# 2023秋“大数据软件设计与实践”课程报告

|  |  |
| --- | --- |
| 姓名： | 刘俊杉 |
| 学号： | 2021112078 |

## 1. 软件功能

实现B+树的基本操作，包括点查找索引项、区间查找、插⼊索引项、删除索引项。实现B+树的可视化展示。

## 2. 设计方案

1. 定义B+树的数据结构。

2. 实现B+树的基本操作，包括点查找索引项、区间查找、插⼊索引项、删除索引项。

3. 实现B+树的可视化展示。

## 3. 软件实现

插入节点

1. **def** search(self, element: int):
2. is\_leaf = len(self.children) == 0
3. idx = 0
5. **for** key **in** self.keys:
6. **if** element == key **and** is\_leaf:
7. **return** key
8. **elif** element < key:
9. **break**
10. **elif** element == key:
11. idx += 1
12. **break**
13. idx += 1
15. **if** self.children:
16. **return** self.children[idx].search(element)
17. **else**:
18. **return** -1

插入节点

1. **def** insert(self, element: int):
2. insert\_idx = 0
4. **for** key **in** self.keys:
5. **if** element < key:
6. **break**
7. insert\_idx += 1
9. **if** self.children:
10. self.children[insert\_idx].insert(element)
11. **else**:
12. self.keys = self.keys[:insert\_idx] + [element] + self.keys[insert\_idx:]
14. **if** len(self.keys) == self.order:
15. self.promote()

融合节点

1. **def** promote(self):
2. is\_leaf = len(self.children) == 0
4. middle\_idx = math.floor((len(self.keys) - 1) / 2)
5. middle\_key = self.keys[middle\_idx]
6. copy\_idx = middle\_idx **if** is\_leaf **else** middle\_idx + 1
8. right\_tree = BPlusTreeNode(self.order)
9. right\_tree.keys = self.keys[copy\_idx:]
10. right\_tree.children = self.children[copy\_idx:]
11. # Update parents on the right tree
12. **for** c **in** right\_tree.children:
13. c.parent = right\_tree
15. self.keys = self.keys[:middle\_idx]
16. self.children = self.children[: copy\_idx]
18. **if** is\_leaf:
19. right\_tree.next\_node = self.next\_node
20. self.next\_node = right\_tree
21. **else**:
22. self.next\_node = None
24. **if** **not** self.parent:
25. self.parent = BPlusTreeNode(self.order)
26. self.parent.children = [self]
28. right\_tree.parent = self.parent
30. insert\_idx = self.parent.children.index(self)
31. self.parent.keys = (
32. self.parent.keys[:insert\_idx] + [middle\_key] + self.parent.keys[insert\_idx:]
33. )
34. self.parent.children = (
35. self.parent.children[: insert\_idx + 1]
36. + [right\_tree]
37. + self.parent.children[insert\_idx + 1:]
38. )
40. # Parentes são promovidos de forma recursiva
41. **if** len(self.parent.keys) == self.order:
42. self.parent.promote()

## 4. 软件界面

