For a collection of data, many applications need to rearrange the data in a certain order.

Problem Statement:
- Given an array of N values, arrange the values into ascending order

Best Sorting Algorithm
- The least number of comparisons
Sorting Algorithm

- Selection Sort
- Insertion Sort
- Bubble Sort
- Shell Sort
- Quick Sort
- Merge Sort
- Heap Sort
- Radix Sorting

Selection Sort

- Most naïve sorting approach
  - Simplest logic, but kind of worst performance

- Algorithm
  - There are two sub-arrays: sorted and unsorted
  - Find the smallest element in the unsorted array
  - Exchange the smallest one with the last one in the sorted array.
  - Repeat these steps until the unsorted array is empty
Selection Sort

Original Data:

| 6 | 3 | 1 | 9 | 2 |

unsorted

| 1 | 3 | 6 | 9 | 2 |

sorted unsorted

| 1 | 2 | 6 | 9 | 3 |

sorted unsorted

for (i = 0; i < N - 1; i++)
{
    min = i;
    for (j = i+1; j < N; j++)
        if (a[j] < a[min])
            min = j;
    swap(a[i], a[min]);
}
Selection Sort

- **Performance**
  - Total number of comparison

<table>
<thead>
<tr>
<th>‘i’ index</th>
<th># of comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>1</td>
<td>N-1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N-1</td>
<td>1</td>
</tr>
</tbody>
</table>

- $N + (N-1) + (N-2) + ... + 2 + 1 = \sum_{i=1}^{N} t = \frac{N(N+1)}{2}$

Insertion Sort

- Similar sorting with Selection Sort ($N^2$ time complexity), but generally perform better than Selection Sort

- **Algorithm**
  - There are two sub-arrays: sorted and unsorted
  - Given a data from unsorted array, find the right position, comparing each value in the sorted array from right to left.
  - Insert the data to the position in the sorted array
  - Repeat these steps until the unsorted array is empty
Insertion Sort

for (i = 1; i < N; i++)
{
    x = a[i];
    for (j = i; a[j - 1] > x; j--)
        a[j] = a[j - 1];
    a[j] = x;
}
Insertion Sort

- **Performance**
  - Best case: the data is already sorted
    - Linear running time, $N$
  - Worst case: the data is sorted but in reverse order
    - Quadratic running time, $N^2$
  - Average cases
    - Also quadratic, impractical for sorting large volume of data
    - Generally, fastest algorithm for small data.

Resources

- **Sorting Algorithm Animations**
  - [http://www.sorting-algorithms.com](http://www.sorting-algorithms.com)
  - [https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html](https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html)