深蓝学院
shenlanxueyuan.com

第二章作业讲评

主讲人 郝爽

作业1

深蓝学院
shenlanxueyuan.com

```
# --- get the split position ---
point_indices_sorted, _ = sort_key_by_vale(point_indices, db[point_indices, axis])

# 作业1
# 屏蔽开始
# 取排序后中间点
middle_left_idx = math.ceil(point_indices_sorted.shape[0] / 2) - 1
# 取排序后中间点的原始索引
middle_left_point_idx = point_indices_sorted[middle_left_idx]
# 取排序后中间点的值
middle_left_point_value = db[middle_left_point_idx, axis]

#取右边一个点的上述信息
middle_right_idx = middle_left_idx + 1
middle_right_point_idx = point_indices_sorted[middle_right_idx]
middle_right_point_value = db[middle_right_point_idx, axis]
```

作业1

深蓝学院
shenlanxueyuan.com

```
#根节点的值赋值为上述两个点的平均值
root.value = (middle_left_point_value + middle_right_point_value) * 0.5
# === get the split position ===
# 二分，构建左子树
root.left = kdtree_recursive_build(root.left,
                                   db,
                                   point_indices_sorted[0:middle_right_idx],
                                   axis_round_robin(axis, dim=db.shape[1]),
                                   leaf_size)
# 二分，构建右子树
root.right = kdtree_recursive_build(root.right,
                                    db,
                                    point_indices_sorted[middle_right_idx:],
                                    axis_round_robin(axis, dim=db.shape[1]),
                                    leaf_size)
```

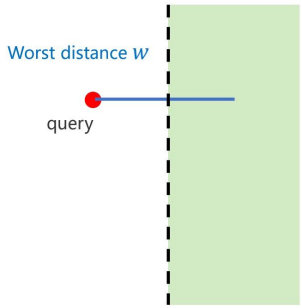
作业2

深蓝学院 shenlanxueyuan.com

```python
if root is None:
    return False

if root.is_leaf():
    # compare the contents of a leaf
    leaf_points = db[root.point_indices, :]
    diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
    for i in range(diff.shape[0]):
        result_set.add_point(diff[i], root.point_indices[i])
    return False
```

```python
# 作业2
# 提示：仍通过递归的方式实现搜索
# 屏蔽开始
# 如果当前axis的值小于根节点的值，则搜索左子树
if query[root.axis] <= root.value:
    kdtree_knn_search(root.left, db, result_set, query)
    if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
        kdtree_knn_search(root.right, db, result_set, query)
else:
    # 如果当前axis的值大于根节点的值，则搜索右子树
    kdtree_knn_search(root.right, db, result_set, query)
    if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
        kdtree_knn_search(root.left, db, result_set, query)
# 屏蔽结束
```



递归查找，如果query数组在axis轴上的值到当前分割面的距离小于当前worstDist（如上图），说明分割面的另一侧，可能有距离更近的点，所以需要递归查找另一侧

作业3

深蓝学院
shenlanxueyuan.com

```
# 作业3
# 提示：通过递归的方式实现搜索
# 屏蔽开始
# 同作业2
if query[root.axis] <= root.value:
    kdtree_radius_search(root.left, db, result_set, query)
    if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
        kdtree_radius_search(root.right, db, result_set, query)
else:
    kdtree_radius_search(root.right, db, result_set, query)
    if math.fabs(query[root.axis] - root.value) < result_set.worstDist():
        kdtree_radius_search(root.left, db, result_set, query)
# 屏蔽结束
```

原理完全与作业2相同，不同点在于，作业2是做k个最近的邻居的查找，而作业3是根据radius去找所有邻居

作业4

深蓝学院
shenlanxueyuan.com

```
# 作业4
# 屏蔽开始

#root有子节点，is_leaf置为False
root.is_leaf = False
#创建8个子节点
children_point_indices = [[] for i in range(8)]
#遍历每一个点
for point_idx in point_indices:
    point_db = db[point_idx]
    #计算当前点该放置到哪个子节点
    morton_code = 0
    #判断该放到x轴的哪一侧
    if point_db[0] > center[0]:
        morton_code = morton_code | 1
    #判断该放到y轴的哪一侧
    if point_db[1] > center[1]:
        morton_code = morton_code | 2
    #判断该放到z轴的哪一侧
    if point_db[2] > center[2]:
        morton_code = morton_code | 4
    #子节点存储点的索引
    children_point_indices[morton_code].append(point_idx)
```

判断当前节点需要放到当前子树的哪一个象限

作业4

深蓝学院
shenlanxueyuan.com

```
# create children
factor = [-0.5, 0.5]
for i in range(8):
    #计算每一个子节点的center坐标
    child_center_x = center[0] + factor[(i & 1) > 0] * extent
    child_center_y = center[1] + factor[(i & 2) > 0] * extent
    child_center_z = center[2] + factor[(i & 4) > 0] * extent
    #子节点的extent
    child_extent = 0.5 * extent
    child_center = np.asarray([child_center_x, child_center_y, child_center_z])

    #递归创建子节点的八叉树
    root.children[i] = octree_recursive_build(root.children[i],
                                              db,
                                              child_center,
                                              child_extent,
                                              children_point_indices[i],
                                              leaf_size,
                                              min_extent)
# 屏蔽结束
```

如果满足非叶子节点的条件，首先根据center来判断每一个点属于8个子树里面的哪一个，然后分别遍历每一个子树，构造子树的center和extent，根据children_point_indices去递归建子树

作业5

深蓝学院
shenlanxueyuan.com

```
# 作业5
# 提示：尽量利用上面的inside、overlaps、contains等函数
# 屏蔽开始
if contains(query, result_set.worstDist(), root):
    # compare the contents of the octant
    leaf_points = db[root.point_indices, :]
    diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
    for i in range(diff.shape[0]):
        result_set.add_point(diff[i], root.point_indices[i])
    # don't need to check any child
    return False

if root.is_leaf and len(root.point_indices) > 0:
    # compare the contents of a leaf
    leaf_points = db[root.point_indices, :]
    diff = np.linalg.norm(np.expand_dims(query, 0) - leaf_points, axis=1)
    for i in range(diff.shape[0]):
        result_set.add_point(diff[i], root.point_indices[i])
    # check whether we can stop search now
    return inside(query, result_set.worstDist(), root)

# no need to go to most relevant child first, because anyway we will go through all children
for c, child in enumerate(root.children):
    if child is None:
        continue
    if False == overlaps(query, result_set.worstDist(), child):
        continue
    if octree_radius_search_fast(child, db, result_set, query):
        return True

# 屏蔽结束
```

通过判断query点到当前子树的最远角点的距离a，与radius搜索半径比较，如果a小于r的话，说明当前子树里面的点符合搜索条件，将当前子树的所有点进行更新

作业6

深蓝学院
shenlanxueyuan.com

```
# 作业6
# 屏蔽开始
# go to the relevant child first
morton_code = 0
if query[0] > root.center[0]:
    morton_code = morton_code | 1
if query[1] > root.center[1]:
    morton_code = morton_code | 2
if query[2] > root.center[2]:
    morton_code = morton_code | 4

if octree_radius_search(root.children[morton_code], db, result_set, query):
    return True

# check other children
for c, child in enumerate(root.children):
    if c == morton_code or child is None:
        continue
    if False == overlaps(query, result_set.worstDist(), child):
        continue
    if octree_radius_search(child, db, result_set, query):
        return True

# 屏蔽结束
```

按照八叉树依次遍历+递归去搜索，理论上作业5速度应该更快

作业7

深蓝学院
shenlanxueyuan.com

```
# 作业7
# 屏蔽开始
# go to the relevant child first
# 根据query找到当前所属象限
morton_code = 0
if query[0] > root.center[0]:
    morton_code = morton_code | 1
if query[1] > root.center[1]:
    morton_code = morton_code | 2
if query[2] > root.center[2]:
    morton_code = morton_code | 4

#去八叉树的相应象限中递归查找
if octree_knn_search(root.children[morton_code], db, result_set, query):
    return True

# check other children
# 上面的octree_knn_search返回False，代表query_offset + radius < extent
for c, child in enumerate(root.children):
    # 如果前面已经搜索过某一个子节点，或者子节点为空
    if c == morton_code or child is None:
        continue
    # 如果搜索半径与子节点没有交集
    if False == overlaps(query, result_set.worstDist(), child):
        continue
    if octree_knn_search(child, db, result_set, query):
        return True

# 屏蔽结束
```

作业7的功能是在八叉树中进行knn的查找，注意与作业6的区别在于一个是knn，一个是radiusNN

benchmark

深蓝学院
shenlanxueyuan.com

```python
#scipy knn查找
def scipy_kdtree_search(tree:KDTree,result_set:KNNResultSet,point: np.ndarray):
    scipy_dist,scipy_index=tree.query(point,result_set.capacity)
    for index, dist_index in enumerate(result_set.dist_index_list):
        dist_index.distance = scipy_dist[index]
        dist_index.index = scipy_index[index]
    return result_set

# brute force 查找
def brute_search(db: np.ndarray,result_set:KNNResultSet, query: np.ndarray):
    diff = np.linalg.norm(np.expand_dims(query, 0) - db, axis=1)
    nn_index = np.argsort(diff)
    nn_dist = diff[nn_index]
    for index, dist_index in enumerate(result_set.dist_index_list):
        dist_index.distance = nn_dist[index]
        dist_index.index = nn_index[index]
    return result_set
```

scipy的knn search 和 暴力搜索的实现

benchmark

深蓝学院
shenlanxueyuan.com

从总共12万多个点里面，采样出3万个点，做了以下时间对比

建树方式	建树时间(ms)	knn查找时间(ms)	radius查找时间(ms)	scipy查找时间(ms)	暴力查找时间(ms)
八叉树	1251.035	40510.457	107920.170	26197.830	105806.846
kdtree	58.997	57284.087	82518.147	27213.291	107428.113

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening