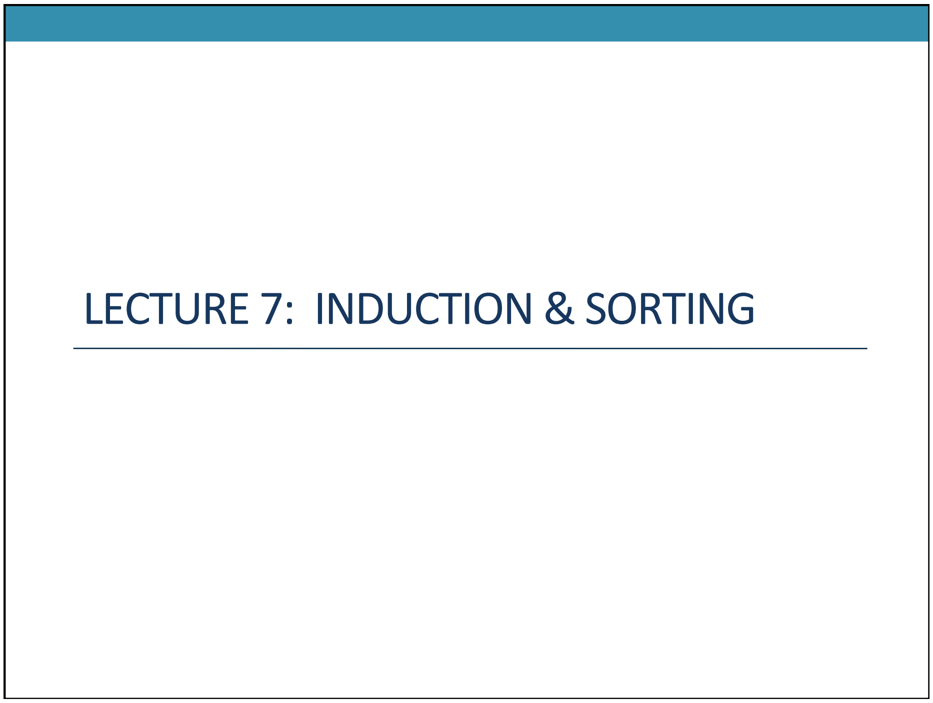
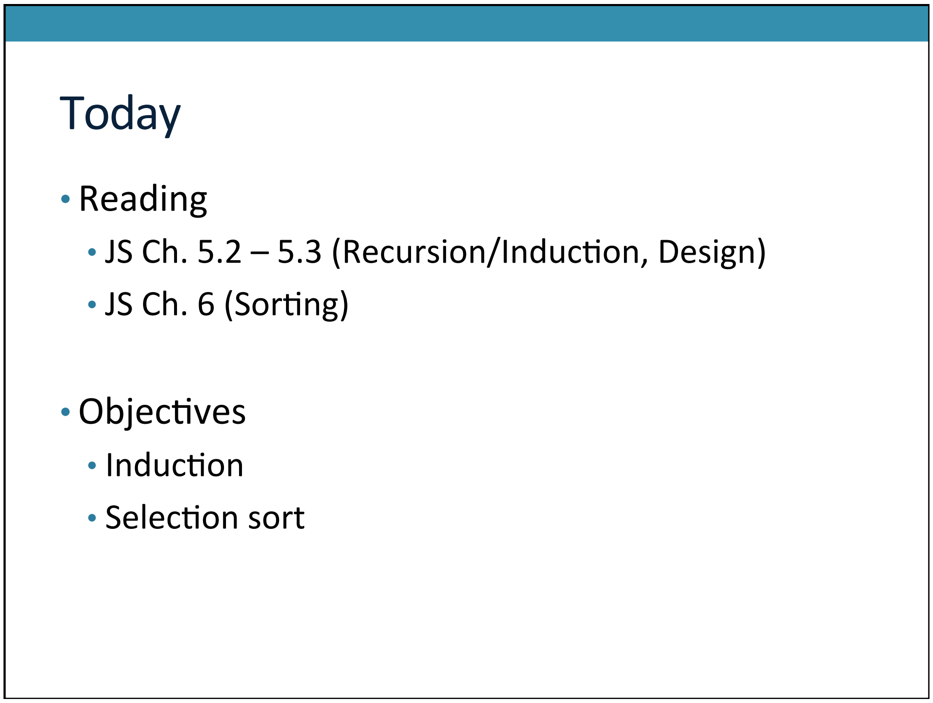


LECTURE 7: INDUCTION & SORTING



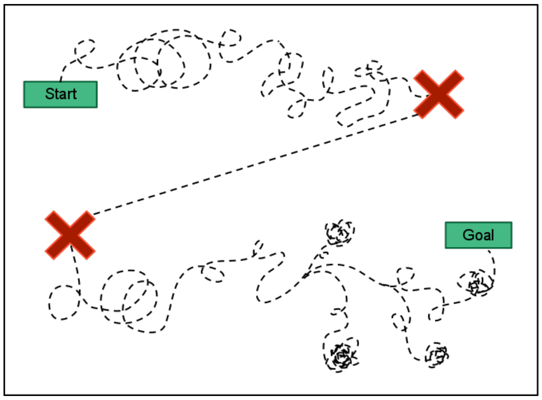
Today

- Reading
  - JS Ch. 5.2 – 5.3 (Recursion/Induction, Design)
  - JS Ch. 6 (Sorting)
- Objectives
  - Induction
  - Selection sort

How to be successful in CS062?

- Set aside enough time!

Roadmap



Announcements

- Quiz Friday on Big-O and induction
- 2/13 Harvey Mudd Career Fair
- What data structure for this week’s assgnt?

Induction

- A mathematical technique for proving
  - mathematical statements over the natural numbers
  - correctness of algorithms
- A recursive proof

Induction

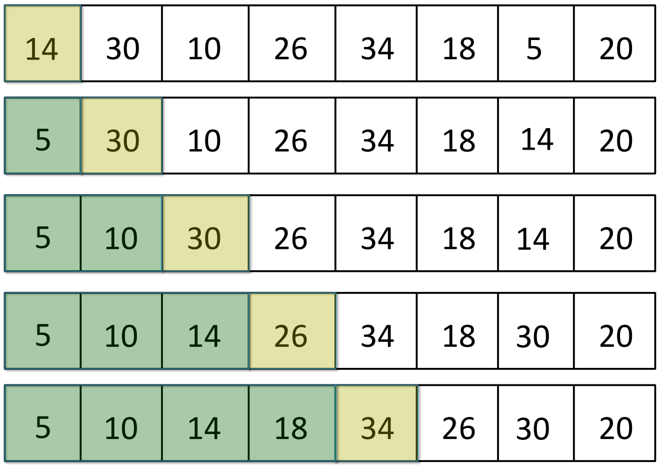
- Let P(n) be some proposition
- To prove P(n) is true for all n ≥ 0
  - (Step One) Base case: Prove P(n) for n = 0
  - (Step Two) Assume P(n) is true for any n = k, k ≥ 0
  - (Step Three) Use this assumption to prove P(n) for n=k+1.

Induction

- Mathematical Examples
  - Prove 0+1+2 + … + n = [n(n+1)]/2 for all n ≥ 0
  - Prove $2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$ for all n ≥ 0
  - Prove $2^n < n!$ for all n ≥ 4

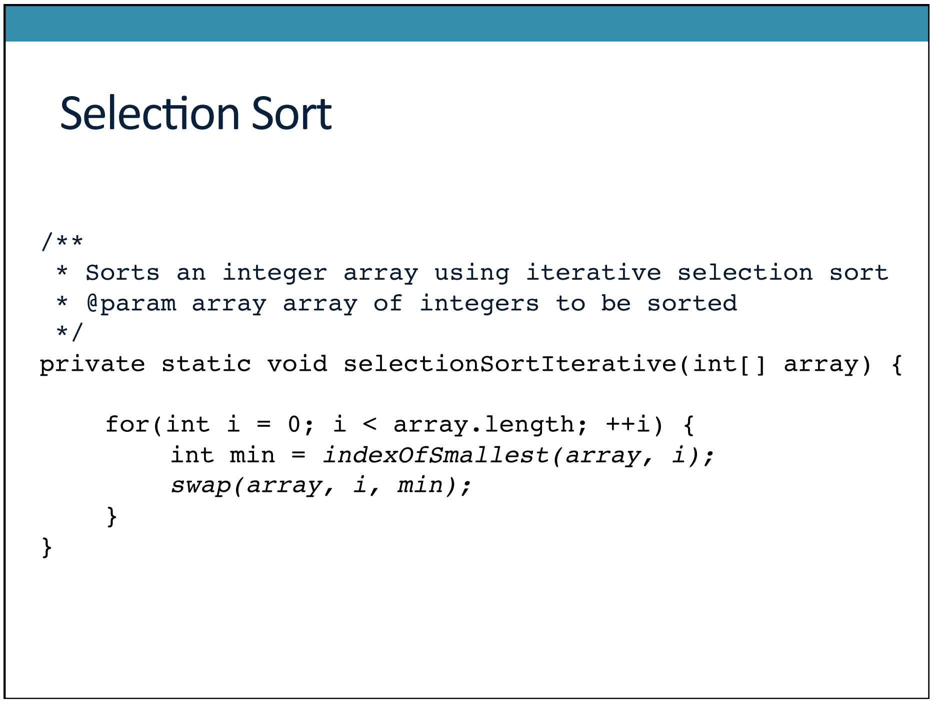
- Induction can also be used to analyze a method or algorithm

Selection Sort



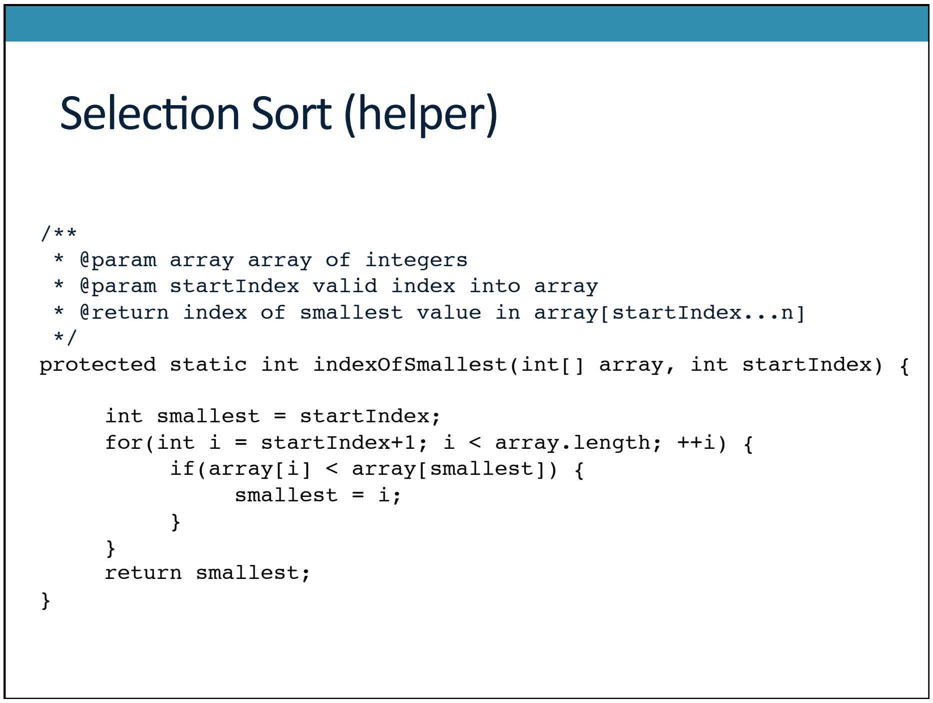
14	30	10	26	34	18	5	20
5	30	10	26	34	18	14	20
5	10	30	26	34	18	14	20
5	10	14	26	34	18	30	20
5	10	14	18	34	26	30	20

1. Find smallest
2. Swap
3. Repeat



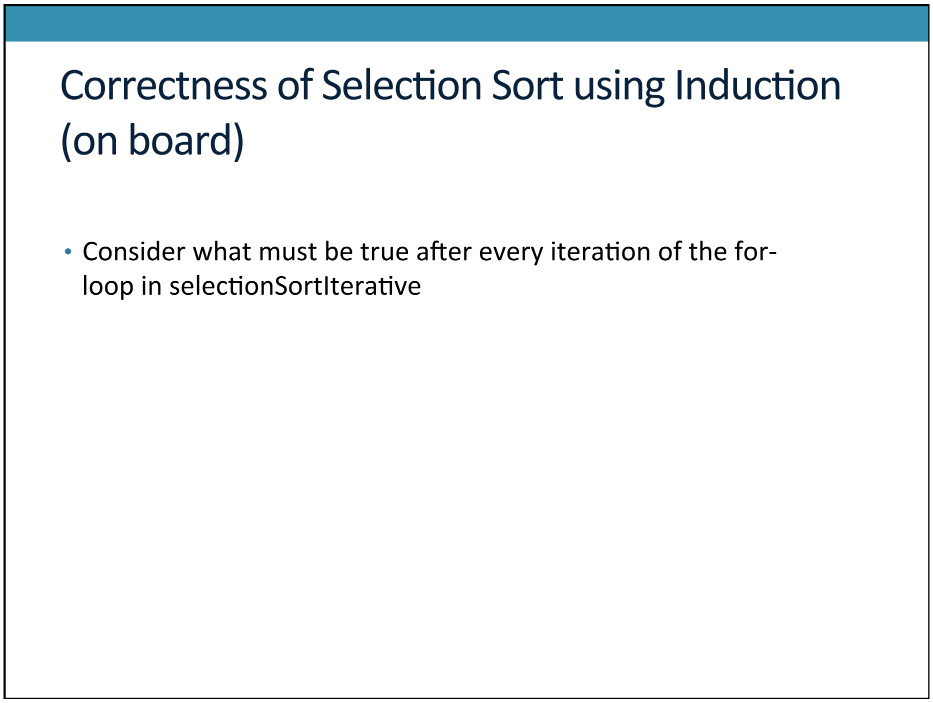
Selection Sort

```
/**
 * 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);
    }
}
```



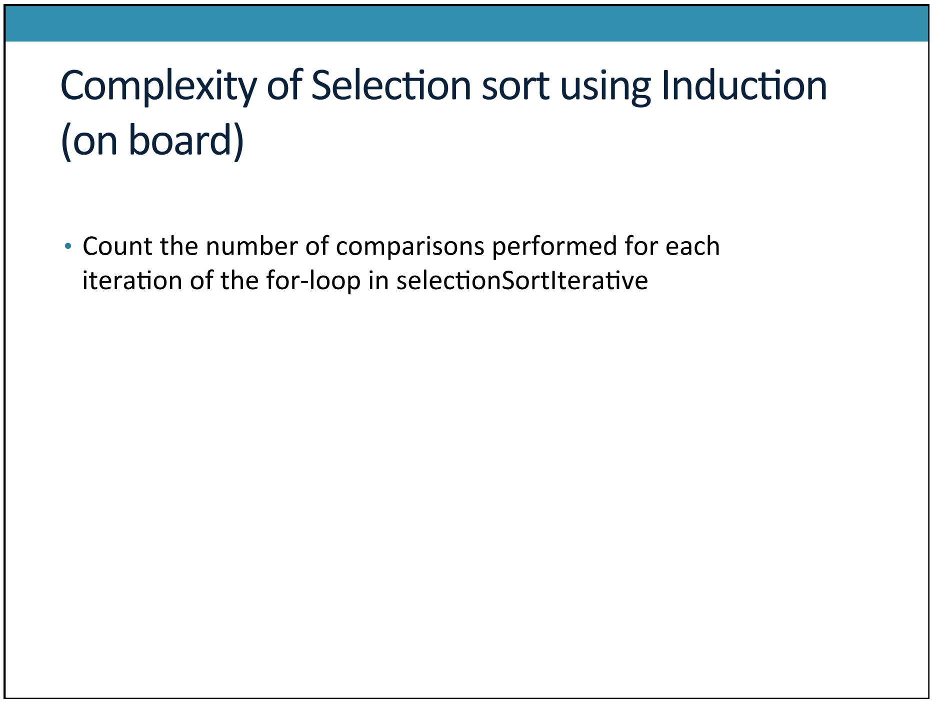
Selection Sort (helper)

```
/**
 * @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;
}
```



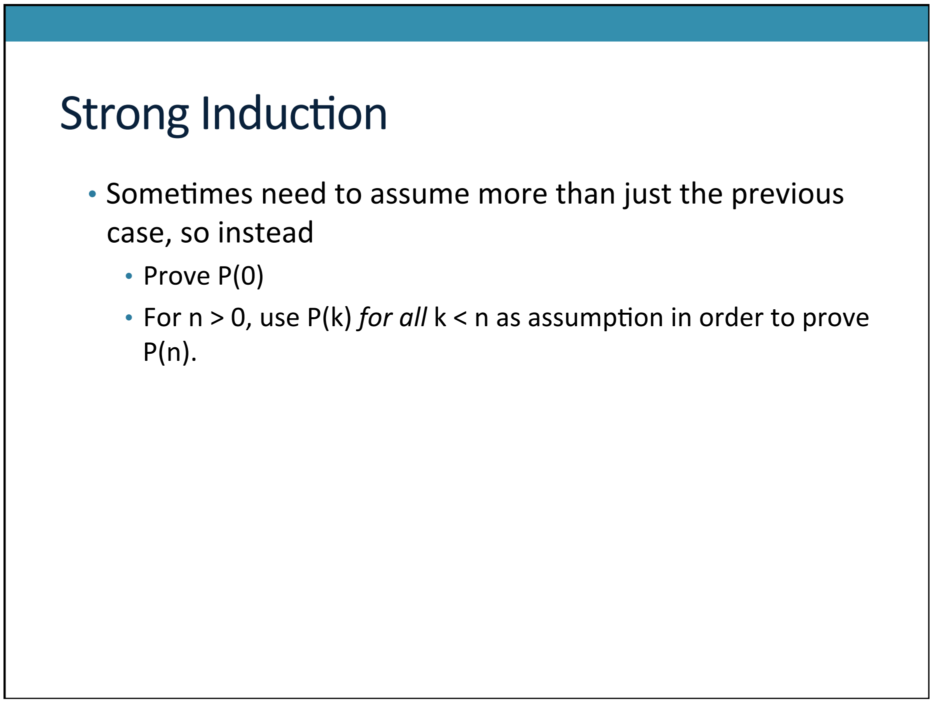
Correctness of Selection Sort using Induction (on board)

- 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



Strong Induction

- Sometimes need to assume more than just the previous case, so instead
  - Prove P(0)
  - For n > 0, use P(k) *for all* k < n as assumption in order to prove P(n).