Group Exercise #4

Due 11:59PM, Monday April 12

Instructions:

1) Find which group you are in from the following list (it is re-randomized since the last group).
    a. Group 1 – Tuva, Talha, Nemed
    b. Group 2 – Garrett, Keith, Alora
    c. Group 3 – Malachi, Kevin, Jacobo
    d. Group 4 – Eddie, David M, Jonathan
    e. Group 5 – Kylie, Cale, Oksana
    f. Group 6 – Aiden, ANDY, Alejandra
    g. Group 7 – Megan, Liam, Luke
    h. Group 8 – Marshall, Hayden, David S
    i. Group 9 – Nicole, Jonah
2) Go to the discord server, introduce yourself in the channel for your group, and work out among your group who will work on which questions. Note that all channels are public.
3) Before the deadline discuss answers for each question in the group until there is consensus.
4) Create a video or videos of your solutions and upload to your channel.

Questions

1. Mystery Sort

The following code sorts an array of numbers. It uses no comparisons! We never compare two elements in the array against each other!

    a. Would you consider this a non-comparison based sorting algorithm? Why or why not?
    b. Describe how the algorithm works.
    c. Give the runtime of the algorithm.

```
import math
def magicSort(a,i,j):
   x = a[i]
   y = a[j]
   a[i] = (x + y - math.sqrt((x-y)**2))/2
   a[j] = (x + y + math.sqrt((x-y)**2))/2

def main():
   a = [10.0, 20.0, 5.0, 25.0, 3.5, 41.25, 2.5, 22.0, 12.0]
   print(a)
   for i in range(0,len(a)):
     for j in range(0,len(a)-1):
       magicSort(a,j,j+1)
   print(a)

main()
```

2. Sorting in Action

Below are the contents of an array as three algorithms are sorting it. Each line does not necessarily represent the next step in the algorithm, but only some later step during the course of the algorithm's execution. Key elements are depicted in **bold**. Identify each algorithm.

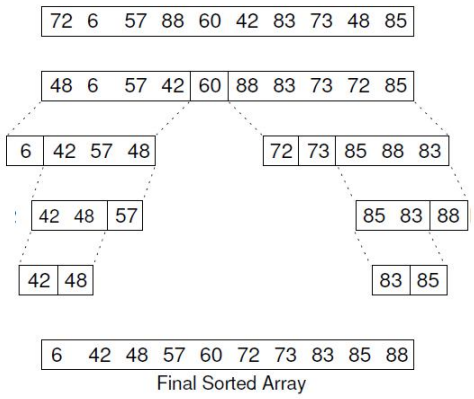
Algorithm 1

12	39	2	94	23	77	52	9
12	**39**	2	94	23	77	52	9
2	12	39	94	23	77	52	9
2	12	39	**94**	23	77	52	9
2	12	**23**	39	94	77	52	9
2	12	23	39	**77**	94	52	9
2	12	23	39	**52**	77	94	9
2	**9**	12	23	39	52	77	94

Algorithm 2

12	39	2	94	23	77	52	9
94	**39**	**77**	**12**	**23**	**2**	**52**	**9**
77	39	52	12	23	2	9	**94**
52	39	9	12	23	2	**77**	94
39	23	9	12	2	**52**	77	94
23	12	9	2	**39**	52	77	94
12	2	9	**23**	39	52	77	94
9	2	**12**	23	39	52	77	94
2	**9**	12	23	39	52	77	94

Algorithm 3



3. Radix Sort

   Watch this animation of radix sort (on the page, click the radix sort button to start the animation).

   https://www.cs.usfca.edu/~galles/visualization/RadixSort.html

   What sorting algorithm is used to sort each digit?

4. Quicksort

   Using the specific implementation of quicksort given in the colab notebook, what is the Big-O performance when all n values of the input array are the same? For example, A = [100, 100, 100, 100, 100, 100]

5. Hash Functions

   Consider the "bad" hash function for English words that was presented in the colab notebook, which simply adds up the ASCII value of each character and then computes the modulus based on the hash table size. Can we compensate for the poor hash function by making the hash table bigger? Comment on the how much this approach improves performance as the hash table size gets larger and larger.

6. Hash Code for Street Addresses

   Design a hash function for street addresses, e.g. "3211 Providence Drive". Give a persuasive argument why your hash function will be good for a variety of addresses.