**CS241 Sorting Algorithm Worksheets**

**Bubble Sort**: By swapping adjacent values, move the highest value to the right. Repeat this process to move the second highest value to the right. Continue until the smallest value remains on the left in the first position.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

swap\_pos = 0 (go right) sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 5 | 4 | 3 |

swap\_pos = 1 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 4 | 5 | 3 |

swap\_pos = 2 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 4 | 3 | 5 |

swap\_pos = 3 sort\_pos = 4 (go left)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 3 | 5 |

swap\_pos = 0 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 3 | 5 |

swap\_pos = 1 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

swap\_pos = 2 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

swap\_pos = 0 sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

swap\_pos = 1 sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

swap\_pos = 0 sort\_pos = 1

for \_\_\_\_\_sort\_pos\_\_\_\_\_\_ in \_\_\_\_range(len(numbers) - 1, 0, -1)\_\_\_\_\_ :

for \_\_\_\_\_swap\_pos\_\_\_\_\_ in \_\_\_\_\_range(sort\_pos)\_\_\_\_\_\_\_\_\_\_\_\_:

if numbers[swap\_pos] > numbers[swap\_pos + 1]:

# Swap the 2 numbers

**Performance: O(\_N^2\_)**

Extra Practice:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 3 | 4 | 9 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 7 | 4 | 9 | 1 |

swap\_pos = 0 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 9 | 1 |

swap\_pos = 1 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 9 | 1 |

swap\_pos = 2 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 1 | 9 |

swap\_pos = 3 sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 1 | 9 |

swap\_pos = 0 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 1 | 9 |

swap\_pos = 1 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 1 | 7 | 9 |

swap\_pos = 2 sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 1 | 7 | 9 |

swap\_pos = 0 sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 1 | 4 | 7 | 9 |

swap\_pos = 1 sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 9 |

swap\_pos = 0 sort\_pos = 1

**Selection Sort**: Search the list for the highest number and swap it so it’s to the right. Repeat this process to find the second highest value and swap it so it’s to the to the right. Continue until the smallest value remains on the left in the first position.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 |

sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 5 |

sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 5 |

sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

sort\_pos = 1

**Performance: O(\_N^2\_)**

Extra Practice:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 3 | 4 | 9 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 3 | 4 | 1 | 9 |

sort\_pos = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 9 |

sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 9 |

sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 9 |

sort\_pos = 1

**Insertion Sort**: Assume the left is sorted. Starting with the second element, swap to the left to insert the number in the correct order. Continue for all numbers to the right.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

Think sorted right place

Don’t need to be swap to get into the box

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

sort\_pos = 1

1 not in right place 5

1 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 4 | 3 |

sort\_pos = 2

4 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 5 | 3 |

sort\_pos = 3

3 5

3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

sort\_pos = 4

**Performance: O(\_N^2\_)**

Extra Practice:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 3 | 4 | 9 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 7 | 4 | 9 | 1 |

sort\_pos = 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 9 | 1 |

sort\_pos = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | 4 | 7 | 9 | 1 |

sort\_pos = 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 7 | 9 |

sort\_pos = 4

**Merge Sort**: Recursively split up the list into smaller lists. A list of size 1 is already sorted. After splitting up the list, merge it back together again by merging two small sorted lists together. Merging two sorted lists is not difficult:

1 3 5 9 merge with 4 6 10 12

Smallaest of the smallest

Left position right position

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 4 | 5 | 6 | 9 | 10 | 12 |

Use Merge Sort to sort:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2 | 5 | 1 |  | 4 | 3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 |  | 1 |  | 4 |  | 3 |

MERGE

SPLIT

|  |  |  |
| --- | --- | --- |
| 2 |  | 5 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 |  | 1 |  | 4 |  | 3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 5 |  | 3 | 4 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |

**Performance: O(\_N \* log\_N\_\_\_\_\_\_)**

Extra Practice:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 1 | 7 | 2 | 5 | 9 | 3 | 8 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

SPLIT

MERGE

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |