**求平面上n个顶点的最近点对问题**

SA20225085 朱志儒

## 实验内容

编程实现求平面上n个顶点的最近点对问题。

对于平面上的四个顶点，A(0,0)、B(1,3)、C(1,0)、D(2,2)，距离最近的两个顶点为A和C，距离为1。

对文件data.txt的所有点，求距离最近的一对及其距离。

程序输入：

字符串文件data.txt。

程序输出：

输出距离最近的一对顶点编号，及其距离值。

## 实验目的

熟练掌握分治算法思想

## 算法设计思路

第一步：数据预处理

因为这些点的位置是随机产生并保存在二维数组中，所以我们得先将这些点，按照x坐标从小到大排序，调整它们在二维数组中的次序。

第二步：划分中轴线

把这些点在平面上分成左右两边。选择最中间的两个元素，求出它俩x坐标的平均值，设置为中轴线的坐标。

第三步：求半边最小距离

左半边和右半边的求最小距离的方法是一样的。假如我们现在求的是左半边，那就把左半边也看成一个整体，我们再把它分成左右两半，依次往下递归，越分越小，当平面只剩下3个点时就不再切分。

第四步：求中间的最小距离

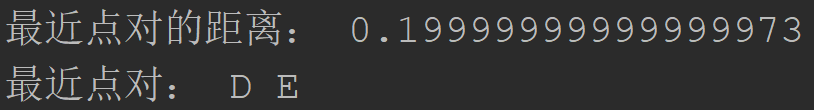
我们只需要考查中轴线左右两边距离小于d的点。理由：距离中轴线大于d的那些点，它们和另一个半边的点的距离，肯定大于d，考查他们就没有意义了。

## 源码+注释

1. **def** readFile():
2. file = open('data.txt', 'r')
3. pair = []
4. **for** line **in** file.readlines():
5. temp = line.strip().split(',')
6. temp[0] = float(temp[0][2:])
7. temp[1] = float(temp[1][1:-1])
8. temp.append(line[0])
9. pair.append(temp)
10. **return** pair
12. **def** cmpX(a, b):
13. **if** b[0] < a[0]:
14. **return** -1
15. **if** a[0] < b[0]:
16. **return** 1
17. **return** 0
19. **def** cmpY(a, b):
20. **if** b[1] < a[1]:
21. **return** -1
22. **if** a[1] < b[1]:
23. **return** 1
24. **return** 0
26. **def** computeDistance(x, y):
27. **return** math.sqrt((x[0] - y[0]) \* (x[0] - y[0]) + (x[1] - y[1]) \* (x[1] - y[1]))
29. **def** search(xpair):
30. minLength = 9999999
31. pair = None
32. **for** i **in** range(len(xpair) - 1):
33. **for** j **in** range(i + 1, len(xpair)):
34. length = computeDistance(xpair[i], xpair[j])
35. **if** length < minLength:
36. minLength = length
37. pair = [xpair[i][2], xpair[j][2]]
38. **return** minLength, pair
40. **def** divide(pairs, pivot):
41. pl = []
42. pr = []
43. **for** point **in** pairs:
44. **if** point[0] < pivot:
45. pl.append(point)
46. **elif** point[0] > pivot:
47. pr.append(point)
48. **else**:
49. pl.append(point)
50. pr.append(point)
51. **return** pl, pr
53. **def** solution(xpair, ypair):
54. num = len(xpair)
55. **if** num <= 3:
56. **return** search(xpair)
57. **if** num % 2 == 1:
58. xmidline = xpair[int(num / 2)][0]
59. **else**:
60. xmidline = (xpair[int(num / 2)][0] + xpair[int(num / 2) - 1][0]) / 2
61. xl, xr = divide(xpair, xmidline)
62. yl, yr = divide(ypair, xmidline)
63. llen, lpair = solution(xl, yl)
64. rlen, rpair = solution(xr, yr)
65. **if** llen < rlen:
66. minlen = llen
67. minpair = lpair
68. **else**:
69. minlen = rlen
70. minpair = rpair
71. y = []
72. **for** point **in** ypair:
73. **if** abs(point[0] - xmidline) <= minlen:
74. y.append(point)
75. i = 0
76. **while** (i == 0 **or** i < len(y) - 7):
77. **for** j **in** range(1, 8):
78. **if** i + j >= len(y):
79. **break**
80. length = computeDistance(y[i], y[i + j])
81. **if** length < minlen:
82. minlen = length
83. minpair = [y[i][2], y[i + j][2]]
84. i += 1
85. **return** minlen, minpair
87. **if** \_\_name\_\_ == '\_\_main\_\_':
88. pair = readFile()
89. xpair = copy.deepcopy(sorted(pair, key=functools.cmp\_to\_key(cmpX), reverse=True))
90. ypair = copy.deepcopy(sorted(pair, key=functools.cmp\_to\_key(cmpY), reverse=True))
91. minlen, minpair = solution(xpair, ypair)
92. **print**("最近点对的距离：", minlen)
93. **print**("最近点对：", minpair[0], minpair[1])

## 算法测试结果

读取字符串文件data.txt，程序输入如下：

****

## 实验过程中遇到的困难及收获

本次实验算法比较复杂，实现过程用时较长。