Insertion Sort

Insertion Sort Algorithm (ascending)

- Check element (store in temp variable)
- If larger than the previous element, leave it
- If smaller than the previous element, shift previous larger elements down until you reach a smaller element (or beginning of array). Insert element.

Insertion Sort Algorithm (ascending)

- 64 54 18 87 35
  - 54 less than 64
  - Shift down and insert 54
- 54 64 18 87 35
  - 18 less than 64
  - 18 less than 54
  - Shift down and insert 18
- 18 54 64 87 35
  - 87 greater than 64
  - Go to next element
- 18 54 64 87 35
  - 35 less than 87
  - 35 less than 64
  - 35 less than 54
  - 35 greater than 18
  - Shift down and insert 35
- 18 35 54 64 87

Insertion Sort Example(ascending)

- Write out each step as you sort this array of 7 numbers (in ascending order)
- 72 4 17 5 5 64 55
- 4 72 17 5 5 64 55
- 4 17 72 5 5 64 55
- 4 5 17 72 5 64 55
- 4 5 5 17 72 64 55
- 4 5 5 17 64 72 55
- 4 5 5 17 55 64 72

Shifting

- Need a temporary variable to store a value while we shift.

  value = arr[j];
- ```
  while (j > 0 && arr[j-1] >= value)
  {
      arr[j] = arr[j-1];
      j--;
  }
  ```

Insertion Sort Code (ascending)

```
public static void insertionSort(int[] arr) {
    int j = 0;
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        j = i;
        while (j > 0 && arr[j-1] >= value) {
            arr[j] = arr[j-1];    //shift
            j--;
        }
        arr[j] = value;  //insert
    }
}
```
```

Insertion Sort Algorithm (descending)

- Check element (store in temp variable)
- If smaller than the previous element, leave it
- If larger than the previous element, shift previous smaller elements down until you reach a larger element (or beginning of array). Insert element.

Insertion Sort Algorithm (descending)

- 58 79 87 57 81
  - 79 greater than 58
  - Shift down and insert 79
- 79 58 87 57 81
  - 87 greater than 58
  - 87 greater than 79
  - Shift down and insert 87
- 87 79 58 57 81
  - 57 less than 58
  - Go to next element
- 87 79 58 57 81
  - 81 greater than 57
  - 81 greater than 58
  - 81 greater than 79
  - 81 less than 87
  - Shift down and insert 81
- 87 81 79 58 57

Insertion Sort Example(descending)

- Write out each step as you sort this array of 7 numbers (in descending order)
- 72 4 17 5 5 64 55
- 72 4 17 5 5 64 55
- 72 17 4 5 5 64 55
- 72 17 5 4 5 64 55
- 72 17 5 5 4 64 55
- 72 64 17 5 5 4 55
- 72 64 55 17 5 5 4

Insertion Sort Code (descending)

```
public static void insertionSort(int[] arr) {
    int j = 0;
    for (int i = 1; i < arr.length; i++) {
        int value = arr[i];
        j = i;
        while (j > 0 && arr[j-1] <= value) {
            arr[j] = arr[j-1];    //shift
            j--;
        }
        arr[j] = value;  //insert
    }
}
```

Insertion Sort Efficiency

- n^2 *comparisons*
  - n is the number of elements in array
- $O(n^2)$ *time complexity*
  - Big O notation, will talk about this later
- Inefficient for large arrays

Why use it?

- Has almost all the advantages of selection sort
- Very efficient when data is close to being in order already
  - Time complexity becomes O(n)
- Very inefficient when data is really out of order (i.e. reversed)