Complexity Calculations and Examples

Notes from Dr. Charlie Obimbo
Revised by Yan Yan.

Contents

1. Example on input size and complexity
2. Sorting algorithms and their complexities
    1. Big-O, Big Omega, & Theta

Question

Computer solves problem of complexity $T(n) \in \Theta(n^3)$ & size $n = 1000$ in 1 minute. What size would be solved in 1 hour?

$$\frac{c * (1000)^3}{c * (x)^3} = \frac{1}{60}$$

$$(x)^3 = 60 * (1000)^3$$

x = 3914

Note that $x \in N$

Sorting

- **Sorting**: Rearranging the values in an array or collection into a specific order (usually into their "natural ordering").
  - – one of the fundamental problems in computer science
- Input: A sequence of n objects
  - $s = \langle a_1, a_2, \ldots, a_n \rangle$
- Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Sorting

- can be solved in many ways:
  - some are faster/slower than others
  - some use more/less memory than others
  - some work better with specific kinds of data

- comparison-based sorting: determining order by comparing pairs of elements:
    - `<`, `>`, `compareTo`, …

Sorting Algorithms

- **bubble sort**: swap adjacent pairs that are out of order
- **selection sort**: look for the smallest element, move to front
- **insertion sort**: build an increasingly large sorted front portion
- **merge sort**: recursively divide the array in half and sort it
- **quick sort**: recursively partition array based on a middle value

Selection Sorting

- Orders a list of values by repeatedly putting the smallest or largest unplaced value into its final position.
  - Look through the list to find the smallest value.
  - Swap it so that it is at index 0.
  - Look through the list to find the second-smallest value.
  - Swap it so that it is at index 1.

    ...

  - Repeat until all values are in their proper places.

Selection sort example

- Initial array:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	22	18	12	-4	27	30	36	50	7	68	91	56	2	85	42	98	25

- After 1st, 2nd, and 3rd passes:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	**-4**	18	12	**22**	27	30	36	50	7	68	91	56	2	85	42	98	25

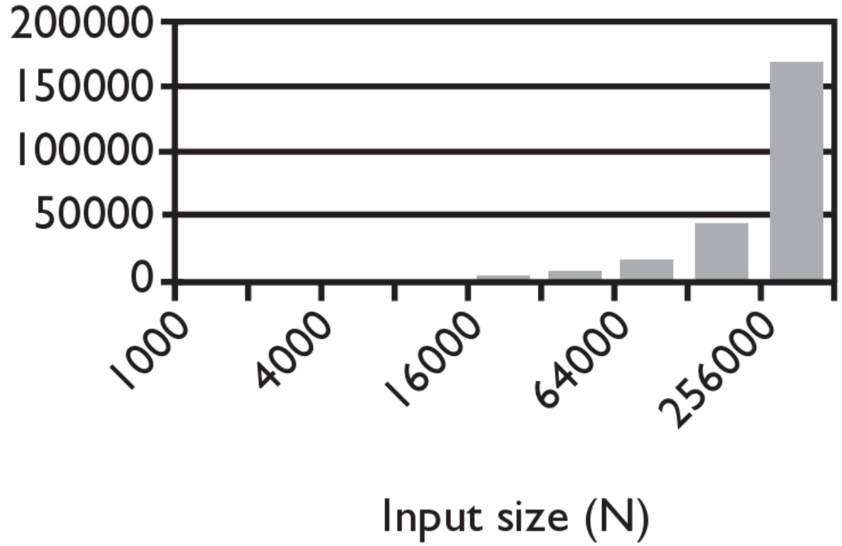
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	**2**	12	22	27	30	36	50	7	68	91	56	**18**	85	42	98	25

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	2	**7**	22	27	30	36	50	**12**	68	91	56	18	85	42	98	25

Selection sort runtime

- What is the complexity class (Big-O) of selection sort?

N	Runtime (ms)
1000	0
2000	16
4000	47
8000	234
16000	657
32000	2562
64000	10265
128000	41141
256000	164985



Similar algorithms

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	22	18	12	-4	27	30	36	50	7	68	91	56	2	85	42	98	25

- **bubble sort**: Make repeated passes, swapping adjacent values

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	18	12	-4	22	27	30	36	7	50	68	56	2	85	42	91	25	98

22 $\longrightarrow$ 50 $\rightarrow$ 91 $\longrightarrow$ 98 $\rightarrow$

- **insertion sort**: Shift each element into a sorted sub-array

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
value	-4	12	18	22	27	30	36	50	7	68	91	56	2	85	42	98	25

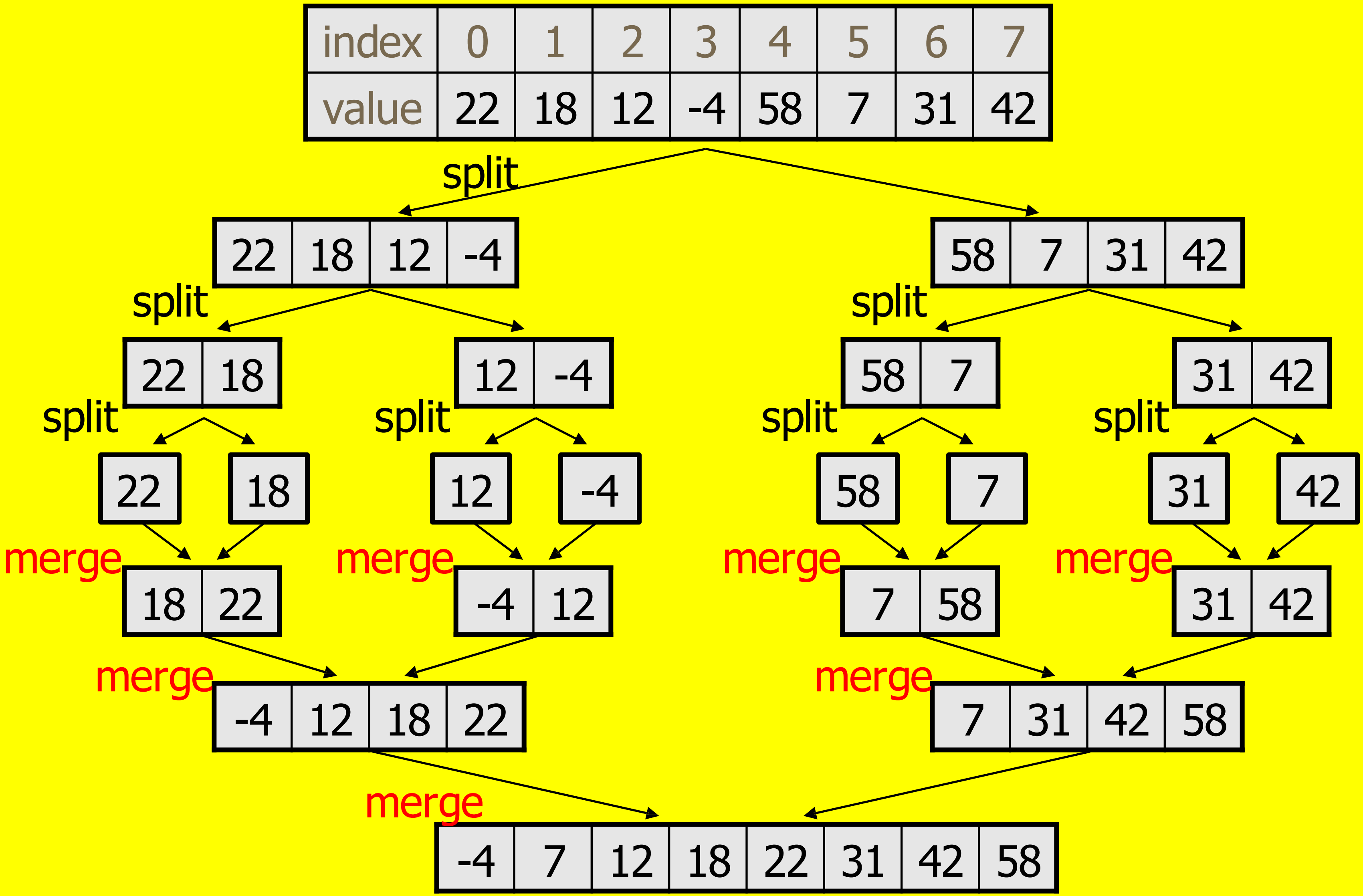
sorted sub-array (indexes 0-7)

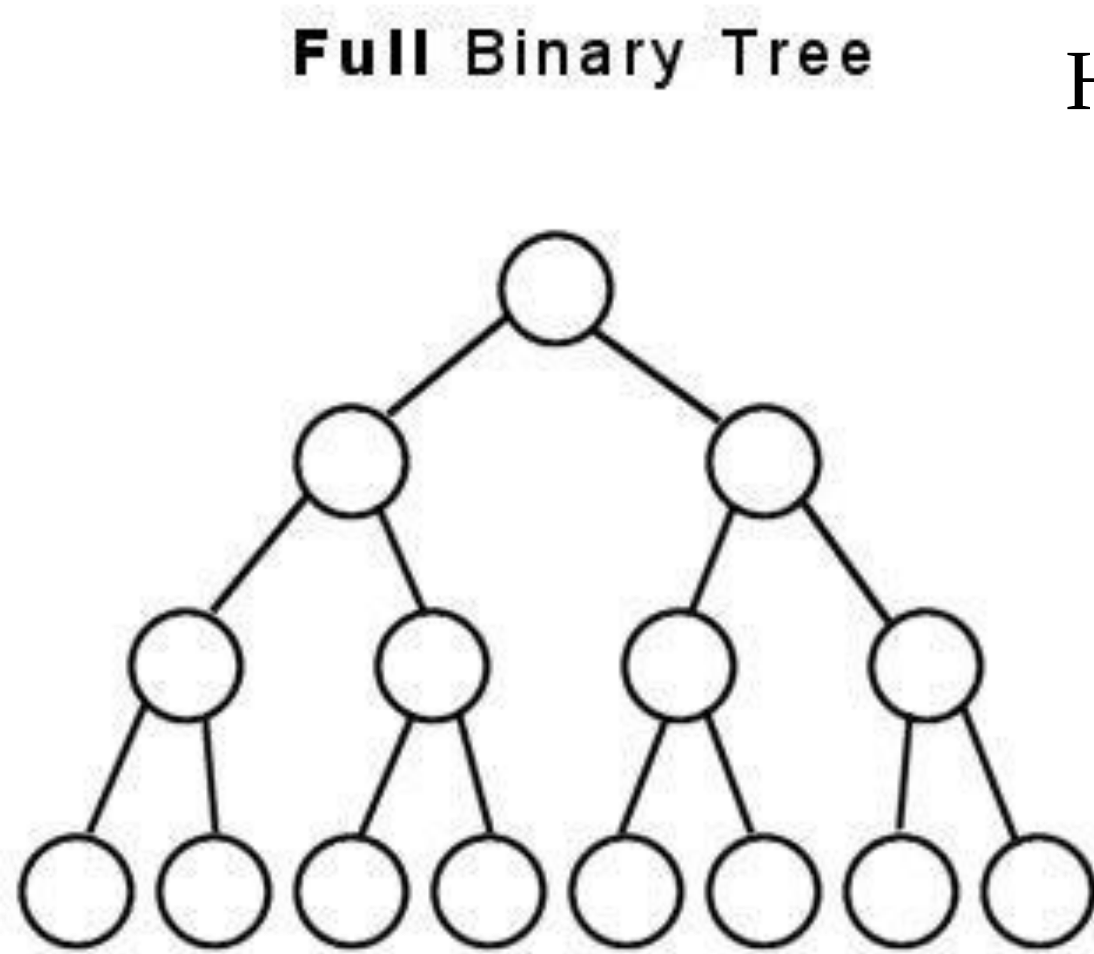
$\longleftarrow$ 7

Merge sort

- Repeatedly divides the data in half, sorts each half, and combines the sorted halves into a sorted whole.
  - Divide the list into two roughly equal halves.
  - Sort the left half.
  - Sort the right half.
  - Merge the two sorted halves into one sorted list.
  - Often implemented recursively.
  - An example of a "divide and conquer" algorithm.
    - Invented by John von Neumann in 1945

Merge sort example





Height	# of Nodes	**Tot # of Nodes**
0	1	1
1	2	3
2	4	7
3	8	15

Sorting Algorithms Complexity Summary

Algorithm	Time Complexity		Space Complexity
	Best	Worst	Worst
Bubble Sort	$\Omega(n)$	$O(n^2)$	$O(1)$
Selection Sort	$\Omega(n^2)$	$O(n^2)$	$O(1)$
Insertion Sort	$\Omega(n)$	$O(n^2)$	$O(1)$
Merge Sort	$\Omega(n \log(n))$	$O(n \log(n))$	$O(n)$
Quick Sort	$\Omega(n \log(n))$	$O(n^2)$	$O(n)$

References and Useful Resources

- Merge Sort
  - https://www.geeksforgeeks.org/time-and-space-complexity-analysis-of-merge-sort/
- Time complexity of sorting algorithms
  - https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/
  - https://www.boardinfinity.com/blog/time-complexity-of-sorting-algorithms/

That’s about this lecture!