Selection Sort

Algorithm:

Invariant: **S** is sorted.
Start with empty list **S** and unsorted list **I** of **n** items.
for(int i = 0; i < n; i++) {
X <- item in **I** with smallest key
Remove X from **I**.
Append X to end of **S**
}

Alternative Algorithm Explanation:

This is the way a child would sort a bunch of cups. Find smallest cup, put in new pile. Keep doing this until sorted.

Properties:

- O(n^2) time
  - Even in best case
- Worse than insertion sort
  - Always run in n^2 time regardless of input
- Very simple
- In-Place Algorithm
- Replace **I** with Heap = Heap Sort

Selection Sort - Spreadsheet

7	3	9	5
I			
Walk through I, find minimum item.			
Minimum Item: 3			
Swap with first item of I			
3	7	9	5
S	I		
Walk through I, find minimum item.			
Minimum Item: 5			
Swap with first item of I			
3	5	9	7
S		I	
Keep Doing This Until Sorted			

Another way to look at the more exact time complexity of this algorithm is to visualize the array and what happens at each step.

This algorithm really runs in $\geq O\left(\frac{n^2}{2}\right)$ with constants.

This is because each time you do a findMin() operation, you are reducing the next findMin() operation by one item.

Another way to look at this is:

$$\sum_{i=0}^{n-1} i$$