HUFFMAN CODES

In Algorithm

201935058 백민우

202031348 민정원

202033762 장민호

202135514 김미소

202135553 윤세현

CONTENTS

- WHAT IS HUFFMAN CODE?

HUFFMAN CODES

- IMPLEMENTATION METHOD
- PSEUDO CODE

In Algorithm

201935058 백민우

202031348 민정원

202033762 장민호

202135514 김미소

202135553 윤세현

WHAT IS HUFFMAN CODE ?

Fixed Length Code

Giving a code of equal length into each character

ex) ASCII CODE

A : 1000001

B : 1000010

C : 1000011

24 BIT

vs

Variable Length Code

Giving a code of variable length into each character

ex) HUFFMAN CODE

A : 01

B : 10

C : 111

7 BIT

WHAT IS HUFFMAN CODE ?

HUFFMAN CODE

No data loss during encoding or decoding

By. David A. Huffman in 1951

Huffman code is a lossless data compression algorithm that produces prefix codes from the frequency of characters.

a code that is made so that no code is a prefix to another code

WHAT IS
HUFFMAN
CODE ?

GREEDY ALGORITHM

An approach for solving a problem by selecting the best option available at the moment

IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

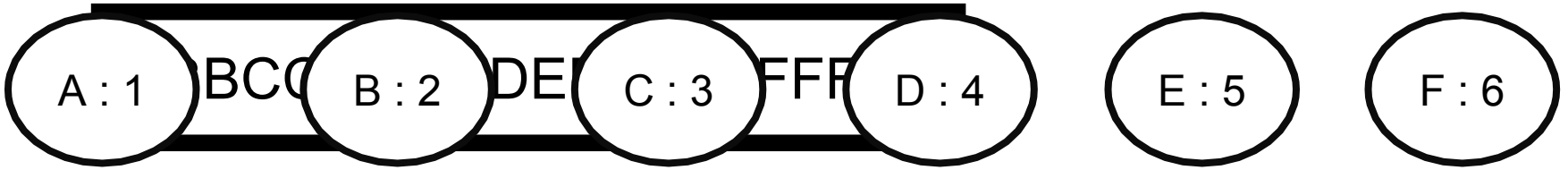
C : 3

D : 4

E : 5

F : 6

1. Create the nodes that include the frequency of the each character



A : 1

B : 2

C : 3

D : 4

E : 5

F : 6

IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

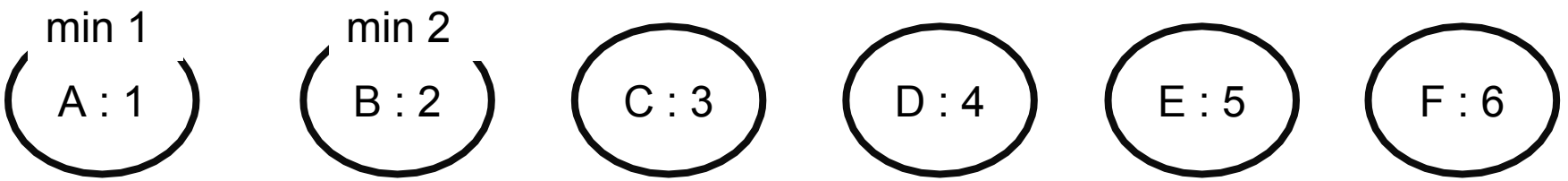
C : 3

D : 4

E : 5

F : 6

2. Compare frequencies to find the least frequently and the second least frequently



IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

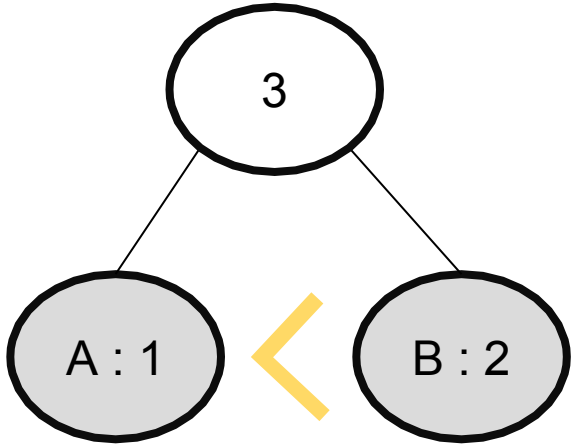
C : 3

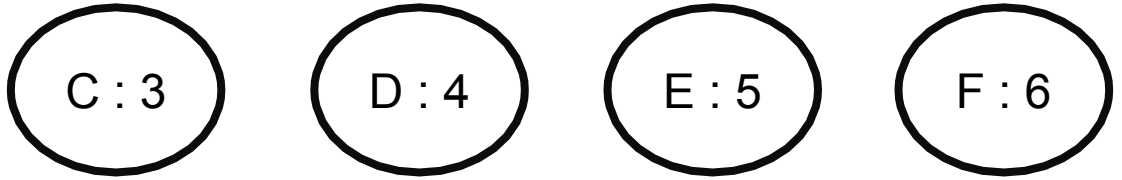
D : 4

E : 5

F : 6

3-1. Of the two nodes, the smaller one is placed on the left and the larger one on the right.





IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

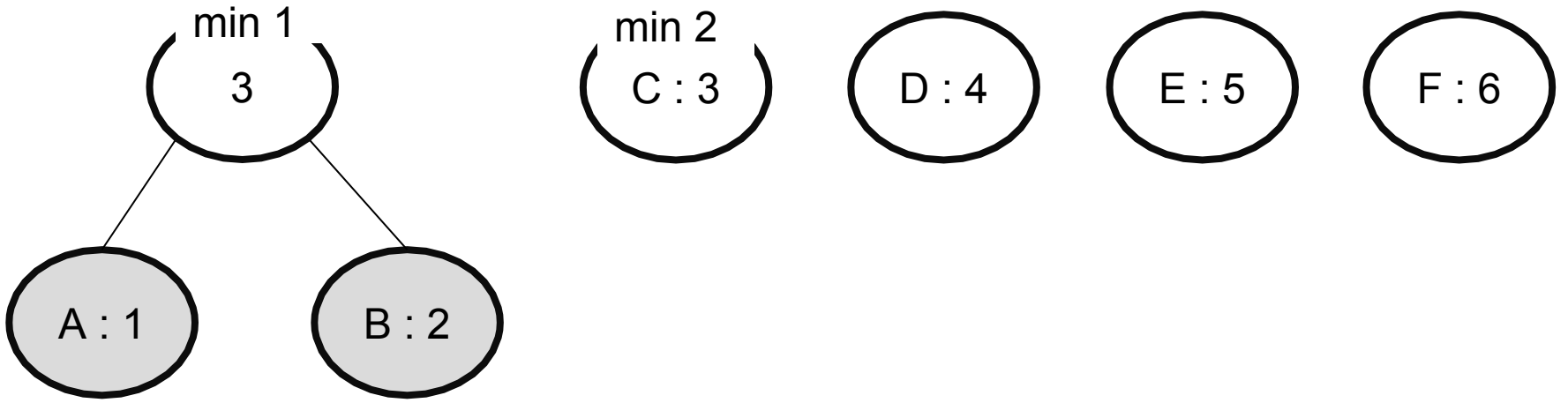
C : 3

D : 4

E : 5

F : 6

3. Create the new parent node that include the sum of the two nodes



IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

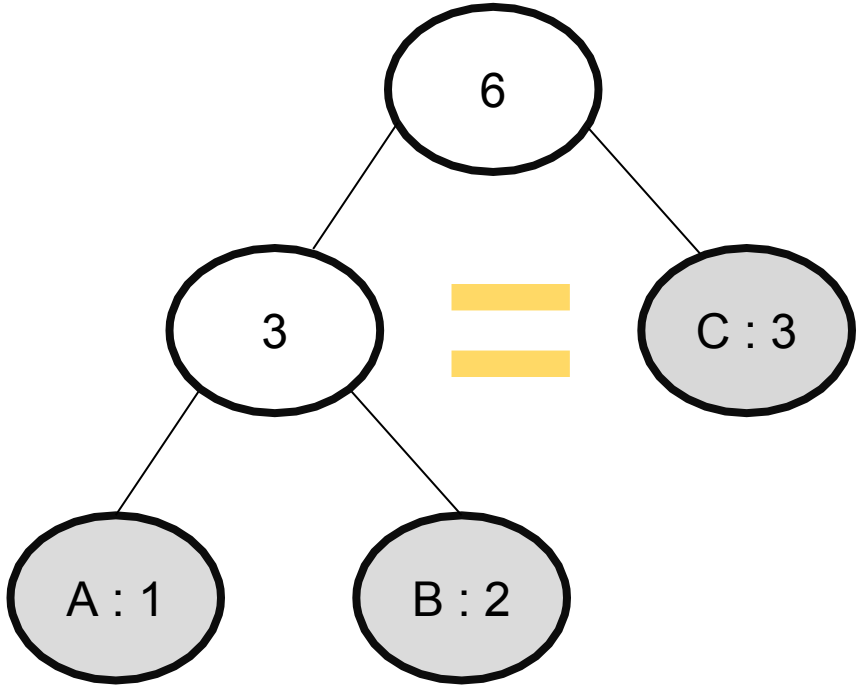
C : 3

D : 4

E : 5

F : 6

3-2. If the frequency is the same, the smaller tree size is preferred.



D : 4

E : 5

F : 6

IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

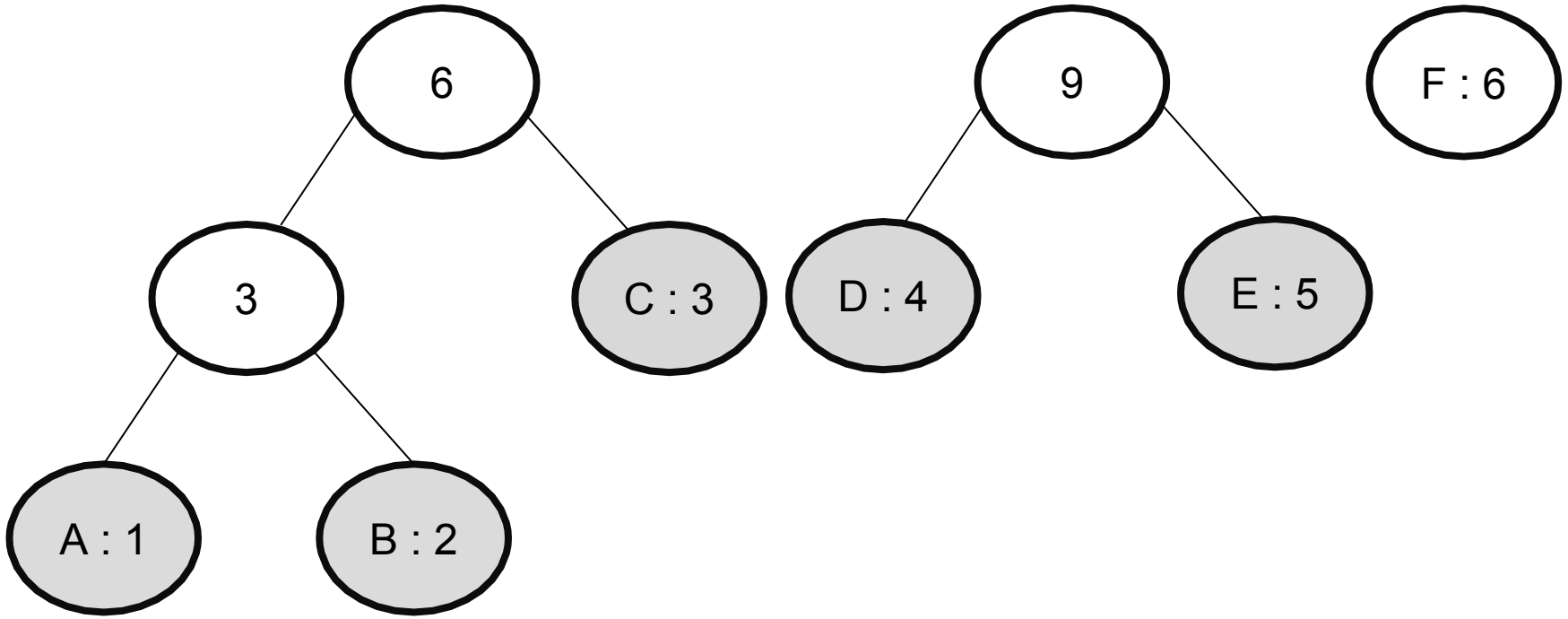
C : 3

D : 4

E : 5

F : 6

3. Create the new parent node that include the sum of the two nodes



IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

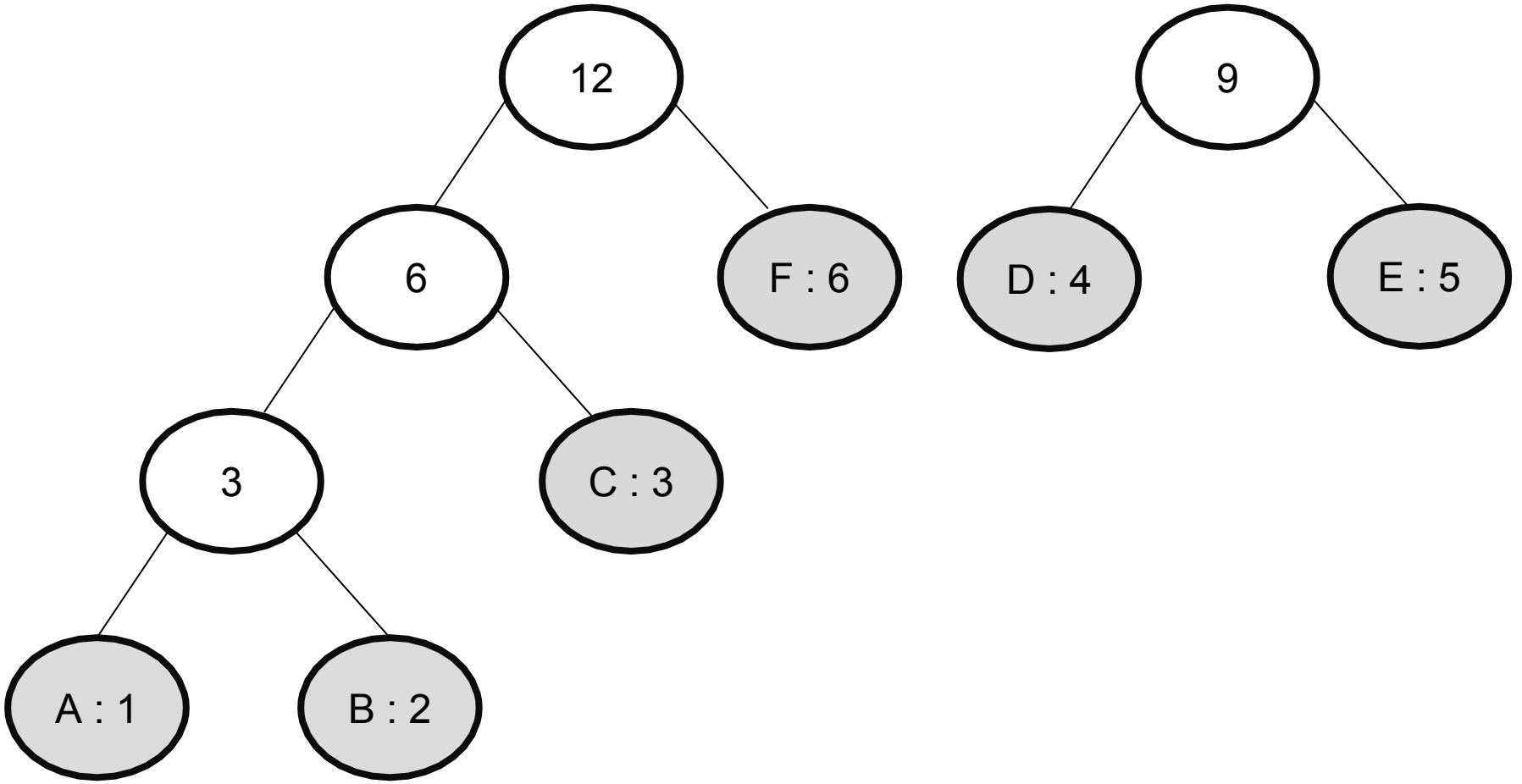
C : 3

D : 4

E : 5

F : 6

3. Create the new parent node that include the sum of the two nodes



IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

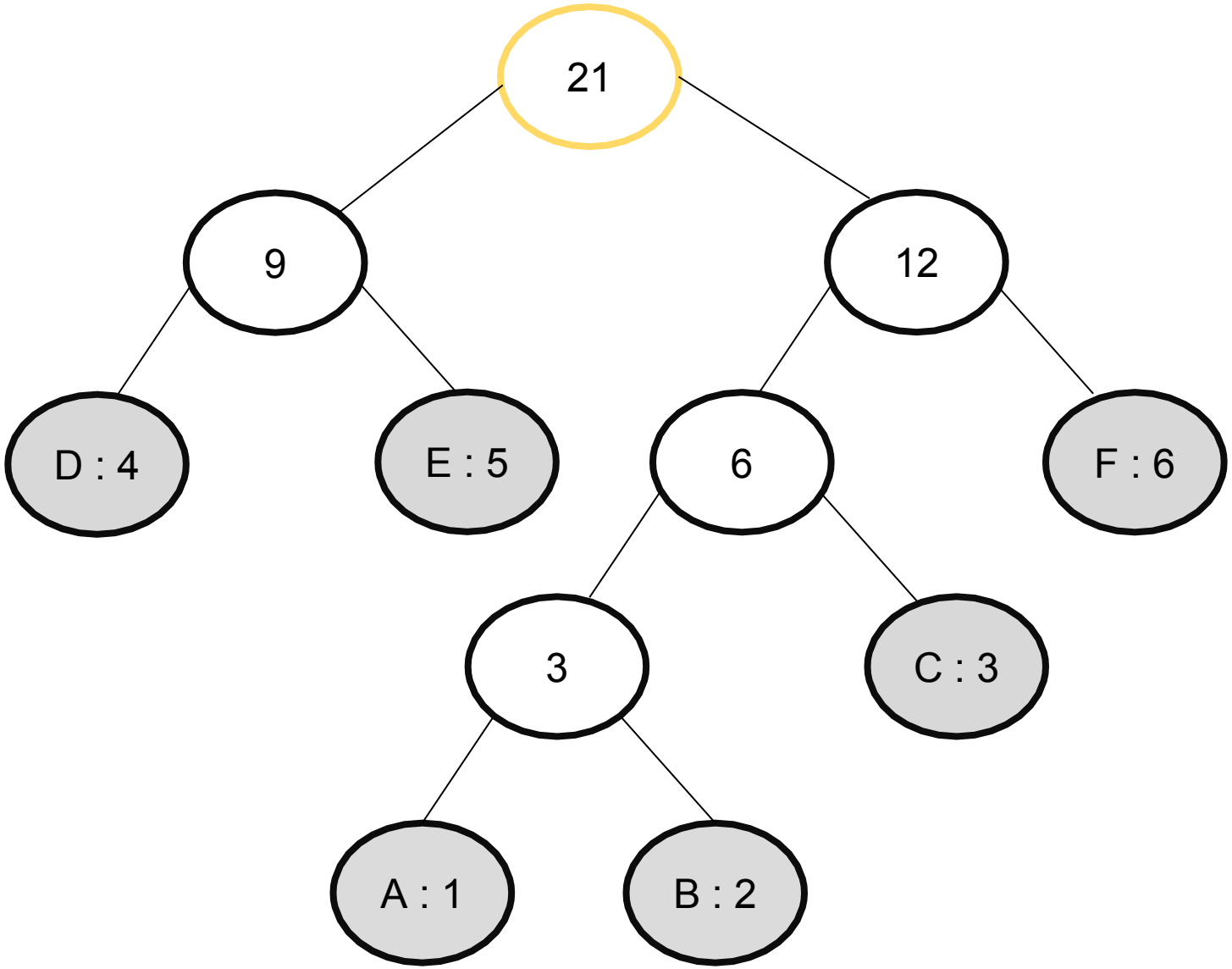
C : 3

D : 4

E : 5

F : 6

4. Repeat until there is one node left to compare.



IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1

B : 2

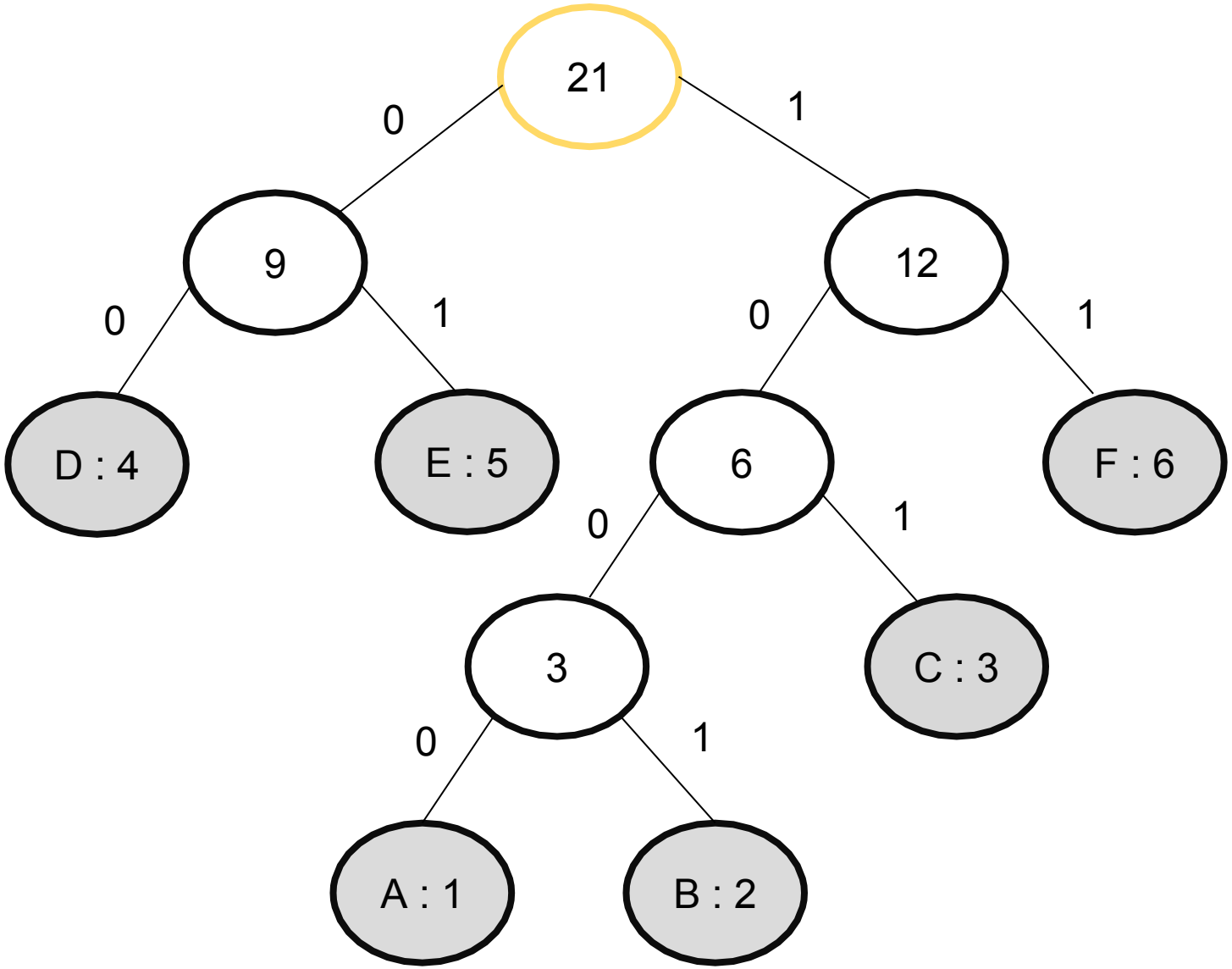
C : 3

D : 4

E : 5

F : 6

5. The tree node has a weight of 0 on the left and 1 on the right.

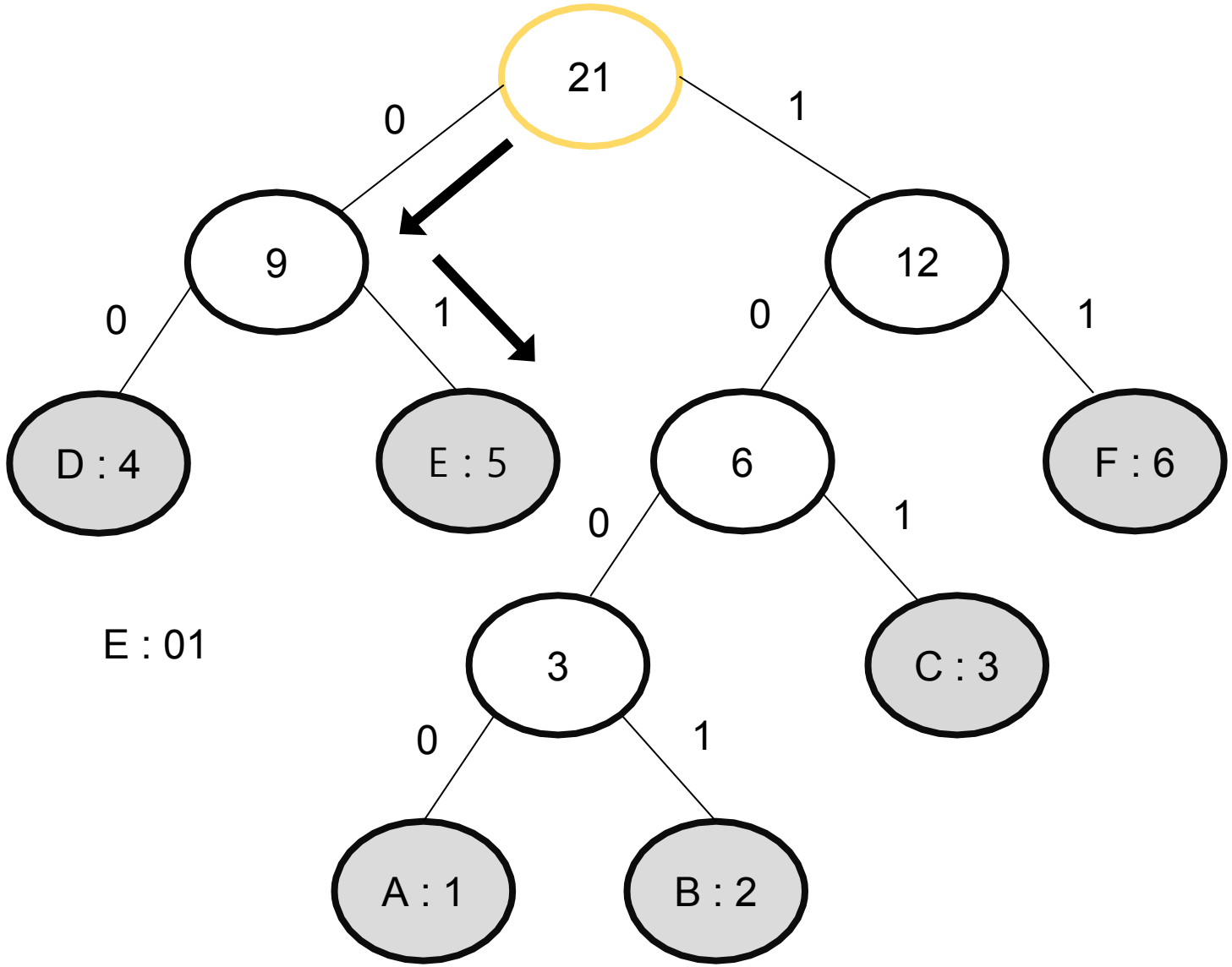


IMPLEMENTATION METHOD

ABBCCCDDDDEEEEEFFFFFF

A : 1
B : 2
C : 3
D : 4
E : 5
F : 6

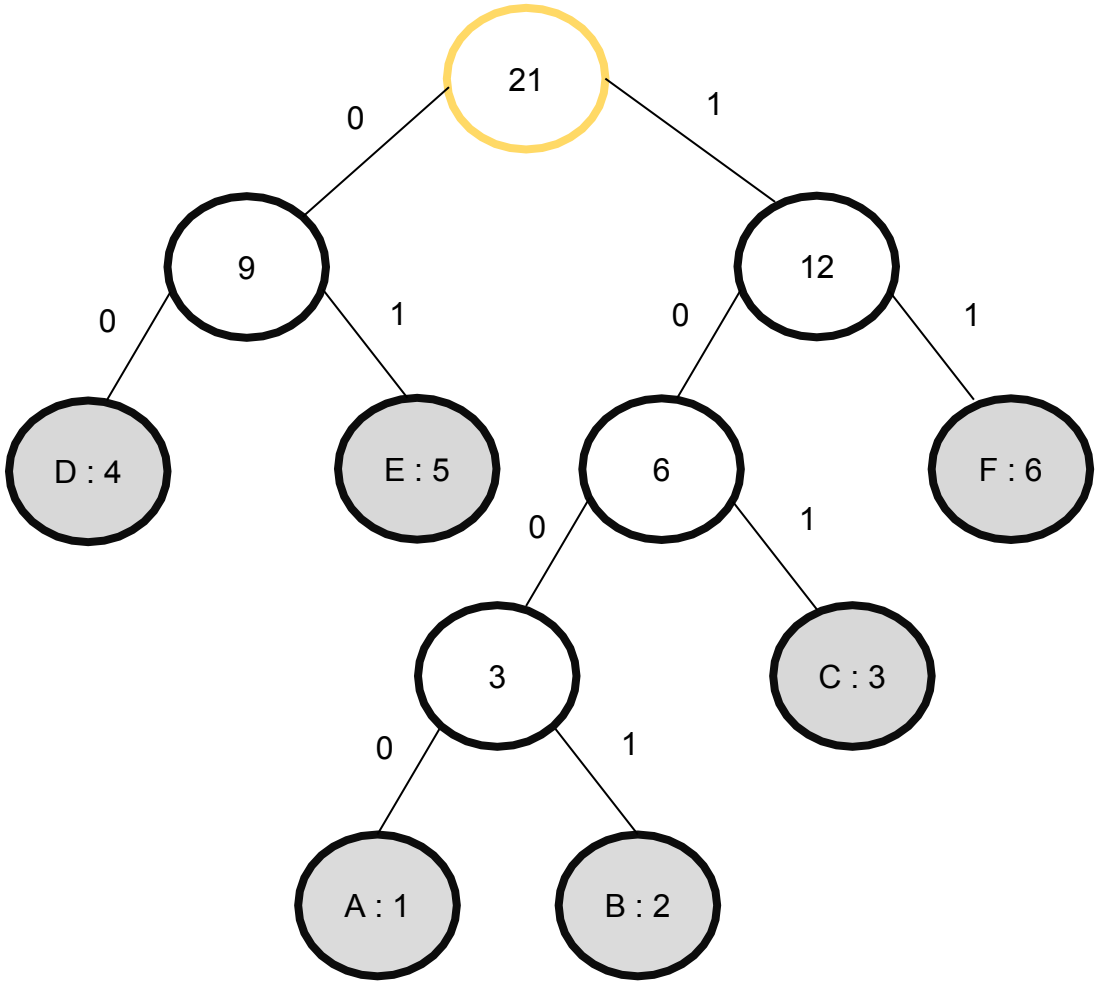
6. When characters are searched from the root, the sum of the weights of the past edges becomes Huffman code.



E : 01

IMPLEMENTATION METHOD

6. When characters are searched from the root, the sum of the weights of the past edges becomes Huffman code.



ABBCCCDDDDEEEEEFFFFF
F

A : 1000	VS	A : 000
B : 1001		B : 001
C : 101		C : 010
D : 00		D : 011
E : 01		E : 100
F : 11		F : 101
51BIT		63BIT

IMPLEMENTATION METHOD

Short binary codes (Huffman codes) are given to characters with high frequency,

Long binary codes are given to characters with low frequency to increase compression efficiency.

PSEUDO CODE

```
HUFFMAN(C)
n = length of C
Queue = C

for i = 1 to n-1
        allocate a new node z
    z.right = x = Extract_Min(Queue)
    z.left = y = Extract_Min(Queue)
    z.freq = x.freq + y.freq
    Insert(Q,z)

return Extract_Min(queue)
```

The time complexity is O(nlogn) because it takes O(n) time to construct heap as an initial priority queue and n-1 extract_min(Queue) O(logn) in the for statement.

(This is the time complexity based on the above code, and the overall time complexity also includes generating n nodes and storing frequencies O(n), and returning O(1) of tree root)

EXAMPLE - BAEKJOON

문제

소설가인 김대전은 소설을 여러 장(chapter)으로 나누어 쓰는데, 각 장은 각각 다른 파일에 저장하곤 한다. 소설의 모든 장을 쓰고 나서는 각 장이 쓰여진 파일을 합쳐서 최종적으로 소설의 완성본이 들어있는 한 개의 파일을 만든다. 이 과정에서 두 개의 파일을 합쳐서 하나의 임시파일을 만들고, 이 임시파일이나 원래의 파일을 계속 두 개씩 합쳐서 파일을 합쳐나가고, 최종적으로는 하나의 파일로 합친다. 두 개의 파일을 합칠 때 필요한 비용(시간 등)이 두 파일 크기의 합이라고 가정할 때, 최종적인 한 개의 파일을 완성하는데 필요한 비용의 총 합을 계산하시오.

예를 들어, C1, C2, C3, C4가 네 개의 장을 수록하고 있는 파일이고, 파일 크기가 각각 40, 30, 30, 50 이라고 하자. 이 파일들을 합치는 과정에서, 먼저 C2와 C3를 합쳐서 임시파일 X1을 만든다. 이때 비용 60이 필요하다. 그 다음으로 C1과 X1을 합쳐 임시파일 X2를 만들면 비용 100이 필요하다. 최종적으로 X2와 C4를 합쳐 최종파일을 만들면 비용 150이 필요하다. 따라서, 최종의 한 파일을 만드는데 필요한 비용의 합은 60+100+150=310 이다. 다른 방법으로 파일을 합치면 비용을 줄일 수 있다. 먼저 C1과 C2를 합쳐 임시파일 Y1을 만들고, C3와 C4를 합쳐 임시파일 Y2를 만들고, 최종적으로 Y1과 Y2를 합쳐 최종파일을 만들 수 있다. 이때 필요한 총 비용은 70+80+150=300 이다.

소설의 각 장들이 수록되어 있는 파일의 크기가 주어졌을 때, 이 파일들을 하나의 파일로 합칠 때 필요한 최소비용을 계산하는 프로그램을 작성하시오.

입력

프로그램은 표준 입력에서 입력 데이터를 받는다. 프로그램의 입력은 T개의 테스트 데이터로 이루어져 있는데, T는 입력의 맨 첫 줄에 주어진다.각 테스트 데이터는 두 개의 행으로 주어지는데, 첫 행에는 소설을 구성하는 장의 수를 나타내는 양의 정수 K (3 ≤ K ≤ 1,000,000)가 주어진다. 두 번째 행에는 1장부터 K장까지 수록한 파일의 크기를 나타내는 양의 정수 K개가 주어진다. 파일의 크기는 10,000을 초과하지 않는다.

출력

프로그램은 표준 출력에 출력한다. 각 테스트 데이터마다 정확히 한 행에 출력하는데, 모든 장을 합치는데 필요한 최소비용을 출력한다.

예제 입력 1 복사

```
2
4
40 30 30 50
15
1 21 3 4 5 35 5 4 3 5 98 21 14 17 32
```

예제 출력 1 복사

```
300
826
```

13975번: 파일 합치기 3 (acmicpc.net)

SOURCES

https://syj-computer.tistory.com/40

https://www.codesdope.com/course/algorithms-huffman-codes/

https://mirrorofcode.tistory.com/67

https://ccsuniversity.ac.in/bridge-library/pdf/MCA-DS-NEXT-MCA-212.pdf

https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

THANK YOU

201935058 백민우

202031348 민정원

202033762 장민호

202135514 김미소

202135553 윤세현