MERGESORT

CS340

Merge Sort

- Insertion sort is **incremental**: the first part of the array is sorted. Keep it sorted when adding a new number.
- Another technique for developing algorithms is **divide and conquer**.
- Divide and conquer
  - **divides** a problem into smaller subproblems (divide)
  - solves those subproblems (**conquer**)
  - **combines** the results (combine)

Merge Sort

To sort A[p..r]

- **Divide** by splitting into two subarrays A[p..q] and A[q+1..r], where q is the halfway point of A[p..r].
- **Conquer** by **recursively** sorting the two subarrays A[p..q] and A[q+1..r].
- **Combine** by merging the two sorted subarrays A[p..q] and A[q+1..r] to produce a single sorted subarray A[p..r].

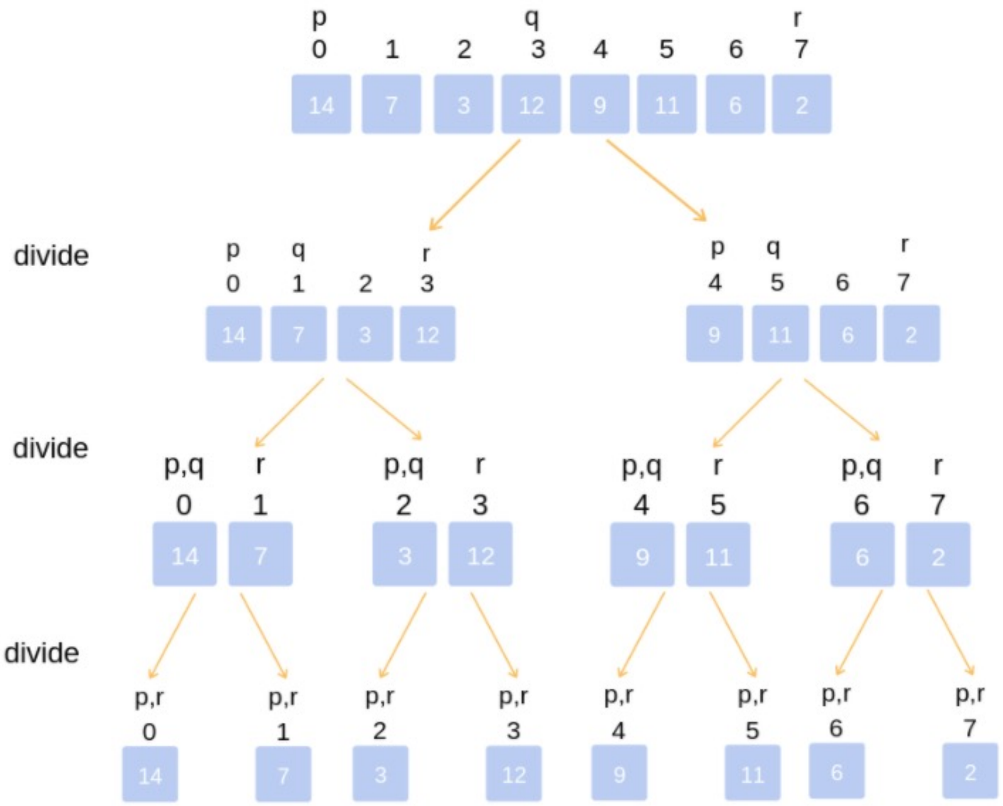
Merge Sort

```
MERGE-SORT(A, p, r)
  if p < r                          // check for base case
      q = ⌊(p + r)/2⌋               // divide
      MERGE-SORT(A, p, q)           // conquer
      MERGE-SORT(A, q + 1, r)       // conquer
      MERGE(A, p, q, r)             // combine
```

Initial call: Merge-Sort(A, 1, n)

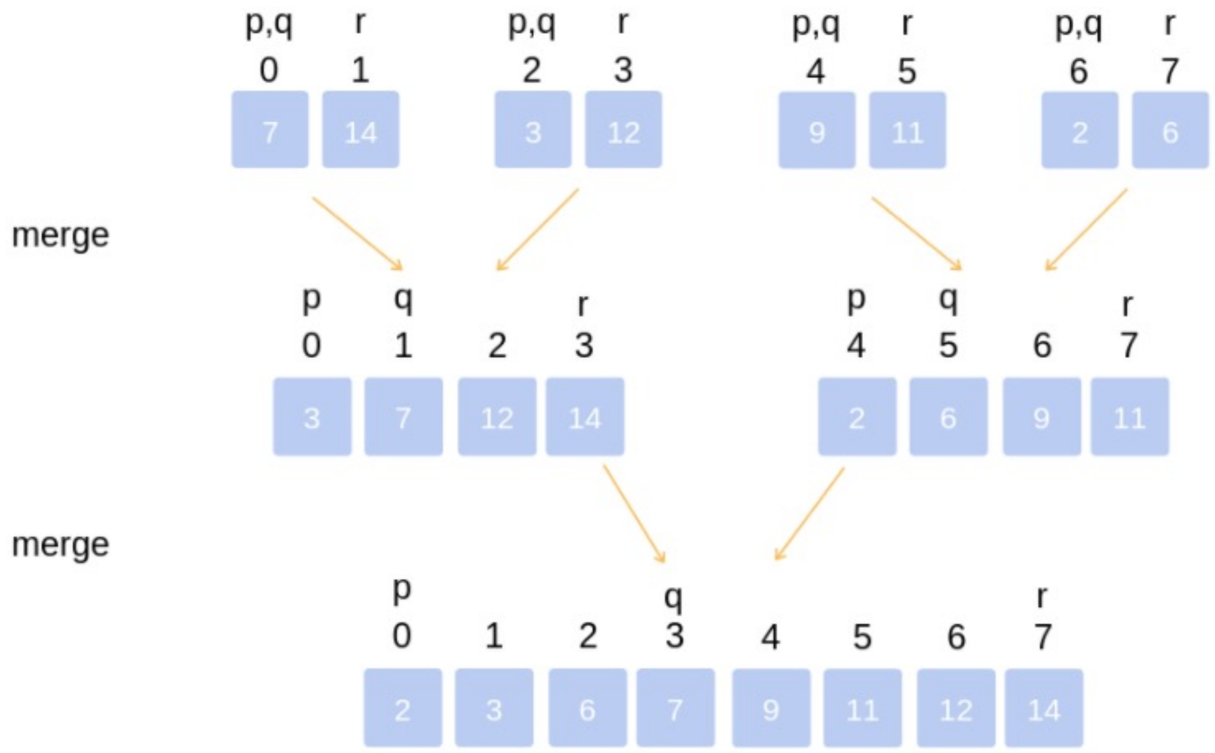
Merge Sort

- First, divide



Merge Sort

- Then merge



The Merge Procedure

- **Input**: Array A and indices p, q, r such that
  - $p \leq q < r$.
  - Subarray A[p..q] is sorted and subarray A[q+1..r] is sorted. By the restrictions on p, q, r, neither subarray is empty.
- **Output**: The two subarrays are merged into a single sorted subarray in A[p..r].

Merging two sorted arrays

What is the time complexity of MERGE() ?

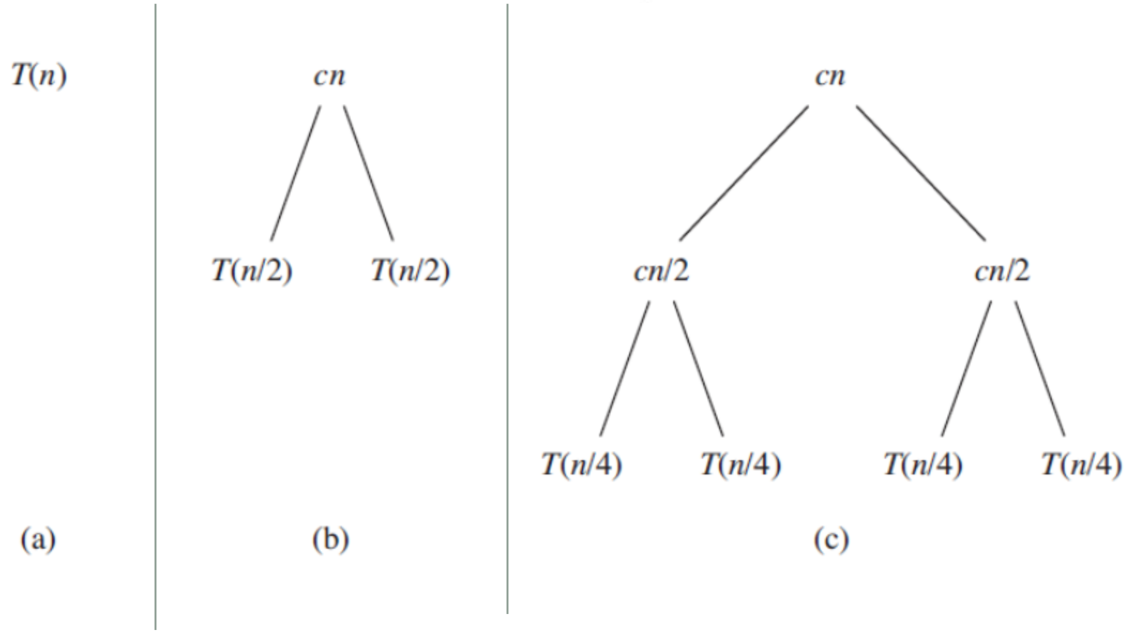
```
MERGE(A, p, q, r)
  n_1 = q - p + 1
  n_2 = r - q
  let L[1..n_1 + 1] and R[1..n_2 + 1] be new arrays
  for i = 1 to n_1
      L[i] = A[p + i - 1]
  for j = 1 to n_2
      R[j] = A[q + j]
  L[n_1 + 1] = ∞
  R[n_2 + 1] = ∞
  i = 1
  j = 1
  for k = p to r
      if L[i] ≤ R[j]
          A[k] = L[i]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

Time complexity of Merge Sort

Describe the complexity with a **recurrence equation** →

$$T(n) = \begin{cases} c & \text{if } n = 1\,, \\ 2T(n/2) + cn & \text{if } n > 1\,. \end{cases}$$

Draw a recursion tree

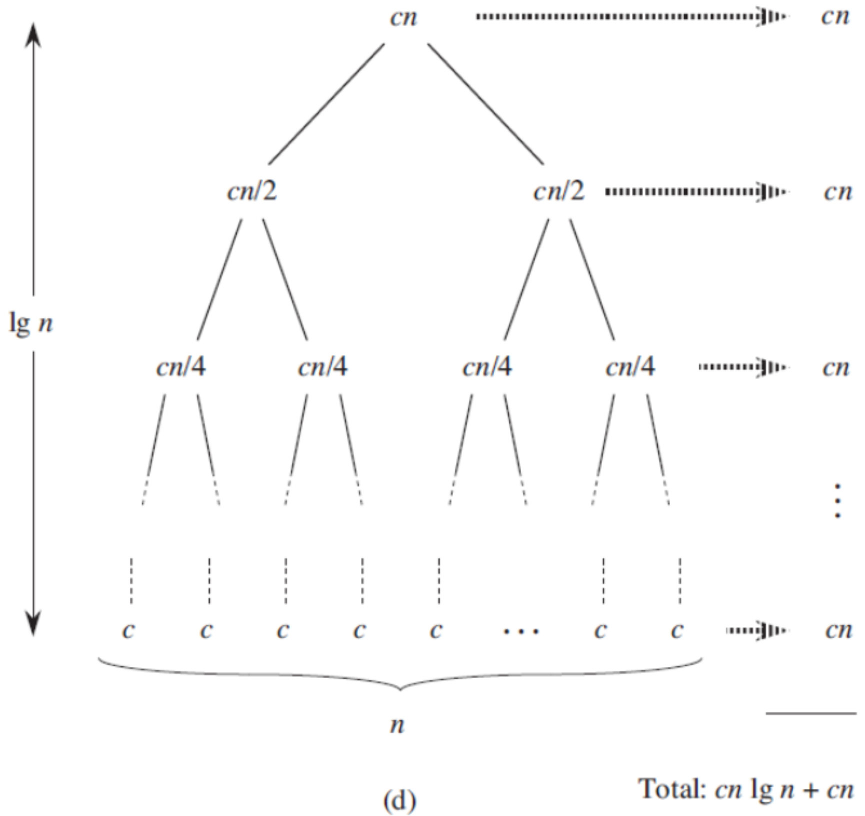


Complexity of Merge Sort

1. How many levels is the tree?
2. What is the cost at each level?

Complexity = $\Theta(n \lg n)$

How does this compare to insertion sort?



(d)