Methodic and Practical Foundations of Computer Science 1

01-Overview

Sayed Ahmad Sahim

Kandahar University
Computer Science Faculty

algprog17@gmail.com

October 15, 2017

Objectives

1. Sorting Algorithms
2. Divide-and-Conquer
3. Merge Sort Approach
4. Merge Sort
   - Example: n Power of 2
   - Merging
   - Example: MERGE(A, 9, 12, 16)
   - Merge - Pseudocode
   - Running Time of Merge
5. Analyzing Divide-and Conquer Algorithms
6. MERGE-SORT Running Time
7. Home Work
8. Question

Sorting Algorithms

- Insertion sort:
  - Design approach: incremental
  - Sorts in place: Yes
  - Best case: θ(n)
  - Worst case: $\theta(n^2)$
- Bubble sort
  - Design approach: incremental
  - Sorts in place: Yes
  - Worst case: $\theta(n^2)$

Sorting Algorithms

- Selection sort
  - Design approach: incremental
  - Sorts in place: Yes
  - Worst case: $\theta(n^2)$
- Merge sort
  - Design approach: divide and conquer
  - Sorts in place: No
  - Running time: *Lets See*

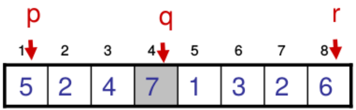
Divide-and-Conquer

- **Divide** the problem into a number of sub-problems
  - Similar sub-problems of smaller size
- **Conquer** the sub-problems
  - Solve the sub-problems *recursively*
  - Sub-problem size small enough $\rightarrow$ solve the problems in straightforward manner
- **Combine** the solutions of the sub-problems
  - Obtain the solution for the original problem

Merge Sort Approach

- To sort an array A[p . . r]:
- Divide:
  - Divide the n-element sequence to be sorted into two subsequences of n/2 elements each
- Conquer:
  - Sort the subsequences recursively using merge sort
  - When the size of the sequences is 1 there is nothing more to d
- Combine:
  - Merge the two sorted subsequences

Merge Sort

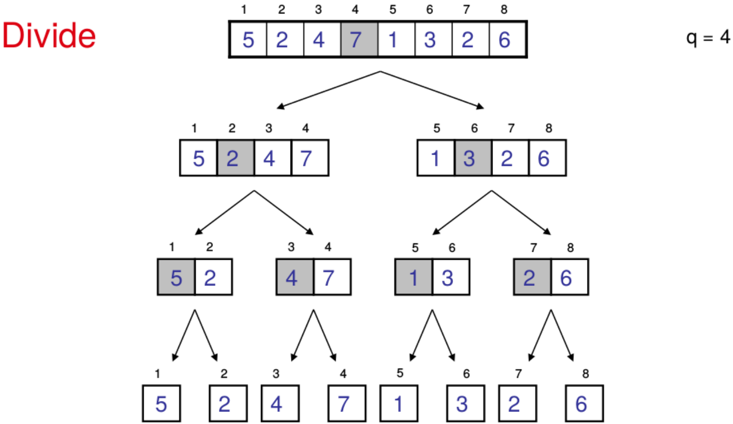


Alg.: MERGE-SORT(A, p, r)

if p < r	▷ Check for base case
then $q \leftarrow \lfloor (p+r)/2 \rfloor$	▷ Divide
MERGE-SORT(A, p, q)	▷ Conquer
MERGE-SORT(A, q + 1, r)	▷ Conquer
MERGE(A, p, q, r)	▷ Combine

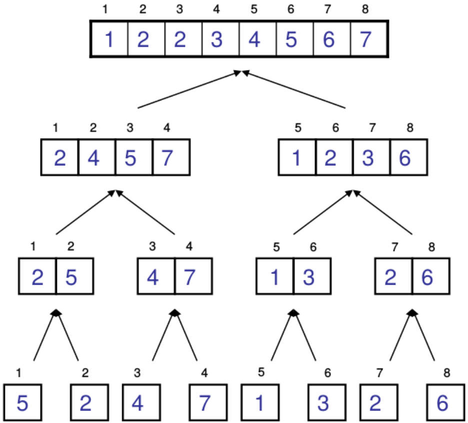
- Initial call: MERGE-SORT(A, 1, n)

Example: n Power of 2

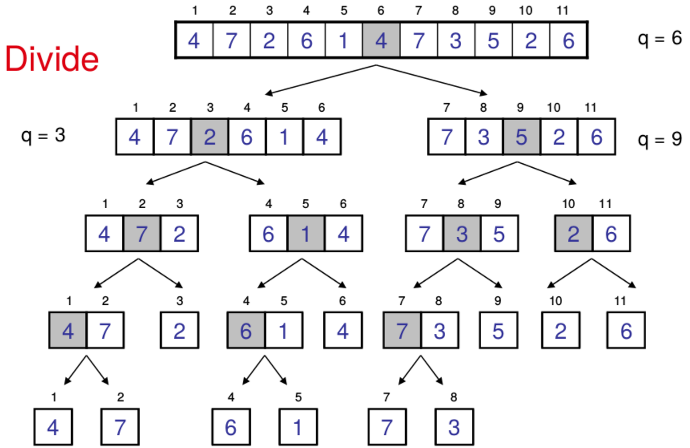


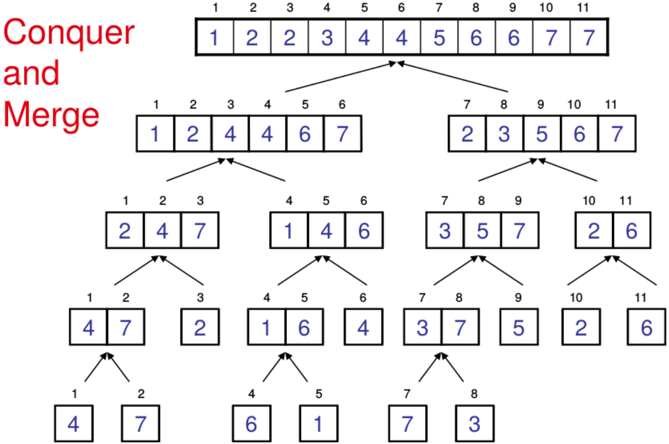
Example: n Power of 2

Conquer and Merge

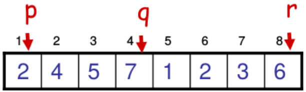


Example: n Power of 2




Conquer and Merge
1 2 3 4 5 6 7 8 9 10 11
1 2 2 3 4 4 5 6 6 7 7
1 2 3 4 5 6
1 2 4 4 6 7
7 8 9 10 11
2 3 5 6 7
1 2 3
2 4 7
4 5 6
1 4 6
7 8 9
3 5 7
10 11
2 6
1 2
4 7
3
2
4 5
1 6
6
4
7 8
3 7
9
5
10
2
11
6
1
4
2
7
4
6
5
1
7
7
8
3

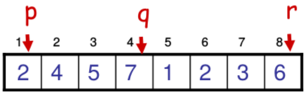
Merging

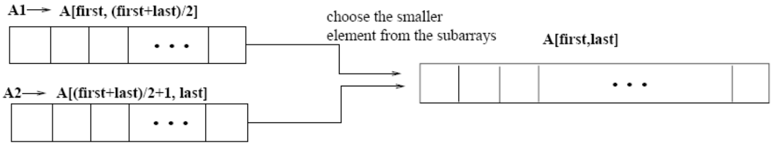


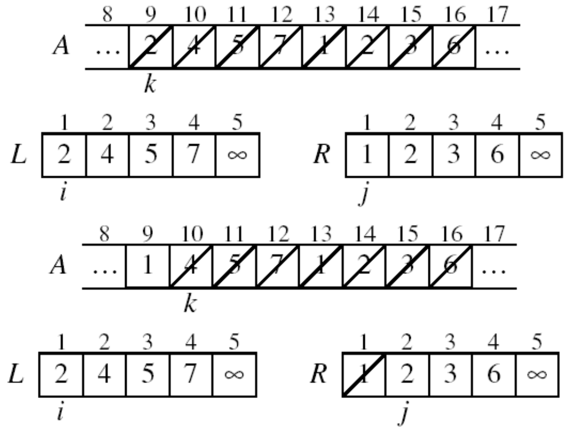
- Input: Array A and indices p, q, r such that $p \leq q < r$
  - Subarrays A[p . . q] and A[q + 1 . . r] are sorted
- Output: One single sorted subarray A[p . . r]

Merging

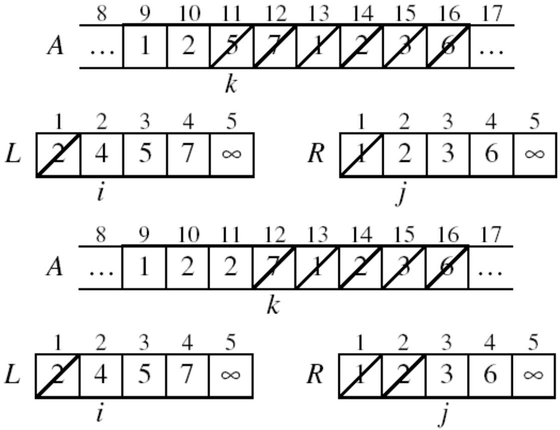
- Idea for merging:
  - Two piles of sorted cards
    - Choose the smaller of the two top cards
    - Remove it and place it in the output pile
  - Repeat the process until one pile is empty
  - Take the remaining input pile and place it face-down onto the output pile



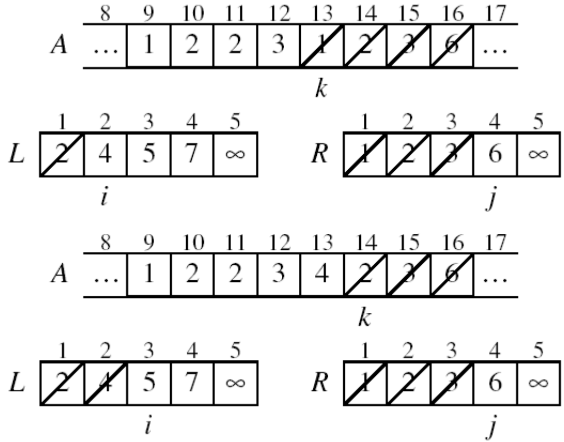



8 9 10 11 12 13 14 15 16 17
A … 2 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j
8 9 10 11 12 13 14 15 16 17
A … 1 4 5 7 1 2 3 6 …
k
1 2 3 4 5
L 2 4 5 7 ∞
i
1 2 3 4 5
R 1 2 3 6 ∞
j

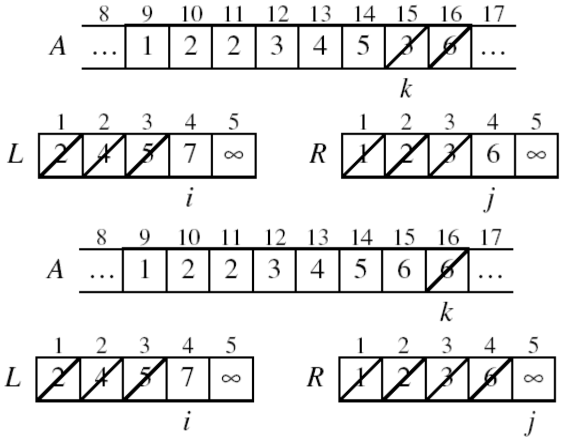
Example: MERGE(A, 9, 12, 16)



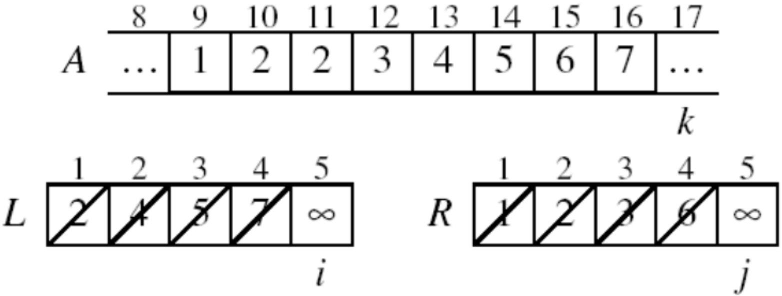
Example: Cont



Example: Cont



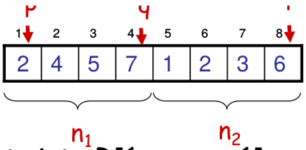
Example: Cont



Done!

Alg.: MERGE(A, p, q, r)

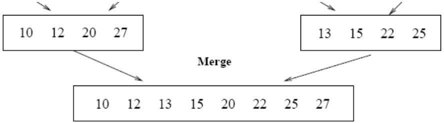
1. Compute n_1 and n_2
2. Copy the first n_1 elements into $L[1\,..\,n_1+1]$ and the next n_2 elements into $R[1\,..\,n_2+1]$
3. $L[n_1+1] \leftarrow \infty$; $R[n_2+1] \leftarrow \infty$
4. $i \leftarrow 1$; $j \leftarrow 1$
5. **for** $k \leftarrow p$ **to** r
6. **do if** $L[\,i\,] \le R[\,j\,]$
7. **then** $A[k] \leftarrow L[\,i\,]$
8. $i \leftarrow i + 1$
9. **else** $A[k] \leftarrow R[\,j\,]$
10. $j \leftarrow j + 1$



	p			q	
L	2	4	5	7	∞
	q + 1			r	
R	1	2	3	6	∞

Running Time of Merge (assume last for loop)

- Initialization (copying into temporary arrays):
  - $\theta(n_1 + n_2) = \theta(\text{n})$
- Adding the elements to the final array:
  - n iterations, each taking constant time $\rightarrow \theta(\text{n})$
- Total time for Merge:
  - $\theta(\text{n})$



Analyzing Divide-and Conquer Algorithms

- The recurrence is based on the three steps of the paradigm:
  - $T(n)$ running time on a problem of size n
  - Divide the problem into a subproblems, each of size n/b: takes $D(n)$
  - Conquer (solve) the subproblems $aT(n/b)$
  - Combine the solutions $C(n)$

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c \\ aT(n/b) + D(n) + C(n) & \text{otherwise} \end{cases}$$

MERGE-SORT Running Time

- Divide:
  - compute q as the average of p and r: $D(n) = \theta(1)$
- Conquer:
  - recursively solve 2 subproblems, each of size $n/2 \rightarrow 2T\ (n/2)$
- Combine:
  - MERGE on an n-element subarray takes $\theta(n)$ time $\rightarrow C(n) = \theta(n)$

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

Home Work

- Solve the Recurrence

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

- Use Masters Theorem:
- Compare n with $f(n) = cn$
- Case 2: $T(n) = \theta(n \lg n)$

Home Work

- Problem: Sort a file of huge records with tiny keys
- Example application: Reorganize your MP-3 files
- Which method to use and why?
  - merge sort
  - selection sort
  - bubble sort
  - a custom algorithm for huge records/tiny keys
  - insertion sort

Home Work

- Problem: Sort a huge randomly-ordered file of small records Application: Process transaction record for a phone company
- Which method to use and why?
  - merge sort
  - selection sort
  - bubble sort
  - a custom algorithm for huge records/tiny keys
  - insertion sort

Home Work

- Problem: sort a file that is already almost in order
- Applications:
  - Re-sort a huge database after a few changes
  - Doublecheck that someone else sorted a file
- Which method to use and why?
  - merge sort
  - selection sort
  - bubble sort
  - a custom algorithm for huge records/tiny keys
  - insertion sort

Home Work

- create an array with at least ten elements
- Implement the following algorithms in java to sort the array.
  - merge sort
  - selection sort
  - bubble sort
  - insertion sort

Question