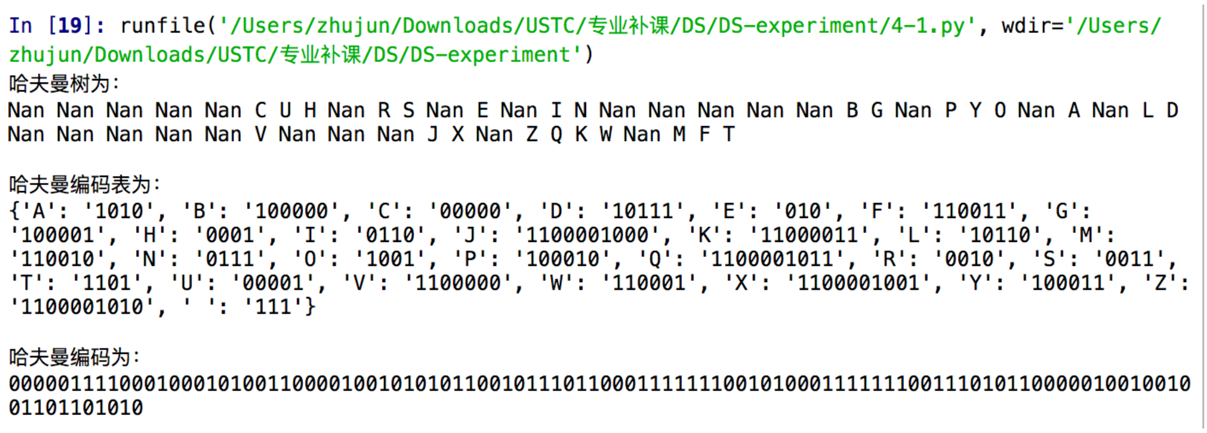
2018年8月5日 星期日

数据结构第四次实验

4-1.py

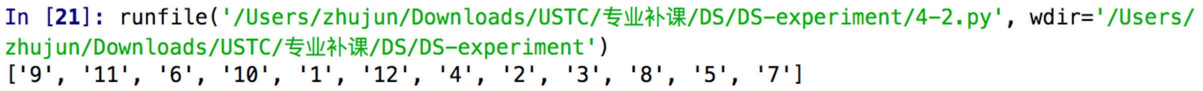
```
'''本题属于哈夫曼编码。首先先要建立一颗哈夫曼树，其所有叶子节点为字符。然后通过遍历哈夫曼树确定
  叶子结点以及对应的哈夫曼编码，最后通过查找编码表即可实现字符串的压缩。'''
# 本题使用的数据结构有：
# PriorityQueue: 优先级队列，通过该结构快速选择最小的两个节点组成一个新的节点并加入
#                队列，直到队列中仅剩下一个元素为止，该元素即为哈夫曼树的根节点
# Node: 哈夫曼树的节点， 注意要实现__it__函数以实现优先队列的比较

from queue import PriorityQueue
def getItems():
    # 返回最初的编码频数表
    dicts = {
            'A': 64, 'B': 13, 'C': 22, 'D': 32, 'E': 103, 'F': 21, 'G': 15,
            'H': 47, 'I': 57, 'J': 1,  'K': 5,  'L': 32,  'M': 20, 'N': 57,
            'O': 63, 'P': 15, 'Q': 1,  'R': 48, 'S': 51,  'T': 80,
            'U': 23, 'V': 8,  'W': 18, 'X': 1,  'Y': 16,  'Z': 1,
            ' ': 168
        }
    return dicts

class Node():
    '''哈夫曼树的节点'''
    def __init__(self, data, freq, left = None, right = None, father = None):
        self._data = data
        self._freq = freq
        self._left = left
        self._right = right
        self._father = father

    def isleft(self):
        return self._left._father == self

    def __lt__(self, other):
        return self._freq < other._freq

def getCode(head, s, dicts):
    # 遍历的过程中存储编码表， 注意不要用if-else
    if head._left == None and head._right == None:
        dicts[head._data] = s
        return
    getCode(head._left, s+'0', dicts)
    getCode(head._right, s+'1', dicts)

def traversal(head):
    # 遍历哈夫曼树
    if head is None:
        return
    print(head._data, end=' ')
    traversal(head._left)
    traversal(head._right)

if __name__ == '__main__':
    q = PriorityQueue(30)
    dicts = getItems()
    for k, v in dicts.items():
        q.put((v, Node(k, v)))
    # 建立哈夫曼树
    while q.qsize() >= 2:
        node1 = q.get()[1]
        node2 = q.get()[1]
        fnode = Node('Nan', node1._freq + node2._freq, node1, node2)
        node1._father = fnode
        node2._father = fnode
        q.put((fnode._freq, fnode))
    hf = q.get()[1]

    print('哈夫曼树为：')
    traversal(hf)
    print('\n')
    # 实现哈夫曼编码
    s = 'C PROGRAM IS MY FAVORITE'
    rs = ''
    getCode(hf, '', dicts)
    for i in s:
        rs += dicts[i]
    print('哈夫曼编码表为：')
    print(dicts)
    print()
    print('哈夫曼编码为：')
    print(rs)
```



```
In [19]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/4-1.py', wdir='/Users/
zhujun/Downloads/USTC/专业补课/DS/DS-experiment')
哈夫曼树为:
Nan Nan Nan Nan Nan C U H Nan R S Nan E Nan I N Nan Nan Nan Nan Nan B G Nan P Y O Nan A Nan L D
Nan Nan Nan Nan Nan V Nan Nan Nan J X Nan Z Q K W Nan M F T

哈夫曼编码表为:
{'A': '1010', 'B': '100000', 'C': '00000', 'D': '10111', 'E': '010', 'F': '110011', 'G':
'100001', 'H': '0001', 'I': '0110', 'J': '1100001000', 'K': '11000011', 'L': '10110', 'M':
'110010', 'N': '0111', 'O': '1001', 'P': '100010', 'Q': '1100001011', 'R': '0010', 'S': '0011',
'T': '1101', 'U': '00001', 'V': '1100000', 'W': '110001', 'X': '1100001001', 'Y': '100011', 'Z':
'1100001010', ' ': '111'}

哈夫曼编码为:
000001111000100010100110000100101010110010111011000111111100101000111111100111010110000010010010
01101101010
```

4-2.py

```
'''拓扑排序， 每次从入度表中挑选入度为0的课程进行学习.注意判断有环图的情况'''
# 本题使用的数据结构有：
# 字典g： 实现图的邻接表存储
# 列表indegrees： 存储每个节点的入度
# 列表q： 存储入度为0的顶点，当列表为空，说明学习完毕

g = {
        '1': ['2', '3', '4', '12'],
        '2': ['3'],
        '3': ['5', '7', '8'],
        '4': ['5'],
        '5': ['7'],
        '6': ['8'],
        '7': [],
        '8': [],
        '9': ['10', '11', '12'],
        '10':['12'],
        '11':['6'],
        '12':[]
    }
indegrees = dict((u, 0) for u in g.keys())
for u in g.keys():
    for v in g[u]:
        indegrees[v] += 1
q = [ u for u in g.keys() if indegrees[u] == 0 ]
s = []
k = 0
while q:
    u = q.pop()
    k += 1
    s.append(u)
    for v in g[u]:
        indegrees[v] -= 1
        if indegrees[v] == 0:
            q.append(v)
if k != len(g):
    print('图有环')
else:
    print(s)
```



```
In [21]: runfile('/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment/4-2.py', wdir='/Users/zhujun/Downloads/USTC/专业补课/DS/DS-experiment')
['9', '11', '6', '10', '1', '12', '4', '2', '3', '8', '5', '7']
```