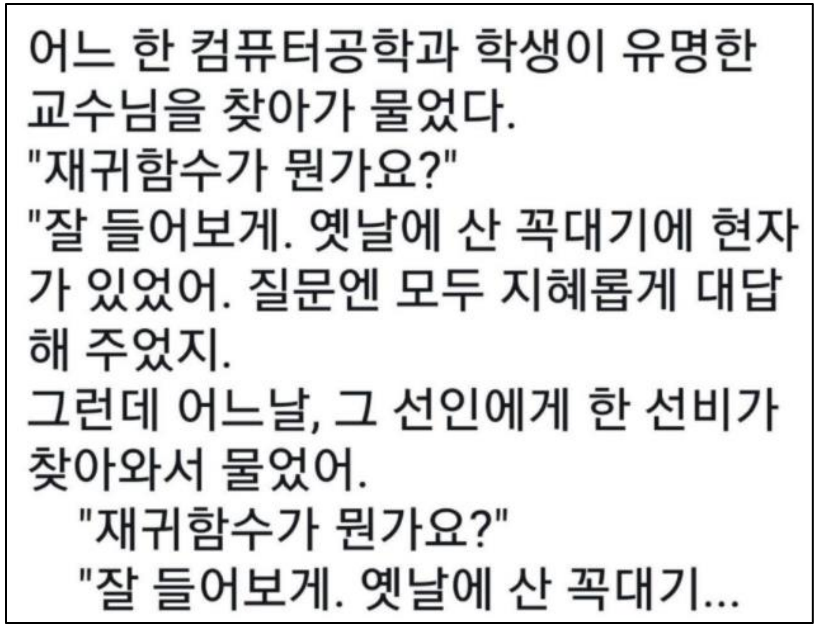
재귀 - 문제 6번



어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.

"재귀함수가 뭔가요?"

"잘 들어보게. 옛날에 산 꼭대기에 현자가 있었어. 질문엔 모두 지혜롭게 대답해 주었지.

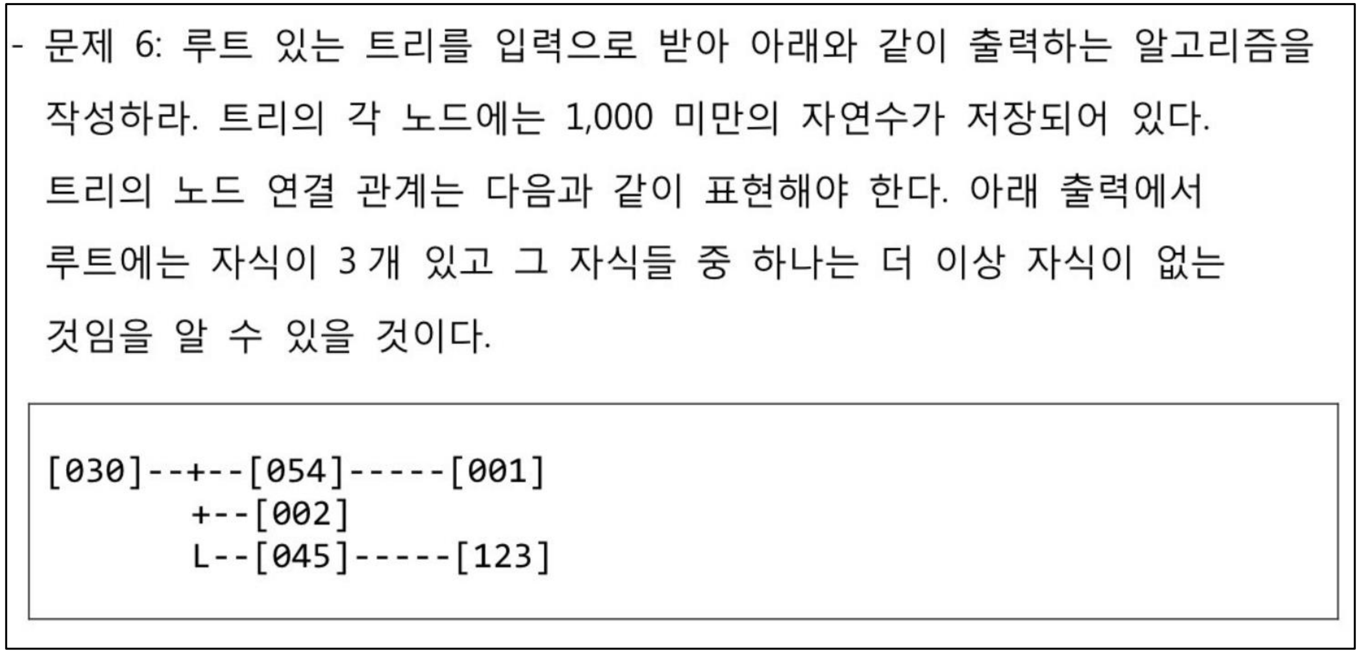
그런데 어느날, 그 선인에게 한 선비가 찾아와서 물었어.

"재귀함수가 뭔가요?"

"잘 들어보게. 옛날에 산 꼭대기...

- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```



- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

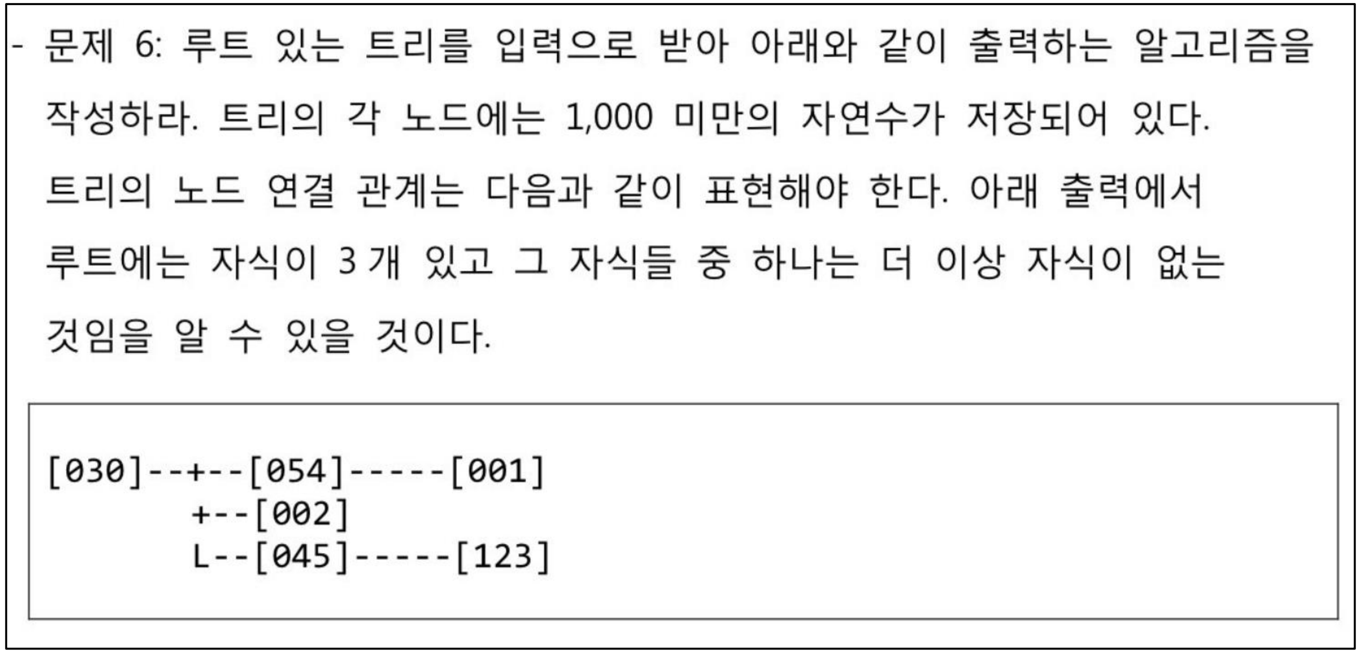
```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```

문제가 길지만

정리를 해보면

노드를 입력할테니
그에 맞게 트리를 그려봐!

라고 할 수 있다.



- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```

문제가 길지만

정리를 해보면

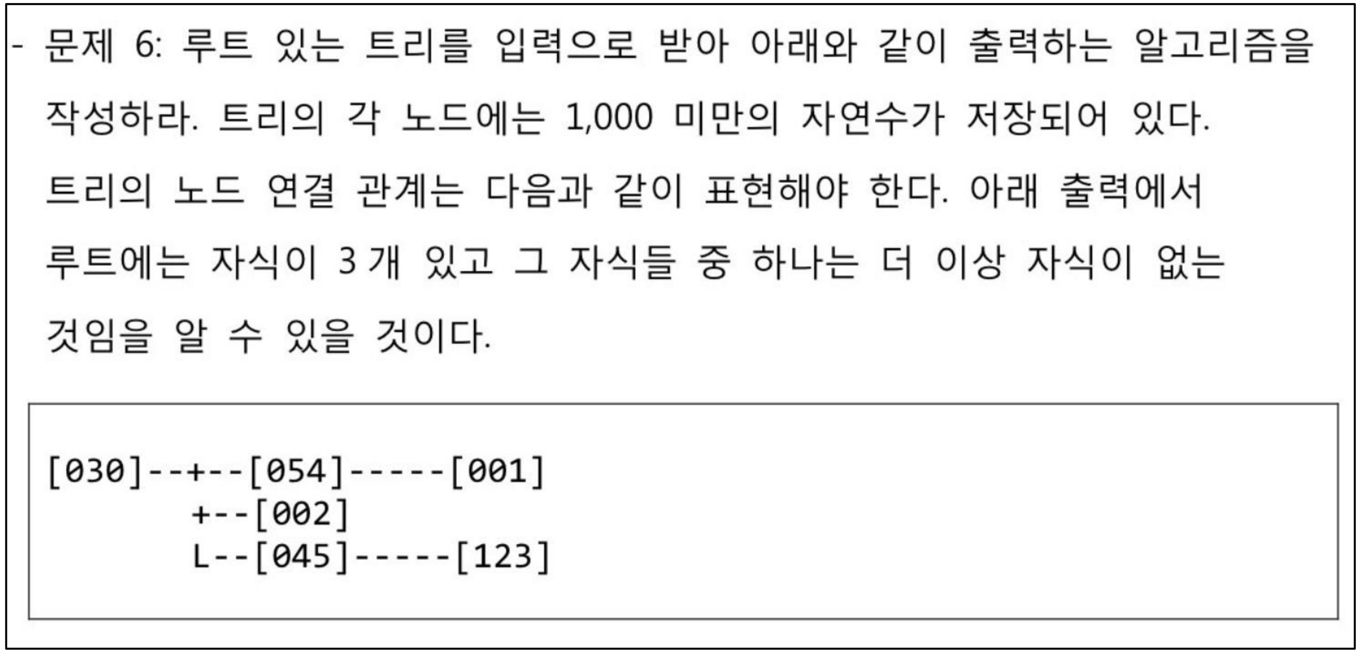
노드를 입력할테니
그에 맞게 트리를 그려봐!

라고 할 수 있다.

문제를 풀기위한 두 가지 규칙

[입력은 이렇게 받습니다.]

[중복되는 노드는 없다고 가정한다.]



- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```

문제가 길지만

정리를 해보면

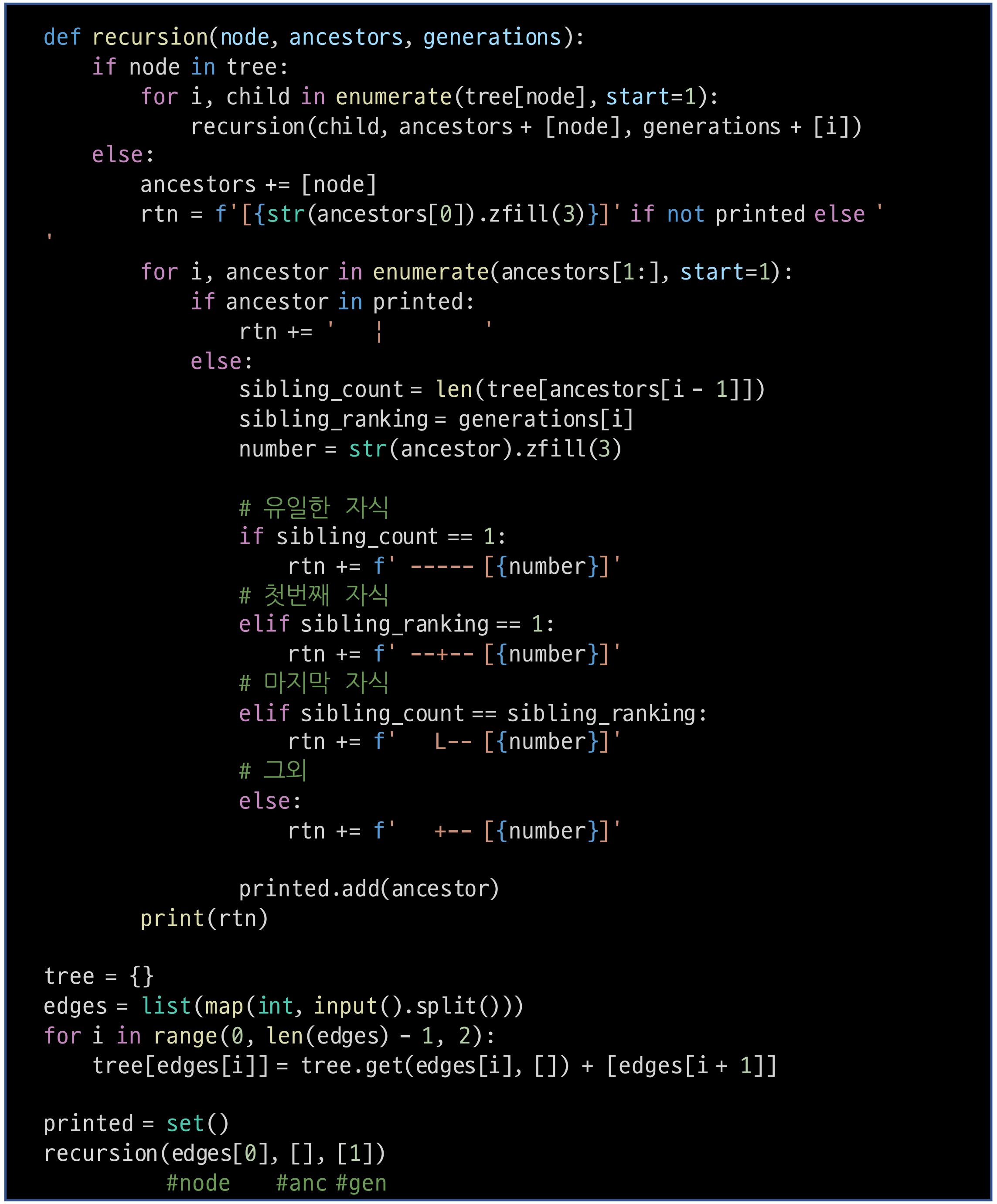
노드를 입력할테니
그에 맞게 트리를 그려봐!

라고 할 수 있다.

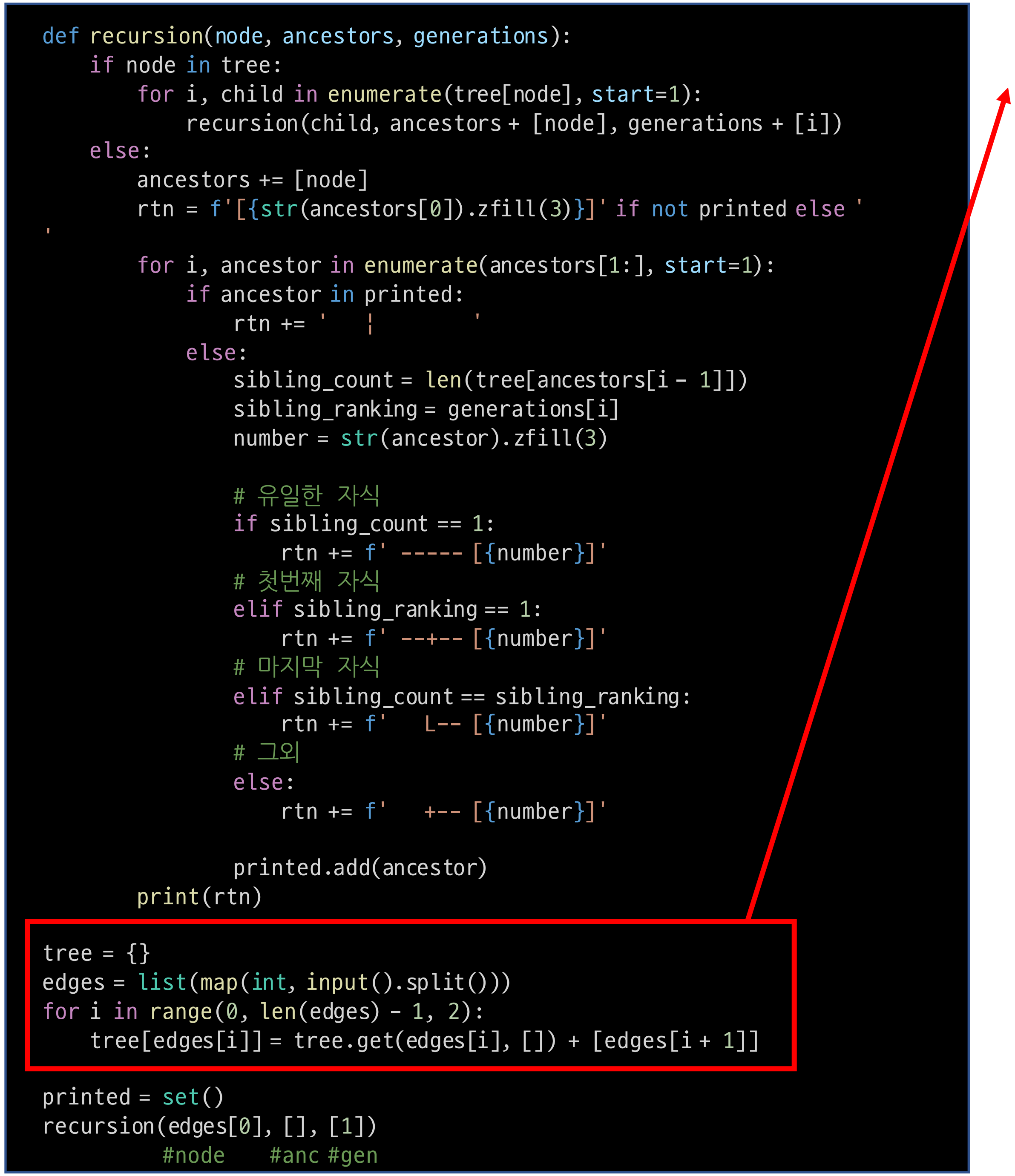
문제를 풀기위한 두 가지 규칙

[입력은 이렇게 받습니다.]
30 54 30 2 30 45 54 1 54 3 45 123 1 101 1 102 3 103

[중복되는 노드는 없다고 가정한다.]



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```



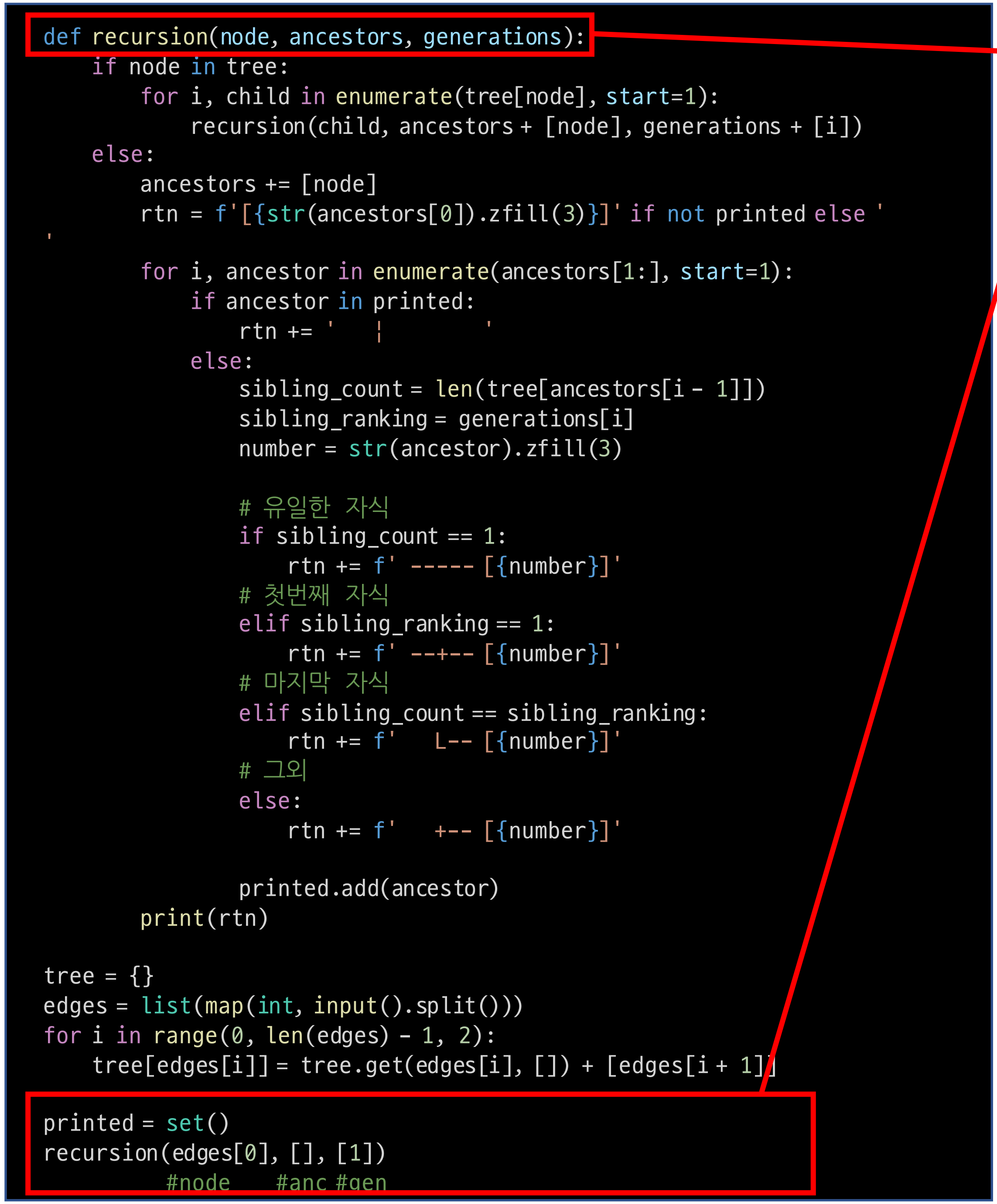
```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'
        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦         '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

30 54 30 2 30 45 54 1 54 3 45 123 1 101 1 102 3 103

```
tree = {

}

tree = {
    30: [54]
}

tree = {
    30: [54, 2]
}
```

⬇

```
tree = {
    30: [54, 2, 45],
    54: [1, 3],
    45: [123],
    1: [101, 102],
    3: [103]
}
```



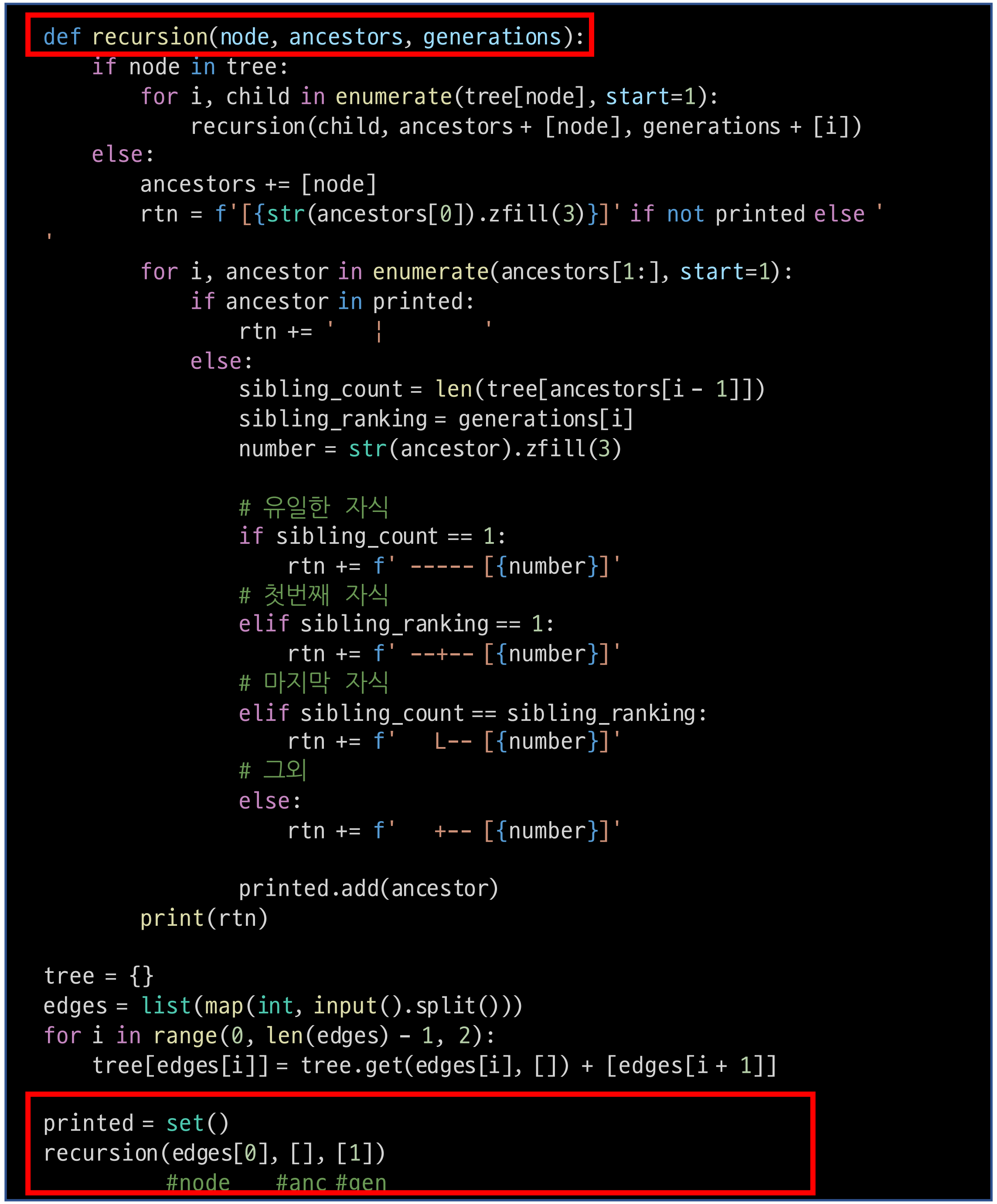
```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

중요한 개념 세 가지

1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦               L-- [102]
                    L-- [003] ----- [103]
```



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'
        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

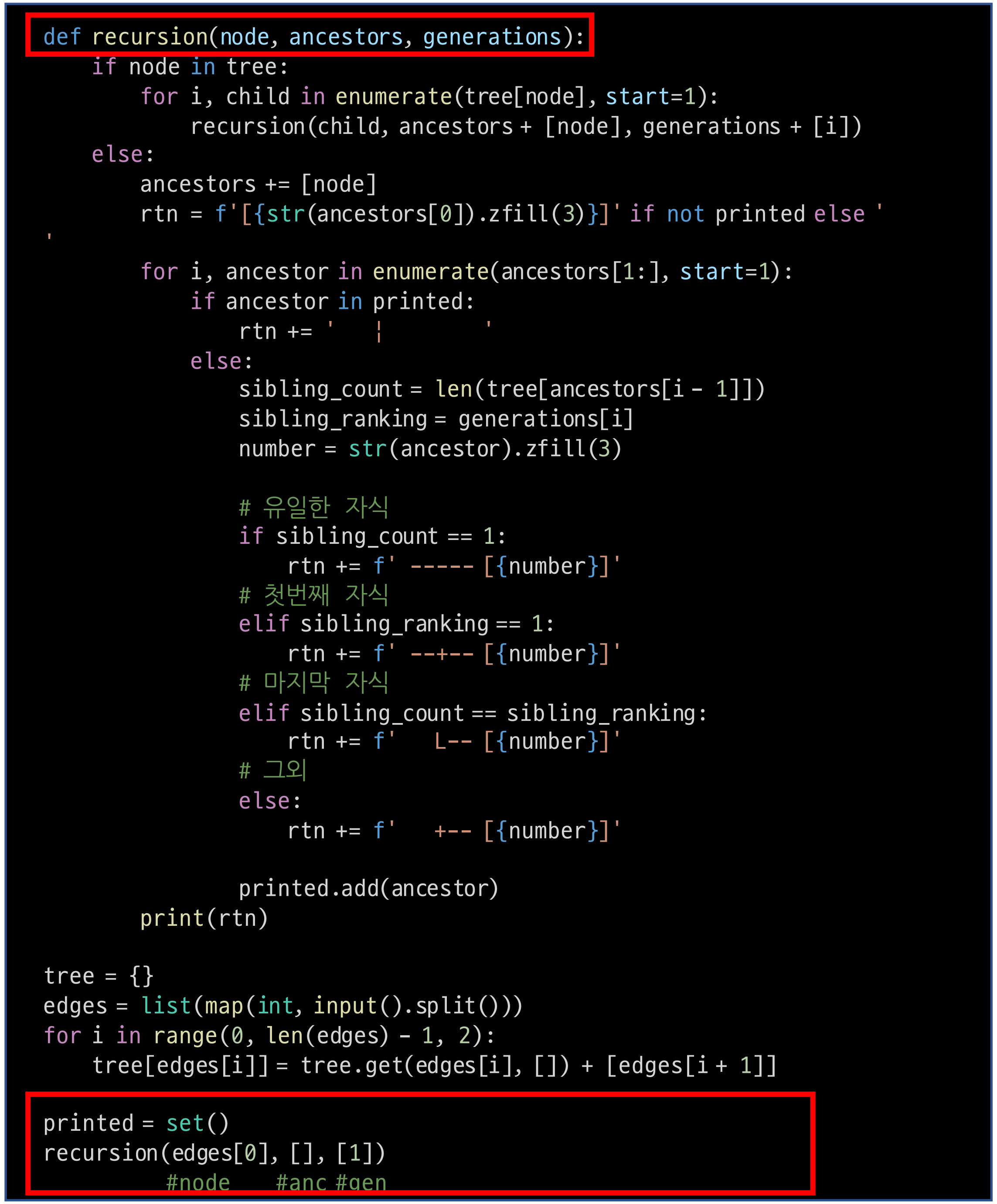
중요한 개념 세 가지

1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦           L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}
```



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

중요한 개념 세 가지

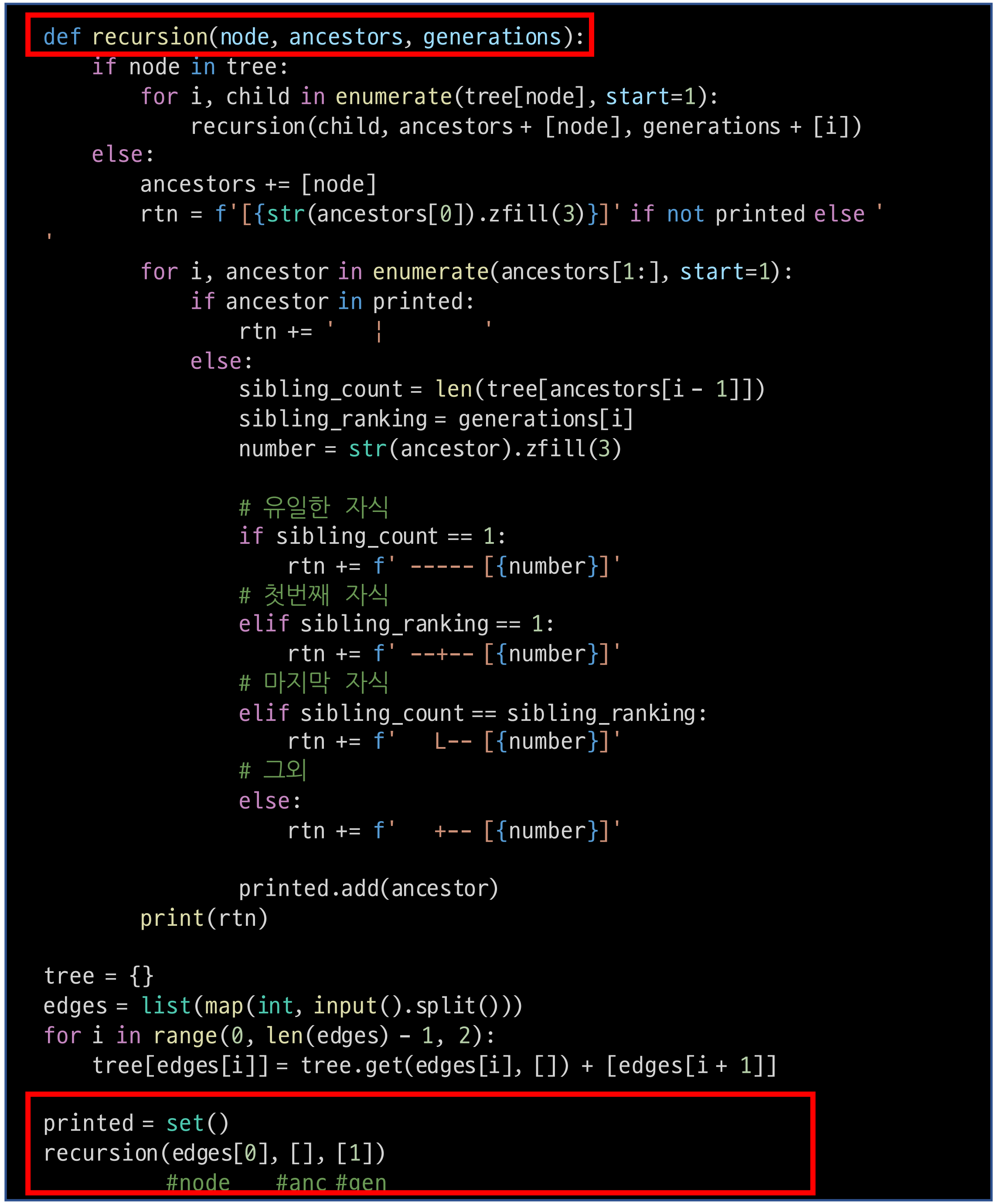
1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦           L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}
```

recursion(30, [], [1])



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'
        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

중요한 개념 세 가지

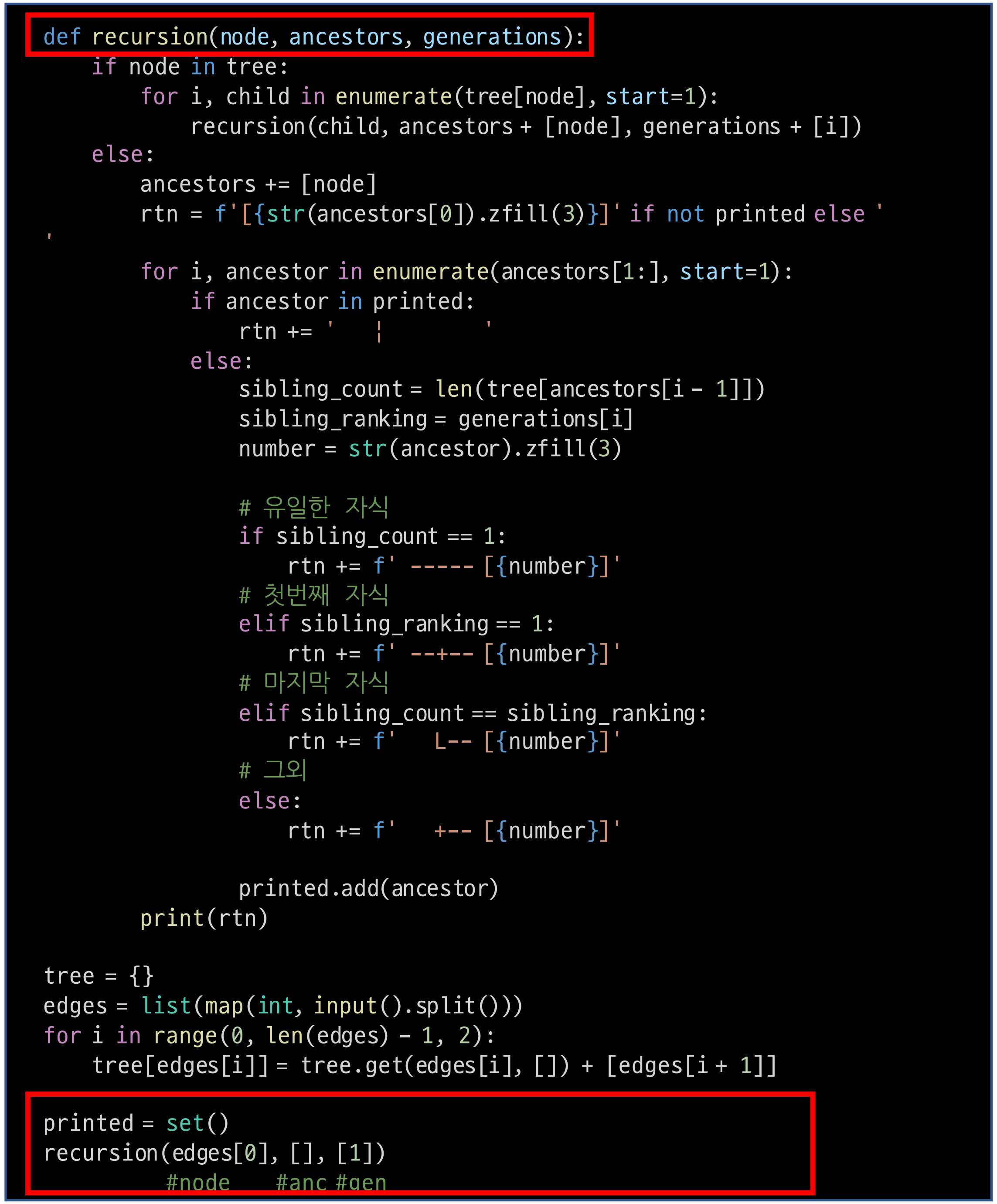
1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}
```

recursion(30, [], [1])



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

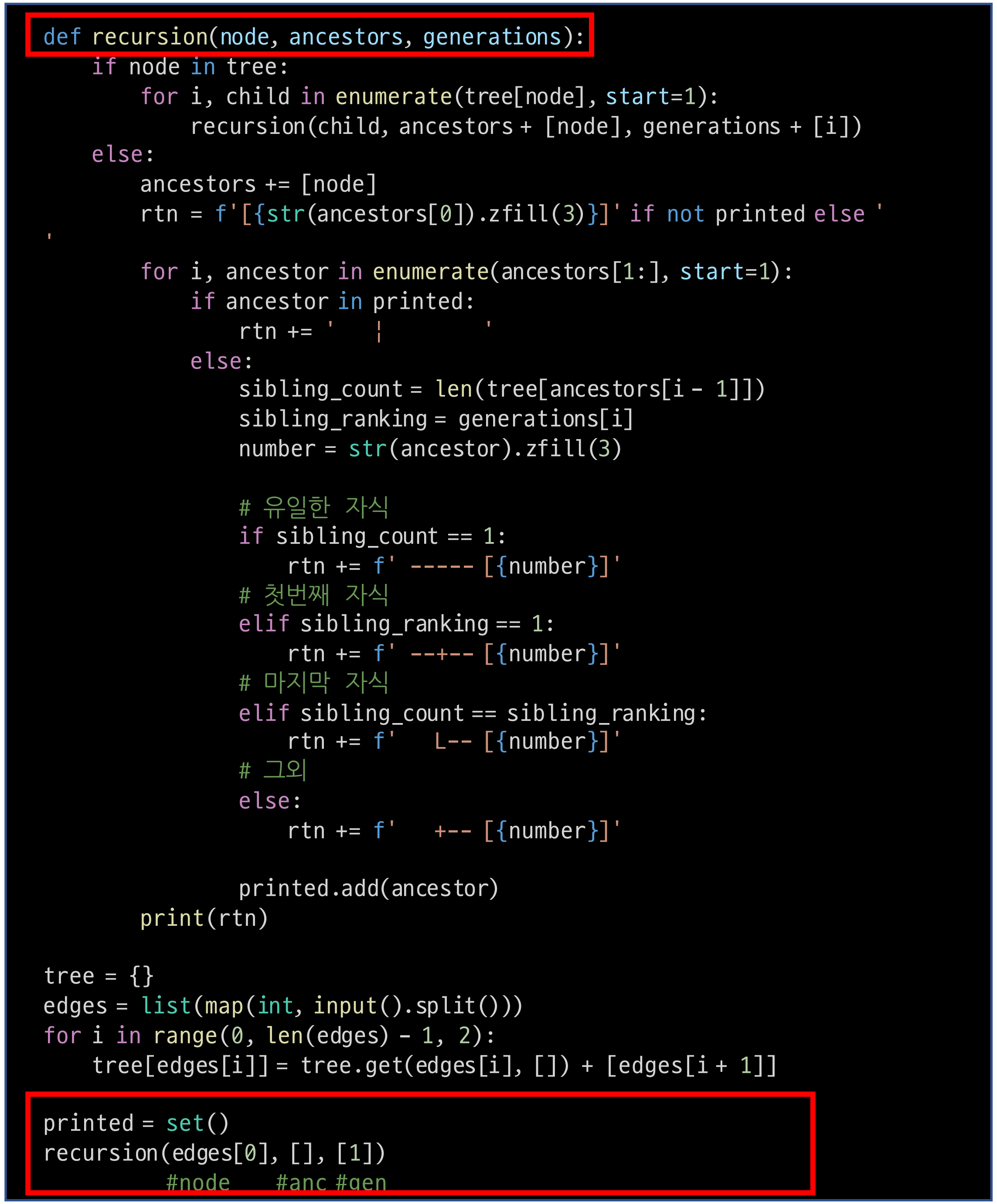
중요한 개념 세 가지

1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],           recursion(30, [], [1])
    54: [1, 3],         recursion(54, [30], [1, 1])
    1: [101, 102],
    3: [103]
}
```



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

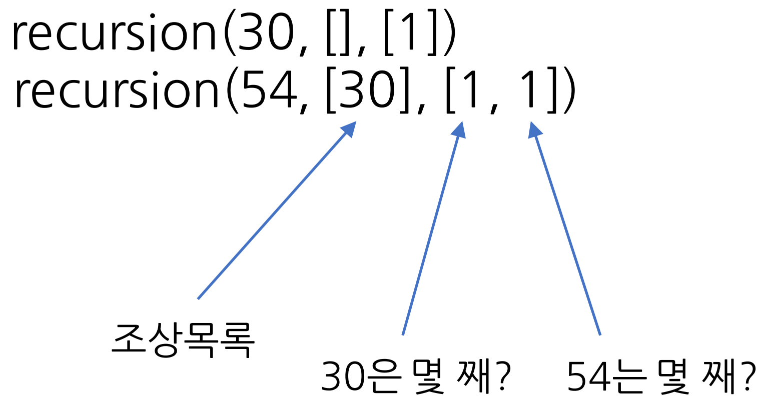
중요한 개념 세 가지

1. printed
2. ancestors
3. generations

결과 미리보기

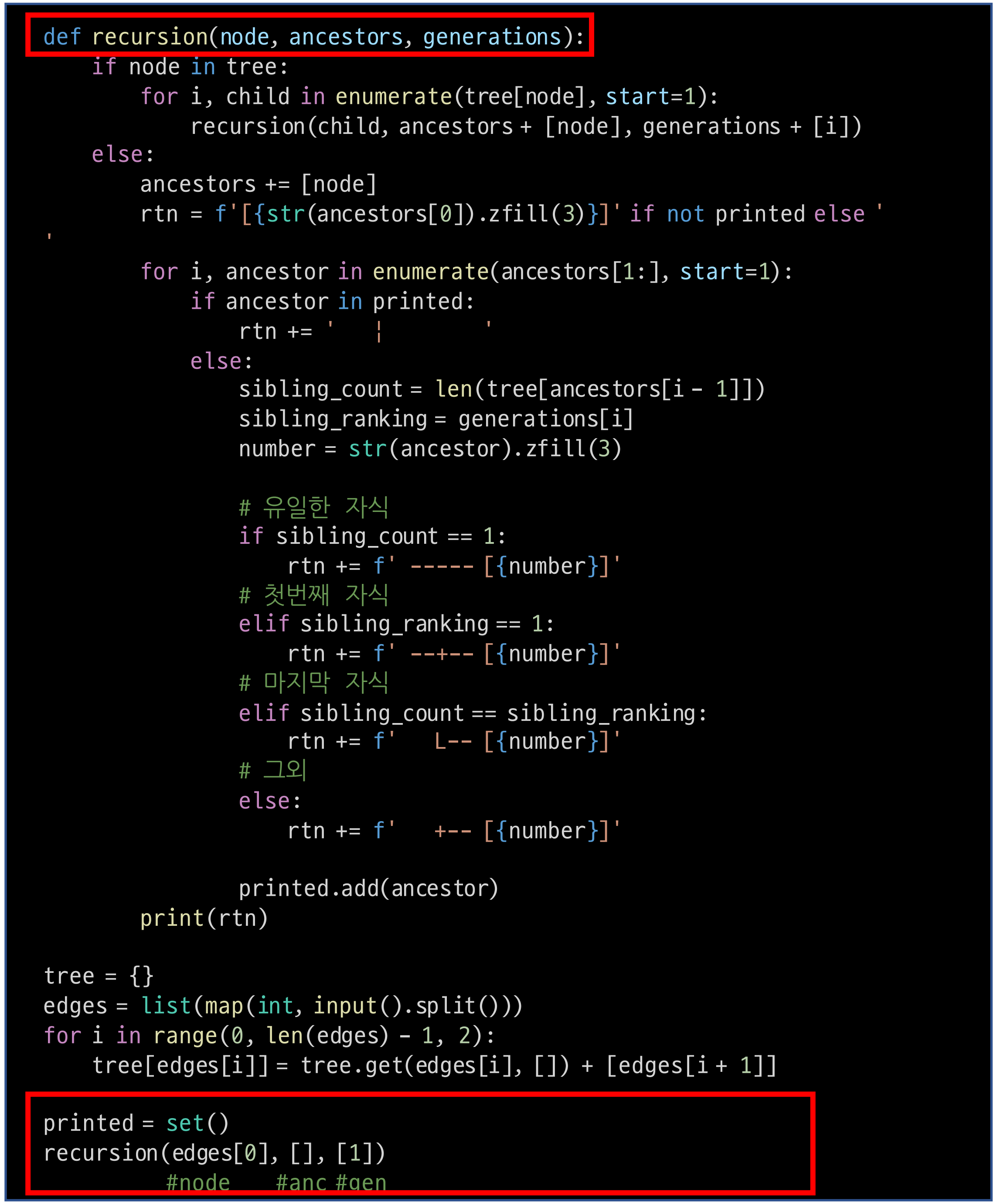
```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}
```



recursion(30, [], [1])

recursion(54, [30], [1, 1])



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '    ¦       '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

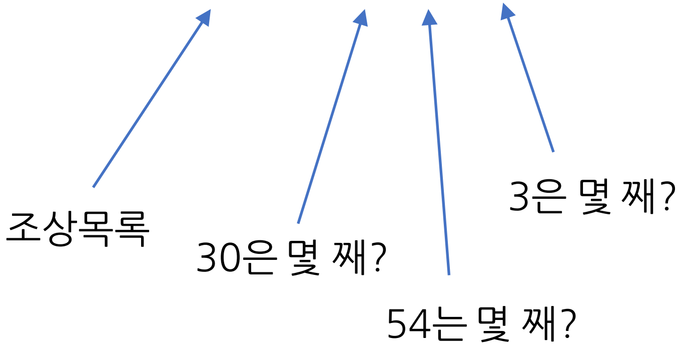
중요한 개념 세 가지

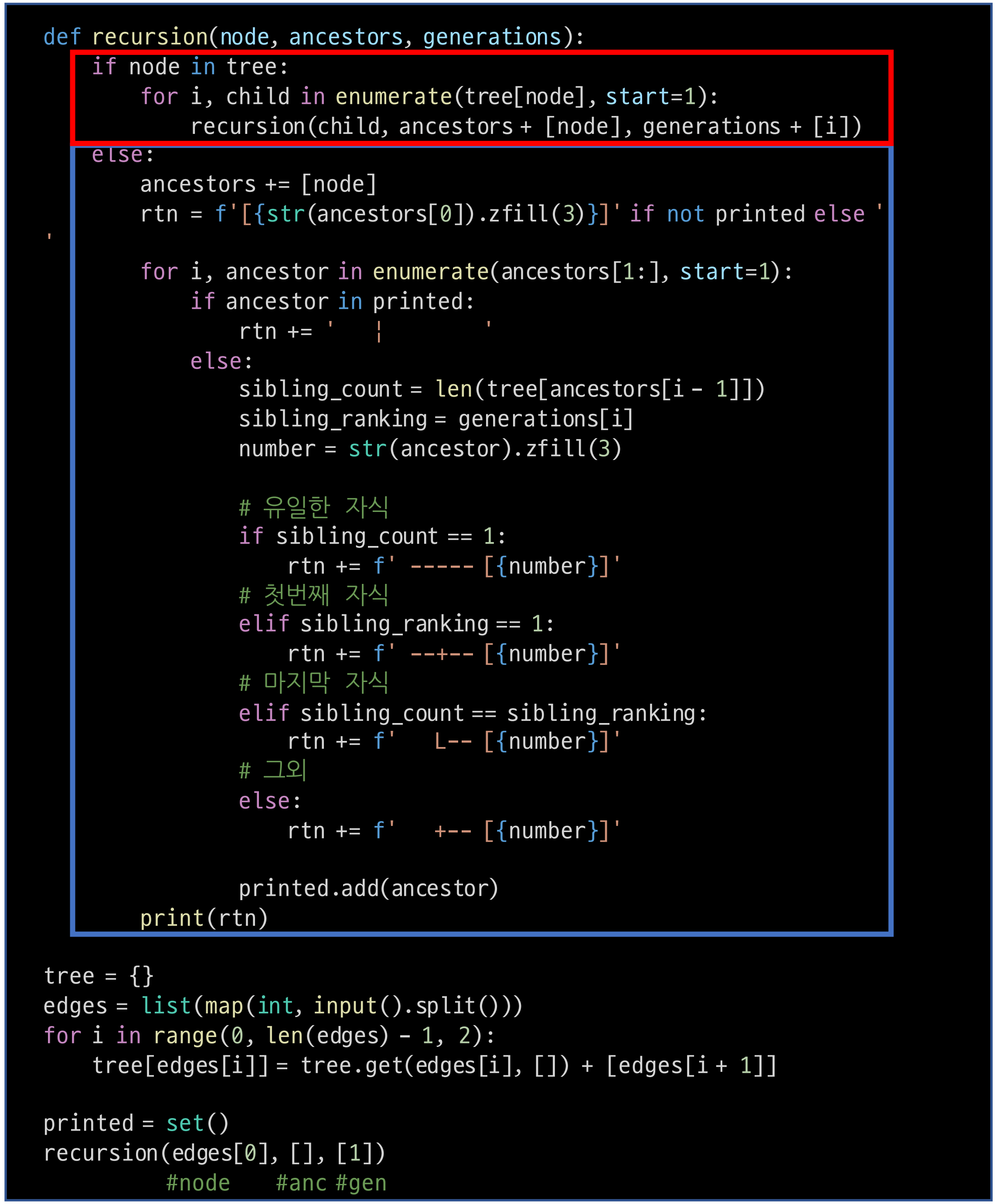
1. printed
2. ancestors
3. generations

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦           L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],           recursion(30, [], [1])
    54: [1, 3],         recursion(54, [30], [1, 1])
    1: [101, 102],      recursion(1, [30, 54], [1, 1, 1])
    3: [103]            recursion(3, [30, 54], [1, 1, 2])
}
```





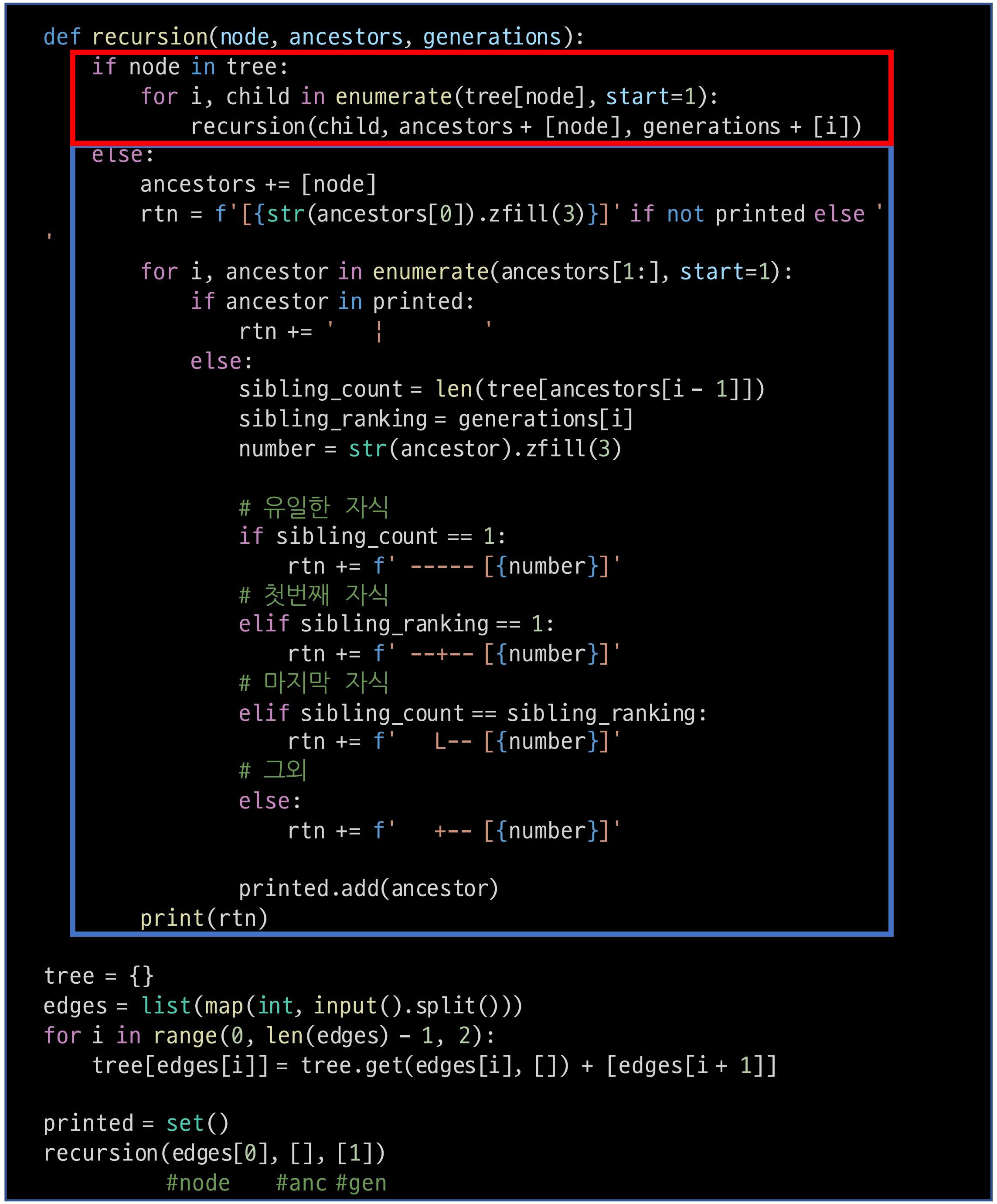
```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦          '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}
```

recursion(30, [], [1])

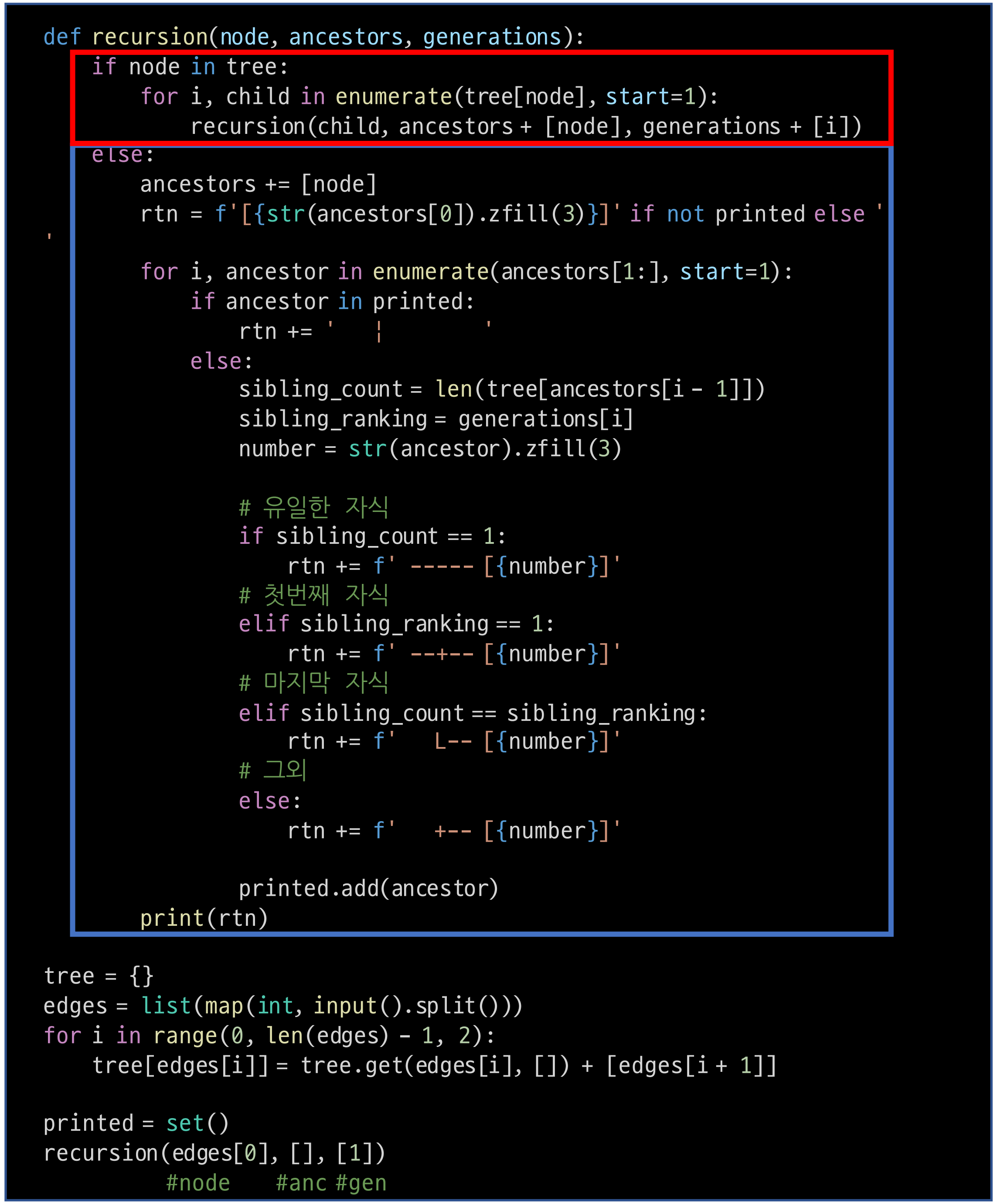


```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦         '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
           #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦                 L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],           recursion(30, [], [1])
    54: [1, 3],         recursion(54, [30], [1, 1])
    1: [101, 102],
    3: [103]
}
```

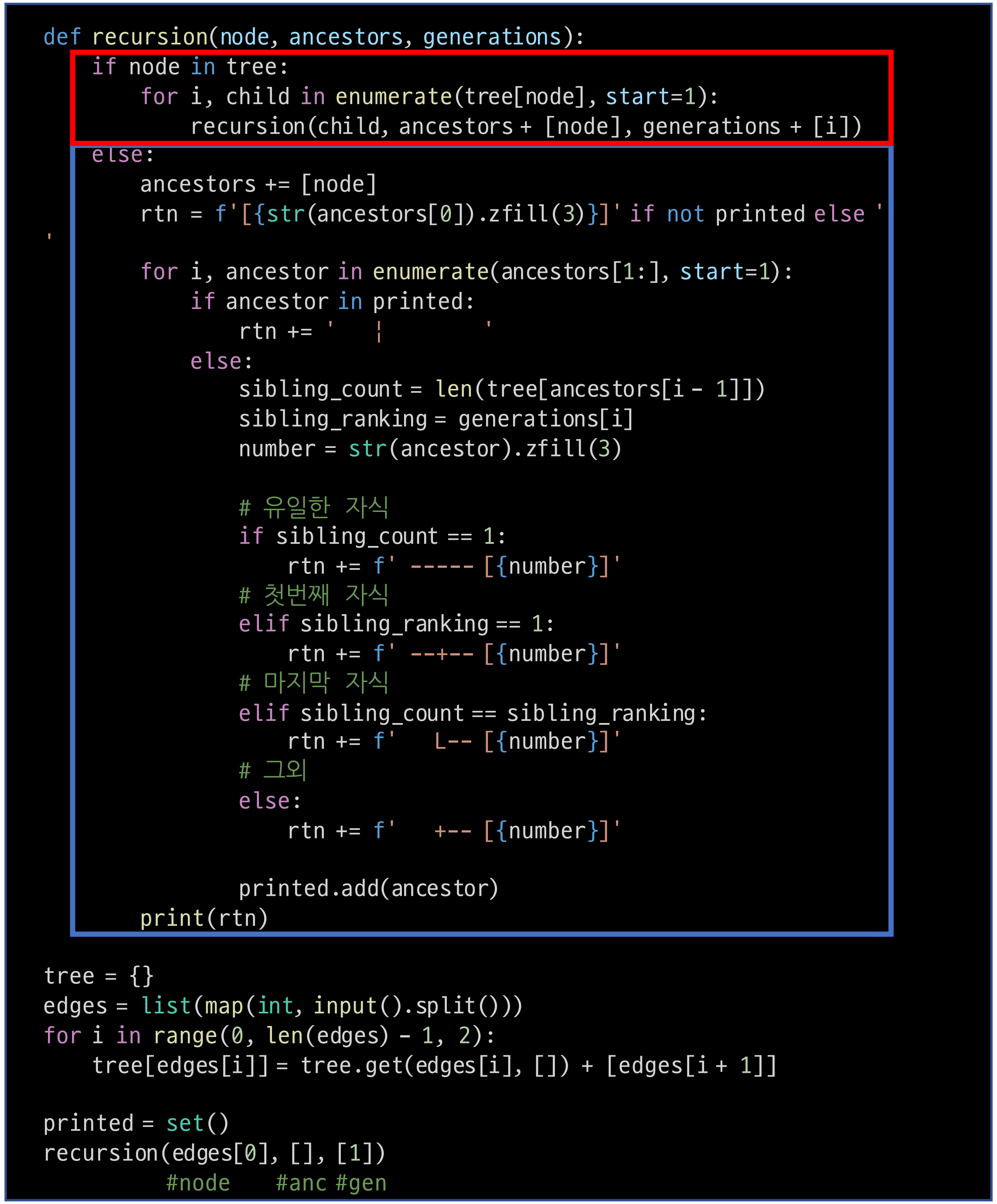


```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '    ¦         '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],           recursion(30, [], [1])
    54: [1, 3],         recursion(54, [30], [1, 1])
    1: [101, 102],      recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

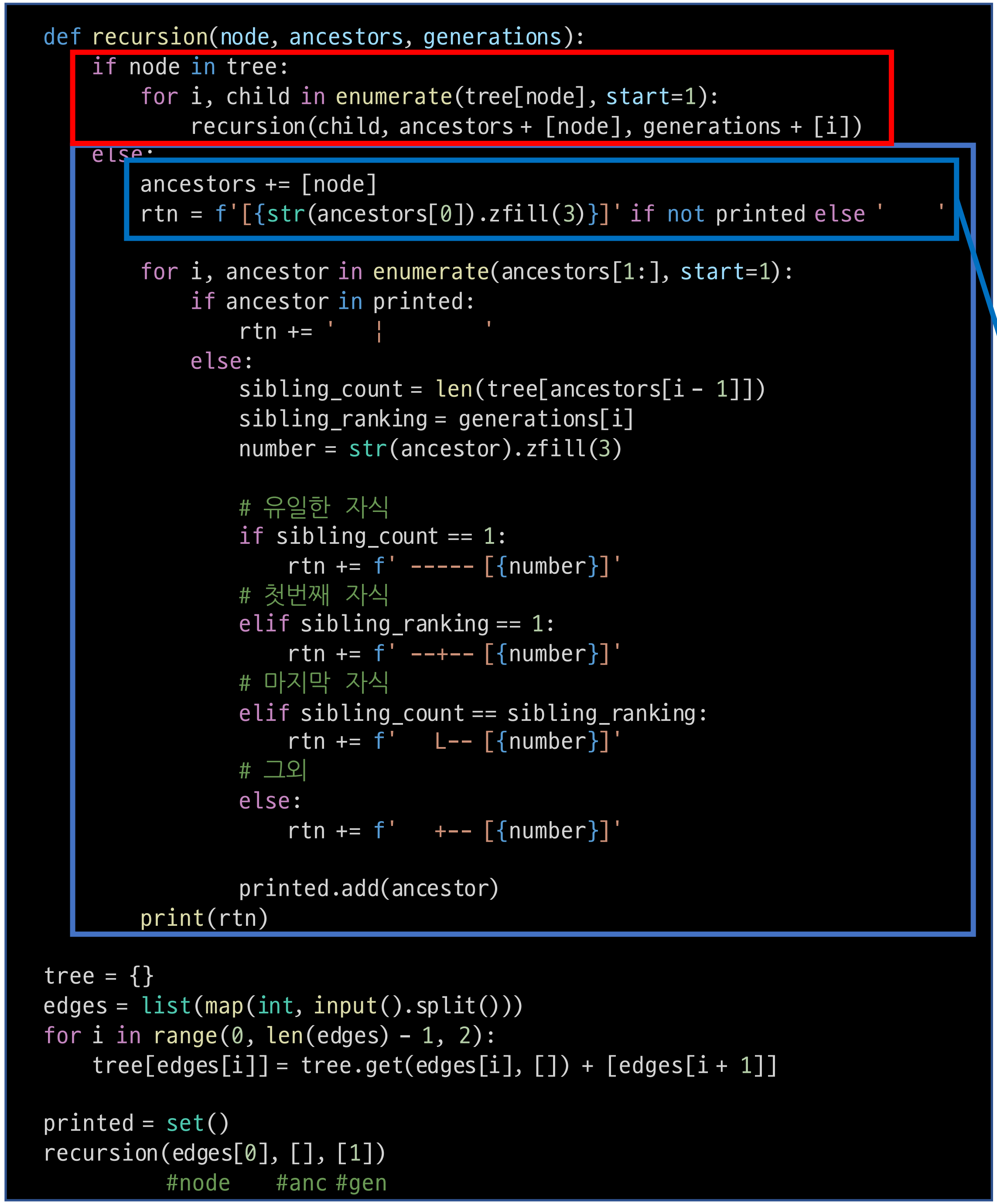


```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '
'

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '    ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node     #anc #gen
```

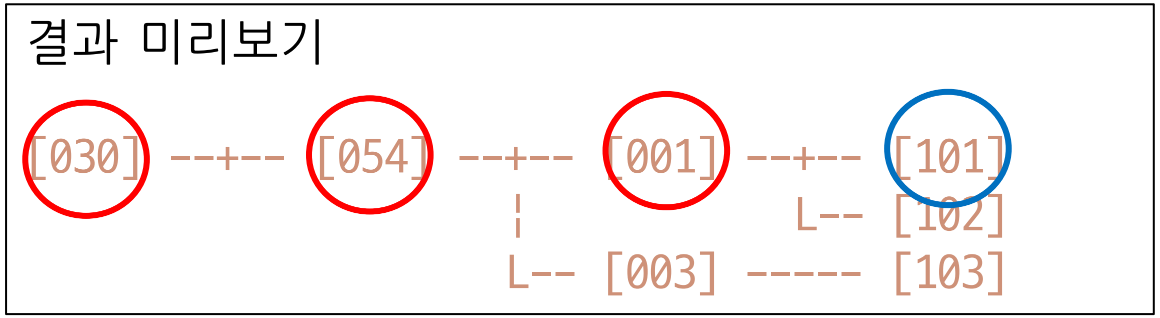
결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦           L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],            recursion(30, [], [1])
    54: [1, 3],          recursion(54, [30], [1, 1])
    1: [101, 102],       recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
    ?    recursion(101, [30, 54, 1], [1, 1, 1, 1])
```



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```



결과 미리보기

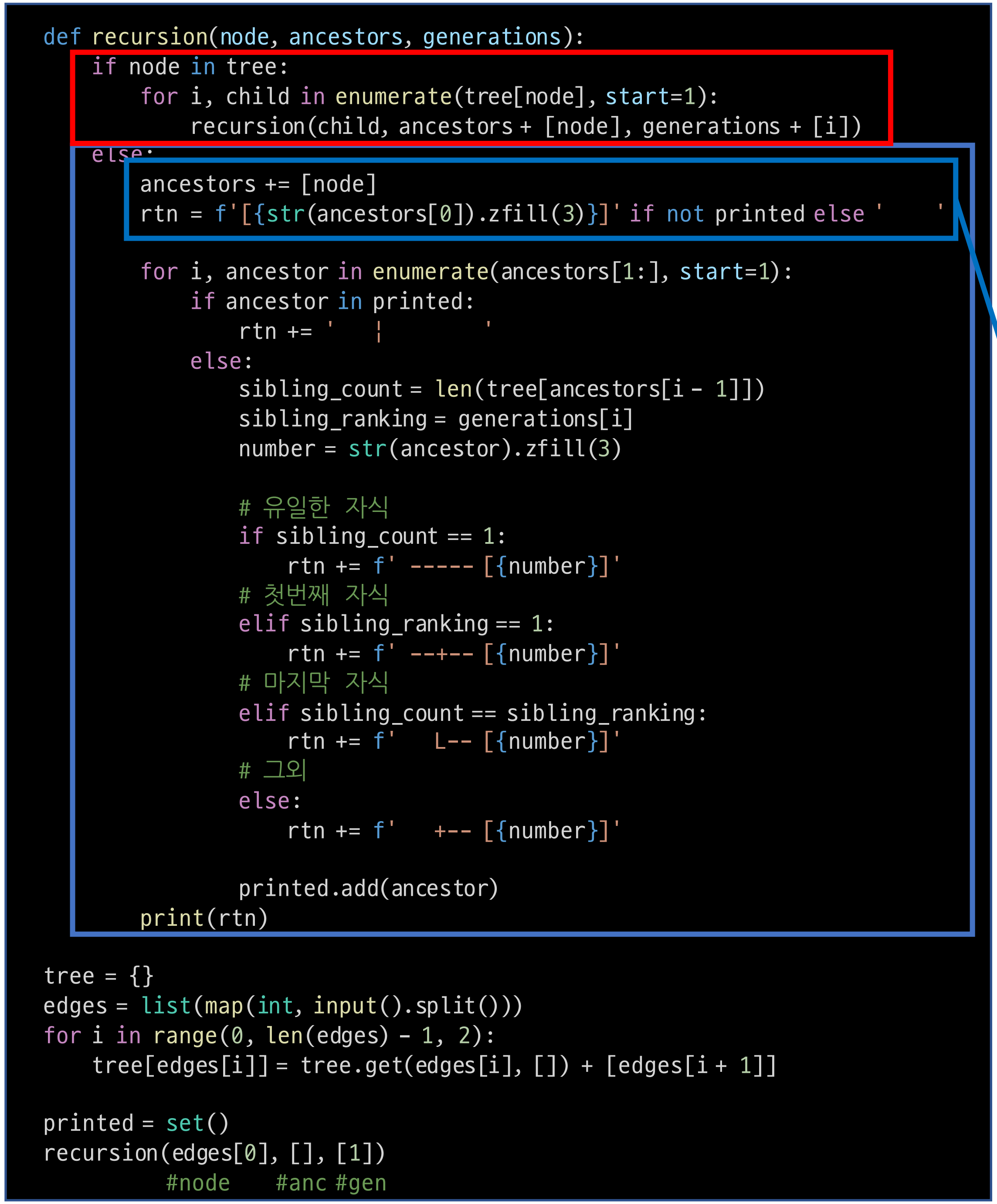
```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

tree = {
  30: [54], recursion(30, [], [1])
  54: [1, 3], recursion(54, [30], [1, 1])
  1: [101, 102], recursion(1, [30, 54], [1, 1, 1])
  3: [103]
}
  recursion(101, [30, 54, 1], [1, 1, 1, 1])

ancestors = [30, 54, 1, 101]

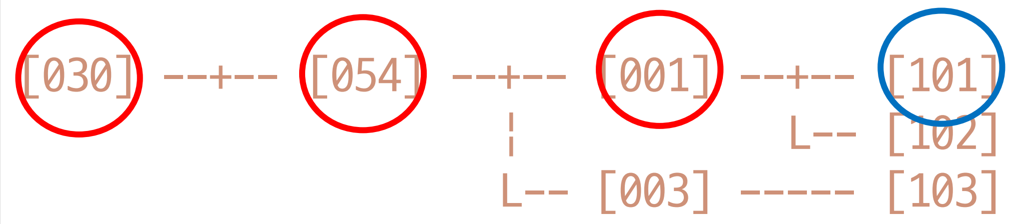
중요한 개념 세 가지

1. printed
2. ancestors
3. generations



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '    ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기



```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

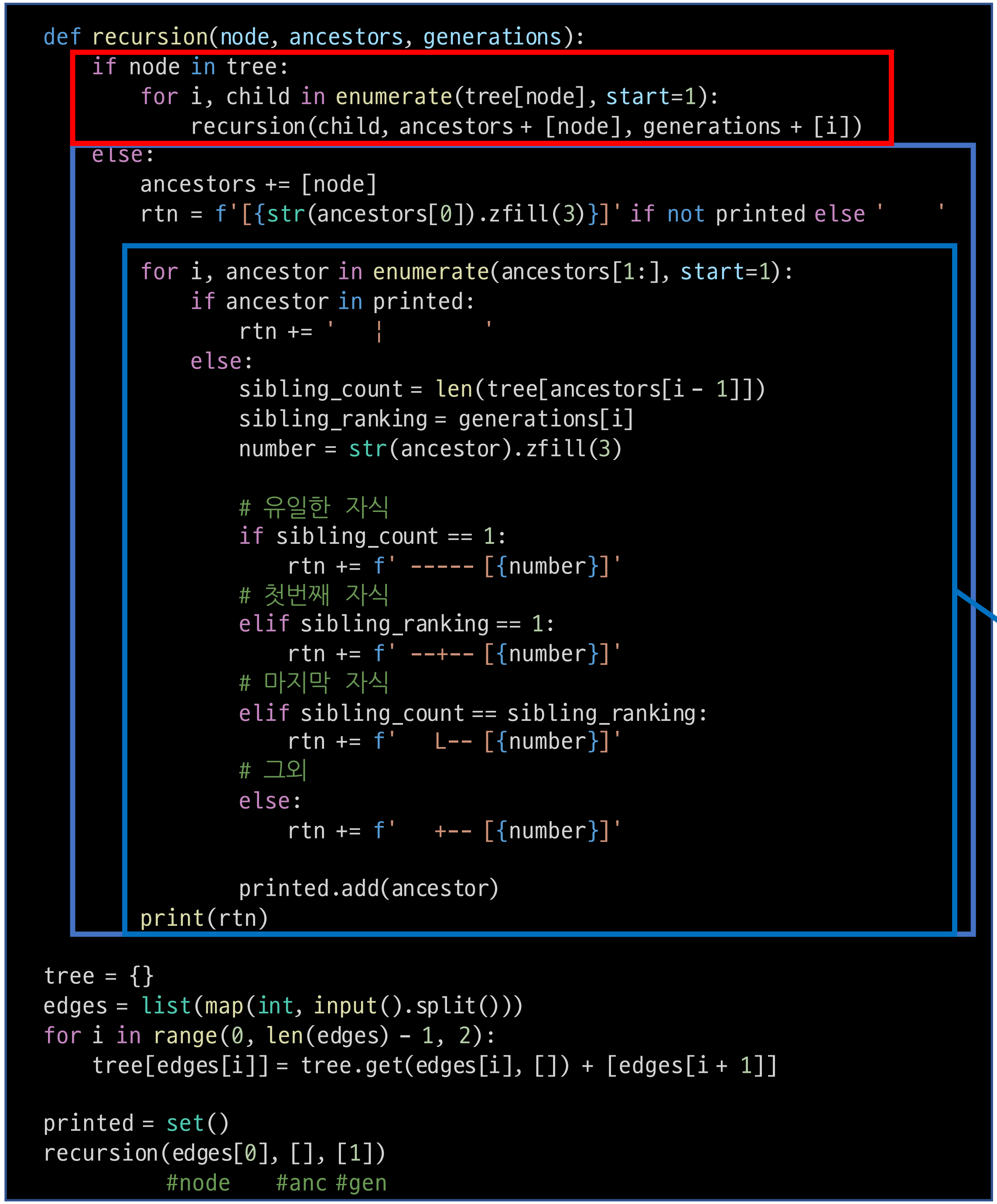
```
tree = {
    30: [54],            recursion(30, [], [1])
    54: [1, 3],          recursion(54, [30], [1, 1])
    1: [101, 102],       recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
    recursion(101, [30, 54, 1], [1, 1, 1, 1])
```

ancestors = [30, 54, 1, 101]

현재 print 된게 아무것도 없으니까

`rtn = f'[{str(ancestors[0]).zfill(3)}]'`

루트노드인 30을
“030” 의 형태로 만들자



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦       '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],            recursion(30, [], [1])
    54: [1, 3],          recursion(54, [30], [1, 1])
    1: [101, 102],       recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

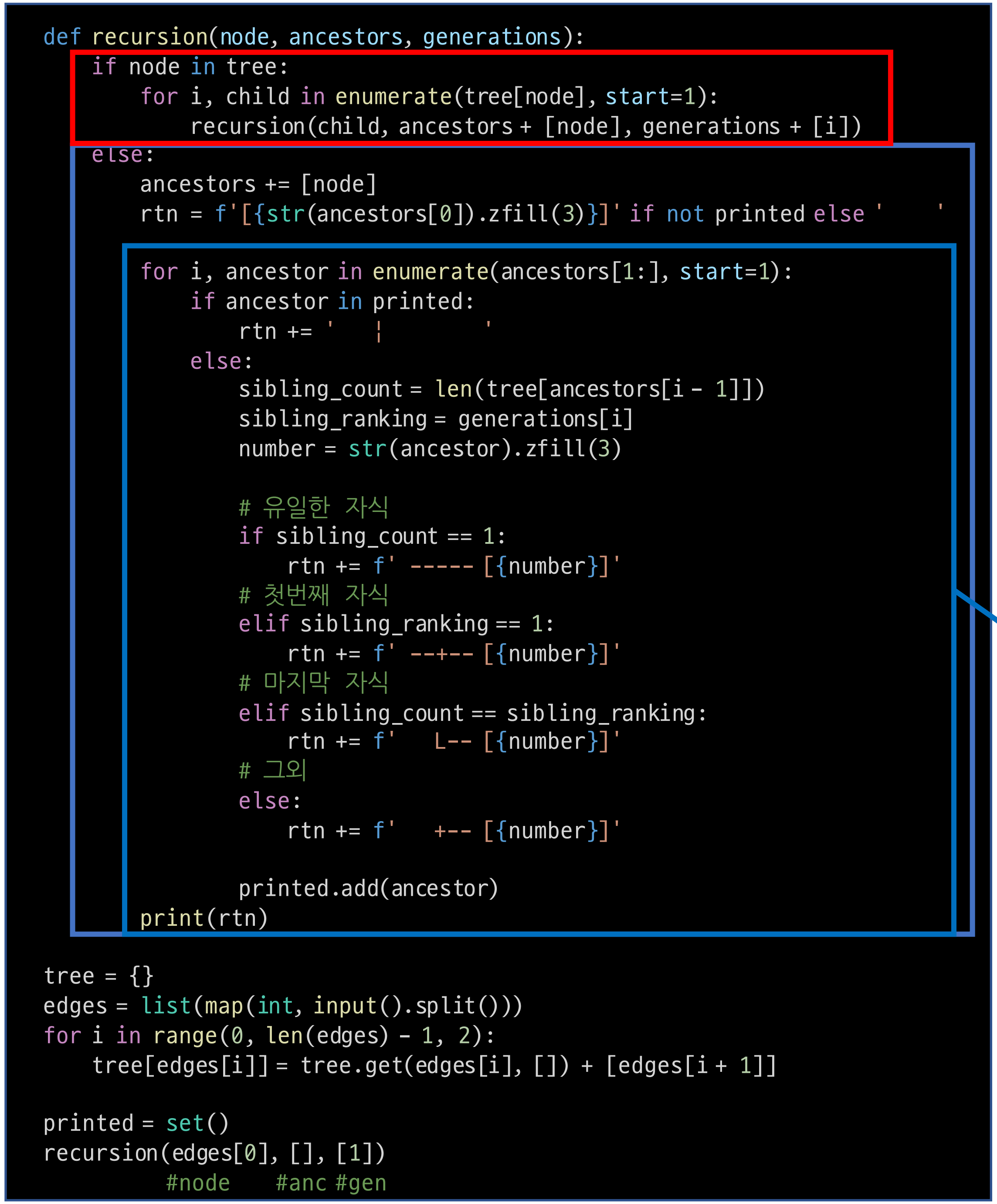
ancestors = [30, 54, 1, 101]

ancestor = 30

ancestor = 54

ancestor = 1

ancestor = 101



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],          recursion(30, [], [1])
    54: [1, 3],        recursion(54, [30], [1, 1])
    1: [101, 102],     recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

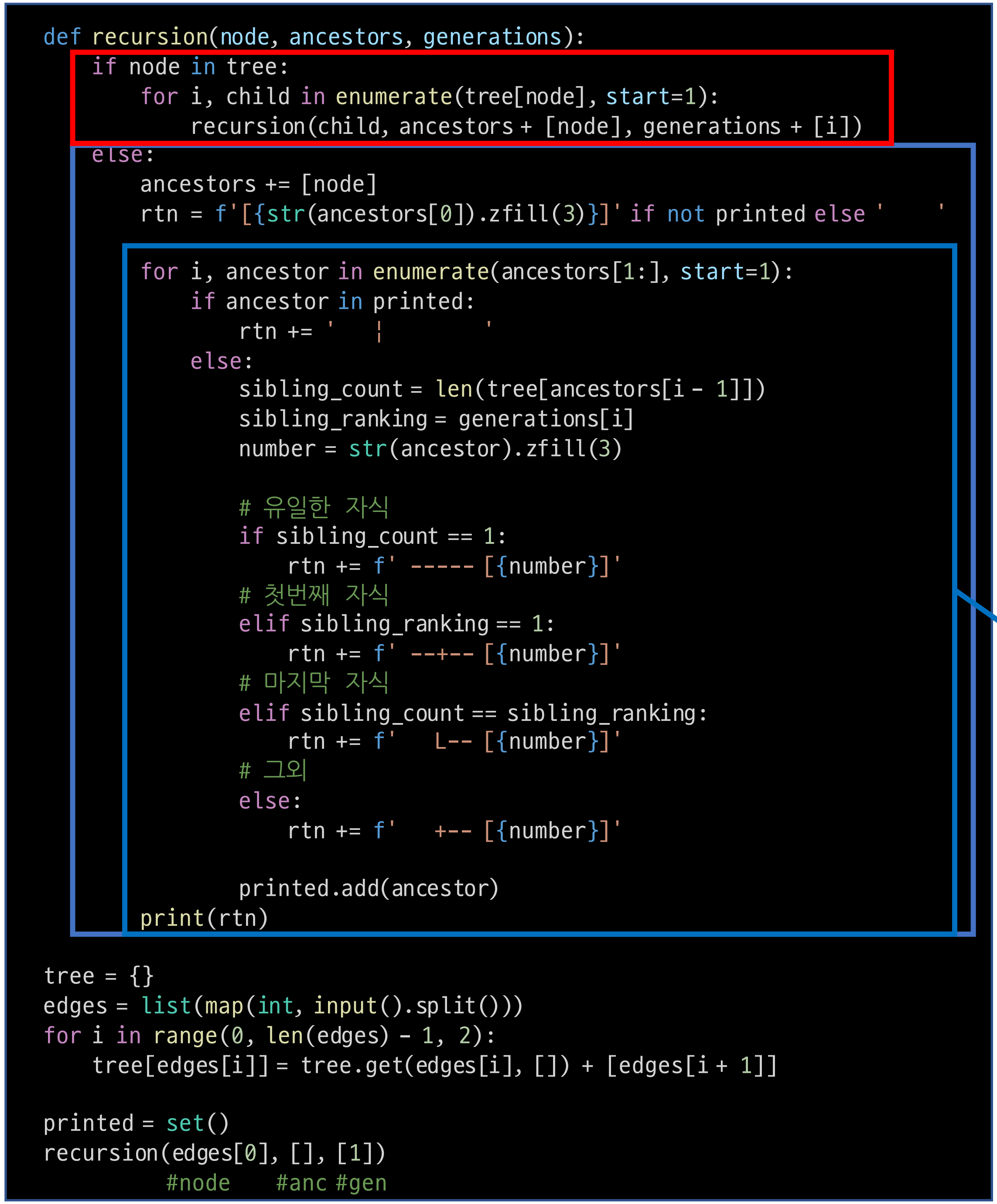
ancestors = [30, 54, 1, 101]

~~ancestor = 30~~

ancestor = 54

ancestor = 1

ancestor = 101



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦       '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

tree = {
    30: [54],
    54: [1, 3],
    1: [101, 102],
    3: [103]
}

recursion(30, [], [1])
recursion(54, [30], [1, 1])
recursion(1, [30, 54], [1, 1, 1])

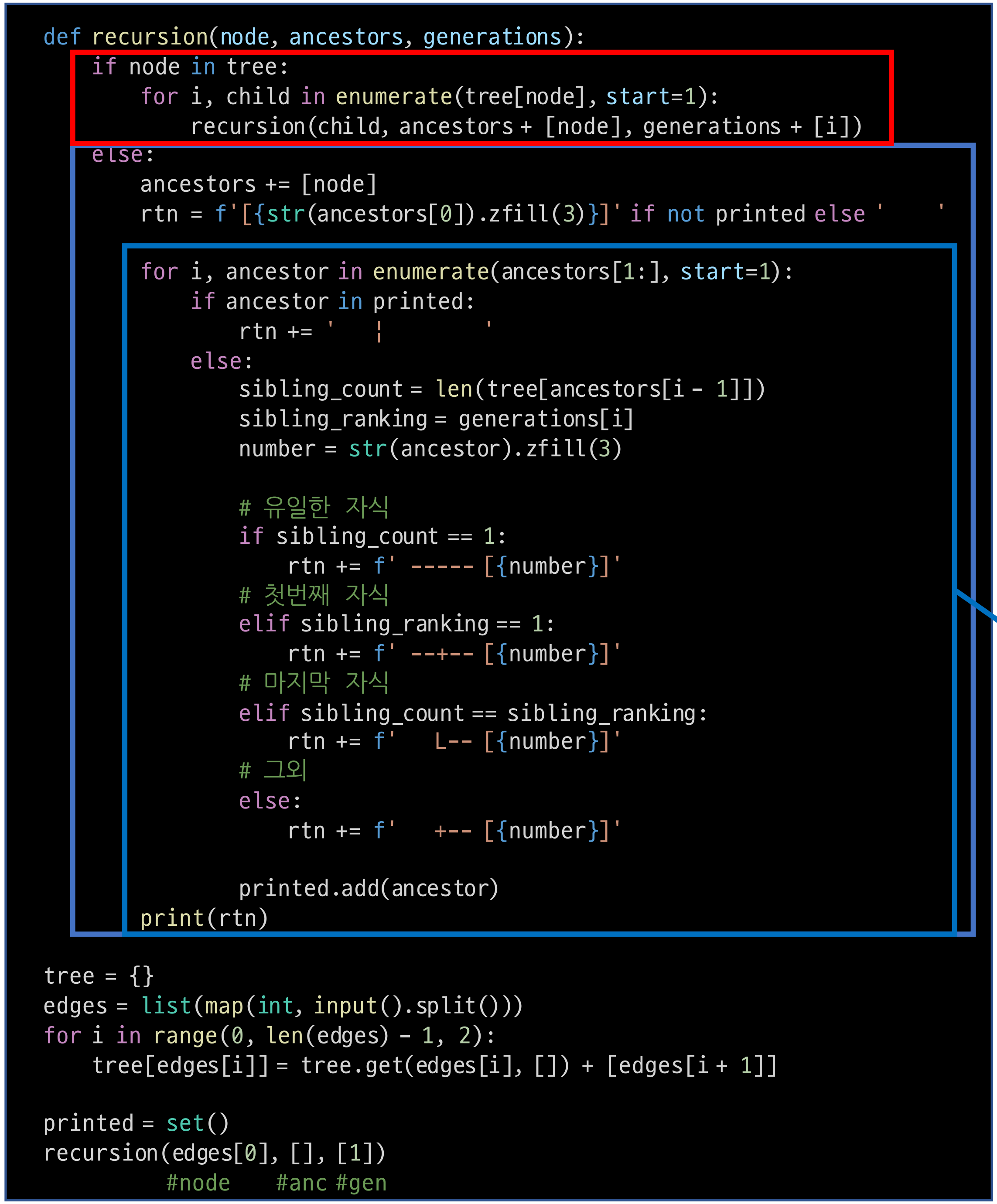
ancestors = [30, 54, 1, 101]

~~ancestor = 30~~

ancestor = 54   in printed?? -> 이미 출력 했니?

ancestor = 1

ancestor = 101



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦       '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

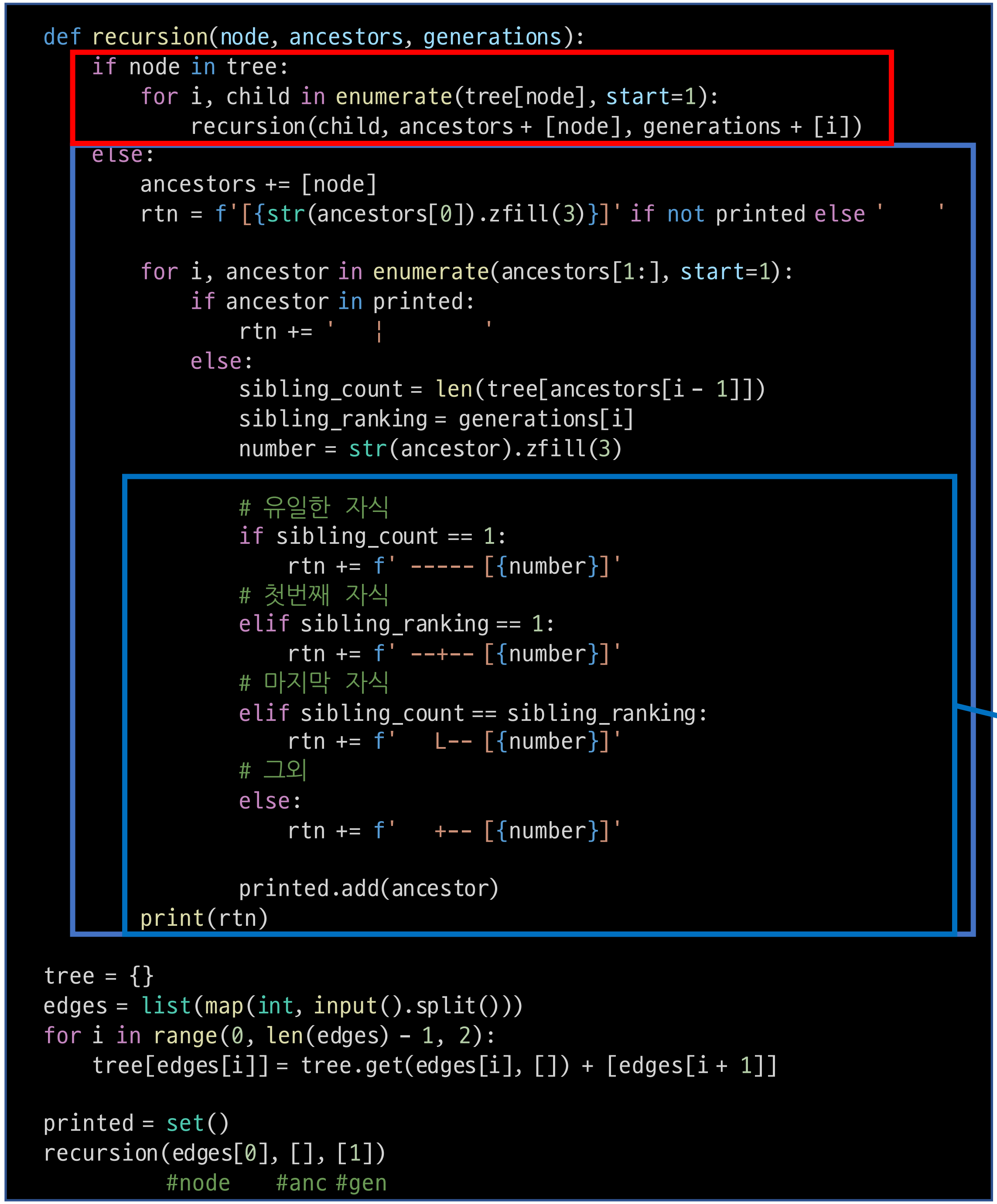
```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],          recursion(30, [], [1])
    54: [1, 3],        recursion(54, [30], [1, 1])
    1: [101, 102],     recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

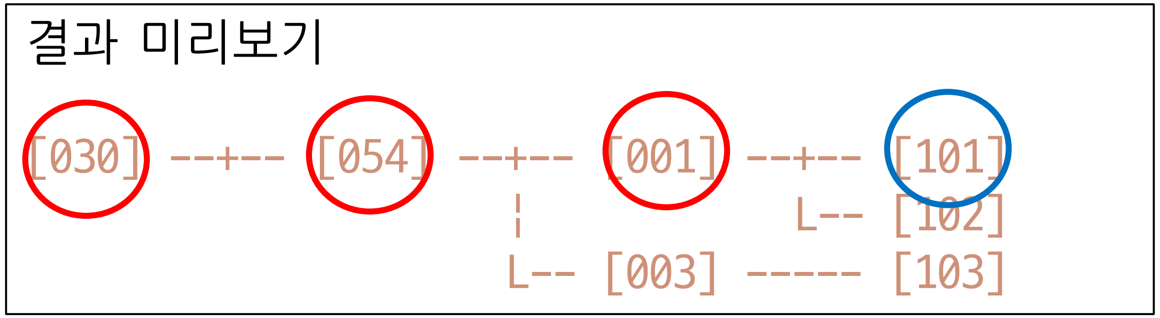
ancestors[1:] = [54, 1, 101]
generations = [1, 1, 1, 1]

i = 1
ancestor = 54

len(tree[ancestors[i - 1]]) = 부모의 자식이 몇 명?
generations[i] = 나는 몇 째 자식?
str(ancestor).zfill(3) = ‘054’



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```



결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦              L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],            recursion(30, [], [1])
    54: [1, 3],          recursion(54, [30], [1, 1])
    1: [101, 102],       recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

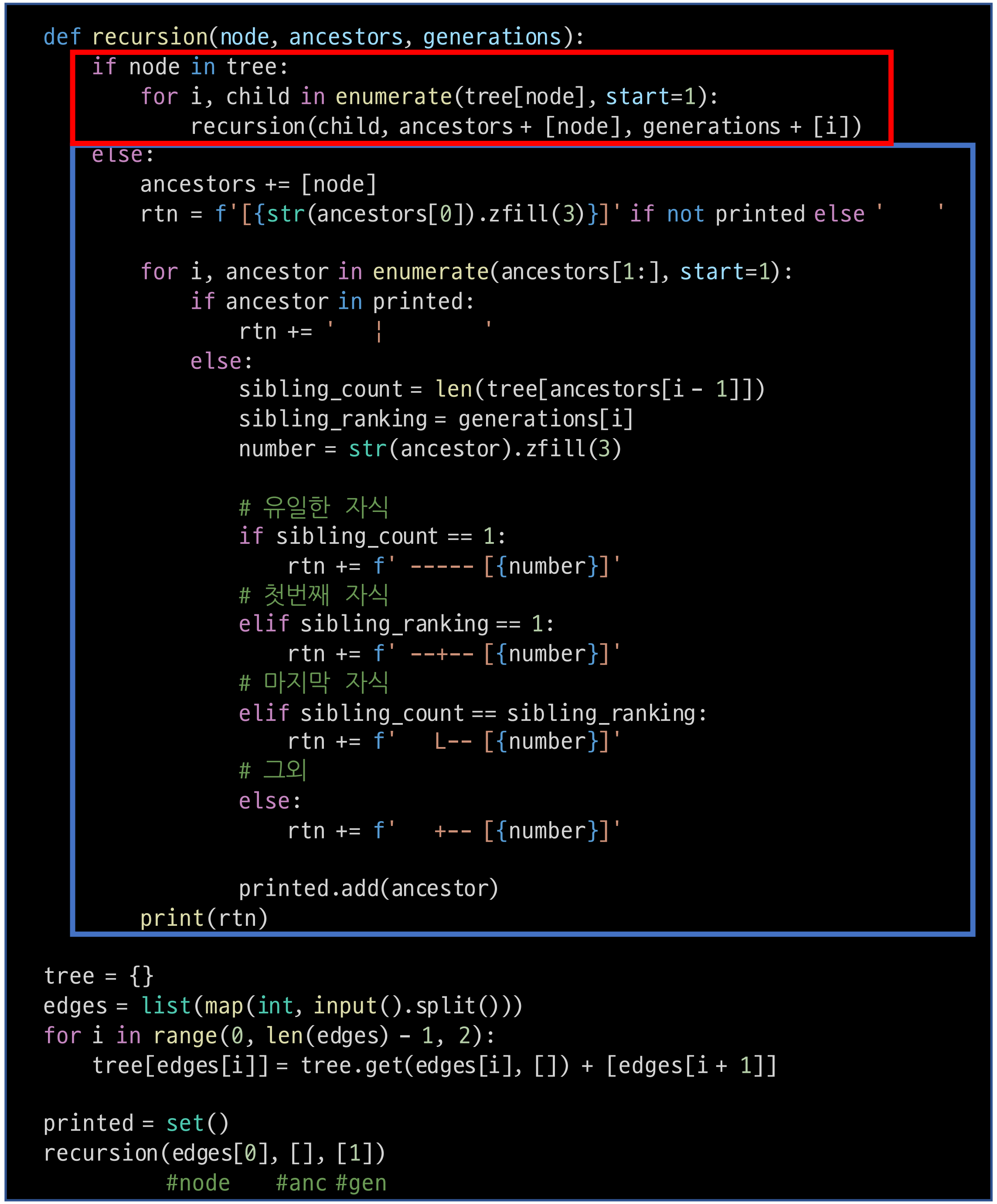
i = 1

ancestor = 54

sibling_count = 부모의 자식이 몇 명?

sibling_ranking = 나는 몇 째 자식?

number = ‘054’



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '      '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'    L-- [{number}]'
                # 그외
                else:
                    rtn += f'    +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

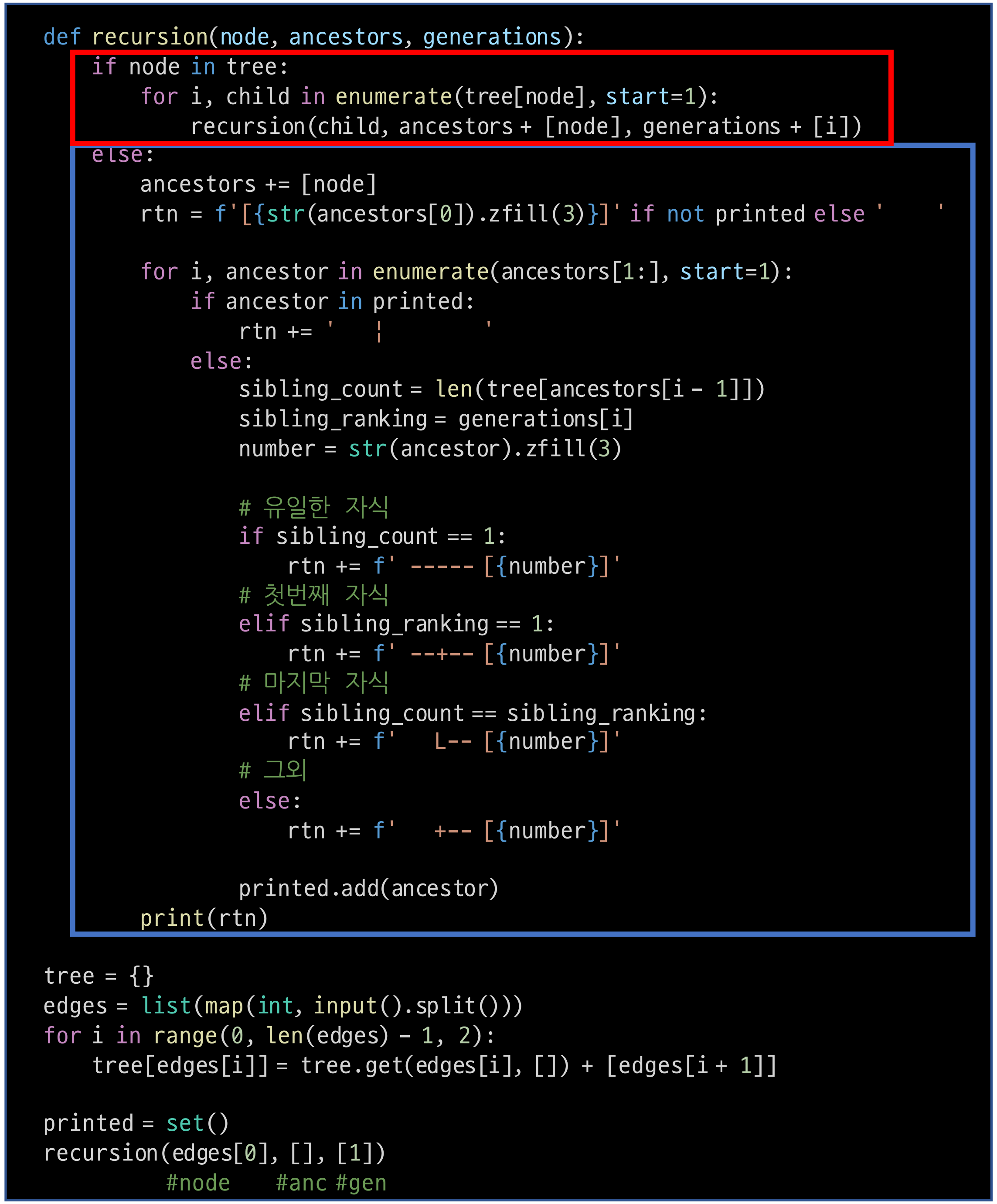
결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦               L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],             recursion(30, [], [1])
    54: [1, 3],           recursion(54, [30], [1, 1])
    1: [~~101~~, 102],    recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

현재까지 출력결과

```
[030] --+-- [054] --+-- [001] --+-- [101]
```



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦         '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

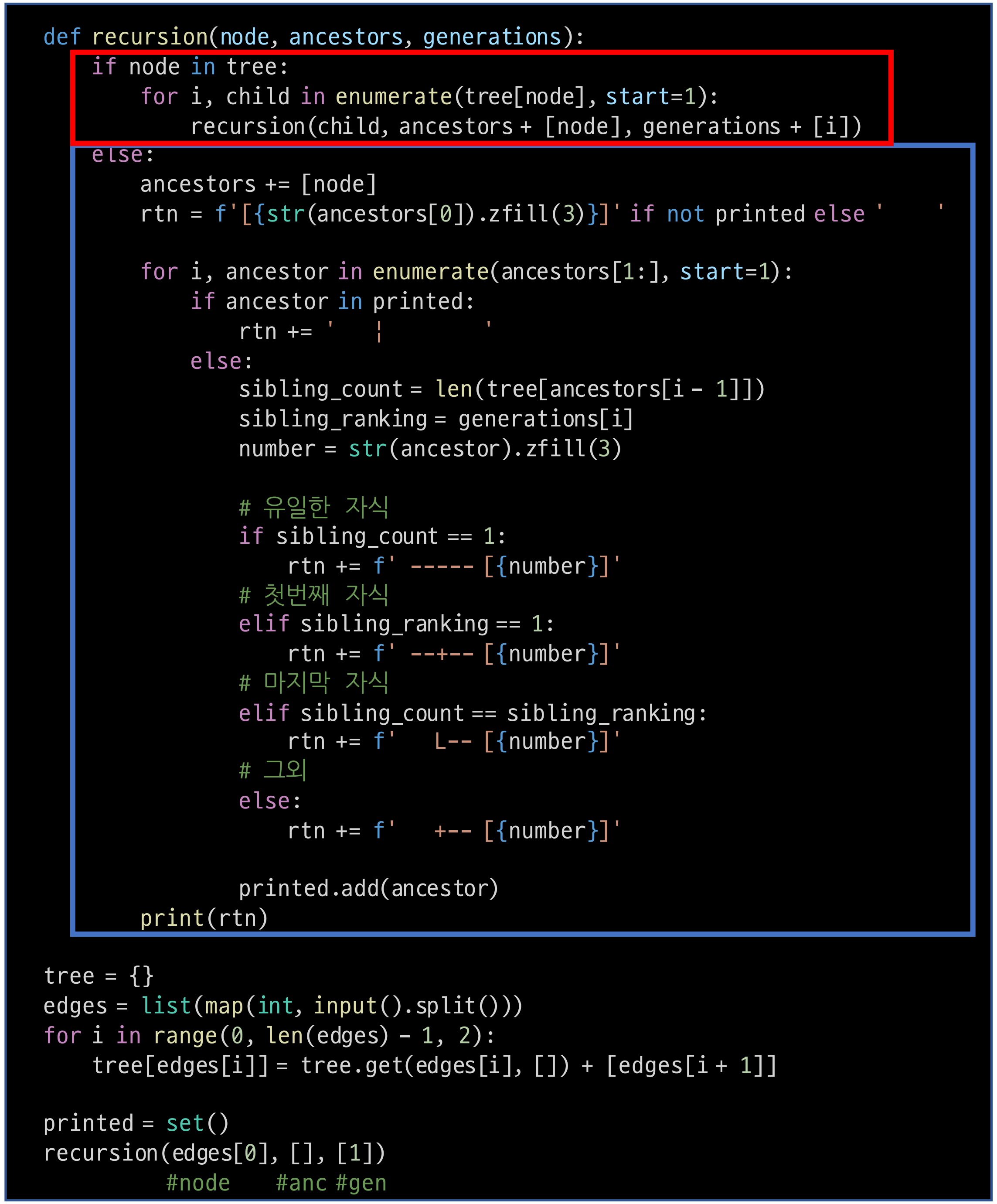
```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦               L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],
    54: [1, 3],
    1: [~~101~~, 102],
    3: [103]
}
```

```
recursion(30, [], [1])
recursion(54, [30], [1, 1])
recursion(1, [30, 54], [1, 1, 1])
```

현재까지 출력결과

```
[030] --+-- [054] --+-- [001] --+-- [101]
```

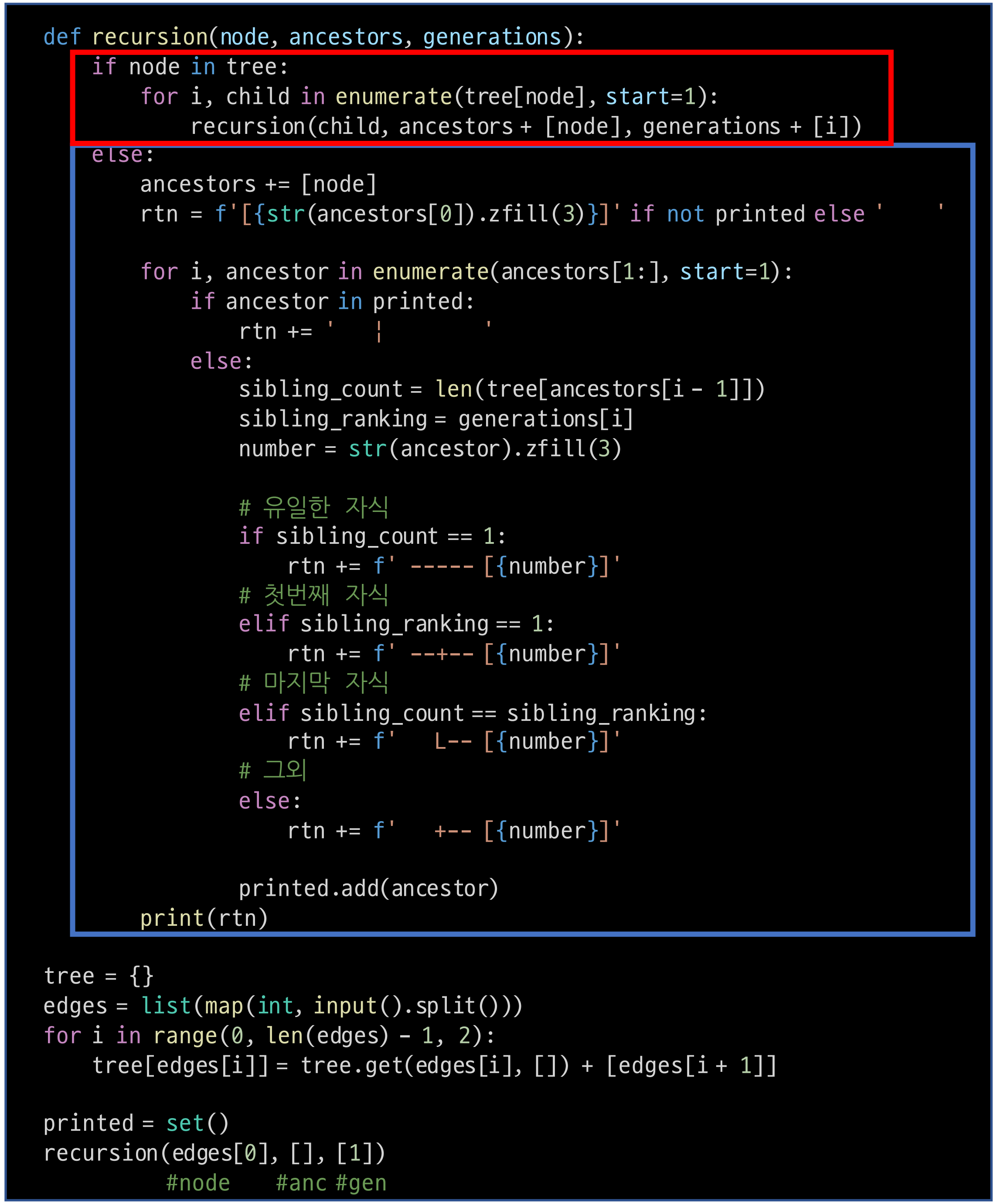


```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦             L-- [102]
                    L-- [003] ----- [103]
```

```
tree = {
    30: [54],             recursion(30, [], [1])
    54: [1, 3],           recursion(54, [30], [1, 1])
    1: [~~101~~, ~~102~~],        recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```



```
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```

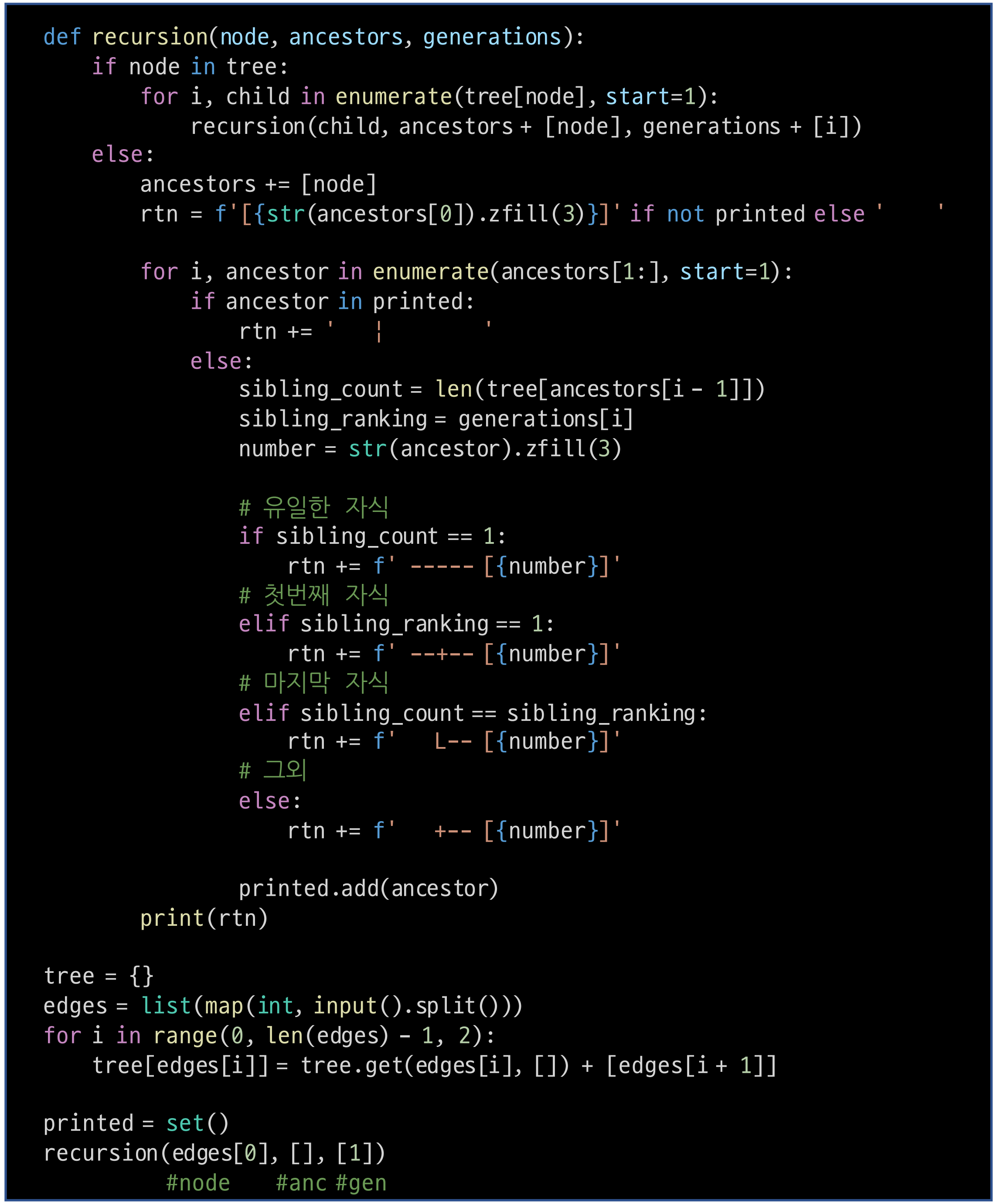
결과 미리보기

```
[030] --+-- [054] --+-- [001] --+-- [101]
                    ¦           L-- [102]
                    L-- [003] ----- [103]
```

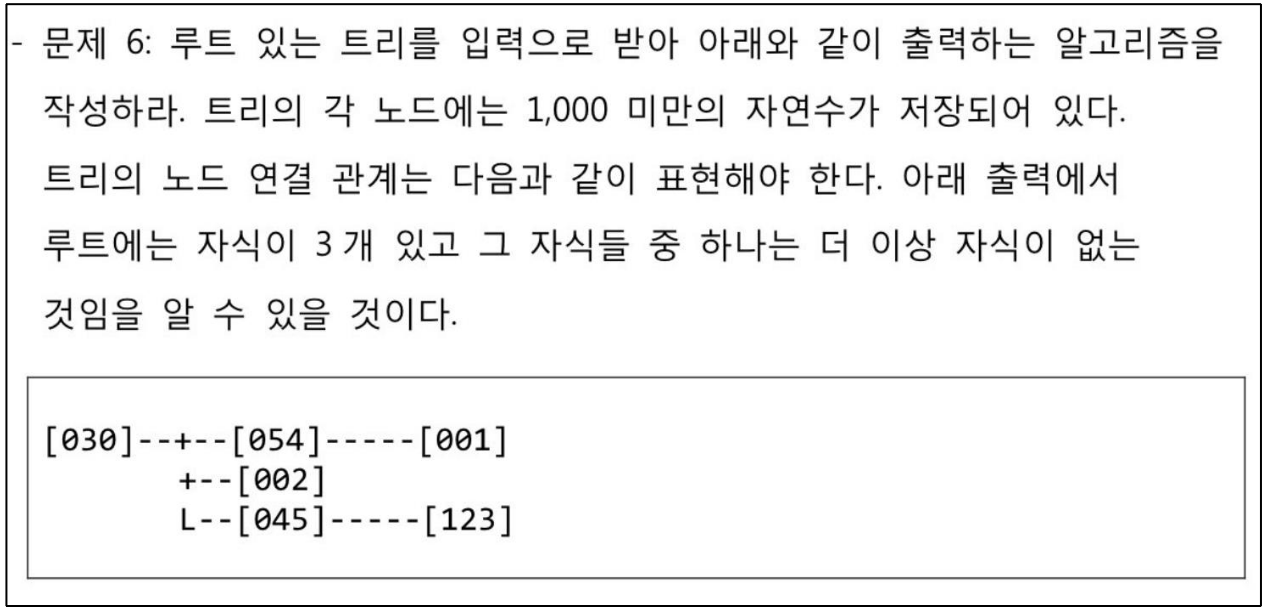
```
tree = {
    30: [54],           recursion(30, [], [1])
    54: [1, 3],         recursion(54, [30], [1, 1])
    1: [1✖, 1✖2],       recursion(1, [30, 54], [1, 1, 1])
    3: [103]
}
```

현재까지 출력결과

```
[030] --+-- [054] --+-- [001] --+-- [101]
                                L-- [102]
```



```python
def recursion(node, ancestors, generations):
    if node in tree:
        for i, child in enumerate(tree[node], start=1):
            recursion(child, ancestors + [node], generations + [i])
    else:
        ancestors += [node]
        rtn = f'[{str(ancestors[0]).zfill(3)}]' if not printed else '     '

        for i, ancestor in enumerate(ancestors[1:], start=1):
            if ancestor in printed:
                rtn += '   ¦        '
            else:
                sibling_count = len(tree[ancestors[i - 1]])
                sibling_ranking = generations[i]
                number = str(ancestor).zfill(3)

                # 유일한 자식
                if sibling_count == 1:
                    rtn += f' ----- [{number}]'
                # 첫번째 자식
                elif sibling_ranking == 1:
                    rtn += f' --+-- [{number}]'
                # 마지막 자식
                elif sibling_count == sibling_ranking:
                    rtn += f'   L-- [{number}]'
                # 그외
                else:
                    rtn += f'   +-- [{number}]'

                printed.add(ancestor)
        print(rtn)

tree = {}
edges = list(map(int, input().split()))
for i in range(0, len(edges) - 1, 2):
    tree[edges[i]] = tree.get(edges[i], []) + [edges[i + 1]]

printed = set()
recursion(edges[0], [], [1])
          #node    #anc #gen
```



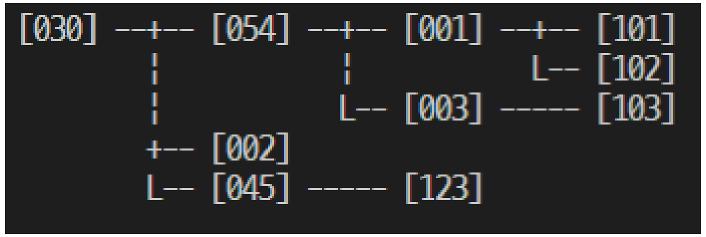
- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```

입력예시

30 54 30 2 30 45 54 1 54 3 45 123 1 101 1 102 3 103

결과예시



```
[030] --+-- [054] --+-- [001] --+-- [101]
        ¦           ¦               L-- [102]
        ¦           L-- [003] ----- [103]
        +-- [002]
        L-- [045] ----- [123]
```