K-D树

k-d树(k维树的简称)是一种空间划分的数据结构，用于组织k维空间中的点。请基于kd tree.py或KDTree.java完成代码。在这个项目中，我们只考虑二维的点。注意，您不能使用任何第三方库。

1.解释现有代码(5分)。

2.实现并解释insert()和range()(英文写作风格，22分)。

3.范围查询时间复杂度分析(5分)。

4.可视化k-d树方法和naive方法之间的时间性能(5分)。

5. (奖励)实现最近邻查询(5分)

还要注意的是，如果你不能在Q2中解释你的代码，那么你将有可能在Q2中失去所有的分数。

**def** range(self, rectangle: Rectangle) -> List[Point]:  
 *"""range query"""* result = []  
  
 **def** \_dfs\_range(node:Node,depth:int=0):  
 **if** node **is None**:  
 **return  
 if** rectangle.is\_contains(node.location):  
 result.append(node.location)  
 axis = depth % 2  
 **if** node.location[axis] > rectangle.lower[axis]:  
 \_dfs\_range(node.left, depth + 1)  
 **if** node.location[axis] <= rectangle.upper[axis]:  
 \_dfs\_range(node.right, depth + 1)  
 *# if depth%2==0:  
 # if node.location[0]>rectangle.lower[0]:  
 # \_dfs\_range(node.left,depth+1)  
 # if node.location[0]<=rectangle.upper[0]:  
 # \_dfs\_range(node.right,depth+1)  
 # else:  
 # if node.location[0]>rectangle.lower[1]:  
 # \_dfs\_range(node.left,depth+1)  
 # if node.location[0]<=rectangle.upper[1]:  
 # \_dfs\_range(node.right,depth+1)* \_dfs\_range(self.\_root)  
 **return** result

*# 中位数index=list长度除2、向下取整*

优先队列

我们已经学习了如何用数组表示堆。在这个项目中，我们将用一个单链表来实现它，其中每个节点定义为：

class Node:

def \_\_init\_\_(self, key=None, next=None):

self.key = key

self.next = next

**1.**基于单链表设计一个完整的二叉树ADT，将第一个元素的索引赋值为o。给定一个索引i，请执行三个操作分别得到它的父、左子和右子(8分)。

**2.**基于上面的完整二叉树，设计一个最小优先级队列。具体来说，你应该实现并解释insert()和delMin()(英文写作风格，14分)。

**3.**在最小优先级队列中分析上述方法的时间复杂度(10分)。

**4.**为基于链表的堆执行一个简单的性能基准测试，并将结果可视化(5分)。

**5.** (奖励)基于你的链表堆，用graphviz绘制相应的树结构(5分)。

请注意，如果你不能在第二节解释你的代码，那么你将有可能在第二节失去所有的分数。