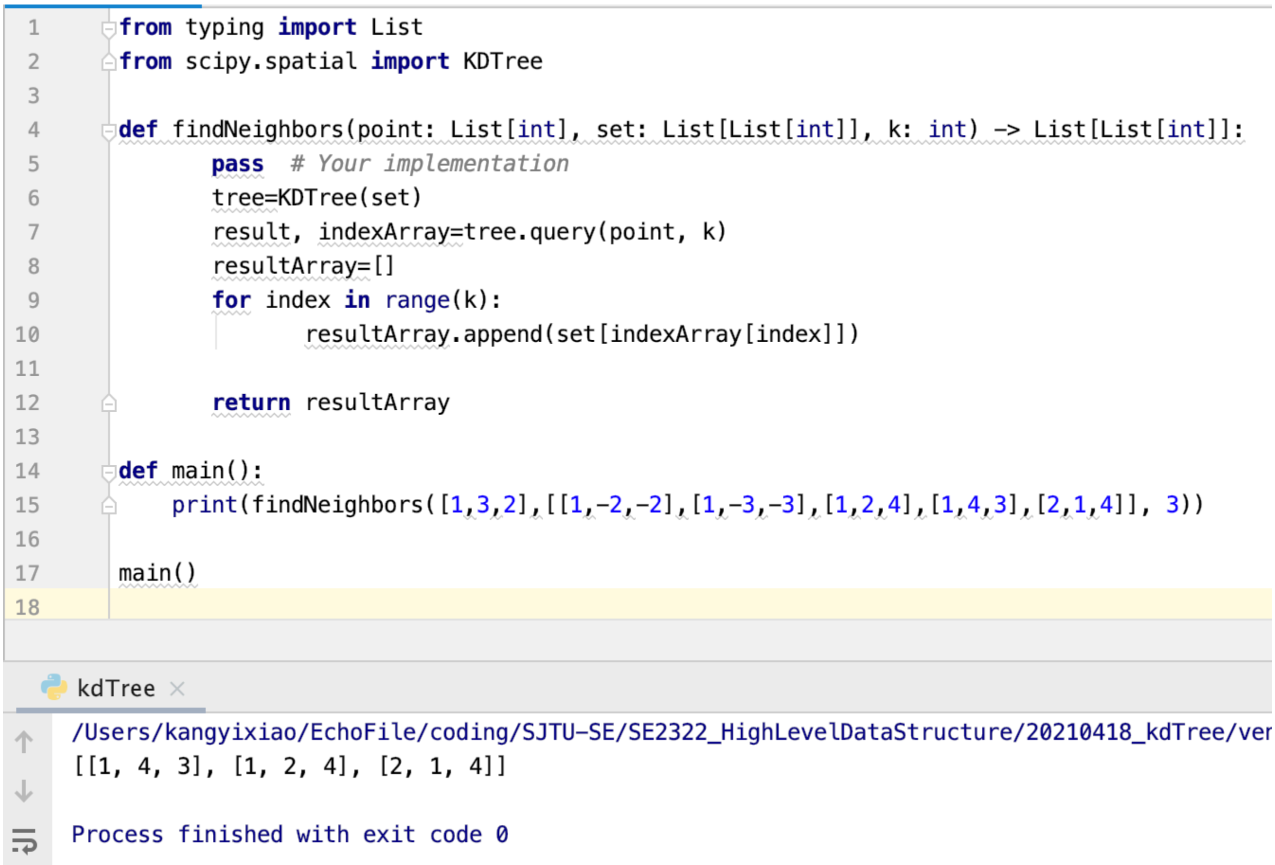
kdTree

康艺潇 518431910002

```
from typing import List
from scipy.spatial import KDTree

def findNeighbors(point: List[int], set: List[List[int]], k: int) ->
List[List[int]]:
        pass  # Your implementation
        tree=KDTree(set)
        result, indexArray=tree.query(point, k)
        resultArray=[]
        for index in range(k):
                resultArray.append(set[indexArray[index]])

        return resultArray

def main():
    print(findNeighbors([1,3,2],[[1,-2,-2],[1,-3,-3],[1,2,4],[1,4,3],[2,1,4]],
3))

main()
```

结果



```python
from typing import List
from scipy.spatial import KDTree

def findNeighbors(point: List[int], set: List[List[int]], k: int) -> List[List[int]]:
        pass  # Your implementation
        tree=KDTree(set)
        result, indexArray=tree.query(point, k)
        resultArray=[]
        for index in range(k):
                resultArray.append(set[indexArray[index]])

        return resultArray

def main():
    print(findNeighbors([1,3,2],[[1,-2,-2],[1,-3,-3],[1,2,4],[1,4,3],[2,1,4]], 3))

main()
```

```
/Users/kangyixiao/EchoFile/coding/SJTU-SE/SE2322_HighLevelDataStructure/20210418_kdTree/ver
[[1, 4, 3], [1, 2, 4], [2, 1, 4]]

Process finished with exit code 0
```

时间复杂度分析

构造kdTree **时间复杂度**：`O(d * n * log(n))`

得到最近点的集合时间复杂度: `O(k * log(n))`

构造一棵d维的kdTree 的时间复杂度为O(d*n*log(n)), kdTree支持查找最近的点，每次查找后把这个点从集合中删去，每次查找的时间复杂度为：O(log(n)),所以总时间复杂度为：O(k*log(n))

因此总的时间复杂度为`O(d * n * log(n))`+`O(k * log(n))` = `O(n*log(n))`