RangeList Function

- A two-dimensional array named `resultRange` is maintained to manage the result values

When `add` operator

```
add(range) {
  // when range is an empty array
  if (range.length === 0) {
    return;
  } else {
    let i = 0;

    // Case 1: When the maximum value of the element in resultRange is less than t
    // Example: resultRange is [[1, 5]], range is [10, 20]
    while (i < this.resultRange.length && this.resultRange[i][1] < range[0]) {
      i++;
    }

    // Case2: When the minimum value of the element in resultRange is less than or
    // Example: resultRange is [[1, 5], [10, 20]], range is [20, 20]
    while (i < this.resultRange.length && this.resultRange[i][0] <= range[1]) {
      range = [Math.min(range[0], this.resultRange[i][0]), Math.max(range[1], this
      this.resultRange.splice(i, 1);
    }

    // Insert range into resultRange
    this.resultRange.splice(i, 0, range);
  }
}
```

When `remove` operator

```
remove(range) {
  // When resultRange is empty or range is empty
  if (this.resultRange.length === 0 || range.length === 0) return;

  // left is the minimum value in the range;
  // right is the maximum value in the range;
  let left = range[0];
  let right = range[1];
```

```
    // min is the minimum value in the resultResult
    // max is the maximum value in the resultResult
    let min = this.resultRange[0][0], max = this.resultRange[this.resultRange.length

    // Case1: When the min more than the right or the max less than the left, exit r
    // Example: rangeResult is [[1, 5], [10, 20]], range is [30, 40]
    if (min > right || max < left) return;

    // In the loop
    for (let i = 0; i < this.resultRange.length; i++) {
      // currentRange is the current element in the resultResult
      let currentRange = this.resultRange[i];

      // Case2: the currentRange[0] less than or equal the left and the currentRange
      // Example1: currentRange is [10, 20], range is [10, 10]
      // Example2: currentRange is [10, 20], range is [20, 20]
      // Example3: currentRange is [10, 20], range is [10, 20]
      // Example4: currentRange is [10, 29], range is [15, 17]
      if (currentRange[0] <= left && currentRange[1] >= right) {
        if (currentRange[0] === left && currentRange[1] !== right) {
          currentRange[0] = right;
        } else if (currentRange[0] !== left && currentRange[1] === right) {
          currentRange[1] = left;
        } else if (currentRange[0] === left && currentRange[1] === right)  {
          this.resultRange.splice(i, 1);
        } else{
          let temp = [right, currentRange[1]];
          currentRange[1] = left;
          this.resultRange.splice(i + 1, 0, temp);
        }
        return;
      }

      // Case3: the currentRange[0] less than the left and the currentRang[1] more t
      // Example: currentRange is [1, 5], range is [3, 19]
      if (currentRange[0] < left && currentRange[1] >= left && currentRange[1] < rig
        currentRange[1] = left;
        continue;
      }
      // Case4: the currentRange[0] less than or equal right and more than left and
      // Example: currentRange is [10, 21], range is [10, 11]
      if (left < currentRange[0] && currentRange[0] <= right && currentRange[1] > ri
        currentRange[0] = right;
        return;
      }
      // Case5: the currentRange[0] more than and equal left and the currentRange[1]
      // Example: currentRange is [11, 15], range is [3, 19]
      if (currentRange[0] > left && currentRange[1] <=right) {
        this.resultRange.splice(i, 1);
        i--;
      }
    }
  }
```

When `print` operator

```
  print() {
    let i = 0;

    // Convert output format
    // Example: [[1, 5], [10, 20]] convert to [1, 5) [10, 20)
    let result = "";
    while (i < this.resultRange.length) {
      result = result + " " + "[" + this.resultRange[i][0] + ", " + this.resultRange
      i++;
    }
    console.log(result);
    return result;
  }
```

> Thinking in JackDan