

Lab 04: Algorithm Analysis

CS 0445: Data Structures

TAs: Jon Rutkauskas
Brian Nixon

<http://db.cs.pitt.edu/courses/cs0445/current.term/>

Sep 30, 2019
University of Pittsburgh, Pittsburgh, PA



Big O notation - Review

- Mathematical notation to describe complexity of an algorithm compared to the size of the input
 - how many extra steps as the input grows
- Used to classify algorithms according to:
 - **Runtime – Main topic of discussion**
 - Memory usage



Big O notation - Review

Recall the formal definition from lecture:

The function $f(n)$ is $O(g(n))$ if:

- for some positive real number, c
- for some positive integer n_0
- $f(n) \leq c * g(n)$ for all $n \geq n_0$

That is, $c * g(n)$ is an **upper bound** on $f(n)$, for **sufficiently large n**



Big O Notation – Broken Down

Our actual function

Must be a sufficiently large n (larger than n_0)

$$f(n) \leq c * g(n) \text{ for all } n \geq n_0$$

Some multiplicative constant c

The growth rate E.g., $n, n^2, \log n$, etc.



Big O Notation – Broken Down

$$f(n) \leq c * g(n) \text{ for all } n \geq n_0$$

- This boils down to...
 - Ignore multiplicative constants
 - Drop lower order terms
 - Take our worst growth rate, $g(n)$.



Big O notation - Review

- What's the Big O of:
 - n^2
 - $2 + \log n$
 - $2 * n$
 - $4 * n^2 + 2 * n$
 - $2^n + 2 * n$



Big O notation - Review

- What's the Big O of:

- $n^2 \longrightarrow O(n^2)$

- $2 + \log n$

- $2 * n$

- $4 * n^2 + 2 * n$

- $2^n + 2 * n$



Big O notation - Review

- What's the Big O of:

- $n^2 \longrightarrow O(n^2)$

- $2 + \log n \longrightarrow O(\log n)$

- $2 * n$

- $4 * n^2 + 2 * n$

- $2^n + 2 * n$



Big O notation - Review

- What's the Big O of:

- $n^2 \longrightarrow O(n^2)$

- $2 + \log n \longrightarrow O(\log n)$

- $2 * n \longrightarrow O(n)$

- $4 * n^2 + 2 * n$

- $2^n + 2 * n$



Big O notation - Review

- What's the Big O of:

- $n^2 \longrightarrow O(n^2)$

- $2 + \log n \longrightarrow O(\log n)$

- $2 * n \longrightarrow O(n)$

- $4 * n^2 + 2 * n \longrightarrow O(n^2)$

- $2^n + 2 * n$



Big O notation - Review

- What's the Big O of:

- $n^2 \longrightarrow O(n^2)$

- $2 + \log n \longrightarrow O(\log n)$

- $2 * n \longrightarrow O(n)$

- $4 * n^2 + 2 * n \longrightarrow O(n^2)$

- $2^n + 2 * n \longrightarrow O(2^n)$



Analyzing Code

```
int counter = 0;  
for (int i=0; i < n ; i++) {  
    counter++;  
}
```



Analyzing Code

```
int counter = 0;
```

```
for (int i=0; i < n ; i++) {  
    counter++;  
}
```

Will repeat n
times



Analyzing Code

```
int counter = 0;
```

```
for (int i=0; i < n ; i++) {  
    counter++;  
}
```

Will repeat n
times

$O(n)$



Analyzing Code

```
int counter = 0;  
for (int i=0; i < n ; i++) {  
    counter++;  
    int j = 6;  
    int k = 10;  
    int m = k*j;  
}
```



Analyzing Code

```
int counter = 0;  
for (int i=0; i < n ; i++) {  
    counter++;  
    int j = 6;  
    int k = 10;  
    int m = k*j;  
}
```

Will repeat n
times

$O(n)$

These additional operations do not affect how many times this loop repeats
so they do not affect the growth rate



Analyzing Code

```
for (int i=0; i < n ; i++) {  
    for (int j=0; j < n ; j++) {  
        counter++;  
    }  
}
```



Analyzing Code

```
for (int i=0; i < n ; i++) {  
    for (int j=0; j < n ; j++) {  
        counter++;  
    }  
}
```

Will repeat
 n times

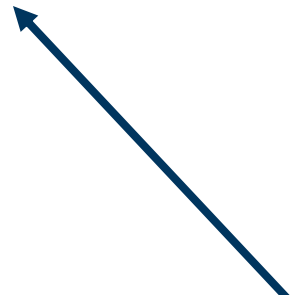
Will repeat
 n times

$O(n^2)$



Analyzing Code

```
for (int i=0; i < n ; i++) {  
    print_mult_row(i, n);  
}
```



```
public void print_mult_row(int mul, int size) {  
    for ( int j = 0; j <= size; j++ ) {  
        System.out.print(mul*j + "\t");  
    }  
}
```



Analyzing Code

```
for (int i=0; i < n ; i++) {  
    print_mult_row(i, n);  
}
```

Will repeat
 n times

$O(n^2)$

```
public void print_mult_row(int mul, int size) {  
    for ( int j = 0; j <= size; j++ ) {  
        System.out.print(mul*j + "\t");  
    }  
}
```

Will repeat
 n times



How to analyze code

- Remember to look at the loops
 - Determine how many times they'll repeat compared to n
 - Multiply the runtime of inner loops by the runtime of outer loops



Worksheet

