Case Western Reserve University
Est. 1826

Merge Sort

Prof. Dianne Foreback

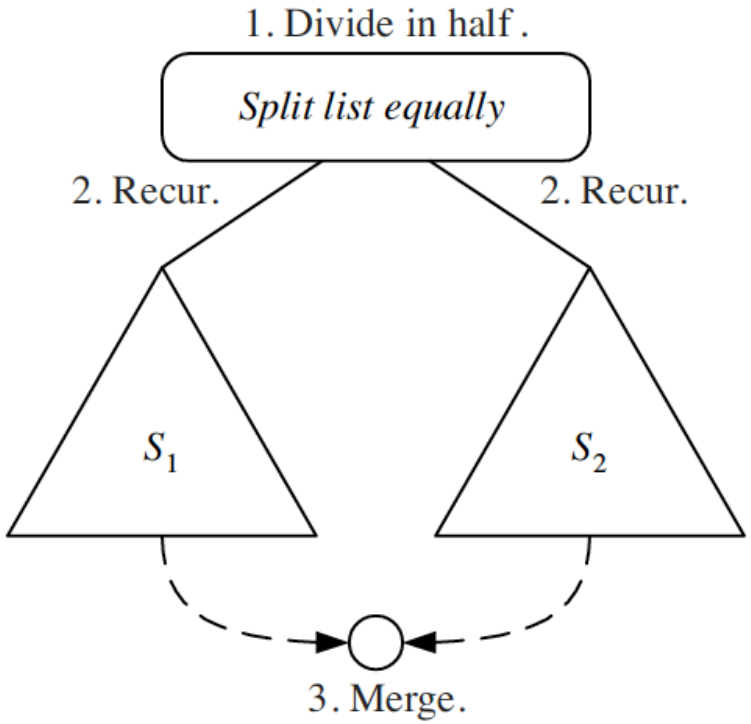
Information on slides originating from *Algorithm Design & Applications* by Michael T. Goodrich and Roberto Tamassia, © 2015 John Wiley & Sons, Inc.Goodrich and Tamassia, ISBN: 978-1118335918.

Reading Material

- ***Algorithm Design & Applications*** **by Michael T. Goodrich and Roberto Tamassia**
  - **Chapter 8 Section 8.1 (Note, the book has a typeo for the MergeSort algorithm. I fixed it in these slides.)**

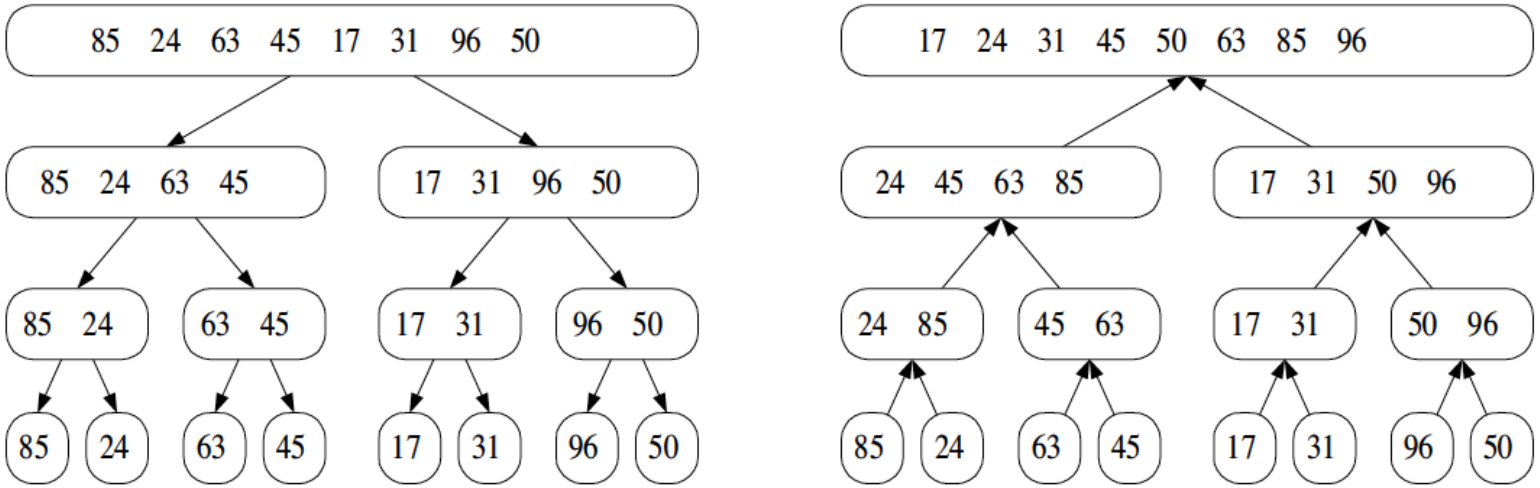
Divide-and-Conquer

- Divide-and conquer is a general algorithm design paradigm:
  - Divide: divide the input data S in two disjoint subsets S_1 and S_2
  - Recur: solve the subproblems associated with S_1 and S_2
  - Conquer: combine the solutions for S_1 and S_2 into a solution for S
- The base case for the recursion are subproblems of size 0 or 1



Merge-Sort

- Merge-sort is a sorting algorithm based on the divide-and-conquer paradigm
- Like heap-sort
  - It has $O(n \log n)$ running time
- Unlike heap-sort
  - It does not use an auxiliary priority queue
  - It accesses data in a sequential manner (suitable to sort data on a disk)



The Merge-Sort Algorithm

- Merge-sort on an input sequence S with n elements consists of three steps:
  - Divide: partition S into two sequences S_1 and S_2 of about $n/2$ elements each
  - Recur: recursively sort S_1 and S_2
  - Conquer: merge S_1 and S_2 into a unique sorted sequence

Algorithm *mergeSort*(S)
  Input sequence S with n elements
  Output sequence S sorted according to C
if $S.size() > 1$
  $(S_1, S_2) \leftarrow partition(S, n/2)$
  $mergeSort(S_1)$
  $mergeSort(S_2)$
  $S \leftarrow merge(S_1, S_2)$

Merging Two Sorted Sequences

- The conquer step of merge-sort consists of merging two sorted sequences A and B into a sorted sequence S containing the union of the elements of A and B
- Merging two sorted sequences, each with $n/2$ elements and implemented by means of a doubly linked list, takes $O(n)$ time

Algorithm merge(S_1, S_2, S):
Input: Two arrays, S_1 and S_2, of size n_1 and n_2, respectively, sorted in non-decreasing order, and an empty array, S, of size at least $n_1 + n_2$
Output: S, containing the elements from S_1 and S_2 in sorted order

$i \leftarrow 1$
$j \leftarrow 1$
while $i \leq n_1$ **and** j $\leq n_2$ **do**
  if $S_1[i] \leq S_2[j]$ **then**
    $S[i + j - 1] \leftarrow S_1[i]$
    $i \leftarrow i + 1$
  else
    $S[i + j - 1] \leftarrow S_2[j]$
    $j \leftarrow j + 1$
while $i \leq n_1$ **do**
  $S[i + j - 1] \leftarrow S_1[i]$
  $i \leftarrow i + 1$
while $j \leq n_2$ **do**
  $S[i + j - 1] \leftarrow S_2[j]$
  $j \leftarrow j + 1$

Type-o in book fixed on these slides. While conditions are not correct in the book.

Thank You !

Questions ?