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。

```
def __init__(self) -> None:
    self.bucket_capacity = 4 #桶的容量
    self.overpoint = 0 #分裂点
    self.init_size = 2 #哈希表初始大小
    self.size = 2 #哈希表大小
    self.level = 1 #分裂轮数
    self.buckets = [{ } for _ in range(self.init_size)]  # 桶数组
    self.id=str(uuid.uuid4())
```

2. 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。

```
def Search(self, key):
    index = self.hash_fun(key, self.level)
    idx=0
    if index < self.overpoint:
        index = self.hash_fun(key, self.level + 1)
    if self.buckets[index].get(key) == None:
        print(str(key)+"键值不存在")
    else:
        print(str(key)+"键值在桶"+str(index))
    return self.buckets[index].get(key)
def Delete(self, key):
    def MergeHash():
        if self.overpoint == 0 :
            self.level -= 1
            self.overpoint = self.init_size / 2
            self.init_size /= 2

        self.overpoint -= 1
        self.size -= 1
        old_bucket = self.buckets[self.size]
        self.buckets = self.buckets[:-1]
        for key in list(old_bucket.keys()):
            index = self.hash_fun(key, self.level)
            value = old_bucket[key]
            self.buckets[index][key] = value

    if self.Search(key) != None:
        index = self.hash_fun(key, self.level)
        if index < self.overpoint:
            index = self.hash_fun(key, self.level + 1)
        bucket = self.buckets[index]
        bucket.pop(key)
        print("从桶" + str(index) + "中删除键值" + str(key))
        if len(bucket) == 0 and self.size >4:
            MergeHash()
```

3. 实现线性哈希表的可视化展示。

```
def visualize(self):
    dot = Digraph(comment='Linear Hash',node_attr={'shape': 'record', 'height': '.1'})
    dot.attr('node', shape='box')
    #dot.node('bucket', style='filled', fillcolor='#40e0d0')
    color_1 = list(map(lambda x: color(tuple(x)), ncolors(100)))
    for i in range(self.size):
        dot.node("node_"+str(i),str(i))
        dot.node('bucket'+str(i), style='filled', fillcolor=color_1[i+random.randint(0,90)])

    for i in range(self.size):
        print(self.buckets[i])
        for key, value in self.buckets[i].items():
            dot.edge("node_"+str(i), 'bucket'+str(i), label=str(key) + " -> " + str(value))
    dot.view()
```

3. 软件实现

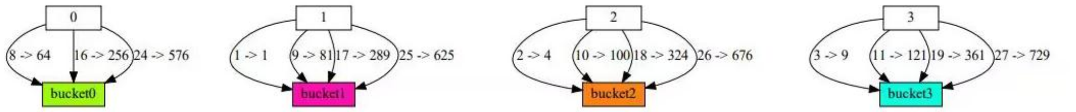
利用 graphviz 定义节点和并且连接节点

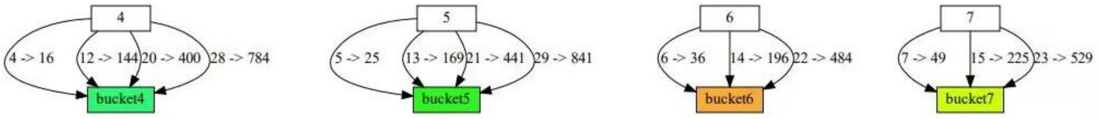
```
digraph {
    node [height=.1 shape=record]
    node [shape=box]
    node_0 [label=0]
    bucket0 [fillcolor="#62F92D" style=filled]
    node_1 [label=1]
    bucket1 [fillcolor="#2EF672" style=filled]
    node_2 [label=2]
    bucket2 [fillcolor="#1AB7FA" style=filled]
    node_3 [label=3]
    bucket3 [fillcolor="#B0F322" style=filled]
    node_4 [label=4]
    bucket4 [fillcolor="#A2F538" style=filled]
    node_5 [label=5]
    bucket5 [fillcolor="#0DF2A0" style=filled]
    node_6 [label=6]
    bucket6 [fillcolor="#32FBCA" style=filled]
    node_7 [label=7]
    bucket7 [fillcolor="#62F92D" style=filled]
    node_0 -> bucket0 [label="8 -> 64"]
    node_0 -> bucket0 [label="16 -> 256"]
    node_0 -> bucket0 [label="24 -> 576"]
    node_1 -> bucket1 [label="1 -> 1"]
    node_1 -> bucket1 [label="9 -> 81"]
    node_1 -> bucket1 [label="17 -> 289"]
    node_1 -> bucket1 [label="25 -> 625"]
    node_2 -> bucket2 [label="2 -> 4"]
    node_2 -> bucket2 [label="10 -> 100"]
    node_2 -> bucket2 [label="18 -> 324"]
    node_2 -> bucket2 [label="26 -> 676"]
    node_3 -> bucket3 [label="3 -> 9"]
    node_3 -> bucket3 [label="11 -> 121"]
    node_3 -> bucket3 [label="19 -> 361"]
    node_3 -> bucket3 [label="27 -> 729"]
    node_4 -> bucket4 [label="4 -> 16"]
    node_4 -> bucket4 [label="12 -> 144"]
    node_4 -> bucket4 [label="20 -> 400"]
    node_4 -> bucket4 [label="28 -> 784"]
    node_5 -> bucket5 [label="5 -> 25"]
    node_5 -> bucket5 [label="13 -> 169"]
    node_5 -> bucket5 [label="21 -> 441"]
    node_5 -> bucket5 [label="29 -> 841"]
    node_6 -> bucket6 [label="6 -> 36"]
    node_6 -> bucket6 [label="14 -> 196"]
    node_6 -> bucket6 [label="22 -> 484"]
    node_7 -> bucket7 [label="7 -> 49"]
    node_7 -> bucket7 [label="15 -> 225"]
    node_7 -> bucket7 [label="23 -> 529"]
}
```

4. 软件界面

```
def test():
    L = Linear_Hash()

    for i in range(1,30,1):
        k = randint(1, 100)
        L.Insert(i,i**2)
    L.print()
    L.visualize()
```


0
8 -> 64
16 -> 256
24 -> 576
bucket0
1
1 -> 1
9 -> 81
17 -> 289
25 -> 625
bucket1
2
2 -> 4
10 -> 100
18 -> 324
26 -> 676
bucket2
3
3 -> 9
11 -> 121
19 -> 361
27 -> 729
bucket3


4
4 -> 16
12 -> 144
20 -> 400
28 -> 784
bucket4
5
5 -> 25
13 -> 169
21 -> 441
29 -> 841
bucket5
6
6 -> 36
14 -> 196
22 -> 484
bucket6
7
7 -> 49
15 -> 225
23 -> 529
bucket7