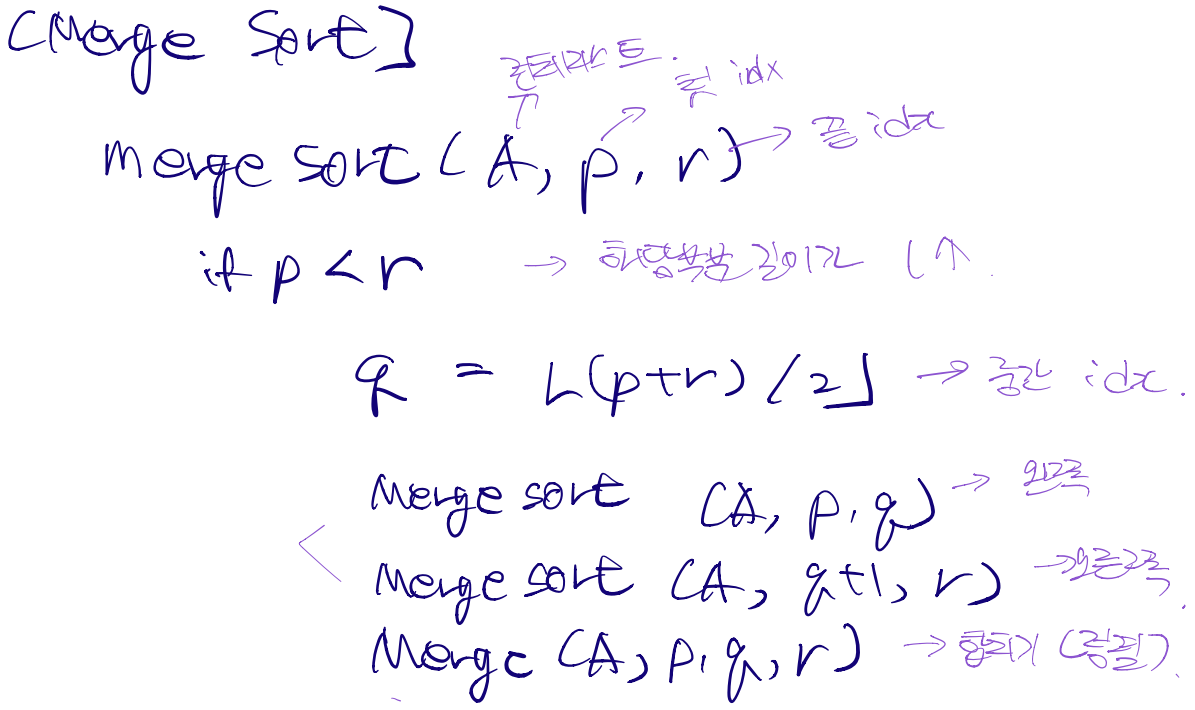
- 문제 2: Merge Sort, 크기 n인 배열을 입력으로 받아,

  배열을 절반으로 두개로 나눈 후,

  각 작은 배열을 재귀적으로 정렬하고,

  그 결과를 Merge한다. Mergesort의 수도 코드를 간략하게 작성해보고 시간 복잡도를 증명하시오.



```
Merge(A, p, q, r)
    n1 = q - p + 1
    n2 = r - q
    let L[1, ... n1+1] and R[1...n2+1] be new arrays.

    for i = 1 to n1
        L[i] = A[p+i-1]
    for j = 1 to n2
        R[j] = A[q+j]
    L[n1+1] = ∞
    R[n2+1] = ∞
    i = 1
    j = 1

    for k = p to r
        if L[i] ≤ R[j]
            A[k] = L[i]
            i = i + 1
        else
            A[k] = R[j]
            j = j + 1
```

- $n_1 = q - p + 1$ → 왼쪽 길이
- $n_2 = r - q$ → 오른쪽 길이
- let L[...], R[...] → 왼쪽, 오른쪽 배열 생성.
- for i / for j → L, R에 값 채우기.
- $L[n_1+1] = \infty$, $R[n_2+1] = \infty$: 한쪽이 다 먼저 A에 들어간 경우 나머지도 다 넣기 위해 비교대상이 필요!
- for k = p to r → k는 A에 대한 idx.
- for문 하나당 한개씩 정렬된 값이 들어감.
- i = i + 1 → L에 대한 idx 증가.
- j = j + 1 → R에 대한 idx 증가.

[시간복잡도]

$$T(n) = T(n/2) + T(n/2) + n$$

$$= 2T\left(\frac{n}{2}\right) + n$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + 3n$$

$$\vdots$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + k \times n$$

let $k = \log_2 n$ ↳ $T(n) = n\,T\left(\frac{n}{n}\right) + n\log_2 n$

$$= n\,T(1) + n\log_2 n$$

$$\Rightarrow O(n \log n)$$

- 문제 4: 위의 소팅 알고리즘에서 수행하는 Swap의 횟수는 최대 몇 번인가?

답 ① swap이 두인자값을 서로 바꾸는 의미인경우

⇒ 0

답 ② swap이 merge할때마다 발생하면

⇒ n-1

답 ③ 2n.

- 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 3개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

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

```
import sys
sys.stdin = open('input.txt')


def dfs(node, connect, w_level, i):
    global h_level
    if i != 0:
        if w_level == 1:
            print('        ' * w_level, end='')
            w_level = 0
        elif w_level > 1:
            for i in range(w_level):
                if i == 0:
                    print('        ', end='')
                    continue
                print('|       ', end='')
            w_level = 0

    zeros = '0' * (3 - len(str(node)))
    if tree[node]:
        if node == root:
            print(f'[{zeros}{node}] --', end='') # 자식이 있는 노드 출력
        else:
            print(f'{connect}-- [{zeros}{node}] --', end='')
        for i in range(len(tree[node])):
            connect = '+'
            if len(tree[node]) == 1:
                connect = '-'
            elif i == len(tree[node]) - 1:
                connect = 'L'
            dfs(tree[node][i], connect, w_level + 1, i)
    else:
        h_level += 1
        print(f'{connect}-- [{zeros}{node}]') # 리프 노드 출력



l = list(map(int, input().split()))
N, E = len(l) // 2 + 1, len(l) // 2 # N: 노드의 수, E: 간선의 수
h_level = 0

tree = {key:[] for key in set(l)}

for i in range(E):
    tree[l[2 * i]].append(l[2 * i + 1])
root = l[0]

print(tree)
dfs(root, '+', 0, 0)
```