Insertion Sort

Algorithm:

Invariant (Property Never Changed): S is sorted
Start with empty list S and unsorted list I of n items
For every item X in I
Insert X into S, in sorted order

Properties:

- O(n^2) Time
- If S is
  - Array
    - In-Place Algorithm
    - Binary Search
      - Find position in log(n) time
      - Take all items to right and shift over
      - Still takes theta(n) time
  - Linked List
    - Theta(n) worst case time to find right position
  - Balanced Binary Search Tree
    - Find item in log(n) time
    - Insert item in log(n) time
    - O(n log(n)) Time
      - Slower than other O(n log(n)) because of hidden constants
- List already almost sorted and can run much faster than quad. time
  - Requires right implementation:
    - When inserting into S
      - Walk backwards until right position found
      - Then Stop
  - Running time proportional to # of inversions
    - Inversion = Pair of # where smaller number comes after bigger number (descending)
- List already sorted runs in O(n) time
  - Since walking backwards
    - Only verifying each element
    - Results in O(n) time
      - Each insertion step only takes O(1) time

7	3	9	5
I			
Partition Array Into Two Separate Lists			
7	3	9	5
S	I		
For this next step, take the "3", and insert it into the correct position. In this case, swap the 3 and 7.			
3	7	9	5
S		I	
For this next step, take the "9", and insert it into the correct position. Already in the correct position			
3	7	9	5
S			I
For this next step, take the "5", and insert it into the correct position. In this case, swap the (5,9), (5,7).			
3	5	7	9
Sorted!			

Insertion Sort - Spreadsheet