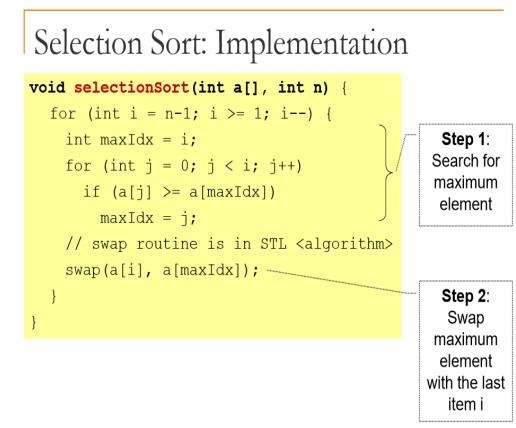
## **Selection Sort**

Selection sort is a sorting algorithm, specifically an in-place comparison sort. Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations.

## It works as follows:

- 1. Find the minimum value in the list.
- 2. Swap it with the value in the first position.

3. Repeat the steps above for remainder of the list (starting at the second position).



**Example:** sort this number (45,20,40,5,15,25,50,35,30,10) using selection sort? (ascending)

Trace of a Selection Sort								
Passes $\rightarrow$	Ι	II	III	IV	V	VI		
A[0] = 45◀	<b>05</b>	<mark>05</mark>	05	<b>05</b>	05	<b>05</b>		
A[1] = 20	<b>20</b> ⁴──	10	10	10	10	<b>10</b>		
A[2] = 40	40	<b>40</b> ◀──	15	<b>15</b>	<b>15</b>	15		
A[3] = 05 ←	45	45	45◀	20	20	20		
A[4] = 15	15	15 🚽	40	40◀	<b>25</b>	25		
A[5] = 25	25	<b>25</b>	25	25	40◀	30		
A[6] = 50	50	<b>50</b>	<b>50</b>	<b>50</b>	<b>50</b>	<b>50</b> <sup>▲</sup>		
A[7] = 35	35	<b>35</b>	35	35	35	35 <b>←</b> ┛		
A[8] = 30	30	30	30	30	<mark>30</mark> ←	40		
A[9] = 10	ل <b>ہ</b> 10	20	لـ 20	45	45	<b>45</b>		

VII	VIII	IX	Sorted Array
05	05	05	05
10	10	<b>10</b>	10
<b>15</b>	15	<b>15</b>	15
20	20	20	20
25	25	<b>25</b>	25
30	30	<b>30</b>	30
35	35	35	35
50	40	<b>40</b>	40
40	50	<b>45</b>	45
45	45 🖵	<b>50</b>	50

Selection sort is very easy to analyze since none of the loops depend on the data in the array.

Selecting the lowest element requires scanning all n elements (this takes n - 1 comparisons) and then swapping it into the first position.

Finding the next lowest element requires scanning the remaining n - 1 elements and so on, for a total of (n - 1) + (n - 2) + ... + 2 + 1 = O(n2) comparisons.

Each of these scans requires one swap for a total of n - 1 swaps (the final element is already in place).

Thus, the comparisons dominate the running time, which is O(n2).