Assignment5

姓名：尹廖丹华

学号：2000092113

院系：心理与认知科学学院

一、 作业题目

1、22275: 二叉搜索树的遍历（3h）

http://cs101.openjudge.cn/practice/22275/

```
class Node():
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

def buildTree(preorder):
    if len(preorder) == 0:
        return None

    node = Node(preorder[0])

    idx = len(preorder)
    for i in range(1, len(preorder)):
        if preorder[i] > preorder[0]:
            idx = i
            break
    node.left = buildTree(preorder[1:idx])
    node.right = buildTree(preorder[idx:])

    return node

def postorder(node):
    if node is None:
        return []
    output = []
    output.extend(postorder(node.left))
    output.extend(postorder(node.right))
    output.append(str(node.val))

    return output

n = int(input())
preorder = list(map(int, input().split()))
print(' '.join(postorder(buildTree(preorder))))
```

2、02524: 宗教信仰（2h40min）

http://cs101.openjudge.cn/dsapre/02524/

```
def init_set(n):
  return list(range(n))

def get_father(x, father):
  if father[x] != x:
    father[x] = get_father(father[x], father)
  return father[x]

def join(x, y, father):
  fx = get_father(x, father)
  fy = get_father(y, father)
  if fx == fy:
    return
  father[fx] = fy

def is_same(x, y, father):
  return get_father(x, father) == get_father(y, father)

def main():
  case_num = 0
```

```
    while True:
        n, m = map(int, input().split())
        if n == 0 and m == 0:
            break
        count = 0
        father = init_set(n)
        for _ in range(m):
            s1, s2 = map(int, input().split())
            join(s1 - 1, s2 - 1, father)
        for i in range(n):
            if father[i] == i:
                count += 1
        case_num += 1
        print(f"Case {case_num}: {count}")

if __name__ == "__main__":
    main()
```

3、04078：实现堆结构（3h40min）

http://cs101.openjudge.cn/practice/04078/

```
class BinHeap:
    def __init__(self):
        self.heapList = [0]
        self.currentSize = 0
```

```
    def percUp(self, i):
        while i // 2 > 0:
            if self.heapList[i] < self.heapList[i // 2]:
                tmp = self.heapList[i // 2]
                self.heapList[i // 2] = self.heapList[i]
                self.heapList[i] = tmp
            i = i // 2

    def insert(self, k):
        self.heapList.append(k)
        self.currentSize = self.currentSize + 1
        self.percUp(self.currentSize)

    def percDown(self, i):
        while (i * 2) <= self.currentSize:
            mc = self.minChild(i)
            if self.heapList[i] > self.heapList[mc]:
                tmp = self.heapList[i]
                self.heapList[i] = self.heapList[mc]
                self.heapList[mc] = tmp
            i = mc

    def minChild(self, i):
        if i * 2 + 1 > self.currentSize:
            return i * 2
        else:
            if self.heapList[i * 2] < self.heapList[i * 2 + 1]:
                return i * 2
            else:
                return i * 2 + 1

    def delMin(self):
        retval = self.heapList[1]
        self.heapList[1] = self.heapList[self.currentSize]
        self.currentSize = self.currentSize - 1
        self.heapList.pop()
        self.percDown(1)
        return retval

    def buildHeap(self, alist):
        i = len(alist) // 2
        self.currentSize = len(alist)
        self.heapList = [0] + alist[:]
        while (i > 0):
            self.percDown(i)
            i = i - 1

n = int(input().strip())
bh = BinHeap()
for _ in range(n):
    inp = input().strip()
    if inp[0] == '1':
        bh.insert(int(inp.split()[1]))
    else:
        print(bh.delMin())
```

4、22161: 哈夫曼编码树（4h30min）

http://cs101.openjudge.cn/practice/22161/

```
import heapq

class Node:
    def __init__(self, weight, char=None):
        self.weight = weight
        self.char = char
        self.left = None
        self.right = None

    def __lt__(self, other):
        if self.weight == other.weight:
            return self.char < other.char
        return self.weight < other.weight

def build_huffman_tree(characters):
    heap = []
    for char, weight in characters.items():
        heapq.heappush(heap, Node(weight, char))

    while len(heap) > 1:
        left = heapq.heappop(heap)
        right = heapq.heappop(heap)
        merged = Node(left.weight + right.weight, min(left.char, right.char))
        merged.left = left
        merged.right = right
        heapq.heappush(heap, merged)
```

```
    return heap[0]

def encode_huffman_tree(root):
    codes = {}

    def traverse(node, code):
        if node.left is None and node.right is None:
            codes[node.char] = code
        else:
            traverse(node.left, code + '0')
            traverse(node.right, code + '1')

    traverse(root, '')
    return codes

def huffman_encoding(codes, string):
    encoded = ''
    for char in string:
        encoded += codes[char]
    return encoded

def huffman_decoding(root, encoded_string):
    decoded = ''
    node = root
    for bit in encoded_string:
        if bit == '0':
            node = node.left
        else:
            node = node.right

        if node.left is None and node.right is None:
            decoded += node.char
            node = root
    return decoded

n = int(input())
characters = {}
for _ in range(n):
    char, weight = input().split()
    characters[char] = int(weight)

huffman_tree = build_huffman_tree(characters)

codes = encode_huffman_tree(huffman_tree)

strings = []
while True:
    try:
        line = input()
        strings.append(line)

    except EOFError:
        break

results = []
for string in strings:
    if string[0] in ('0','1'):
```

```
        results.append(huffman_decoding(huffman_tree, string))
    else:
        results.append(huffman_encoding(codes, string))

for result in results:
    print(result)
```

二、学习的感想与收获

还是没有学会树的写法，又是看着答案理解的一周…

最近 ddl 还是一样的多，都没能有时间好好理解代码的写法，哭了

这周有时间再继续学习！！！