Brian Zhu

Professor Rayana

CSE 214.02

11 September 2017

HW #1 Problem 1 Time Complexities

1a.
```
public static boolean isPair(int[] inputArr, int sum)
{
    for (int i = 0; i < inputArr.length - 1; i++)
    {
        for (int j = i + 1; j < inputArr.length; j++)
        {
            if (inputArr[i] + inputArr[j] == sum)
            {
                return true;
            }
        }
    }

    return false;
}
```
n

((n – 1)(n + 2)) / 2

n(n - 1) / 2

1

1

The total number of operations for input array of size n in the worst-case scenario is $n + \frac{(n-1)(n+2)}{2} + \frac{n(n-1)}{2} + 2$. The time complexity in Big O-notation is $O(n^2)$.

1b.
```
public static boolean isTriplet(int[] inputArr, int sum)
{
    for (int i = 0; i < inputArr.length - 2; i++)
    {
        for (int j = i + 1; j < inputArr.length - 1; j++)
        {
            for (int k = j + 1; k < inputArr.length; k++)
            {
                if (inputArr[i] + inputArr[j] + inputArr[k] == sum)
                {
                    return true;
                }
            }
        }
    }

    return false;
}
```
n - 1

((n – 2)(n + 1)) / 2

$((n^3 – 7n + 6)) / 6$

(n-2)(n-1)(n) / 6

1

1

The total number of operations for input array of size n in the worst-case scenario is n + $\frac{(n-2)(n+1)}{2} + \frac{(n^3-7n+6)}{6} + \frac{(n-2)(n-1)(n)}{6}$ + 1. The time complexity in Big O-notation is $O(n^3)$.

1c.

```
for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            double sum = 0;
            for (int k = 0; k < n; k++) {
                sum += a[i][k] * b[k][j];
            }
            c[i][j] = sum;
        }
    }
```

n + 1

$(n^2 + n)$

n^2

$(n^3 + n^2)$

(n^3)

(n^2)

The total number of operations of input size n in closed form is $2n^3 + 4n^2 + 2n + 1$. The time complexity in Big O-notation is $O(n^3)$. The space complexity in Big O-notation is $O(n^2)$.