




Lecture 8: Induction and Sorting

The graphic consists of a large blue square on the left containing a white plus sign. To its right are four smaller squares arranged in a 2x2 grid: orange (top-left), green (top-right), purple (bottom-left), and red (bottom-right).

+ Today



- Reading
 - JS Ch. 6
- Objectives
 - Selection sort
 - Merge Sort

The slide features a blue plus sign followed by the word 'Today'. To the right is a vertical bar with a thin green line on the left and a wider blue section on the right. The main content is a bulleted list with blue square markers.

+ Selection Sort

| | | | | | | | |
|----|----|----|----|----|----|---|----|
| 14 | 30 | 10 | 26 | 34 | 18 | 5 | 20 |
|----|----|----|----|----|----|---|----|

| | | | | | | | |
|---|----|----|----|----|----|----|----|
| 5 | 30 | 10 | 26 | 34 | 18 | 14 | 20 |
|---|----|----|----|----|----|----|----|

| | | | | | | | |
|---|----|----|----|----|----|----|----|
| 5 | 10 | 30 | 26 | 34 | 18 | 14 | 20 |
|---|----|----|----|----|----|----|----|

| | | | | | | | |
|---|----|----|----|----|----|----|----|
| 5 | 10 | 14 | 26 | 34 | 18 | 30 | 20 |
|---|----|----|----|----|----|----|----|

| | | | | | | | |
|---|----|----|----|----|----|----|----|
| 5 | 10 | 14 | 18 | 34 | 26 | 30 | 20 |
|---|----|----|----|----|----|----|----|

1. Find smallest
2. Swap
3. Repeat

+ Selection Sort

```
/**
 * Sorts an integer array using iterative selection sort
 * @param array array of integers to be sorted
 */
private static void selectionSortIterative(int[] array) {

    for(int i = 0; i < array.length; ++i) {
        int min = indexOfSmallest(array, i);
        swap(array, i, min);
    }
}
```

+ Selection Sort (helper)

```
/**
 * @param array array of integers
 * @param startIndex valid index into array
 * @return index of smallest value in array[startIndex...n]
 */
protected static int indexOfSmallest(int[] array, int startIndex) {
    int smallest = startIndex;
    for(int i = startIndex+1; i < array.length; ++i) {
        if(array[i] < array[smallest]) {
            smallest = i;
        }
    }
    return smallest;
}
```

+ Correctness of Selection Sort using Induction (on board)

- Consider what must be true after every iteration of the for-loop in selectionSortIterative

+ Complexity of Selection sort using Induction (on board)



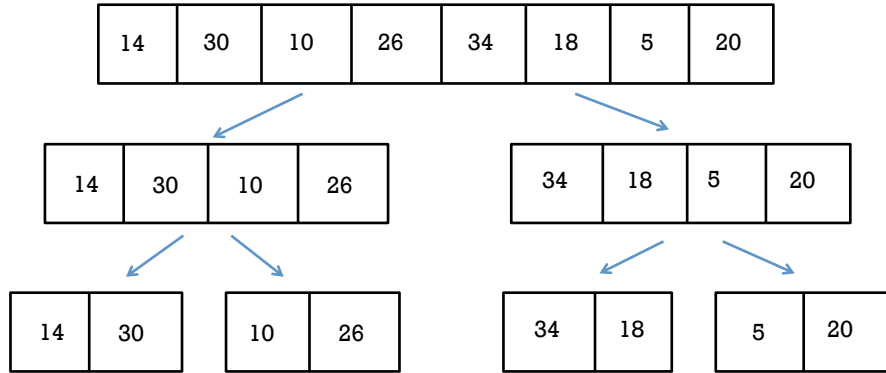
- Count the number of comparisons performed for each iteration of the for-loop in `selectionSortIterative`

+ Divide and Conquer

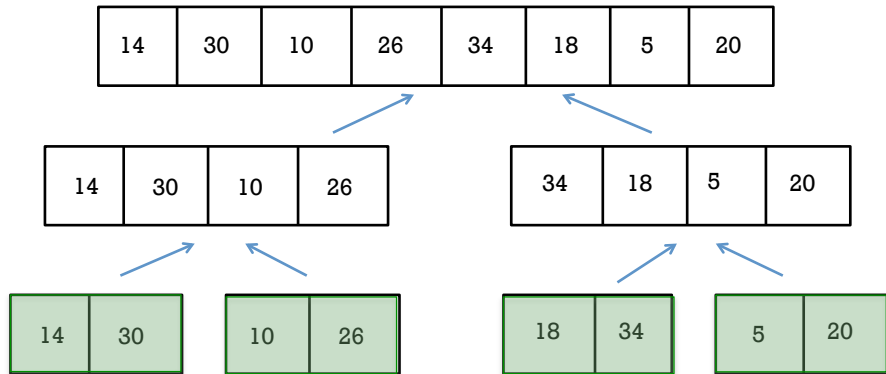


- Divide-and-conquer is a common approach for solving problems
 - Divide the problem into smaller subproblems until the subproblems are so small that the solution is trivial
 - Combine solutions to smaller subproblems to create solution to larger problem
 - Recursion!

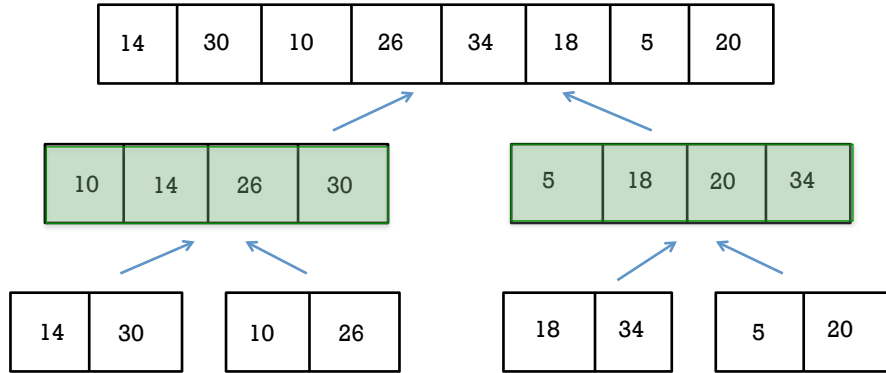
+ MergeSort



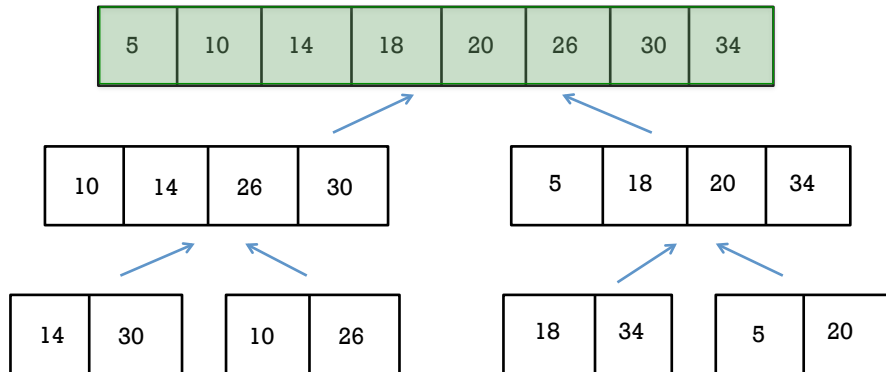
+ MergeSort



+ MergeSort



+ MergeSort



+ Complexity of Merge Sort



- Let $f(n)$ denote the number of comparisons performed by Merge Sort on an array of size n .
- Then we have the following recurrence relation:

$$f(n) = 2f(n/2) + n$$

- **Claim: $f(n) = n \log_2 n + n$**

+ Complexity of Merge Sort



- **Claim: $f(n) = n \log_2 n + n$**
- **Proof by Strong Induction**
- **Base Case**
 - Prove for $n = 1$
- **Inductive Hypothesis**
 - $f(m) = m \log_2 m + m$ **for all** $m < n$
 - Now show that $f(n) = n \log_2 n + n$