# Input: An array A and indices i and j.

# Output: An array where A[i] and A[j] have been swapped.

def Swap(A, i, j):

    temp = A[i]

    A[i] = A[j]

    A[j] = temp

# Input: An array A of integers.

# Output: Array A sorted in increasing order.

def SelectionSort(A):

    compared\_count = 0

    swap\_count = 0

    # Starts from the end of the list and goes to the beggining, stepping by -1 each time.

    for i in range(len(A)-1, 0, -1):

        m = i

        for j in range(i-1, -1, -1):

            # Checks if A[j] is larger than A[m] instead of smaller.

            if A[j] > A[m]:

                m = j

            # Adds one for each comparison made.

            compared\_count += 1

        Swap(A, i, m)

        # Adds one for each swap made.

        swap\_count += 1

        print(A)

        print(f"Compared count: {compared\_count}")

        print(f"Swapped count: {swap\_count}")

        # Resets variables to zero each time through the loop.

        compared\_count = 0

        swap\_count = 0

A1 = [63, 44, 17, 77, 20, 6, 99, 84, 52, 39]

A2 = [84, 52, 39, 6, 20, 17, 77, 99, 63, 44]

A3 = [99, 84, 77, 63, 52, 44, 39, 20, 17, 6]

SelectionSort(A1)

print("\n")

SelectionSort(A2)

print("\n")

SelectionSort(A3)

Output:

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

# Input: An array A and indices i and j.

# Output: An array where A[i] and A[j] have been swapped.

def Swap(A, i, j):

 temp = A[i]

 A[i] = A[j]

 A[j] = temp

# Input: An array A of integers.

# Output: An array A sorted in increasing order.

def BubbleSort(A):

    # Initializes all counts to zero.

    total\_comparisons = 0

    total\_swaps = 0

    compared\_count = 0

    swap\_count = 0

    for i in range(len(A)-1):

        # Creates a flag for when the array is sorted.

        swapped = False

        for j in range(len(A)-i-1):

            if A[j+1] < A[j]:

                Swap(A, j+1, j)

                # Counts each time a swap is made.

                swap\_count += 1

                # If a swap is made this will stay true.

                swapped = True

            # Counts each comparison made.

            compared\_count += 1

        # Counts the total swaps and comparisons throughout the sorting.

        total\_comparisons += compared\_count

        total\_swaps += swap\_count

        print(A)

        print(f"Compared count: {compared\_count}")

        print(f"Swapped count: {swap\_count}")

        # Resets variables to zero each time through the loop.

        compared\_count = 0

        swap\_count = 0

        # If there is no swap this will breat out of the loop and the array will be sorted.

        if not swapped:

            break

    print(f"Total compared: {total\_comparisons}")

    print(f"Total swapped: {total\_swaps}")

A4 = [44, 63, 77, 17, 20, 99, 84, 6, 39, 52]

A5 = [52, 84, 6, 39, 20, 77, 17, 99, 44, 63]

A6 = [6, 17, 20, 39, 44, 52, 63, 77, 84, 99]

BubbleSort(A4)

print("\n")

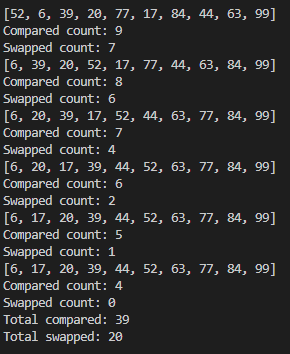
BubbleSort(A5)

print("\n")

BubbleSort(A6)

Output:

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

# Input: An x value and the p power it is being raised to.

# Output: returns a calculation of x\*x p times.

def power(x, p):

    ans = 1

    for i in range(p):

        # Starts at 1 then multiplies x by itself p times when the function is called.

        ans \*= x

    return ans

# Input: An array A of integers in a polynomial and a given x value.

# Output: A solved polynomial where the index represented the power of x.

def evaluate(A, x):

    polynomial\_length = len(A)

    # Initializes the solved value to zero.

    solved = 0

    for i in range(polynomial\_length):

        # For the length of the array x is raised to the index of A.

        power\_raised = A[i] \* power(x, i)

        # Adds power raised to solved as it iterates through the array.

        solved += power\_raised

    return f"{solved:.2f}"

A = [12.3, 40.7, -9.1, 7.7, 6.4, 0, 8.9]

x = 5.4

print(evaluate(A, x))

Output:

227295.86

Text

Description automatically generated

Asymptotic Analysis:

Line Cost Count

4 c1 1

5 c2 n

7 c3 n

8 c4 1

12 c5 1

14 c6 1

15 c7 n

17 c8 n

19 c9 n

20 c10 1

T(n) = (c2 + c3 + c7 + c8 + c9)n + (c1 + c4 + c5 + c6 + c10)1

<= c11n + c12\*1

= c13n

T(n) = O(n)