# 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

实现线性哈希表的基本操作，包括点查找索引项、插⼊索引项、删除索引项。实现线性哈希表的可视化展示。

## 2. 设计方案

1. 定义线性哈希表的数据结构。
2. **def** \_\_init\_\_(self) -> None:
3. self.bucket\_capacity = 4 #桶的容量
4. self.overpoint = 0 #分裂点
5. self.init\_size = 2 #哈希表初始大小
6. self.size = 2 #哈希表大小
7. self.level = 1 #分裂轮数
8. self.buckets = [{ } **for** \_ **in** range(self.init\_size)]  # 桶数组
9. self.id=str(uuid.uuid4())
10. 实现线性哈希表的基本操作，包括点查找索引项、插⼊索引项、删除索引项。
11. **def** Search(self, key):
12. index = self.hash\_fun(key, self.level)
13. idx=0
14. **if** index < self.overpoint:
15. index = self.hash\_fun(key, self.level + 1)
16. **if** self.buckets[index].get(key) == None:
17. **print**(str(key)+"键值不存在")
18. **else**:
19. **print**(str(key)+"键值在桶"+str(index))
20. **return** self.buckets[index].get(key)
21. **def** Delete(self, key):
22. **def** MergeHash():
23. **if** self.overpoint == 0 :
24. self.level -= 1
25. self.overpoint = self.init\_size / 2
26. self.init\_size /= 2
28. self.overpoint -= 1
29. self.size -= 1
30. old\_bucket = self.buckets[self.size]
31. self.buckets = self.buckets[:-1]
32. **for** key **in** list(old\_bucket.keys()):
33. index = self.hash\_fun(key, self.level)
34. value = old\_bucket[key]
35. self.buckets[index][key] = value
37. **if** self.Search(key) != None:
38. index = self.hash\_fun(key, self.level)
39. **if** index < self.overpoint:
40. index = self.hash\_fun(key, self.level + 1)
41. bucket = self.buckets[index]
42. bucket.pop(key)
43. **print**("从桶" + str(index) + "中删除键值" + str(key))
44. **if** len(bucket) == 0 **and** self.size >4:
45. MergeHash()

3. 实现线性哈希表的可视化展示。

1. def visualize(self):
2. dot = Digraph(comment='Linear Hash',node\_attr={'shape': 'record', 'height': '.1'})
3. dot.attr('node', shape='box')
4. #dot.node('bucket', style='filled', fillcolor='#40e0d0')
5. color\_1 = list(map(lambda x: color(tuple(x)), ncolors(100)))
6. for i in range(self.size):
7. dot.node("node\_"+str(i),str(i))
8. dot.node('bucket'+str(i), style='filled', fillcolor=color\_1[i+random.randint(0,90)])
10. for i in range(self.size):
11. print(self.buckets[i])
12. for key, value in self.buckets[i].items():
13. dot.edge("node\_"+str(i), 'bucket'+str(i), label=str(key) + " -> " + str(value))
14. dot.view()

## 3. 软件实现

利用graphviz 定义节点和并且连接节点

1. digraph {
2. node [height=.1 shape=record]
3. node [shape=box]
4. node\_0 [label=0]
5. bucket0 [fillcolor="#62F92D" style=filled]
6. node\_1 [label=1]
7. bucket1 [fillcolor="#2EF672" style=filled]
8. node\_2 [label=2]
9. bucket2 [fillcolor="#1AB7FA" style=filled]
10. node\_3 [label=3]
11. bucket3 [fillcolor="#B0F322" style=filled]
12. node\_4 [label=4]
13. bucket4 [fillcolor="#A2F538" style=filled]
14. node\_5 [label=5]
15. bucket5 [fillcolor="#0DF2A0" style=filled]
16. node\_6 [label=6]
17. bucket6 [fillcolor="#32FBCA" style=filled]
18. node\_7 [label=7]
19. bucket7 [fillcolor="#62F92D" style=filled]
20. node\_0 -> bucket0 [label="8 -> 64"]
21. node\_0 -> bucket0 [label="16 -> 256"]
22. node\_0 -> bucket0 [label="24 -> 576"]
23. node\_1 -> bucket1 [label="1 -> 1"]
24. node\_1 -> bucket1 [label="9 -> 81"]
25. node\_1 -> bucket1 [label="17 -> 289"]
26. node\_1 -> bucket1 [label="25 -> 625"]
27. node\_2 -> bucket2 [label="2 -> 4"]
28. node\_2 -> bucket2 [label="10 -> 100"]
29. node\_2 -> bucket2 [label="18 -> 324"]
30. node\_2 -> bucket2 [label="26 -> 676"]
31. node\_3 -> bucket3 [label="3 -> 9"]
32. node\_3 -> bucket3 [label="11 -> 121"]
33. node\_3 -> bucket3 [label="19 -> 361"]
34. node\_3 -> bucket3 [label="27 -> 729"]
35. node\_4 -> bucket4 [label="4 -> 16"]
36. node\_4 -> bucket4 [label="12 -> 144"]
37. node\_4 -> bucket4 [label="20 -> 400"]
38. node\_4 -> bucket4 [label="28 -> 784"]
39. node\_5 -> bucket5 [label="5 -> 25"]
40. node\_5 -> bucket5 [label="13 -> 169"]
41. node\_5 -> bucket5 [label="21 -> 441"]
42. node\_5 -> bucket5 [label="29 -> 841"]
43. node\_6 -> bucket6 [label="6 -> 36"]
44. node\_6 -> bucket6 [label="14 -> 196"]
45. node\_6 -> bucket6 [label="22 -> 484"]
46. node\_7 -> bucket7 [label="7 -> 49"]
47. node\_7 -> bucket7 [label="15 -> 225"]
48. node\_7 -> bucket7 [label="23 -> 529"]
49. }

## 4. 软件界面

1. **def** test():
2. L = Linear\_Hash()
4. **for** i **in** range(1,30,1):
5. k = randint(1, 100)
6. L.Insert(i,i\*\*2)
7. L.**print**()
8. L.visualize()

图示

中度可信度描述已自动生成

图示

描述已自动生成