1.

```
Function conquer(integer array arr, integer start, integer end, integer answer)
    middle <- (start + end - 1) / 2

    // Left array stores the left half from array and right for the right half
    Left <- arr[start to middle]
    Right <- arr[middle to end]

    // index for the arrays
    left index <- 0
    right index <- 0
    array index <- start

    // bound of the index of the array
    length of left <- middle - start
    length of right <- end - middle

    // do the general case
    while left index < length of left and right index < length of right

        // when a pair is reverse
        if Left[left index] > Right[right index]
            arr[array index] <- Left[left index]

            // add the number of reverse pairs due to current left index
            answer <- answer + right index
            left index <- left index + 1
        else
            arr[array index] <- Right[right index]
            right index <- right index + 1

        array index <- array index + 1

    // handle the left half if needed
    while left index < length of left
        arr[array index] <- Left[left index]
        left index <- left index + 1
        array index <- array index + 1

    // handle the right half if needed
    while right index < length of right
        arr[array index] <- Right[right index]
        right index <- right index + 1
        array index <- array index + 1
```

```
Function divide(integer array arr, integer start, integer end, integer &answer)

    // only continue to divide if end is greater than start
    if end > start
        middle <- (start + end - 1) / 2

        // left half
        divide(arr, start, middle, ans)

        // right half
        divide(arr, middle, end, ans)

        // conquer current case
        conquer(arr, start, end, ans)


Function main
    arr <- [integer array]
    answer <- 0
    divide(arr, 0, length of array - 1, ans)
    print(ans)
```

2.

To simplify, ignore boundary (floor & ceiling) and assume base case is constant for some n.

$$T(n) = \begin{cases} O(1) & , n = 1 \\ 2T(\frac{n}{2}) + O(n) & , n \geq 2 \end{cases}$$

$$\Rightarrow T(n) \leq 2T(\tfrac{n}{2}) + c \cdot n \leq 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + c \cdot n = 4T(\tfrac{n}{4}) + 2 \cdot c \cdot n$$

$$\leq 4(2T(\tfrac{n}{8}) + c \cdot \tfrac{n}{4}) + 2cn = 8T(\tfrac{n}{8}) + 3c \cdot n \cdots \leq 2^k T(\tfrac{n}{2^k}) + k \cdot c \cdot n$$

$\therefore$ expansion stop at $2^k = n \Rightarrow T(n) \leq n \cdot T(1) + c \cdot n \cdot \log_2 n = O(n) + O(n \log n) = O(n \log n)$ *