图视图中往往占有很大的比重，所以为了能够更好的认知地图，面状符号需要具有较高的辨识度，色彩需要选用常理认知范围内的，在图域内的面状符号需要有相近的饱和度与明度；形状方面需要选择简单易画，且让人易猜得代表地物的；大小既需要与代表地物特征匹配，又需要在整幅图内不喧兵夺主。在定义时也可以参考之前的标准进行定义。边缘线型的选择也会对认知地图造成影响。

9、

1）客观地貌的变化性

2）来自仪器度量精度的误差

3）地理数据处理时误差的累积

4）偶然误差

10、

空间样本点分布的情况与密度是选择插值方法的重要依据，而在如克里金插值等方法中，选择不同的函数、幂又会对结果造成很大的影响，如选择距离权重时若地图边缘样本点数量较少，则结果正确性会很低，而在样本点数量过少的时候，极端条件下若样本点数量为1，要分析的范围极大，则无论使用什么方法都是没有可信度的。

三、

