11.11 Selection Sort

- A simple, but inefficient, sorting algorithm
- First iteration selects the smallest element in the array and swaps it with the first element
- Second iteration selects the second-smallest item (which is the smallest item of the remaining elements) and swaps it with the second element
- The algorithm continues until the last iteration selects the second-largest element and swaps it with the second-tolast index, leaving the largest element in the last index

11.11 Selection Sort (cont.)

- Consider the array

```
34 56 14 20 77 51 93 30 15 52
```

- First iteration determines the smallest element (14) of this array (at index 2), then swaps 14 with 34, resulting in

```
14 56 34 20 77 51 93 30 15 52
```

- Second iteration determines the smallest value of the remaining elements (all elements except 14), which is 15 at index 8 and swaps 15 with 56, resulting in

```
14 15 34 20 77 51 93 30 56 52
```

- Third iteration determines the next smallest value (20) and swaps it with 34.

```
14 15 20 34 77 51 93 30 56 52
```

- Process continues until the array is fully sorted

```
14 15 20 30 34 51 52 56 77 93
```

11.11.1 Selection Sort Implementation

Note: The last two lines of source code in this example have been modified from the print book so you can execute the example inside the notebook.

Function `selection_sort`

- Outer loop iterates `len(data) - 1` times
- `smallest` stores the index of the smallest element in the remaining array
- Initializes `smallest` to the current index `index1`
- Inner loop iterates over remaining elements
  - For each, compares its value to the value of the smallest element
  - If the current element is smaller than the smallest, assigns the current element's index to `smallest`
- When inner loop finishes, `smallest` will contain the index of the smallest element in the remaining array
- Next, we use tuple packing and unpacking to swap the smallest remaining element into the next ordered spot in the array

In [1]:

```
# selectionsort.py
"""Sorting an array with selection sort."""
import numpy as np
from ch11utilities import print_pass

def selection_sort(data):
    """Sort array using selection sort."""
    # loop over len(data) - 1 elements
    for index1 in range(len(data) - 1):
        smallest = index1  # first index of remaining array

        # loop to find index of smallest element
        for index2 in range(index1 + 1, len(data)):
            if data[index2] < data[smallest]:
                smallest = index2

        # swap smallest element into position
        data[smallest], data[index1] = data[index1], data[smallest]
        print_pass(data, index1 + 1, smallest)
```

Function `main`

In [2]:

```
def main():
    data = np.array([34, 56, 14, 20, 77, 51, 93, 30, 15, 52])
    print(f'Unsorted array: {data}\n')
    selection_sort(data)
    print(f'\nSorted array: {data}\n')
```

In [3]:

```
# call main to run the sort
main()
```

```
Unsorted array: [34 56 14 20 77 51 93 30 15 52]

after pass 1: 14  56  34* 20  77  51  93  30  15  52
              --
after pass 2: 14  15  34  20  77  51  93  30  56* 52
              --  --
after pass 3: 14  15  20  34* 77  51  93  30  56  52
              --  --  --
after pass 4: 14  15  20  30  77  51  93  34* 56  52
              --  --  --  --
after pass 5: 14  15  20  30  34  51  93  77* 56  52
              --  --  --  --  --
after pass 6: 14  15  20  30  34  51* 93  77  56  52
              --  --  --  --  --  --
after pass 7: 14  15  20  30  34  51  52  77  56  93*
              --  --  --  --  --  --  --
after pass 8: 14  15  20  30  34  51  52  56  77* 93
              --  --  --  --  --  --  --  --
after pass 9: 14  15  20  30  34  51  52  56  77* 93
              --  --  --  --  --  --  --  --  --

Sorted array: [14 15 20 30 34 51 52 56 77 93]
```

11.11.2 Utility Function print_pass

```
# ch11utilities.py
"""Utility function for printing a pass of the
insertion_sort and selection_sort algorithms"""

def print_pass(data, pass_number, index):
    """Print a pass of the algorithm."""
    label = f'after pass {pass_number}: '
    print(label, end='')

    # output elements up to selected item
    print(' '.join(str(d) for d in data[:index]),
          end=' ' if index != 0 else '')

    print(f'{data[index]}* ', end='') # indicate swap with *

    # output rest of elements
    print(' '.join(str(d) for d in data[index + 1:len(data)]))

    # underline elements that are sorted after this pass_number
    print(f'{" " * len(label)}{"-- " * pass_number}')
```

11.11.3 Big O of the Selection Sort

- Selection sort runs in O(n^2) time
- Function selection_sort uses nested loops
  - The outer one iterates over the first n – 1 elements in the array, swapping the smallest remaining item into its sorted position
  - The inner loop iterates over each item in the remaining array, searching for the smallest element, executing n – 1 times during the first outer loop iteration, n – 2 times during the second, then n – 3, …, 3, 2, 1
- The inner loop will iterate a total of $n(n – 1)/2$ or $(n^2 – n)/2$
- In Big O, smaller terms drop out, and constants are ignored, leaving O(n^2)