



**地理信息系统实验报告**

|  |
| --- |
| **学生姓名： 刘欣豪** |
| **学 号： 2020112921** |
| **班 级： 交通4班** |
| **实验日期： 2022.12** |

1. 实验目的
2. 熟悉PyQGIS的各种函数操作，对读入与写入进行代码编写。
3. 对地理信息系统知识进行有效复习。
4. 融会贯通本课程的知识与操作。
5. 实验环境
   1. 硬件环境
6. 计算机：OMEN by HP Laptop 15
7. CPU: Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz
8. RAM：16GB
   1. 软件环境
9. 操作系统：Windows10家庭中文版
10. 开发工具：QGIS Desktop 3.26.2
11. 实验内容
12. **补齐代码中在read\_shpFile和writeShp两个函数的内容**

read\_shpFile函数采用课堂所给标准代码进行存储转化。

writeShp函数中加入o与d参数表示起讫点用于保存命名存储路径为 ”//output//shortestpath”+o+”\_”+d+”.shp”。

1. **利用Chicago Sketch网络和5条最短路径计算并进行制图展示**

加入random模块，设置random.seed(4)，利用random.randint()得到起讫点。随机选择5个起讫点进行最短路计算并制图展示

更改代码为：

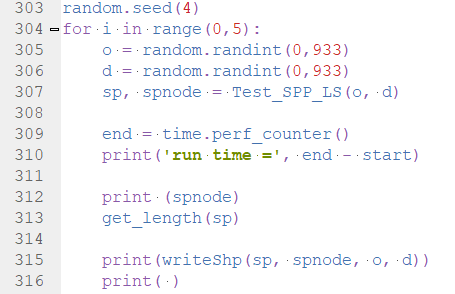


图1：主循环代码

输出为

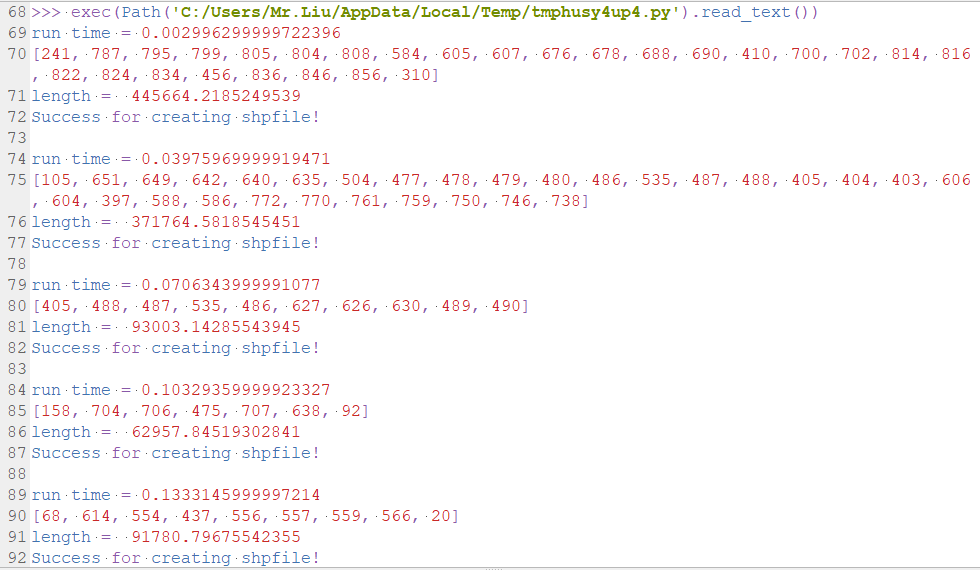


图2：输出

即**Chicago Sketch原网络图为：**

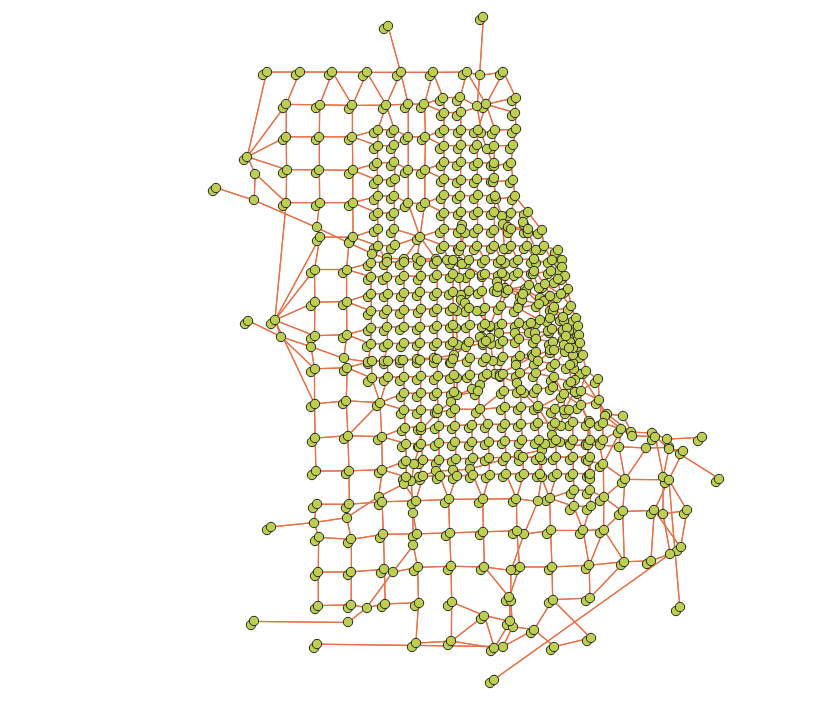


图3：**Chicago Sketch原网络图**

* **od对（68，20）最短路径**

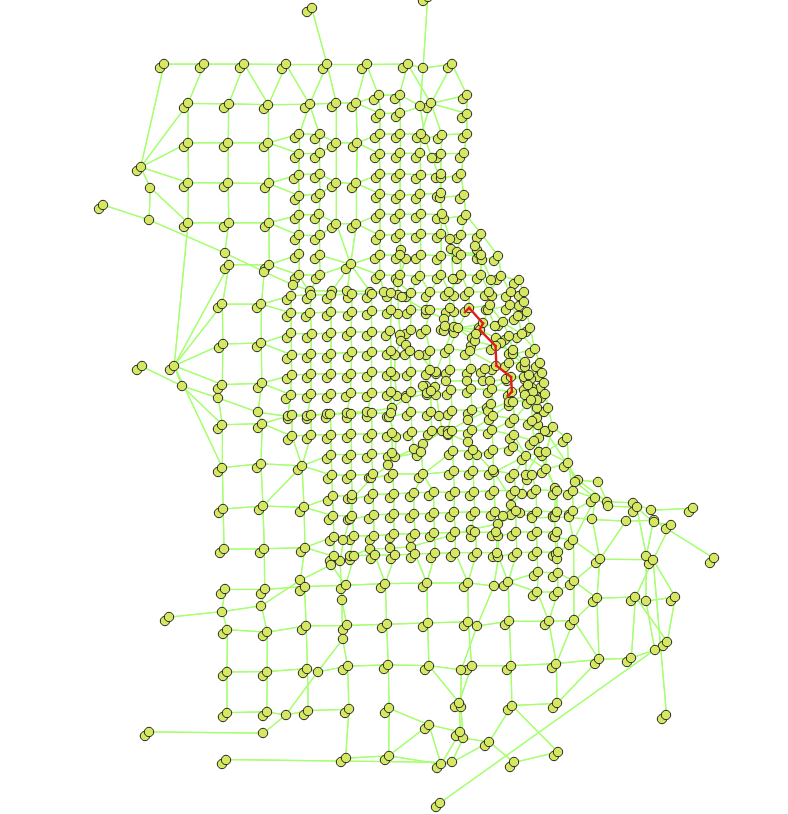


图4.1：**od对（68，24）最短路径**

* **od对（105，738）最短路径**

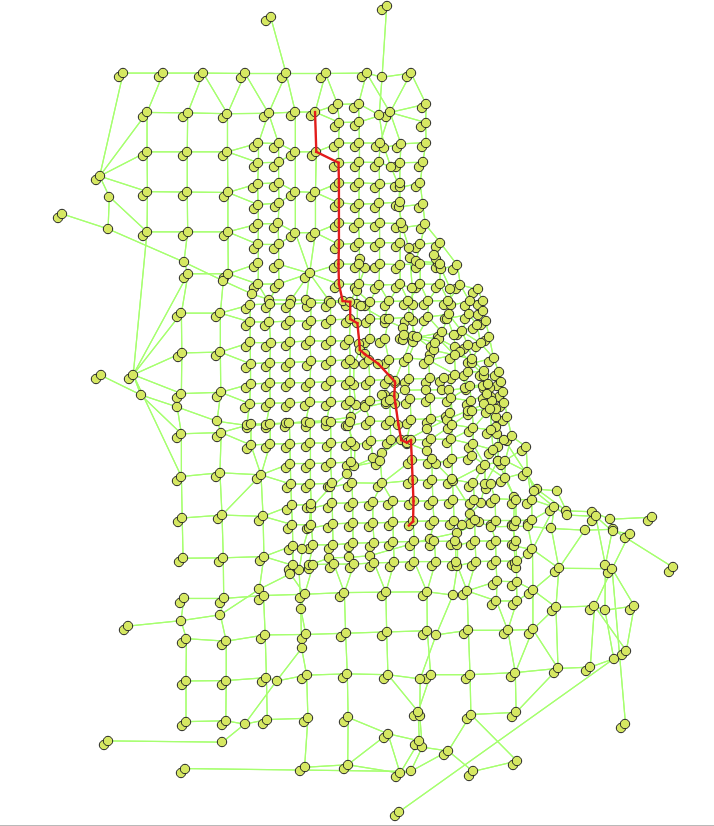


图4.2：**od对（105，738）最短路径**

* **od对（158，92）最短路径**

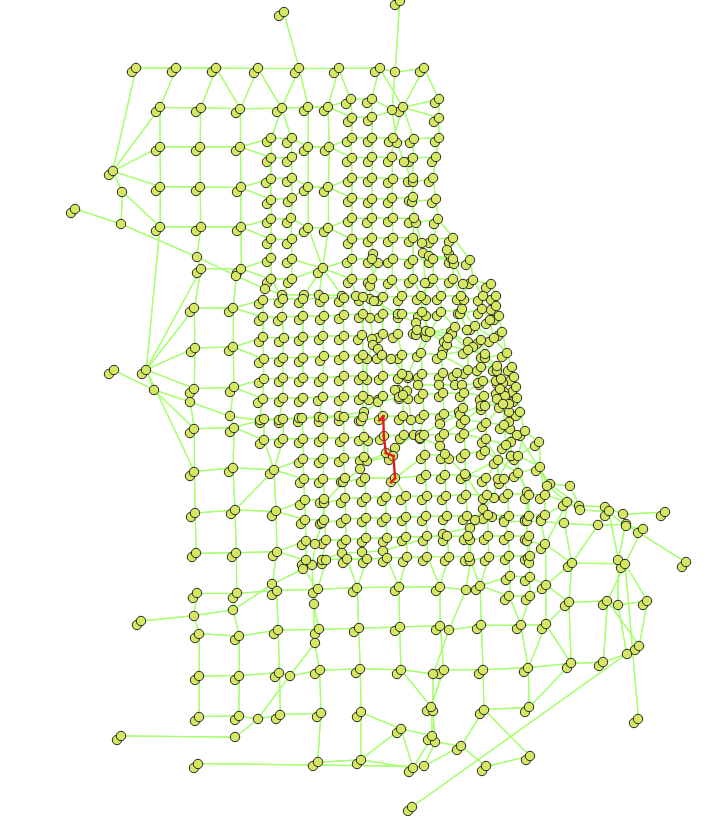


图4.3：**od对（158，92）最短路径**

* **od对（241，310）最短路径**

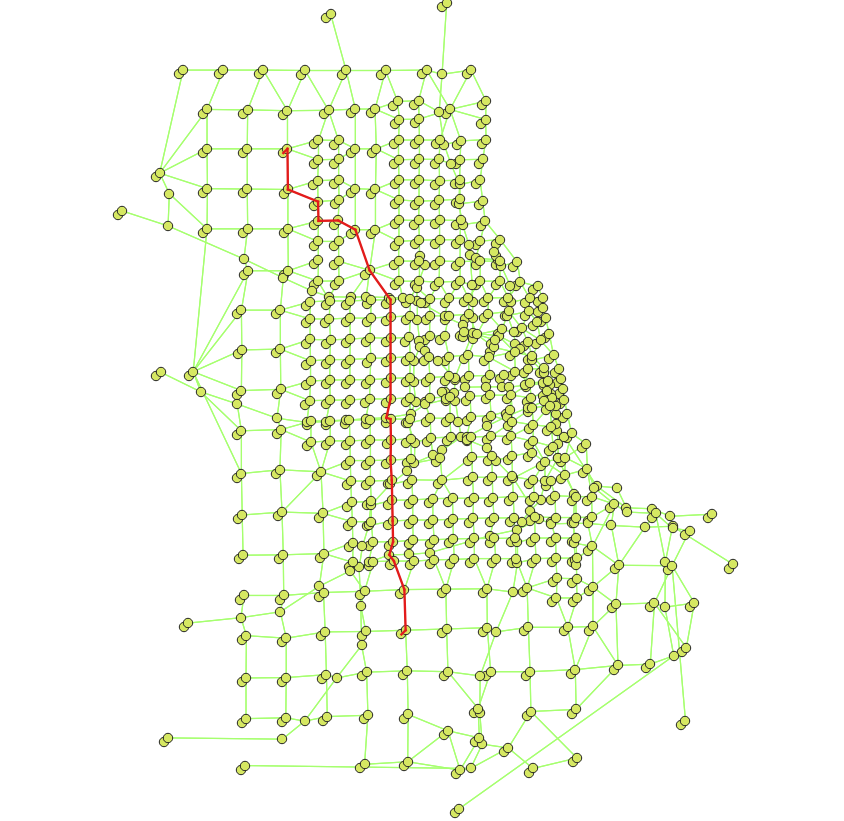


图4.4：**od对（241，310）最短路径**

* **od对（405，490）最短路径**

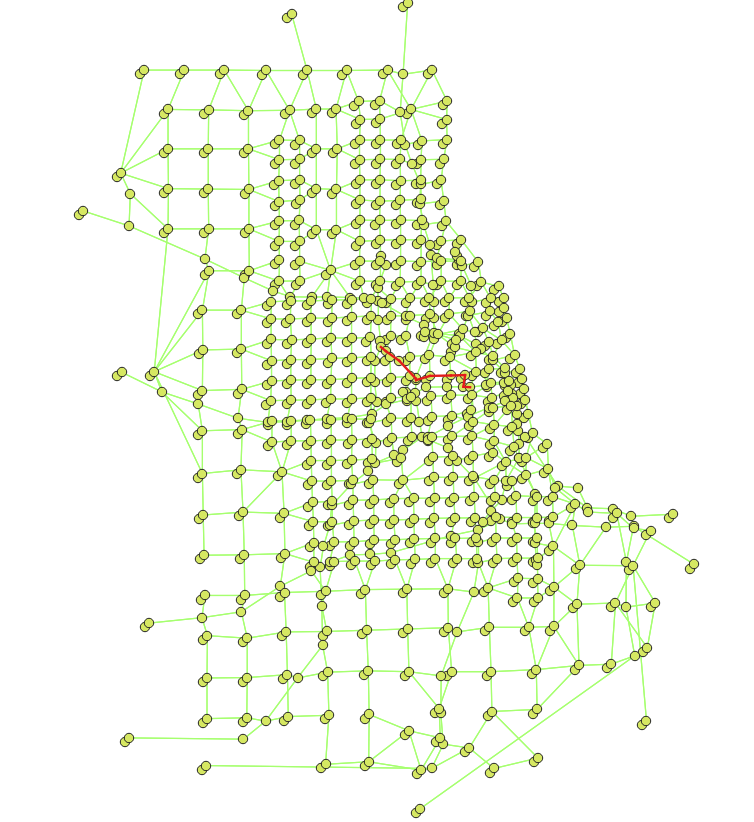


图4.5：**od对（405，490）最短路径**

1. 实验总结

该次实验结合本课程所学知识，充分锻炼了动手能力，提升了交通素养，在制图方面取得了新的进步。十分感谢老师提供的资源与讲解，这门课程对我的帮助巨大，在其它项目的地图可视化工作中得以更好的体现所要反映的内容。

**附录：代码**

1. **import** time
2. **import** math
3. **import** sys
4. **import** random
5. **from** qgis.core **import** \*
7. """
8. 类名：首字母大写 Node Link Origin Destination
9. 承载类的对象的容器：全部字母大写 NODE LINK ORIGIN
10. 其中NODE、LINK为list，0位置废掉，下标代表编号
11. ORIGIN为dictionary，下标代表编号
12. ORIGIN[i].destination为装载Destination对象的字典，ORIGIN[i].destination[j]代表以i为O，
13. 以j为D的Destination对象
14. """
15. ##################################
16. #建立Node,Link,Origin,Destination类
17. ##################################
18. """
19. 属性：
20. node\_id 节点编号; l\_in 流入节点的路段对象构成的列表; l\_out 流出节点的路段对象构成的列表;
21. u 最短路中该节点cost标号; p 最短路中该节点前序节点标号
22. 方法：
23. set\_l\_in 向流入路段列表末尾添加一个路段对象; set\_l\_out 向流出路段列表末尾添加一个路段对象;
24. set\_SPP\_u 设置cost标号; set\_SPP\_p 设置前序节点标号
25. """
26. DEBUGMODE=True
27. #import time
28. **class** Node:
29. **def** \_\_init\_\_(self, node\_id, l\_in\_empty, l\_out\_empty, X, Y):
30. self.node\_id = node\_id
31. self.l\_in = l\_in\_empty
32. self.l\_out = l\_out\_empty
33. self.X = X
34. self.Y = Y
35. **def** set\_l\_in(self, l\_in):
36. self.l\_in.append(l\_in)
37. **def** set\_l\_out(self, l\_out ):
38. self.l\_out.append(l\_out)
39. **def** set\_SPP\_u(self, u):
40. self.u = u
41. **def** set\_SPP\_p(self,p):
42. self.p = p
43. **def** set\_X(self, X):
44. self.X = X
45. **def** set\_Y(self, Y):
46. self.Y = Y
47. **def** set\_Astar\_g(self, g):
48. self.g = g
49. **def** set\_Astar\_h(self, h):
50. self.h = h
51. **def** set\_Astar\_f(self, f):
52. self.f = f
54. **class** Link:
55. **def** \_\_init\_\_(self, link\_id, tail\_node, head\_node, length):
56. self.liAnk\_id = link\_id
57. self.tail\_node = tail\_node
58. self.head\_node = head\_node
59. self.length = length
61. ##################################
62. #读取网络文件
63. ##################################
64. **def** read\_shpFile(Link\_layer, Node\_layer, Nfilename, Lfilename):
65. #补充完整代码
66. Link\_layer = QgsVectorLayer(Link\_layer, Lfilename,'ogr')
67. Node\_layer = QgsVectorLayer(Node\_layer, Nfilename, 'ogr')
69. Link\_features = Link\_layer.getFeatures()
70. Node\_features = Node\_layer.getFeatures()
72. NODE\_COUNT = 0
73. list2 = []
74. **for** feature **in** Node\_features:
75. NODE\_COUNT = NODE\_COUNT + 1
76. geom = feature.geometry()
77. x = geom.asPoint()
78. list2.append(x)
80. NODE = [Node(i + 1, [], [], list2[i][0],list2[i][1]) **for** i **in** range(NODE\_COUNT)]
81. NODE.insert(0,0)
83. LINK\_COUNT = 0
84. list = []
85. **for** feature **in** Link\_features:
86. LINK\_COUNT = LINK\_COUNT + 1
87. geom = feature.geometry()
88. x = geom.asMultiPolyline()
90. find\_tail = False
91. find\_head = False
92. **for** node **in** NODE[1:]:
93. **if** (x[0][0][0] == node.X **and** x[0][0][1]==node.Y):
94. find\_tail = True
95. tail\_id = node.node\_id
96. **if** (x[0][1][0] == node.X **and** x[0][1][1]==node.Y):
97. find\_head = True
98. head\_id = node.node\_id
99. **if** (find\_tail **and** find\_head):
100. **break**
101. list.append([tail\_id, head\_id, geom.length()])
103. LINK = [Link(i+1, list[i][0],list[i][1], list[i][2]) **for** i **in** range(LINK\_COUNT)]
104. LINK.insert(0,0)
106. **for** i **in** range(1,LINK\_COUNT+1):
107. NODE[LINK[i].tail\_node].set\_l\_out(LINK[i])
108. NODE[LINK[i].head\_node].set\_l\_in(LINK[i])
109. **return** (LINK,NODE,NODE\_COUNT,LINK\_COUNT)
111. **def** SPP\_CA(o\_id, node):
112. node\_num = len(node)
113. node[o\_id].set\_SPP\_u(0)
114. **for** t **in** node[1:]:
115. t.set\_SPP\_p(-1)
116. **if** t.node\_id != o\_id:
117. t.set\_SPP\_u(float('inf'))
118. **for** n **in** range(node\_num):#和LC算法区别在于，计算复杂度为O(mn)
119. **for** i **in** node[1:node\_num+1]:
120. **for** ij **in** i.l\_out:
121. j = node[ij.head\_node]
122. **if** j.u > (i.u+ij.length):
123. j.u = (i.u+ij.length)
124. j.p = i#id还是对象
125. shortestpath\_p\_list = [0]
126. **for** t **in** node[1:]:
127. shortestpath\_p\_list.append(t.p)
128. **return** shortestpath\_p\_list
130. **def** SPP\_LC(o\_id,node):
131. #initialize
132. node[o\_id].set\_SPP\_u(0)
133. **for** t **in** node[1:]:
134. t.set\_SPP\_p(-1)
135. **if** t.node\_id != o\_id:
136. t.set\_SPP\_u(float('inf'))
137. #mainloop
138. Q = [node[o\_id]]
139. **while** len(Q) != 0:
140. i = Q[0]
141. **del** Q[0]
142. **for** ij **in** i.l\_out:
143. j = node[ij.head\_node]
144. **if** j.u > (i.u+ij.length):
145. j.u = (i.u+ij.length)
146. j.p = i#id还是对象
147. **if** j **not** **in** Q:
148. Q.append(j)
149. shortestpath\_p\_list = [0]
150. **for** t **in** node[1:]:
151. shortestpath\_p\_list.append(t.p)
152. **return** shortestpath\_p\_list


156. **def** SPP\_LS(o\_id,d\_id,node):
157. #   #initialize
158. node[o\_id].set\_SPP\_u(0)
159. **for** t **in** node[1:]:
160. t.set\_SPP\_p(-1)
161. **if** t.node\_id != o\_id:
162. t.set\_SPP\_u(float('inf'))
163. #mainloop
164. Q = [node[o\_id]]
165. **while** len(Q) != 0:
166. Q.sort(key = **lambda** qq:qq.u)
167. i = Q[0]
168. **if** i.node\_id == d\_id:
169. **break**
170. **del** Q[0]
171. **for** ij **in** i.l\_out:
172. j = node[ij.head\_node]
173. **if** j.u > (i.u+ij.length):
174. j.u = (i.u+ij.length)
175. j.p = i#id还是对象
176. **if** j **not** **in** Q:
177. Q.append(j)
178. shortestpath\_p\_list = [0]
179. **for** t **in** node[1:]:
180. shortestpath\_p\_list.append(t.p)
181. spath = []
182. snode = []
183. head\_n = node[d\_id]
184. snode.append(d\_id)
185. tail\_n = shortestpath\_p\_list[d\_id]
186. **while** tail\_n != -1:
187. **for** l **in** head\_n.l\_in:
188. **if** l.tail\_node == tail\_n.node\_id:
189. spath.insert(0,l)
190. head\_n = tail\_n
191. snode.append(tail\_n.node\_id)
192. tail\_n = shortestpath\_p\_list[head\_n.node\_id]
193. snode.reverse()
194. **return** (spath,snode)
195. #find the path and node list
197. **def** get\_length(Astarsp):
198. sum\_length = 0
199. **for** i **in** Astarsp:
200. sum\_length += i.length
201. **print**('length = ',sum\_length)

204. #test the lable setting algorithm
205. **def** Test\_SPP\_LS(o\_id, d\_id):
206. spath, snode = SPP\_LS(o\_id,d\_id,NODE)
207. **return**(spath, snode)
209. #test the continuous approximation algorithm
210. **def** Test\_SPP\_CA(o\_id,d\_id):
211. Lc\_node = []
212. shortestpath\_p\_list = SPP\_CA(o\_id,NODE)
213. **if** shortestpath\_p\_list[o\_id] == -1:
214. **pass**
215. **else**:
216. **print**('shortestpath\_p\_list is wrong!')
217. shortestpath\_link=[]
218. head\_n = NODE[d\_id]
219. Lc\_node.append(d\_id)
220. tail\_n = shortestpath\_p\_list[d\_id]
221. **while** tail\_n != -1:
222. **for** l **in** head\_n.l\_in:
223. **if** l.tail\_node == tail\_n.node\_id:
224. shortestpath\_link.insert(0,l)
225. head\_n = tail\_n
226. Lc\_node.append(tail\_n.node\_id)
227. tail\_n = shortestpath\_p\_list[head\_n.node\_id]
228. Lc\_node.reverse()
229. **return** (shortestpath\_link,Lc\_node)
231. #test the lable correcting algorithm
232. **def** Test\_SPP\_LC(o\_id,d\_id):
233. Lc\_node = []
234. shortestpath\_p\_list = SPP\_LC(o\_id,NODE)
235. **if** shortestpath\_p\_list[o\_id] == -1:
236. **pass**
237. **else**:
238. **print**('shortestpath\_p\_list is wrong!')
239. shortestpath\_link=[]
240. head\_n = NODE[d\_id]
241. Lc\_node.append(d\_id)
242. tail\_n = shortestpath\_p\_list[d\_id]
243. **while** tail\_n != -1:
244. **for** l **in** head\_n.l\_in:
245. **if** l.tail\_node == tail\_n.node\_id:
246. shortestpath\_link.insert(0,l)
247. head\_n = tail\_n
248. Lc\_node.append(tail\_n.node\_id)
249. tail\_n = shortestpath\_p\_list[head\_n.node\_id]
250. Lc\_node.reverse()
251. **return** (shortestpath\_link,Lc\_node)

254. ##################################
255. # 创建结果的shp文件
256. ##################################
257. **def** writeShp(splink, spnode, o, d):
258. # 补充完整代码
259. o = str(o)
260. d = str(d)
261. layerFields = QgsFields()
262. layerFields.append(QgsField('id',QVariant.Int))
263. layerFields.append(QgsField('tail', QVariant.Int))
264. layerFields.append(QgsField('head', QVariant.Int))
266. addr = sys.path[0]
267. writer = QgsVectorFileWriter('C://Users//Mr.Liu//Desktop//交通信息系统//Gis第四次作业//QGIS\_python\_shortest Qgis version2作业版本//output//shortestpath'+o+'\_'+d+'.shp','UTF-8',layerFields,\
268. QgsWkbTypes.LineString, QgsCoordinateReferenceSystem('EPSG:4326'),'ESRI Shapefile')
270. count = 0
271. **for** link **in** splink:
272. count = count+1
273. feat = QgsFeature()
274. tailnode = QgsPoint(NODE[link.tail\_node].X, NODE[link.tail\_node].Y)
275. headnode = QgsPoint(NODE[link.head\_node].X, NODE[link.head\_node].Y)
276. feat.setGeometry(QgsGeometry.fromPolyline([tailnode, headnode]))
277. feat.setAttributes([count,link.tail\_node-1,link.head\_node-1])
279. writer.addFeature(feat)
280. **del**(writer)
282. **return** "Success for creating shpfile!"
284. #find the path and node list
285. #设置文件名称
286. ##################################
287. Link\_layer = "C://Users//Mr.Liu//Desktop//交通信息系统//Gis第四次作业//QGIS\_python\_shortest Qgis version2作业版本//ChicagoSketch-ex1\_lnk\_s.shp"
288. Node\_layer = "C://Users//Mr.Liu//Desktop//交通信息系统//Gis第四次作业//QGIS\_python\_shortest Qgis version2作业版本//ChicagoSketch-ex1\_nod\_s.shp"
289. Node\_filename = "ChicagoSketch-ex1\_nod\_s" #ChicagoSketch-ex1\_nod\_s , sf-ex1\_nod\_s
290. Link\_filename = "ChicagoSketch-ex1\_lnk\_s" #ChicagoSketch-ex1\_lnk\_s , sf-ex1\_lnk\_s
291. ##################################
292. #调用函数
293. ##################################
294. LINK, NODE, NODE\_COUNT, LINK\_COUNT = read\_shpFile(Link\_layer, Node\_layer, Node\_filename, Link\_filename)
295. ###
296. start = time.perf\_counter()
298. random.seed(4)
299. **for** i **in** range(0,5):
300. o = random.randint(0,933)
301. d = random.randint(0,933)
302. sp, spnode = Test\_SPP\_LS(o, d)
304. end = time.perf\_counter()
305. **print**('run time =', end - start)
307. **print** (spnode)
308. get\_length(sp)
310. **print**(writeShp(sp, spnode, o, d))
311. **print**( )