

20171248 안재형

CAUSWE 2021

Algorithm Course - Class#2

(Prof. Eunwoo Kim)

# Midterm Assignment Report

## Codes

### Time\_Checker.py

```
import time, math, sort_bubble, sort_bucket, sort_insertion, sort_merge,
sort_quick, sort_radix

def printtime(label, time):
    print(format(time, 'f'))

arr = list(range(100, 0, -1))

start = time.time()
sort_bubble.bubblesort(arr)
printtime('bubble', time.time()-start)

# start = time.time()
# result = sort_insertion.insertionsort(arr)
# printtime('insertion', time.time()-start)

# start = time.time()
# sort_merge.mergesort(arr)
# printtime('merge', time.time()-start)

# start = time.time()
# sort_quick.quicksort(arr)
# printtime('quick', time.time()-start)

# start = time.time()
# sort_radix.radixsort(arr)
# printtime('radix', time.time()-start)

# start = time.time()
# sort_bucket.bucketsort(arr, 10)
```

## **bubble sort**

```
def bubblesort(arr):  
    result = arr  
    for i in range(0, len(arr) - 1):  
        for j in range(0, len(arr) - i - 1):  
            if(result[j] > result[j+1]):  
                temp = result[j]  
                result[j] = result[j+1]  
                result[j+1] = temp  
    return result
```

## **insertion sort**

```
def insertionsort(arr):  
    result = arr  
    for i in range(1, len(result)):  
        for j in range(i, 0, -1):  
            if(result[j-1] < result[j]):  
                break  
            else:  
                temp = result[j]  
                result[j] = result[j-1]  
                result[j-1] = temp  
    return result
```

## Merge sort

```
def mergesort(arr):
    if(len(arr) == 1):
        return arr
    else:
        left = arr[:round(len(arr)/2)]
        right = arr[round(len(arr)/2):]

        left = mergesort(left)
        right = mergesort(right)

        resultArr = []
        l = r = 0
        for _ in range(0, len(arr)):

            if(r >= len(right)):
                resultArr.append(left[l])
                l += 1
            if(l >= len(left)):
                for _ in range(r, len(right)):
                    resultArr.append(right[r])
                    r += 1
                break

            else:
                resultArr.append(right[r])
                r += 1
            if(r >= len(right)):
                for _ in range(l, len(left)):
                    resultArr.append(left[l])
                    l += 1
                break
        return resultArr
```

## Quick Sort

```
def quicksort(arr):
    if(len(arr) <= 1):
        return arr
    else:
        x = round(len(arr)/2)
        left = []
        right = []

        for i, i_value in enumerate(arr):
            if(i == x):
                continue
            else:
                if(i_value <= arr[x]):
                    left.append(i_value)
                else:
                    right.append(i_value)
        return quicksort(left) + [arr[x]] + quicksort(right)
```

## Radix Sort

```
def radixsort(arr):
    maxval = max(arr)
    exponent = 1

    result = arr

    while(maxval / exponent >= 1):
        result = countsort(result, exponent)
        exponent *= 10
    return result

def countsort(arr, exponent):
    count = [0,0,0,0,0,0,0,0,0,0,0]
    result = [0 for i in range(len(arr))]

    for i in arr:
        index = round(((i % (exponent * 10)) - (i % (exponent))) / exponent)
        count[index] += 1

    for i, _ in enumerate(count[::-1]):
        count[i+1] += count[i]

    for i in reversed(arr):
        index = round(((i % (exponent * 10)) - (i % (exponent))) / exponent)
        count[index] -= 1
        result[count[index]] = i

    return result
```

## Bucket Sort

```
import math
def bucketsort(arr, buckets):
    result = [[] for i in range(buckets)]
    arr_min = min(arr)
    arr_max = max(arr)
    arr_range = arr_max - arr_min
    bucket_range = arr_range/buckets

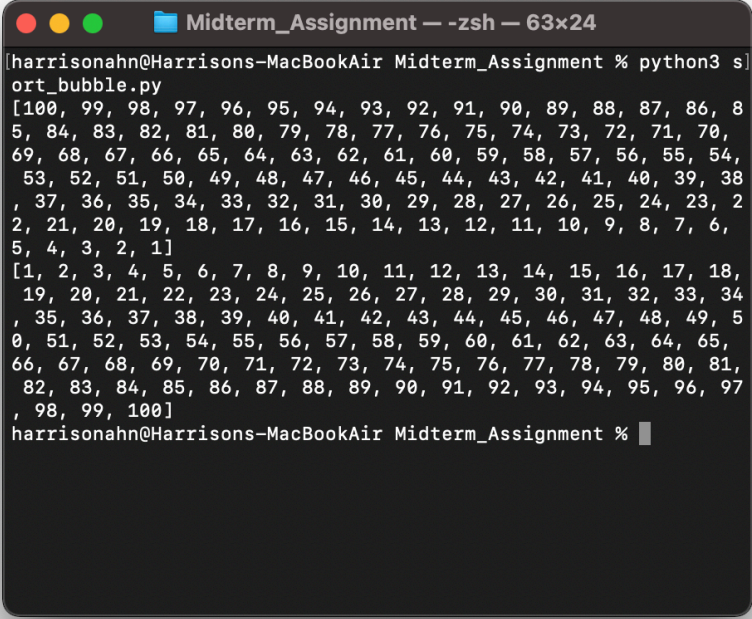
    for i in arr:
        index = math.floor((i-arr_min)/bucket_range)
        if(index == buckets):
            index -= 1
        insert_index = len(result[index])
        for j_index, j in enumerate(result[index]) :
            if(j >= i):
                insert_index = j_index
                break
        result[index].insert(insert_index, i)

    result_return = []
    for i in result:
        result_return += i

    return result_return
```

## Input/Output

### Bubble Sort



A terminal window titled "Midterm\_Assignment — -zsh — 63x24" showing the execution of a Python script named "ort\_bubble.py". The script takes a list of 100 numbers as input and outputs the sorted list. The numbers are sorted in ascending order, from 1 to 100.

```
[harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 s]
ort_bubble.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 8
5, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70,
69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54,
53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38
, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 2
2, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6,
5, 4, 3, 2, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34
, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 5
0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65,
66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81,
82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97
, 98, 99, 100]
harrisonahn@Harrisons-MacBookAir Midterm_Assignment %
```

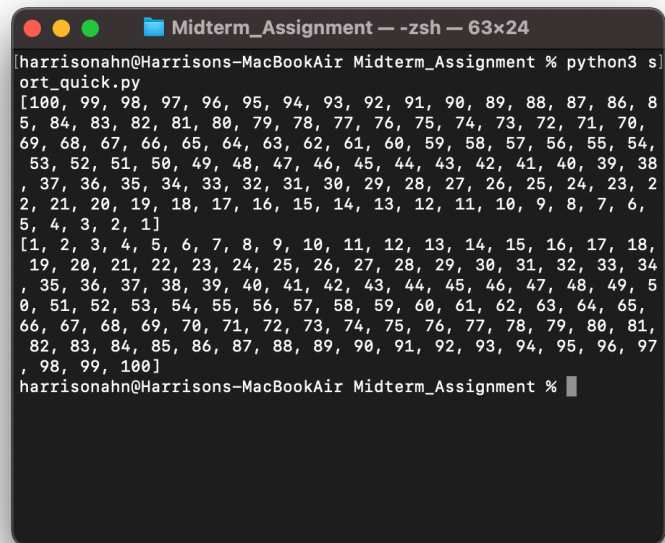
## Insertion Sort

```
Midterm_Assignment --zsh-- 63x24
harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 sort_insertion.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
harrisonahn@Harrisons-MacBookAir Midterm_Assignment %
```

## Merge Sort

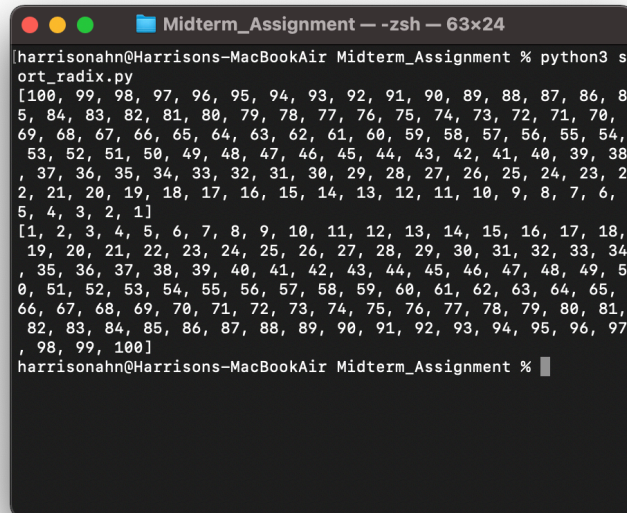
```
Midterm_Assignment --zsh-- 63x24
harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 sort_merge.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
harrisonahn@Harrisons-MacBookAir Midterm_Assignment %
```

## Quick Sort

A terminal window titled "Midterm\_Assignment -- zsh -- 63x24" showing the execution of a Quick Sort program. The user runs "python3 sort\_quick.py" and the program outputs a list of 100 numbers, which are already sorted in ascending order. The list starts with 100, 99, 98, ..., 1.

```
[harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 sort_quick.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[harrisonahn@Harrisons-MacBookAir Midterm_Assignment % ]
```

## Radix Sort

A terminal window titled "Midterm\_Assignment -- zsh -- 63x24" showing the execution of a Radix Sort program. The user runs "python3 sort\_radix.py" and the program outputs a list of 100 numbers, which are already sorted in ascending order. The list starts with 100, 99, 98, ..., 1.

```
[harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 sort_radix.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[harrisonahn@Harrisons-MacBookAir Midterm_Assignment % ]
```

## Bucket Sort

```
Midterm_Assignment --zsh-- 63x24
harrisonahn@Harrisons-MacBookAir Midterm_Assignment % python3 sort_bucket.py
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
harrisonahn@Harrisons-MacBookAir Midterm_Assignment %
```

## Result Table

	Bubble	Insertion	Merge	Quick	Radix	Bucket
100	0.001051	0.000720	0.000305	0.000184	0.000239	0.000105
1000	0.075248	0.069590	0.003497	0.001780	0.002269	0.000795
10000	6.08586	6.097266	0.032284	0.020772	0.030535	0.008429

harrisonahn@Harrisons-MacBookAir Midterm\_Assignment % python3 sort\_bucket.py  
0.001051  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000720  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000305  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000184  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000239  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000105  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %

harrisonahn@Harrisons-MacBookAir Midterm\_Assignment % python3 sort\_bucket.py  
0.075248  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.069590  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.003497  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.001780  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.002269  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.000795  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %

harrisonahn@Harrisons-MacBookAir Midterm\_Assignment % python3 sort\_bucket.py  
6.08586  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
6.097266  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.032284  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.020772  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.030535  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %  
0.008429  
harrisonahn@Harrisons-MacBookAir Midterm\_Assignment %