ISTA 350 Big O Worksheet I Name:

What is the input size in terms of n (and possibly m, l, …), of the following data containers?

A CSV file with 11,291 lines, each line containing 17 comma-separated fields:

A 3D array whose lengths in the 3 dimensions are 10, 4, and 5:

A 3D array whose lengths in the 3 dimensions are i, j, and k:

A dictionary whose values are lists:

A `DataFrame`:

A `Series`:

What is n of the following Python functions? What are the big O's?

```
def weird(arr):
    ''' arr is a 1D array '''
    total = 0
    for x in arr:
        total += x
    for x in arr:
        total += x**2
    for x in arr:
        total += x**3
```

```
def weird(arr):
    ''' arr is a 2D array '''
    total = 0
    for r in range(len(arr)):
        for c in range(len(arr[0])):
          total += arr[r, c]**3
```

```
def is_square(n):
    for i in range(n + 1):
        if i * i == n:
            return True
        elif i * i > n:
            return False
```

```
def products(n):
    result = np.empty(n, n)
    for i in range(n):
        for j in range(n):
            result[i, j] = i * j
```

What is the big O of the following mathematical functions?

$f(n) = 7n + 4 + 24n^4$

$f(n) = \log(n) + 4 + 3n\log(n)$

$f(n) = n + \log(n)$

$f(n) = 12$

Sorting is a fundamental computer science task. Sorts (sorting algorithms) fall mostly into two categories: simple sorts and efficient sorts. An example of a simple sort is bubble sort. Bubble sort is so named because the largest value "bubbles up" to the end of the sequence to be sorted during the first pass. The second largest bubbles up to the second to the last position during the second pass etc. Here is some pseudocode for the algorithm:

```
outer loop: the loop control variables starts at the length of
the list and decrements down to 1.
     inner loop: the loop control variable starts at 1 and
     increments up to the value of the outer loop control
     variable.
          if the value at the position of the inner loop control
          variable minus one is greater than that at the next
          position, switch them.
```

Implement this algorithm. Then do your best to determine the big O of your routine. Assume the worst case (that the `if` executes every time). State your reasoning.