CS240 Fall 2014

Mike Lam, Professor

Algorithm Analysis Exercises

Exercises

- Fill in the appropriate Big-O relation:

$10n \in \_\_ (5n)$

$\log n \in \_\_ (2n)$

$n^2 + n \in \_\_ (10n)$

$5n \log n \in \_\_ (10n)$

$5n \log n \in \_\_ (n^2)$

$n^2 \log n \in \_\_ (n^2 + n \log n)$

$2^n \in \_\_ (n^2)$

Exercises

- Fill in the appropriate Big-O relation:

$$10n \in \Theta(5n)$$

$$\log n \in O(2n)$$

$$n^2 + n \in \Omega(10n)$$

$$5n \log n \in \Omega(10n)$$

$$5n \log n \in O(n^2)$$

$$n^2 \log n \in \Omega(n^2 + n \log n)$$

$$2^n \in \Omega(n^2)$$

Exercises

- Using either definition of Big-O, demonstrate:

$$10n \in \Theta(5n)$$

Exercises

- Using either definition of Big-O, demonstrate:

$$10n \in \Theta(5n)$$

$10n \in O(5n)$

Show that c and n_0 exist such that:
$10n \leq c \cdot 5n$ for all $n > n_0$

$10n \in \Omega(5n)$

Show that c and n_0 exist such that:
$10n \geq c \cdot 5n$ for all $n > n_0$

Exercises

- Using either definition of Big-O, demonstrate:

$$10n \in \Theta(5n)$$

$10n \in O(5n)$

Show that c and n_0 exist such that:
$10n \le c \cdot 5n$ for all $n > n_0$

$n_0 = 1$, $c = 2$

$10n \in \Omega(5n)$

Show that c and n_0 exist such that:
$10n \ge c \cdot 5n$ for all $n > n_0$

$n_0 = 1$, $c = 2$

Exercises

- Using either definition of Big-O, demonstrate:

$$10n \in \Theta(5n)$$

Alternately, show that $\lim_{n \to \infty} \frac{10n}{5n}$ is a constant greater than 0 and less than infinity.

Exercises

- Using either definition of Big-O, demonstrate:

$10n \in \Theta(5n)$

Alternately, show that $\lim_{n \to \infty} \frac{10n}{5n}$ is a constant greater than 0 and less than infinity.

$$\lim_{n \to \infty} \frac{10n}{5n} = \lim_{n \to \infty} 2 = 2$$

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example1(values):
    sum = 0
    for i in values:
        sum += i
    for i in range(20):
        sum += i
    return sum
```

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example1(values):
    sum = 0
    for i in values:
        sum += i
    for i in range(20):
        sum += i
    return sum
```

Additions: 20 + n
Complexity: O(n)

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example2(values):
    sum = 0
    for i in values:
        sum += i
        for j in range(20):
            sum += j
    return sum
```

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example2(values):
    sum = 0
    for i in values:
        sum += i
        for j in range(20):
            sum += j
    return sum
```

Additions: 21n
Complexity: O(n)

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example3(values):
    sum = 0
    for i in values:
        sum += i
        for j in values:
            sum += j
    return sum
```

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example3(values):
    sum = 0
    for i in values:
        sum += i
        for j in values:
            sum += j
    return sum
```

Additions: n^2+n
Complexity: $O(n^2)$

Exercises

- Given these two algorithms, for what values of *n* is Algorithm A faster?

Algorithm A

Additions: n^2+n
Complexity: O(n^2)

Algorithm B

Additions: 21n
Complexity: O(n)

Exercises

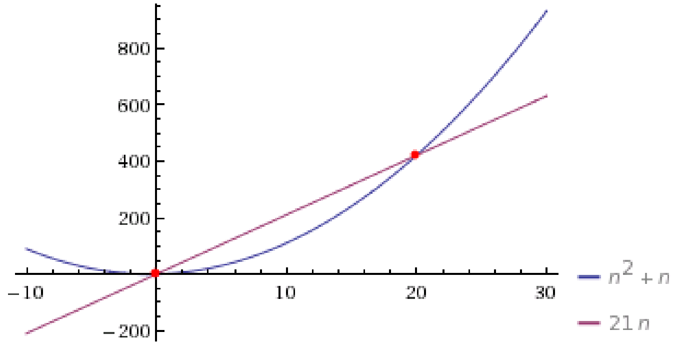
- Given these two algorithms, for what values of *n* is Algorithm A faster?

Algorithm A

Additions: n^2+n
Complexity: $O(n^2)$

Algorithm B

Additions: 21n
Complexity: O(n)



Exercises

- Given these two algorithms, for what values of *n* is Algorithm A faster?

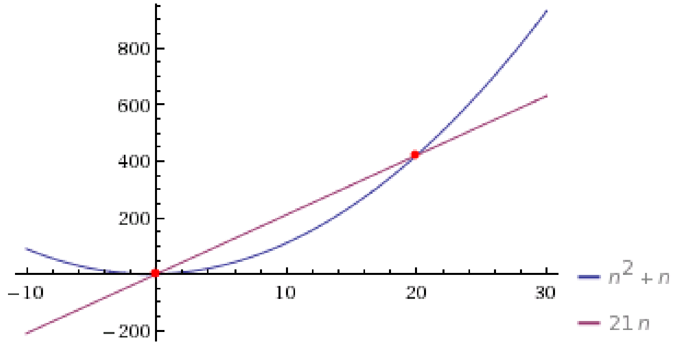
Algorithm A

Additions: n^2+n
Complexity: $O(n^2)$

Algorithm B

Additions: 21n
Complexity: O(n)

Preferred for x<20



Preferred for x>20

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example4(values):
    sum = 0
    for i in range (1000):
        sum = sum + i
    for num in values:
        indx = 1
        while indx <= len(values)
            sum += values[indx-1]
            indx *= 2
    return sum
```

Exercises

- Determine the number of addition operations performed by this function as well as its complexity class.

```
def example4(values):
    sum = 0
    for i in range (1000):
        sum = sum + i
    for num in values:
        indx = 1
        while indx <= len(values)
            sum += values[indx-1]
            indx *= 2
    return sum
```

Additions: 1000 + n $\log_2 n$

Complexity: O(n log n)

Exercises

- Given these two algorithms, for what values of *n* is Algorithm A faster?

Algorithm A	**Algorithm B**
Additions: $49n^2 + 50n$	**Additions: n^3**

Exercises

- Using either definition of Big-Θ demonstrate that $2n^3 + 2n \in \Theta(n^3)$