알고리즘과 문제해결 일반과제3

20181061 전하영

1. 다음 문장에서 각 문자의 빈도수를 구하시오. (공백은 무시하시오)

```python
from collections import Counter
import heapq

# 빈도수 구하기
def get_frequency(text):
    return dict(Counter(text))
```

```python
# 메인함수
def solution(text):
    # 공백 무시, 공백 제거
    text = text.replace(" ","")
    frequency = get_frequency(text)
    print(frequency)
```

```
{'a': 13, 'd': 14, 'f': 12, 's': 10, 'g': 3}
```

2. 이들 문자의 허프만 코드를 위한 허프만 트리를 만드시오.

```python
from collections import Counter
import heapq

# 빈도수 구하기
def get_frequency(text):
    return dict(Counter(text))
```

```python
# 허프만 트리 구성하기
def make_tree(frequency):
    # 여기서 '' 은 아직 할당되지 않은 이진코드
    heap = [[weight, [symbol, '']] for symbol, weight in frequency.items()]
    # min heap 만들기
    heapq.heapify(heap)

    # 노드가 1개 이하로 떨어질 때 까지 반복
    while len(heap) > 1:
        # heap 에서 빈도수가 제일 작은 노드 2개를 순차적으로 꺼내온다.
        lo = heapq.heappop(heap)
        hi = heapq.heappop(heap)
        # [1:] 의 의미는 weight 값을 제외하고 가져오겠단 뜻.


        for pair in lo[1:]:
            # 가져온 노드에 이진코드를 붙여줌. lo 부분이 이진트리에서 앞부분에 해당하므로 0을 앞에 붙여줌
            pair[1] = '0' + pair[1]

        for pair in hi[1:]:
            # 가져온 노드에 이진코드를 붙여줌. hi 부분이 이진트리에서 뒷부분에 해당하므로 1을 앞에 붙여줌
            pair[1] = '1' + pair[1]
        # 상위 노드를 추가함. 상위 노드의 weight는 하위 노드 2개의 weight의 합이다.
        heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])
    # 완성한 허프만 트리를 반환함.
    return sorted(heapq.heappop(heap)[1:], key=lambda p: (len(p[-1]), p))
```

```python
tree = make_tree(frequency)
print(tree)
```

```
[['a', '01'], ['d', '11'], ['f', '00'], ['g', '100'], ['s', '101']]
```

3. 위 문장을 허프만 압축 하시오

```
# 압축하기 (encode)
def encode(tree,text):
    encode_text = ""
    for c in text:
        for node in tree:
            if c in node[0]:
                encode_text = encode_text+node[1]
    return encode_text
```

```
encode_text = encode(tree,text)
print(encode_text)
```

```
01111000110111110001010110111110011101010111000110100010110100111111000101101110011000000101111001000010110010101101
```

01111000110111110001010110111110011101010111000110100010110100111111000101101110011000000101111001000010110010101101

4. 허프만 압축의 압축률은 얼마인가?

```
# 압축률 구하기
def get_compression_rate(plain_text, frequency, tree):
    ascii_text_size = 0
    compression_text_size = 0
    for node in tree:
        ascii_text_size += int(frequency.get(node[0]))*8
        compression_text_size += int(frequency.get(node[0]))*len(node[1])
    return (compression_text_size/ascii_text_size)*100
```

```
compression_rate = get_compression_rate(text,frequency,tree)
print(compression_rate,"%")
```

```
28.125 %
```

5. 압축된 문장을 다시 푸시오

```
# 복원하기 (decode)
def decode(cipher_text,tree):

    def find_key(dict, val):
        return next(key for key, value in dict.items() if value == val)

    plain_text=''
    temp=''
    temp_tree = dict(tree)
    for c in cipher_text:
        temp += c
        try:
            plain_text += find_key(temp_tree,temp)
            temp=''
        except:
            pass
    return plain_text
```

```
plain_text = decode(encode_text,tree)
print(plain_text)
```

```
addfasddfaaasddfdsaadfasfaasfdddfaasdfdfffsdggfsgsas
```

```
solution("add fasd dfa aas dd fd saad fasf aas fdd dfaas df dfff sdgg fsgsas")
```

```
{'a': 13, 'd': 14, 'f': 12, 's': 10, 'g': 3}
[['a', '01'], ['d', '11'], ['f', '00'], ['g', '100'], ['s', '101']]
01111000110111110001010110111110011101010111000110100101101001111110001011011100110000001011110010000101100101011012
28.125 %
addfasddfaaasddfdsaadfasfaasfdddfaasdfdfffsdggfsgsas
```