```
MERGE(A, p, q, r)
  n_1 = q − p + 1
  n_2 = r − q
  let L[1 .. n_1 + 1] and R[1 .. n_2 + 1] be new arrays
  for i = 1 to n_1
      L[i] = A[p + i − 1]
  for j = 1 to n_2
      R[j] = A[q + j]
  L[n_1 + 1] = ∞
  R[n_2 + 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
```

Θ(n) Θ(n₁) Θ(n₂) Θ(r−p)

Runtime of Merge

$$\Theta(n) + \underbrace{\Theta(n_1) + \Theta(n_2)}_{\Theta(n)} + \underbrace{\Theta(r-p)}_{\Theta(n)} = \Theta(n)$$

$n = r - p + 1$

$n_1 + n_2 = q - p + 1 + r - q = r - p + 1 = n$

Merge (A, p, q, r)

Input: - array A w/ valid indices $p \le q \le r$
- $A[p..q]$ is sorted and $A[q+1..r]$ is ~~sorted~~ sorted.

Output:
- The new $A[p..r]$ is a permutation of the old $A[p..r]$
- The new $A[p..r]$ is sorted.

Proof of correctness is an example of using a loop invariant.

Merge Sort correctness

Induction on size of subarray given for sorting: $n = r - p + 1$

Induction Hypothesis:

$P(i)$ = Merge-Sort works correctly for subarrays of size at most i

Base Case: $i = 1$ ✓

Induction Step: assume that $P(i)$ holds.

Need to show that $P(i+1)$ holds

Apply IH, MergeSort on lines 3, 4 work correctly.

Merge sort run-time analysis

Using recursion equation ("recurrence")

$T(n)$ = running time of merge sort on inputs of size n.

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

$2 \cdot T\left(\frac{n}{2}\right)$: lines 3 and 4 (recursive calls).

$\Theta(n)$: merge function

$$T(1) = c \quad \text{or} \quad T(1) = \Theta(1)$$

c: constant

$$T(n) = \Theta(n \log n)$$