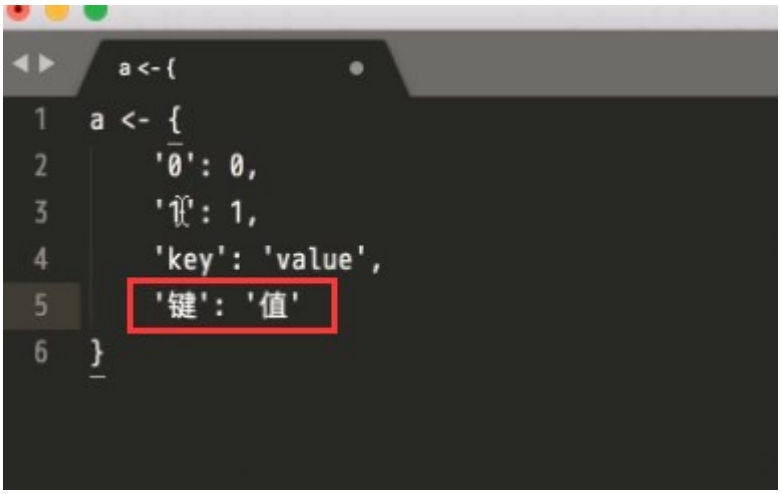
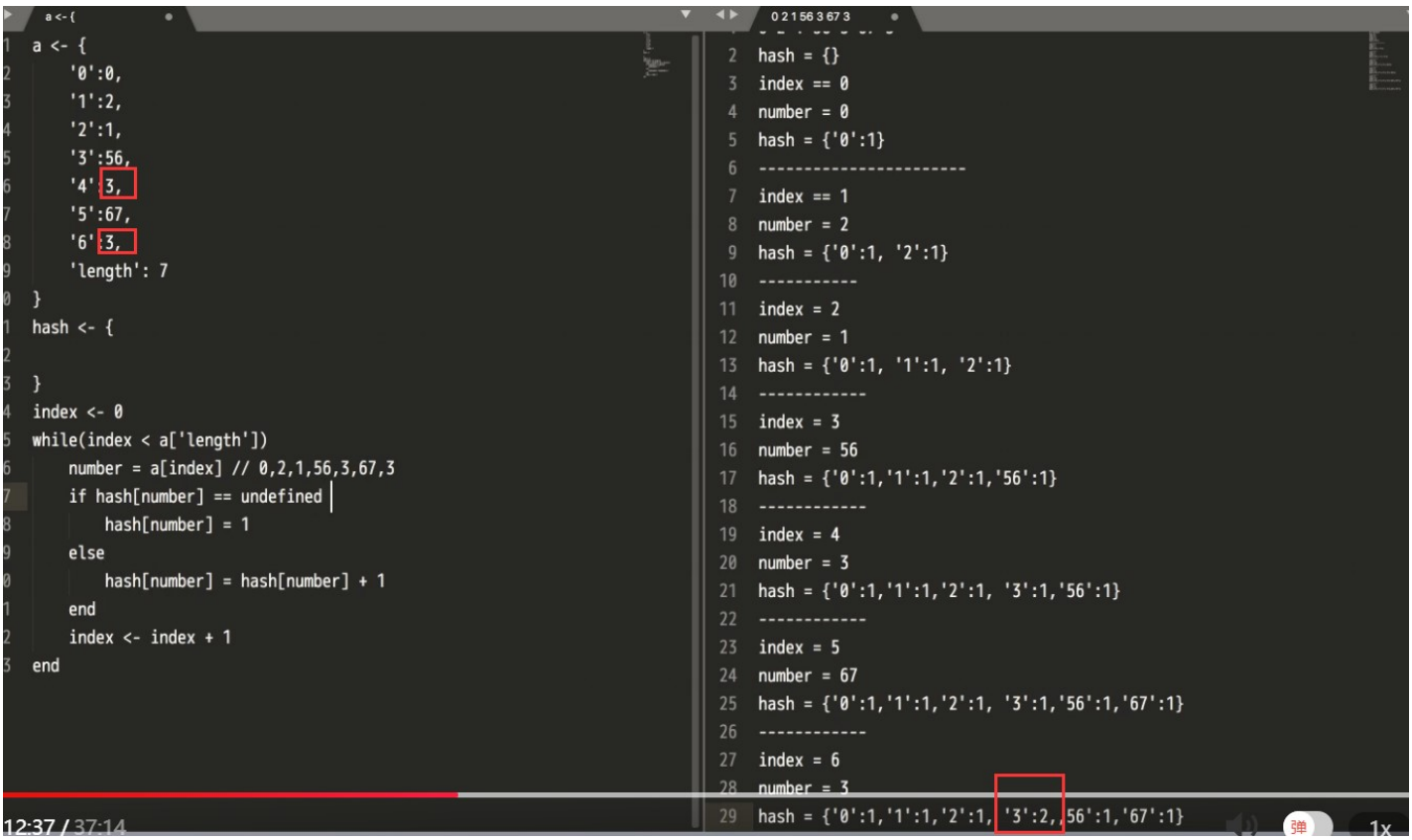
https://github.com/MisterBooo/LeetCodeAnimation
(学习数据结构的一个非常好的地方)

所有满足这种一个键一个值的都可以称为哈希;(数组也是哈希)



随机快排的复杂度:NlogN

但是计数快排可以更快;



数组的一些说明

```
a <- {
    '0':0,
    '1':2,
    '2':1,
    '3':56,
    '4':4
    '5':67,
    '6':3,
    '66' : 8,
    'length': 67
}
hash <- {

}
index <- 0
while(index < a['length'])
    number = a[index] // 0,2,1,56,3,67,3
    if hash[number] == undefined // hash[number] 不存在
        hash[number] = 1
    else // hash[number] 存在
        hash[number] = hash[number] + 1
    end
    index <- index + 1
end
index2 <- 0
max <- findMax(a)
while(index2 < max+1) // index2 == max(67) 是可以出现的, index2<68
```

这里的值其实就是下标的最大值+1，
而非这里数组的key的个数

这边是遍历

过程

```
index <- 0
while(index < a['length'])
    number = a[index] // 0,2,1,56,3,67,3
    if hash[number] == undefined // hash[number] 不存在
        hash[number] = 1
    else // hash[number] 存在
        hash[number] = hash[number] + 1
    end
    index <- index + 1
end
index2 <- 0
max <- findMax(a)
newArr <- {}
while(index2 < max+1)
    count = hash[index2]
    if count == 1
        newArr.push(index2)
    elseif count == 2
        newArr.push(index2)
        newArr.push(index2)
    else count == 3
        newArr.push(index2)
        newArr.push(index2)
        newArr.push(index2)
    end
    index2 <- index2 + 1
end
print newArr
```

```
    '3':2,
    '56':1,
    '67':1
}

入桶
--------
出桶
index2 = 0
newArr = [0]
index2 = 1
newArr = [0,1]
index =2
newArr = [0,1,2]
index2 =3
newArr = [0,1,2,3,3]
index2 = 4
什么也不做
index2 = 5
什么也不做
...
...
index2 = 56
newArr = [0,1,2,3,3,56]
...
index2 = 67
newArr = [0,1,2,3,3,56,67]
```

入桶与出桶

```
index = 6
number = 3
hash = {
    '0':1,
    '1':1,
    '2':1,
    '3':2,
    '56':1,
    '67':1
}

入桶
--------
出桶
index2 = 0
newArr = [0]
index2 = 1
newArr = [0,1]
index =2
newArr = [0,1,2]
index2 =3
newArr = [0,1,2,3,3]
index2 = 4
什么也不做
index2 = 5
什么也不做
...
...
index2 = 56
```

优化

```
hash <- {

}
index <- 0
while(index < a['length'])
    number = a[index] // 0,2,1,56,3,67,3
    if hash[number] == undefined // hash[number] 不存在
        hash[number] = 1
    else // hash[number] 存在
        hash[number] = hash[number] + 1
    end
    index <- index + 1
end
index2 <- 0
max <- findMax(a)
newArr <- {}
while(index2 < max+1)
    count = hash[index2]
    if count != undefined
        countIndex = 0
        while(countIndex < count)
            newArr.push(index2)
            countIndex <- countIndex + 1
        end
    end
    index2 <- index2 + 1
end
```

```
index2 = 56
newArr = [0,1,2,3,3,56]
...
index2 = 67
newArr = [0,1,2,3,3,56,67]

///


    if count == 1
        newArr.push(index2)
    elseif count == 2
        newArr.push(index2)
        newArr.push(index2)
    elseif count == 3
        newArr.push(index2)
        newArr.push(index2)
        newArr.push(index2)
    end
```

35:45 / 37:14

整个代码

```
a <- {'0':0, '1':2, '2':1, '3':56, '4':4 '5':67, '6':3, 'length':
67 }
hash <- {}
index <- 0
while(index < a['length'])
    number = a[index] // 0,2,1,56,3,67,3
    if hash[number] == undefined // hash[number] 不存在
        hash[number] = 1
    else // hash[number] 存在
        hash[number] = hash[number] + 1
    end
    index <- index + 1
end
index2 <- 0
max <- findMax(a)
newArr <- {}
while(index2 < max+1)
    count = hash[index2]
    if count != undefined
        countIndex = 0
        while(countIndex < count)
            newArr.push(index2)
            countIndex <- countIndex + 1
        end
    end
    index2 <- index2 + 1
end
print newArr
```