CS 2022 : DATA STRUCTURES & ALGORITHMS

Lecture 3: Recursion, Divide & Conquer and Merge Sort

Malaka Walpola

Outline

- Recursion
- Divide & Conquer Approach
- Merge Sort
- Analyzing Merge Algorithm

Learning Outcomes

- After successfully studying contents covered in this lecture, students should be able to,
  - explain and develop recursive algorithms
  - explain the divide & conquer algorithm design technique
  - explain the merge sort algorithm

Recursion

- An algorithm or function that **calls itself** directly or indirectly to **solve a smaller version** of its task is recursive
- Recursion occurs until a final call which does not require further recursion
  - Terminating condition(s)

Recursion

- Example Recursive Solutions
  - Factorial
  - Searching in a Linked List
  - Creating a long string by duplicating a string several times
- Recursion Exercises
  - Searching in an array
  - Checking whether a given string is a palindrome

Divide & Conquer Approach

- Approach is Based on Recursion
- Strategy
  - **Divide:** Divide a given problem into smaller sub-problems that are similar to original problem
  - **Conquer:** Solve **a subset** of sub-problem recursively
  - **Combine:** Combine the solutions to sub-problems to get the solution to the original problem

Divide & Conquer Approach

- Examples
  - Binary search
  - Depth-first tree traversals
  - Merge sort
  - Quick sort

MERGE SORT

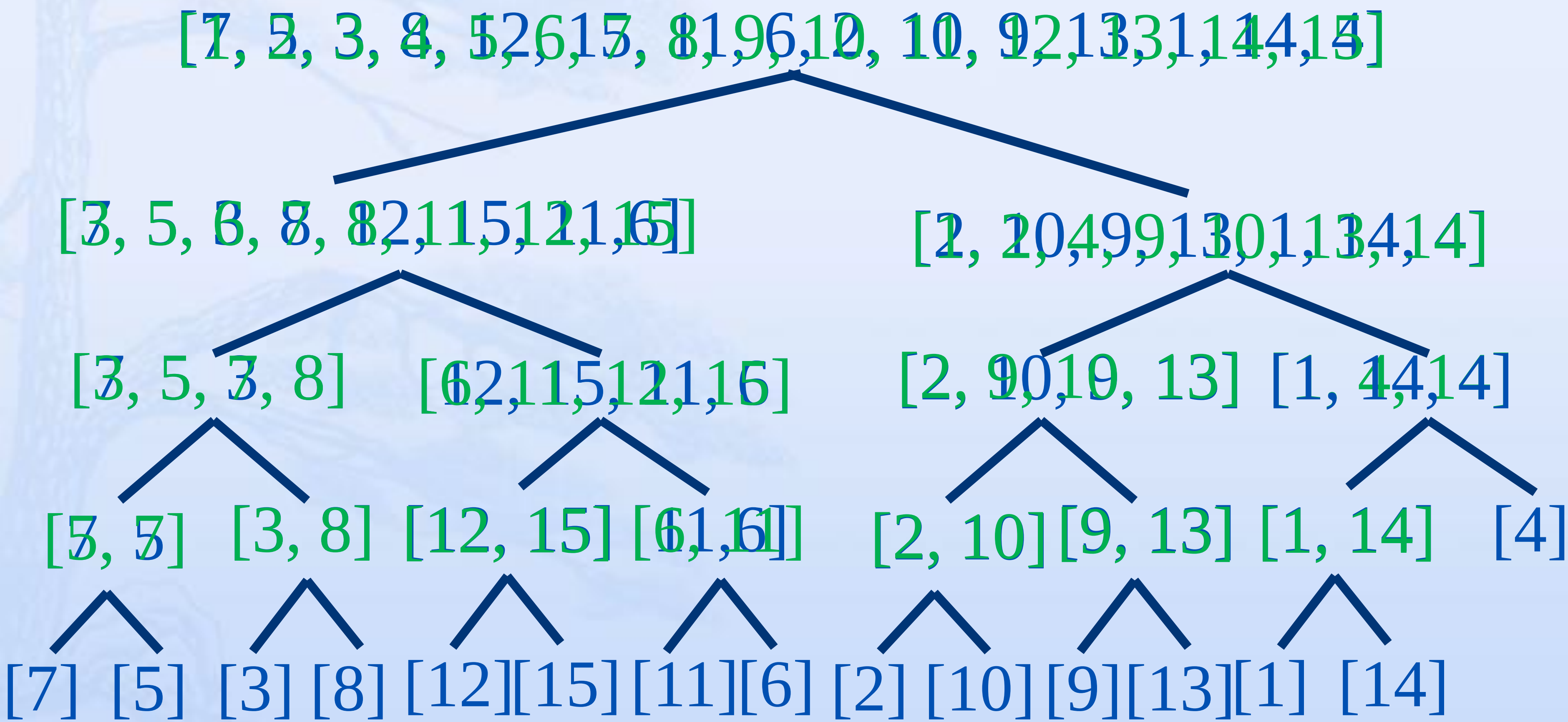
- **Divide:** Divide the *n*-element sequence to be sorted into two subsequences of n/2 elements each
- **Conquer:** Sort the two subsequences recursively using merge sort
- **Combine:** Merge the two sorted subsequences to produce the sorted sequence

Merge Sort

- To sort n numbers
  - if n = 1 done! – **Boundary condition** for recursion
  - Recursively sort 2 lists of numbers $\lceil n/2 \rceil$ and $\lfloor n/2 \rfloor$ elements
  - merge 2 sorted lists

Merge Sort

- The Idea



MERGE SORT

- The Algorithm

```
MERGE-SORT(A, p, r)
  IF p < r
     q ← ⌊(p + r)/2⌋
     MERGE-SORT(A, p, q)
     MERGE-SORT(A, q + 1, r)
     MERGE(A, p, q, r)
```

Merge Algorithm

- The Idea
  - Copy two parts into two new arrays
  - Add infinity to the end of two new arrays
  - Copy the contents of the two arrays into the original array in the sorted order
    - Set i & j to 0
    - Compare the elements at i & j
    - Copy small element to array and increment the corresponding index

Merge Algorithm

MERGE(A, p, q, r)

```
n1 ← q - p + 1
n2 ← r - q
//create arrays L[0………n1 ] and R[0………n2]
for i ← 0 to n1-1
    L[i] ← A[p+i]
for j ← 0 to n2-1
    R[j] ← A[(q+1)+j]
L[n1] ← ∞
R[n2] ← ∞
```

Merge Algorithm

```
i ← 0
j ← 0
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
```

Analysis of Merge Algorithm

Code	Cost	Times
`MERGE(A, p, q, r)`	$F(n)$	1
1. $n_1 \leftarrow q - p + 1$	C_1	1
2. $n_2 \leftarrow r - q$	C_2	1
3. //…………	*0*	1
4. for $i \leftarrow 0$ to n_1-1	C_4	n_1+1
5. L[i] ← A[p+i]	C_5	n_1
6. for $j \leftarrow 0$ to $n_2 -1$	C_6	n_2+1
7. R[j] ← A[(q+1)+j]	C_7	n_2
8. $L[n_1] \leftarrow \infty$	C_8	1
9. $R[n_2] \leftarrow \infty$	C_9	1

n = number of elements to merge

$n = n_1+n_2$

$C_4 = C_6$

$C_5 = C_7$

Analysis of Merge Algorithm

Code	Cost	Times
`10. i ← 0`	C_{10}	1
`11. j ← 0`	C_{11}	1
`12. for k ← p to r`	C_{12}	n+1
`13.     if L[i] ≤ R[j]`	C_{13}	n
`14.         A[k] ← L[i]`	C_{14}	n_3
`15.         i ← i + 1`	C_{15}	n_3
`16.     else`	?	?
`17.         A[k] ← R[j]`	C_{17}	n_4
`18.         j ← j + 1`	C_{18}	n_4

$n = n_3 + n_4$

$C_{14} = C_{17}$

$C_{15} = C_{18}$

Analysis of Merge Algorithm

$$
\begin{aligned}
F(n) = c_1 + c_2 + c_4(n_1+1) + c_5 n_1 \\
+ c_6(n_2+1) + c_7 n_2 + c_8 \\
+ c_9 + c_{10} + c_{11} + c_{12}(n+1) \\
+ c_{13} n + c_{14} n_3 + c_{15} n_3 \\
+ c_{17} n_4 + c_{18} n_4
\end{aligned}
$$

$n = n_1 + n_2$

$C_4 = C_6$

$C_5 = C_7$

$n = n_3 + n_4$

$C_{14} = C_{17}$

$C_{15} = C_{18}$

$$
\begin{aligned}
F(n) = c_1 + c_2 + 2c_4 + c_8 + c_9 + c_{10} + c_{11} + c_{12} \\
+ n(c_4 + c_5 + c_{12} + c_{13} + c_{14} + c_{15})
\end{aligned}
$$

$F(n) \in O(n)$ *where* n = n = *number of elements to merge*

SELF STUDYING

Self Studying

- Reading Assignment
  - Recursion and Divide & Conquer Approaches and Merge Sort: Section 2.3.1 of IA
- Homework
  - Analyze the worst case time complexity of **merge** algorithm. (**`MERGE(A, p, q, r)`**)
- ***You should know Merge sort and merge algorithms well when you come to the next class***

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 3rd Ed. Cambridge, MA, MIT Press, 2009.

[2] S. Baase and Allen Van Gelder, *Computer Algorithms: Introduction to Design and Analysis*, 3rd Ed. Delhi, India, Pearson Education, 2000.

[3] Lecture slides from Prof. Erik Demaine of MIT, available at http://dspace.mit.edu/bitstream/handle/1721.1/37150/6-046JFall-2004/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-046JFall-2004/B3727FC3-625D-4FE3-A422-56F7F07E9787/0/lecture_01.pdf