DSC 40B - Discussion 02

Problem 1.

Compute the best and worst case time complexity for the following code snippets:

a)
```
def f_1(arr1, arr2):
    """`arr1` and `arr2` are two arrays each of size n"""
    n = len(arr1)
    for i in range(n):
        for j in range(n):
            if arr[i] + arr[j] == 0:
                return (i,j)
```

> **Solution:** Discussion timestamp → 1 : 10
> Best case time complexity is $\theta(1)$ when `arr[0] + arr[0] == 0`
> Worst case time complexity if $\theta(n^2)$ when no two elements in the array sum to 0.

b)
```
def insertion_sort(arr):
    """Sort `arr` in ascending order."""
    n = len(arr)
    for i in range(1, n):
        x = arr[i]
        j = i - 1
        # find where to place x
        while j >= 0 and x < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = x
```

> **Solution:** Discussion timestamp → 7 : 26
> Best case time complexity is $\theta(n)$ when all elements in the array are already sorted in increasing order.
> Worst case time complexity if $\theta(n^2)$ when all elements in the array are sorted in decreasing order.

Problem 2.

Compute the average time complexity for the following code snippet:

a)
```
def f_1(arr1, arr2):
    """`arr1` and `arr2` are two arrays each of size n"""
    n = len(arr1)
    for i in range(n):
        for j in range(n):
            if arr[i] + arr[j] == 0:
                return (i,j)
```

> **Solution:** Discussion timestamp → 21 : 20
>
> Each pair (i,j) is equally likely to be the answer. As i and j can range from 0 to (n-1), there are

n^2 possible pairs. Therefore, probability of (i, j) being the solution, $P(i,j) = \frac{1}{n^2}$.
Now, let us analyze the time taken for the algorithm to run if (i,j) is the solution (denoted as T(i, j)).

$$T(i,j) = \sum_{k=1}^{i-1} n + (j+1)$$

So, we get

$$T(0,0) = 1$$
$$T(0,1) = 2$$
$$T(0,2) = 3$$
$$\vdots$$
$$T(0,n-1) = n$$
$$T(1,0) = n+1$$
$$\vdots$$
$$T(1,n-1) = 2n$$
$$T(2,0) = 2n+1$$
$$\vdots$$
$$T(2,n-1) = 3n$$
$$\vdots$$
$$T(n-1,n-1) = n^2$$

Therefore, $\sum_{i,j} T(i,j) = \sum_{k=1}^{n^2} k = \frac{n^2(n^2+1)}{2}$.
The average time complexity is as follows:
$\sum_{i,j} P(i,j)T(i,j) = \sum_{i,j} \frac{1}{n^2} T(i,j) = \frac{1}{n^2} \sum_{i,j} T(i,j) = \frac{1}{n^2} \frac{n^2(n^2+1)}{2} = \theta(n^2)$.

Problem 3.

Provide a tight theoretical lower bound for the problems given below. Provide justification for your answer.

a) Given an array of n numbers, find the sum of the numbers in the array.

Solution: Discussion timestamp $\rightarrow 36:20$
Adding all the elements in the array requires you to visit all the elements at least once. Therefore, the lower bound is $\Omega(n)$.

b) Given a sorted array of $n \geq 2$ numbers, find the second largest number in the array.

> **Solution:** Discussion timestamp $\rightarrow 37:55$
> As the array is sorted, we just need to check the second last element in the array to get the second largest number in the array. This takes constant time. Therefore, the lower bound is $\Omega(1)$.

Problem 4.

Provide the asymptotic time complexity of the operation below using multivariate Θ notation.
Given two d-dimensional vectors $v = [v_1, v_2, ..., v_d]$ and $u = [u_1, u_2, ..., u_d]$, their sum is computed as $v + u = [v_1 + u_1, v_2 + u_2, ..., v_d + u_d]$. Given n vectors over R^d, check if any two of them sum to the zero vector using the brute force method.

> **Solution:** Discussion timestamp $\rightarrow 40:15$
> Computing the sum of two d dimensional vectors takes $\theta(d)$ time. In the brute force method, we compute the sum of every vector with every other vector.
> In the best case, we find two vectors which sum to the zero vector in the very first iteration. Therefore, the best case time complexity would be $\theta(d)$.
> In the worst case, we do not find a pair which sum to the zero vector. There are $\binom{n}{2}$ pairs. We know that $\binom{n}{2} = \theta(n^2)$. Therefore, computing the sum of $\binom{n}{2}$ pairs takes $\theta(n^2 d)$ time.
> The average time complexity is $\theta(n^2 d)$.