CSI 2110 Tutorial (Section A)

Yiheng Zhao
yzhao137@uottawa.ca
Office hour: Fri 13:00 – 14:00
Place: STE 5000G

Big-Oh ($O(g(n))$): upper boundary of function

$$If\ there\ exist\ C > 0\ and\ n_0 > 0\ such\ that$$
$$f(n) \leq C * g(n)\ for\ all\ n > n_0$$

Big-Omega ($\Omega(g(n))$): lower boundary of function

$$If\ there\ exist\ C > 0\ and\ n_0 > 0\ such\ that$$
$$f(n) \geq C * g(n)\ for\ all\ n > n_0$$

Big-Theta ($\Theta(g(n))$): equal to the function

$$If\ there\ exist\ C_1 > 0,\ C_2 > 0, and\ n_0 > 0\ such\ that$$
$$C_1 * g(n) \leq f(n) \leq C_2 * g(n)\ for\ all\ n > n_0$$

R-4.3:

The number of operations executed by algorithms A and B is $40n^2$ and $2n^3$, respectively. Determine n_0, such that A is better than B for $n \geq n_0$

$40n_0^2$ < $2n_0^3$ $\quad (0 \leq n_0)$

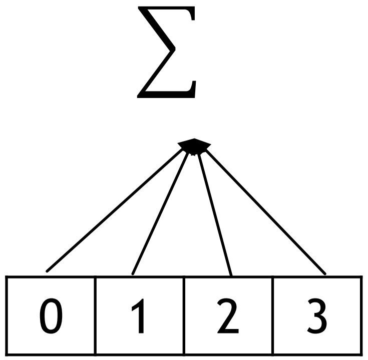
=> $20n_0^2$ < n_0^3 $\quad$ (divided by 2)

=> 20 < n_0 $\quad$ (divided by n_0^2)

R-4.9:

Give a big-Oh characterization, in terms of n, of the running time of the example1 method shown in Code Fragment.

```
/**Returns the sum of the integers in given array.*/
Public static int example1(int[] arr){
    int n = arr.length, total = 0;
    for (int j=0; j<n; j++)
        total += arr[j];
    return total;
}
```

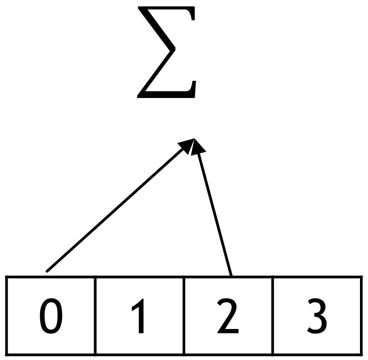


Number of computational operations: n
Big-Oh: O(n)

R-4.10:

Give a big-Oh characterization, in terms of n, of the running time of the example2 method shown in Code Fragment.

```
/**Returns the sum of the integers with even index in given array.*/
Public static int example2(int[] arr){
    int n = arr.length, total = 0;
    for (int j=0; j<n; j+=2)
        total += arr[j];
    return total;
}
```



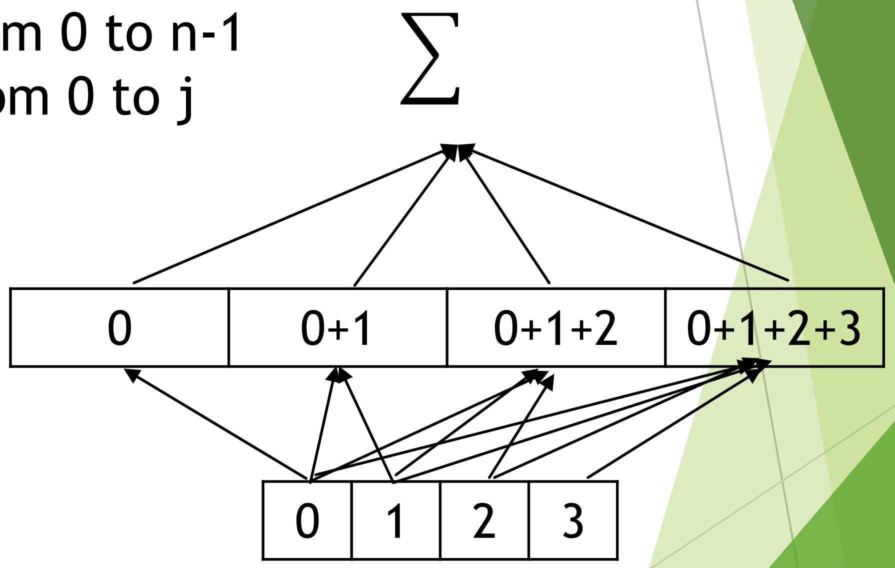
Number of computational operations: $\frac{1}{2}n$

Big-Oh: O(n)

R-4.11:

Give a big-Oh characterization, in terms of n, of the running time of the example3 method shown in Code Fragment.

```
/**Returns the sum of the prefix sums of given array.*/
Public static int example3(int[] arr){
    int n = arr.length, total = 0;
    for (int j=0; j<n; j++)              // loop from 0 to n-1
        for (int k=0; k<=j; k++)         // loop from 0 to j
            total += arr[j];
    return total;
}
```



Number of computational operations:

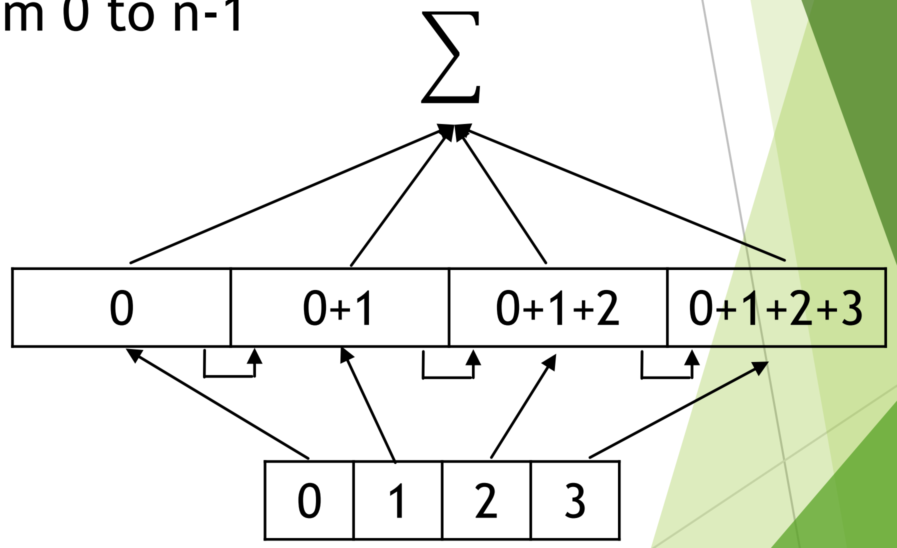
$$1 + 2 + \ldots + n = \frac{(1+n)n}{2}$$
$$= \frac{1}{2}(n^2 + n)$$
$$\leq \frac{1}{2}(n^2 + n^2)$$
$$\leq n^2$$

Big-Oh: $O(n^2)$

R-4.12:

Give a big-Oh characterization, in terms of n, of the running time of the example4 method shown in Code Fragment.

```
/**Returns the sum of the prefix sums of given array.*/
Public static int example4(int[] arr){
    int n = arr.length, prefix = 0, total = 0;
    for (int j=0; j<n; j++)          // loop from 0 to n-1
        prefix += arr[j];
        total += prefix;
    return total;
}
```

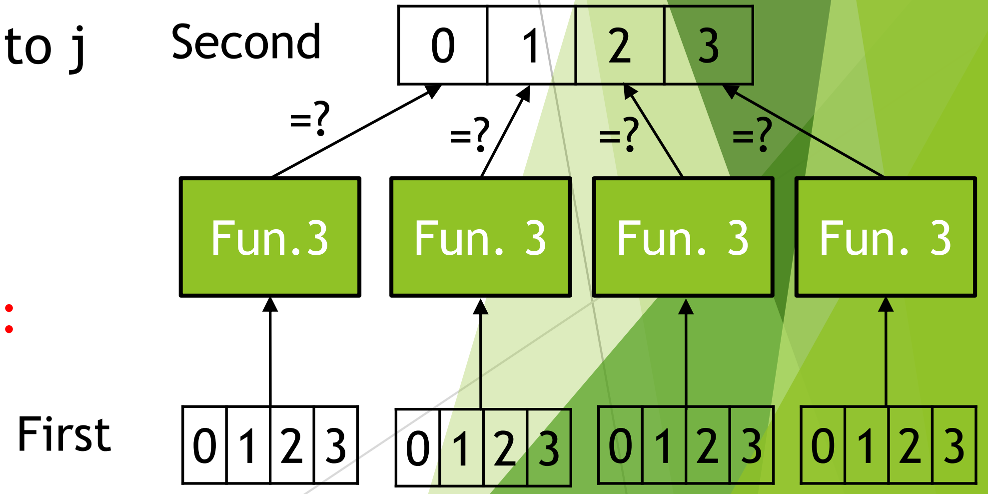


Number of computational operations: 2n
Big-Oh: O(n)

R-4.13:

Give a big-Oh characterization, in terms of n, of the running time of the example5 method shown in Code Fragment.

```
/**Returns the number of times second array stores sum of prefix sums from first.*/
Public static int example5(int[] first, int[] second){  // assume equal-length arrays
      int n = first.length, count = 0;
      for (int i=0; i<n; i++)                  // loop from 0 to n-1
            int total = 0;
            for (int j=0; j<n; j++)        // loop from 0 to n-1
                  for (int k=0; k<=j; k++)      // loop from 0 to j
                        total += first[k];
            if (second[i] == total) count++;
      return count;
}
```



Number of computational operations:

$$n * (1 + 2 + \ldots + n) = n * \frac{(1+n)n}{2}$$

$$= \frac{1}{2}(n^3 + n^2)$$

$$\leq \frac{1}{2}(n^3 + n^3)$$

$$\leq n^3$$

Big-Oh: $O(n^3)$

R-4.16:

Show that if d(n) is O(f(n)) and e(n) is O(g(n)), then d(n) + e(n) is O(f(n) + g(n)).

Since $d(n)$ is $O(f(n))$, there exist $C_1 > 0$ and $n_1 > 0$ such that $d(n) \leq C_1 * f(n)$ for all $n \geq n_1$

Since $e(n)$ is $O(g(n))$, there exist $C_2 > 0$ and $n_2 > 0$ such that $e(n) \leq C_2 * g(n)$ for all $n \geq n_2$

Assign $C = \max(C_1,\ C_2)$ and $n_0 = \max(n_1,\ n_2)$

Then
$$
\begin{aligned}
d(n) + e(n) &\leq C_1 * f(n) + C_2 * g(n) \\
&\leq C * f(n) + C * g(n) \\
&\leq C * (f(n) + g(n))
\end{aligned}
$$
For all $n \geq n_0$

So Big-Oh of $d(n) + e(n)$ is $O(f(n) + g(n))$

Exercise:

C-4.45