William Xiang
CSE214 -- HW1

September 16, 2017
Prof. Shebuti Rayana

Question 1a: Paired Sum

```java
public static boolean pairSum(int[] arr, int sum) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = i + 1; j < arr.length; j++) {
            if (arr[i] + arr[j] == sum) {
                System.out.println("Found pair: " +
                        arr[i] + " + " + arr[j] +
                        " = " + sum);
                return true;
            }
        }
    }
    return false;
}
```

This method would, in worse case scenario, run in time complexity **$O(n^2)$**, because:

- The outer for loop runs n times.
- The inner for loop runs $\sum_{i=0}^{n} n - i - 1$ times in worst case (pair is never found)
    - This series evaluates to some factor of n^2

The n^2 dominates the n with large array sizes, so the final time complexity is **$O(n^2)$**.

Question 1b: Tripled Sum

```java
public static boolean tripletSum(int[] arr, int sum) {
    for (int i = 0; i < arr.length; i++) {
        for (int j = i + 1; j < arr.length; j++) {
            for (int k = i + 2; k < arr.length; k++) {
                if (arr[i] + arr[j] + arr[k] == sum) {
                    System.out.println("Found triplet: " + arr[i] +
                            " + " + arr[j] + " + " +
                            arr[k] + " = " + sum);
                    return true;
                }
            }
        }
    }
    return false;
}
```

This method would, in worse case scenario, run in time complexity **$O(n^3)$**, because:

- We know from the previous example that the middle loop runs some factor of n^2 times in worst case
- Without doing an obscene amount of series calculations, we can logically suggest that the inner loop adds another "layer" to the iteration, and dominates the smaller n terms, resulting in **$O(n^3)$** runtime.
    - I believe the inner loop runs $\sum_{i=0}^{n}(n - i - 1 + \sum_{i=0}^{n} n - i - 2)$ times in worst case scenario

Question 1c: Matrix Multiplication

```
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;
    }
}
```

This method would run in **time** complexity **$O(n^3)$**, because:

- Outer for loop runs n times
- Middle for loop runs $n * n = n^2$ times
- Inner for loop runs $n * n * n = n^3$ times

n^3 dominates smaller n terms, so **$O(n^3)$** runtime.

This method would run in **space** complexity **$O(n^2)$**, because:

- The method accesses three $n * n$ arrays: the 2 matrices to be multiplied and the product matrix
- $3 * n * n = 3n^2 \Rightarrow$ **$O(n^2)$**