题目描述

VLAN是一种对局域网设备进行逻辑划分的技术，为了标识不同的VLAN，引入VLAN ID(1-4094之间的整数)的概念。

定义一个VLAN ID的资源池(下称VLAN资源池)，资源池中连续的VLAN用开始VLAN-结束VLAN表示，不连续的用单个整数表示，所有的VLAN用英文逗号连接起来。

现在有一个VLAN资源池，业务需要从资源池中申请一个VLAN，需要你输出从VLAN资源池中移除申请的VLAN后的资源池。

输入描述

第一行为字符串格式的VLAN资源池，第二行为业务要申请的VLAN，VLAN的取值范围为[1,4094]之间的整数。

输出描述

从输入VLAN资源池中移除申请的VLAN后字符串格式的VLAN资源池，输出要求满足题目描述中的格式，并且按照VLAN从小到大升序输出。

如果申请的VLAN不在原VLAN资源池内，输出原VLAN资源池升序排序后的字符串即可。

备注

输入VLAN资源池中VLAN的数量取值范围为[2-4094]间的整数，资源池中VLAN不重复且合法([1,4094]之间的整数)，输入是乱序的。

用例

输入	1-5 2
输出	1,3-5
说明	原VLAN资源池中有VLAN 1、2、3、4、5，从资源池中移除2后，剩下VLAN 1、3、4、5，按照题目描述格式并升序后的结果为1,3-5。

输入	20-21,15,18,30,5-10 15
输出	
说明	21、30，从资源池中移除15后，资源池中剩下的VLAN为 5、6、7、8、9、10、18、20、21、30，按照题目描述格式并升序后的结果为5-10,18,20-21,30。

输入	5,1-3 10
输出	1-3,5
说明	原VLAN资源池中有VLAN 1、2、3，5，申请的VLAN 10不在原资源池中，将原资源池按照题目描述格式并按升序排序后输出的结果为1-3,5。

题目解析

本题主要考察逻辑分析能力，以及一些边际情况的处理。比如1-2删除1，或者1-3删除3等。

我的解题思路如下：

将输入的第一行转化为一个二维数组，比如用例2第一行输入可以转化为：

[[20,21], [15], [18], [30], [5,10]]

之后，将这个二维数组，按照元素的第0个索引的值进行升序排序，得到如下

[[5,10], [15], [18], [20,21], [30]]

然后从左到右遍历各个区间，看要删除的数remove（第二行输入的数）是否在区间[from, to]内，如果在则删除，删除时需要注意：

- 如果要删除的数，是区间左边界，则区间变为[remove+1, to]
- 如果要删除的数，是区间右边界，则区间变为[from, remove-1]
- 如果要删除的数，在区间内，则区间变为[from, remove-1]，和[remove+1, to]

需要注意的是，如果区间的左右边界相同，比如[3,3]，此时应该将区间变为[3]

Java算法源码

```java
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Scanner;
import java.util.StringJoiner;
import java.util.stream.Collectors;

public class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String[] vlanArr = sc.nextLine().split(",");
    int remove = Integer.parseInt(sc.nextLine());

    System.out.println(getResult(vlanArr, remove));
  }

  public static String getResult(String[] vlanArr, int remove) {
    LinkedList<Integer[]> vlanList =
        Arrays.stream(vlanArr)
            .map(v -> Arrays.stream(v.split("-")).map(Integer::parseInt).toArray(Integer[]::new))
            .sorted((a, b) -> a[0] - b[0])
            .collect(Collectors.toCollection(LinkedList::new));

    for (int i = 0; i < vlanList.size(); i++) {
      Integer[] vlan = vlanList.get(i);

      int from = vlan[0];

      if (vlan.length > 1) {
        int to = vlan[1];

        if (remove < from || remove > to) continue;

        vlanList.remove(i);

        if (remove == from) {
          vlanList.add(i, generateRange(remove + 1, to));
        } else if (remove == to) {
          vlanList.add(i, generateRange(from, remove - 1));
        } else {
          vlanList.add(i, generateRange(remove + 1, to));
          vlanList.add(i, generateRange(from, remove - 1));
        }

        break;
      } else if (from == remove) {
        vlanList.remove(i);
        break;
      }
    }

    StringJoiner ans = new StringJoiner(",");

    vlanList.stream()
        .map(
            vlan -> {
              StringJoiner sj = new StringJoiner("-");
              for (Integer v : vlan) sj.add(v + "");
              return sj.toString();
            })
        .forEach(ans::add);

    return ans.toString();
  }

  public static Integer[] generateRange(int from, int to) {
    if (from < to) {
      return new Integer[] {from, to};
    } else {
      return new Integer[] {from};
    }
  }
}
```

JS算法源码

```javascript
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    let vlanArr = lines[0].split(",");
    let remove = parseInt(lines[1]);

    console.log(getResult(vlanArr, remove));

    lines.length = 0;
  }
});

function getResult(vlanArr, remove) {
  vlanArr = vlanArr
    .map((vlan) => vlan.split("-").map(Number))
    .sort((a, b) => a[0] - b[0]);

  for (let i = 0; i < vlanArr.length; i++) {
    let [from, to] = vlanArr[i];

    if (to != undefined) {
      if (remove < from || remove > to) continue;

      if (remove == from) {
        vlanArr.splice(i, 1, generateRange(remove + 1, to));
      } else if (remove == to) {
        vlanArr.splice(i, 1, generateRange(from, remove - 1));
      } else {
        vlanArr.splice(
          i,
          1,
          generateRange(from, remove - 1),
          generateRange(remove + 1, to)
        );
      }
      break;
    } else if (from === remove) {
      vlanArr.splice(i, 1);
      break;
    }
  }

  return vlanArr.map((vlan) => vlan.join("-")).join(",");
}

function generateRange(from, to) {
  if (from < to) return [from, to];
  else return [from];
}
```

Python算法源码

```python
# 输入获取
vlanArr = list(map(lambda x: list(map(int, x.split("-"))), input().split(",")))
remove = int(input())


def generateRange(start, end):
    if start < end:
        return [start, end]
    else:
        return [start]


# 算法入口
def getResult():
    vlanArr.sort(key=lambda x: x[0])

    for i in range(len(vlanArr)):
        vlan = vlanArr[i]

        start = vlan[0]
        if len(vlan) > 1:
            end = vlan[1]

            if remove < start or remove > end:
                continue

            vlanArr.pop(i)

            if remove == start:
                vlanArr.insert(i, generateRange(remove + 1, end))
            elif remove == end:
                vlanArr.insert(i, generateRange(start, remove - 1))
            else:
                vlanArr.insert(i, generateRange(remove + 1, end))
                vlanArr.insert(i, generateRange(start, remove - 1))

            break
        elif start == remove:
            vlanArr.pop(i)
            break

    return ",".join(list(map(lambda x: "-".join(map(str, x)), vlanArr)))


# 算法调用
```