LECTURE 8: INDUCTION AND SORTING

Today

- Reading
  - JS Ch. 6 (Sorting algorithms)
- Objectives
  - Induction to prove correctness/complexity
    - Selection sort
    - Merge Sort

Induction

- Let P(n) be some proposition
- To prove P(n) is true for all n ≥ 0
  - (Step One) Base case: Prove P(n) for n = 0
  - (Step Two) Assume P(n) is true for some n = k, k ≥ 0
  - (Step Three) Use this assumption to prove P(n) for n=k+1.

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

for loop in SSI →

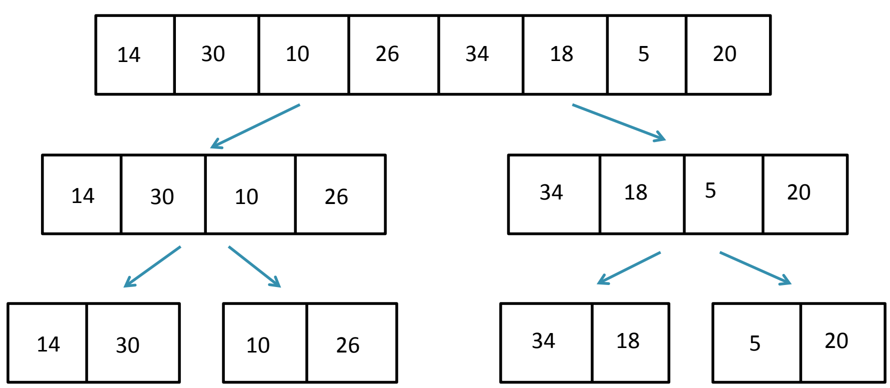
i	start	end	#comparisons
0	1	l-1	(l-1) – 1 + 1 = (l-1)
1	2	l-1	(l-1) – 2 + 1 = (l-2)
2	3	l-1	(l-1) – 3 + 1 = (l-3)
…	…	…	…
l-2	l-1	l-1	(l-1) – (l-1) + 1 = 1
l-1	l	l-1	0

$$(l-1) + (l-2) + (l-3) + \ldots + 2 + 1 = ?$$

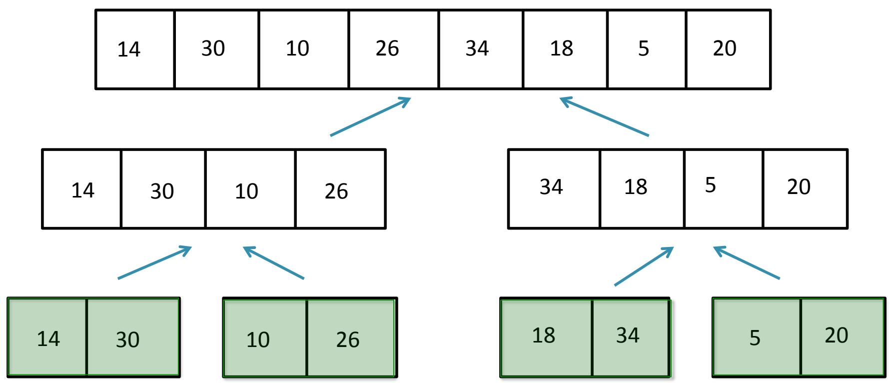
Divide and Conquer

- 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!

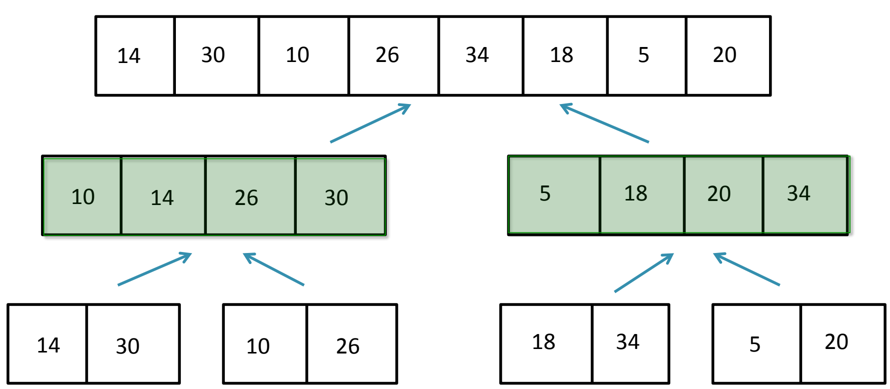
MergeSort



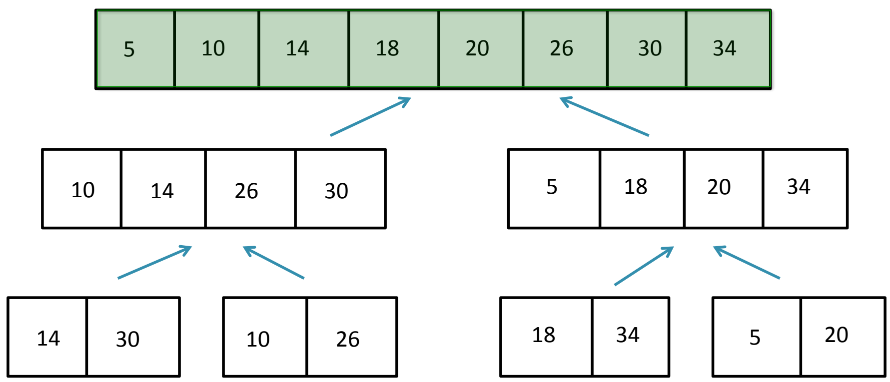
MergeSort



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

Strong Induction

- Sometimes need to assume more than just the previous case
- Assume P(n) is true ***for all n=1, 2,…, k***
- (Usually, just assume true for some n=k)

Complexity of Merge Sort

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