algorithmic analysis

asymptotics, runtime, etc.

slides
bit.ly/abhi-disc

attendance
bit.ly/abhi-attendance

announcements

announcements

1. Project 1 due 3/4 (friday)

announcements

1. Project 1 due 3/4 (friday)
2. HW 4 due 3/1 (tomorrow)

announcements

1. Project 1 due 3/4 (friday)
2. HW 4 due 3/1 (tomorrow)
3. Labs this week are Project 1 Office Hours

announcements

1. Project 1 due 3/4 (friday)
2. HW 4 due 3/1 (tomorrow)
3. Labs this week are Project 1 Office Hours
4. Weekly survey due tomorrow!

cost

cost

- time complexity

cost

- time complexity
- space complexity

cost

- **time complexity**
  - time it takes to run the program if we feed it a certain input
- **space complexity**

cost

- **time complexity**
  - time it takes to run the program if we feed it a certain input
- **space complexity**
  - how much space does running the program take up on our computer?

asymptotics

asymptotics

- evaluate the performance of a program using math

asymptotics

- evaluate the performance of a program using math
- ignore all constants

asymptotics

- evaluate the performance of a program using math
- ignore all constants
- only care about values with reference to the input (denoted as having size ‘N’)

bounds

bounds

- **big O:** upper bound in terms of the input
    - assume conditional statements evaluate to the worst case

bounds

- **big O:** upper bound in terms of the input
  - assume conditional statements evaluate to the worst case
- **big Ω:** lower bound in terms of the input
  - assume conditional statements evaluate to the best case

bounds

- **big O:** upper bound in terms of the input
  - assume conditional statements evaluate to the worst case
- **big Ω:** lower bound in terms of the input
  - assume conditional statements evaluate to the best case
- **big Θ:** the tightest bound, only exists when the upper and lower bounds converge

useful sums

$1 + 2 + 3 + ... + N \ = ?$

useful sums

$1 + 2 + 3 + \ldots + N = N(N - 1)/2$

useful sums

$$1 + 2 + 3 + ... + N = N(N - 1)/2$$

$$= (N^2 - N)/2$$

useful sums

$$1 + 2 + 3 + ... + N = N(N - 1)/2$$

$$= (N^2 - N)/2$$

$$= N^2/2 - N/2$$

useful sums

$1 + 2 + 3 + ... + N = N(N - 1)/2$

$= (N^2 - N)/2$

$= N^2/2 - N/2$

$\Theta(N^2/2 - N/2)$

- remove the constants: $N^2 - N$
- remove “smaller” additive terms: N^2

useful sums

$$1 + 2 + 3 + \ldots + N = N(N - 1)/2$$

$$= (N^2 - N)/2$$

$$= N^2/2 - N/2$$

$\Theta(N^2/2 - N/2) = \Theta(N^2)$

- remove the constants -> $N^2 - N$
- remove “smaller” additive terms -> N^2

useful sums

$1 + 2 + 3 + \ldots + N = \Theta(N^2)$

useful sums

$1 + 2 + 3 + \ldots + N = \Theta(N^2)$

$1 + 2 + 4 + 8 + \ldots + N = ?$

useful sums

$1 + 2 + 3 + \ldots + N = \Theta(N^2)$

$2^0 + 2^1 + 2^2 + \ldots + 2^{N-1} = ?$

useful sums

$1 + 2 + 3 + \ldots + N \;=\; \Theta(N^2)$

$1 + 2 + 4 + 8 + \ldots + N \;=$

2^0 + 2^1 + 2^2 + . . . + 2^N-1 = $\Theta(N)$

useful sums

$$1 + 2 + 3 + \ldots + N = \Theta(N^2)$$

$$1 + 2 + 4 + 8 + \ldots + N = \Theta(N)$$

useful sums

$1 + 2 + 3 + ... + N = \Theta(N^2)$ -> “arithmetic” sum

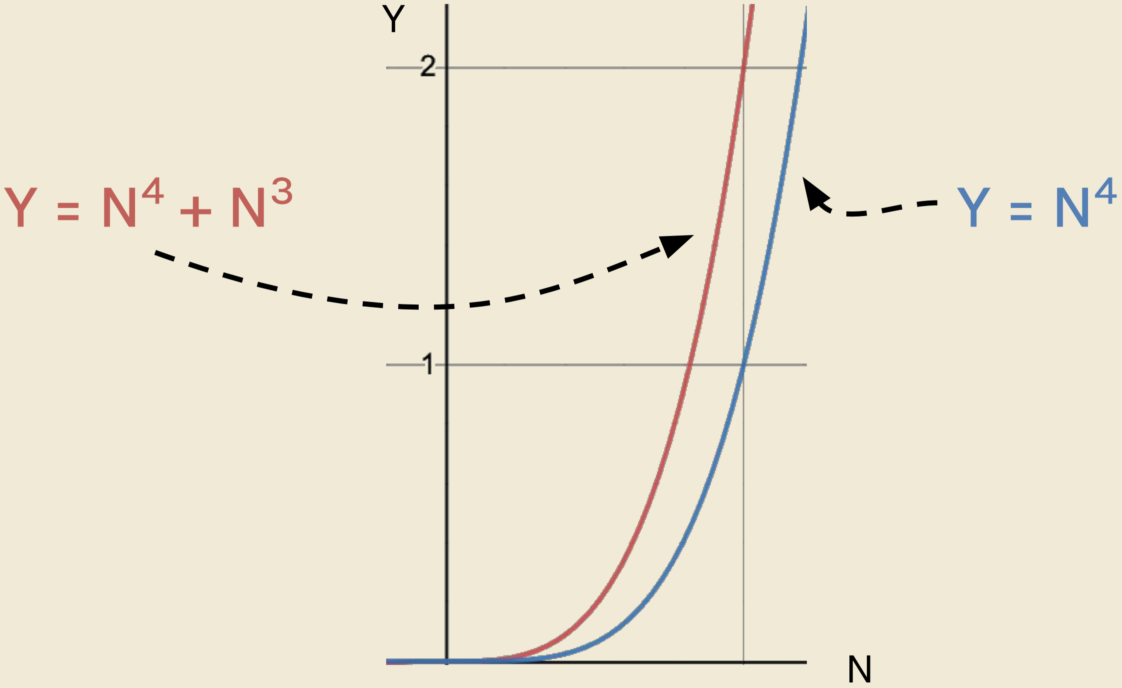
$1 + 2 + 4 + 8 + ... + N = \Theta(N)$ -> “geometric” sum

$$\Theta(N^2 + \log N) = \Theta(N^2)?$$

$$\Theta(N^4 + N^3) = \Theta(N^4)?$$

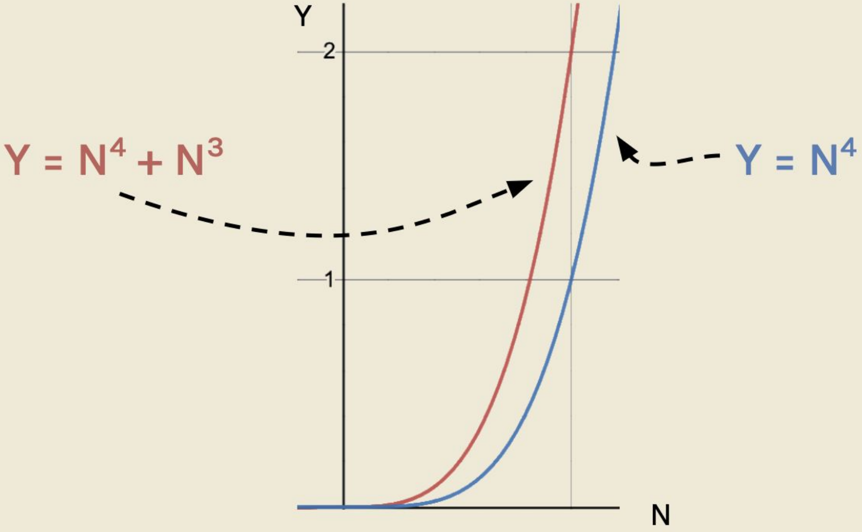
$$\Theta(2^N + N^{314159265359}) = \Theta(2^N)?$$

$$\Theta(N^4 + N^3) = \Theta(N^4)?$$



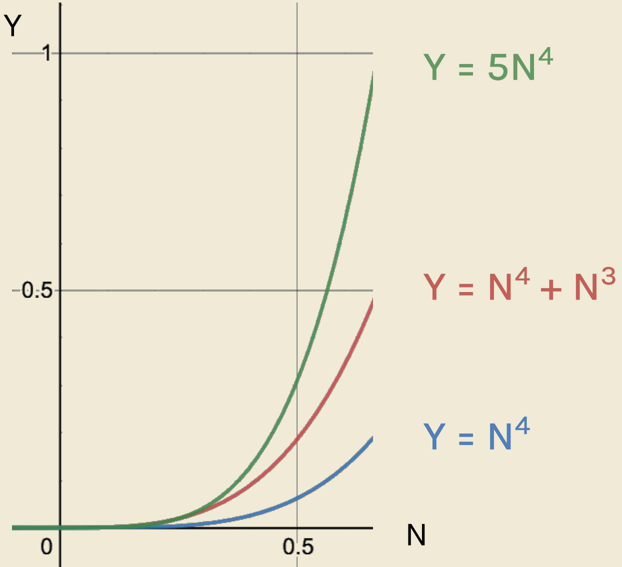
$\Theta(N^4 + N^3) = \Theta(N^4)$? how?

- seems to me like $y = N^4$ *strictly* < $y = N^4 + N^2$



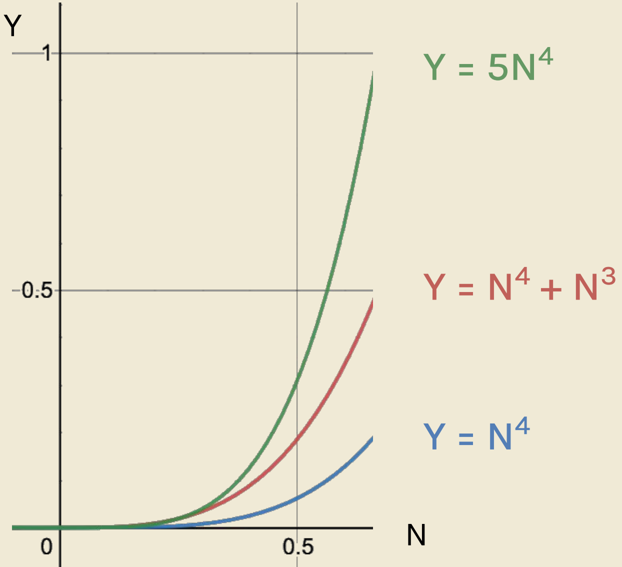
$\Theta(N^4 + N^3) = \Theta(N^4)$? how?

- seems to me like $y = N^4$ *strictly* $<$ $y = N^4 + N^2$
- consider $y = 5N^4$

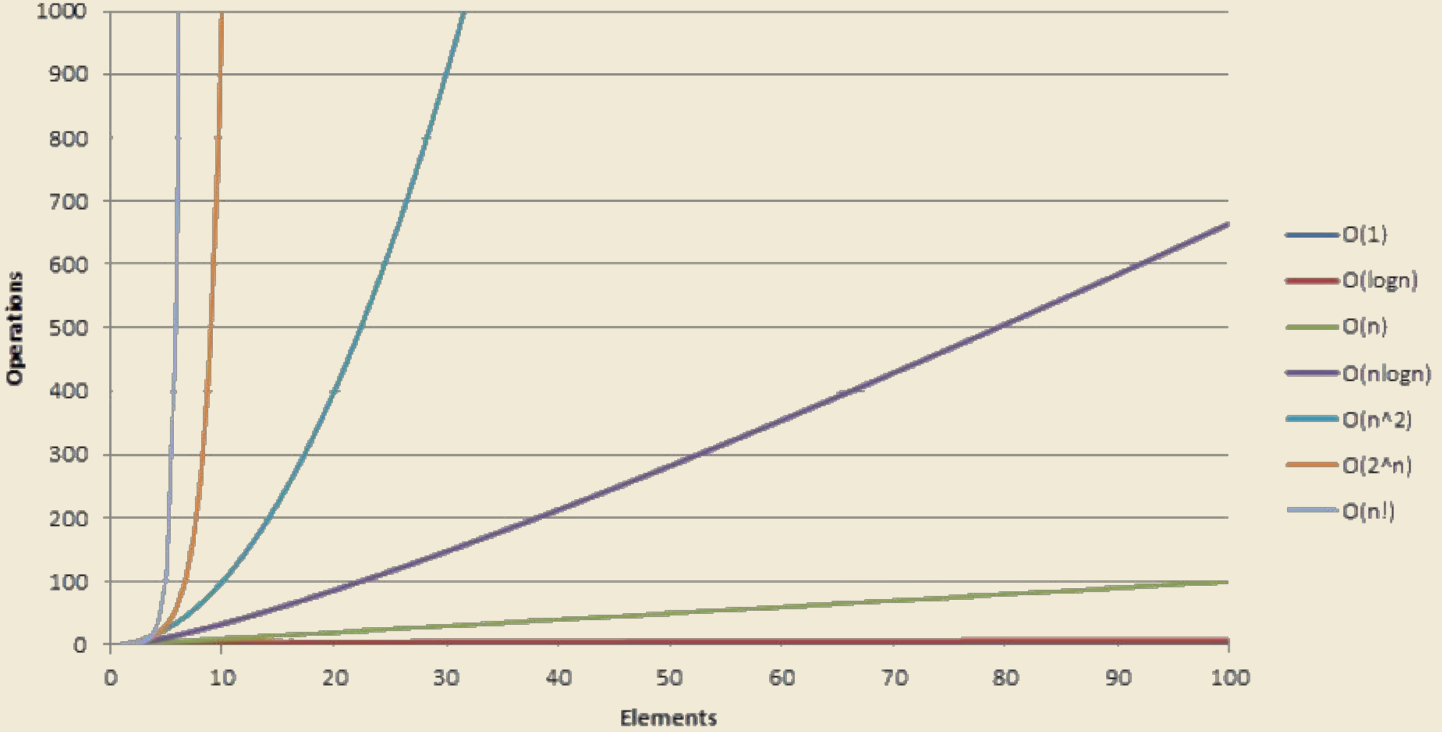


$\Theta(N^4 + N^3) = \Theta(N^4)$? how?

- seems to me like $y = N^4$ *strictly* $< y = N^4 + N^2$
- consider $y = 5N^4$
- $N^4 < N^4 + N^2 < 5N^4$



Big-O Complexity



analyzing a program

- choose an operation to count
- figure out the order of growth of the operation
  - exact counting OR
  - inspection

inspired by josh hug 2019, lec 13

analyzing a program - dup

```
private static boolean dup(int[] A) {
    int N = A.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            if (A[i] == A[j]) {
                return true;
            }
        }
    }
    return false;
}
```

[inspired by josh hug 2019, lec 13]

dup(int[] A) runtime

```
private static boolean dup(int[] A) {
    int N = A.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            if (A[i] == A[j]) {
                return true;
            }
        }
    }
    return false;
}
```

operation to count?

[inspired by josh hug 2019, lec 13]

dup(int[] A) runtime

```java
private static boolean dup(int[] A) {
    int N = A.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            if (A[i] == A[j]) {
                return true;
            }
        }
    }
    return false;
}
```

operation to count

[inspired by josh hug 2019, lec 13]

dup(int[] A) runtime

```java
private static boolean dup(int[] A) {
    int N = A.length;
    for (int i = 0; i < N; i++) {
        for (int j = i + 1; j < N; j++) {
            if (A[i] == A[j]) {
                return true;
            }
        }
    }
    return false;
}
```

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

[inspired by josh hug 2019, lec 13]()

dup(...) exact runtime

$$C = (N - 1) + (N - 2) + \ldots + 2 + 1$$

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

[inspired by josh hug 2019, lec 13]

dup(...) exact runtime

$$C = (N - 1) + (N - 2) + \ldots + 2 + 1$$
$$= 1 + 2 + \ldots + (N - 2) + (N - 1)$$

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

[inspired by josh hug 2019, lec 13]

dup(...) exact runtime

$$C = (N - 1) + (N - 2) + \ldots + 2 + 1$$
$$= 1 + 2 + \ldots + (N - 2) + (N - 1)$$

useful sums

$1 + 2 + 3 + \ldots + N = \Theta(N^2)$ -> “arithmetic” sum

$1 + 2 + 4 + 8 + \ldots + N = \Theta(N)$ -> “geometric” sum

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

[inspired by josh hug 2019, lec 13]()

dup(...) exact runtime

$C = (N - 1) + (N - 2) + \ldots + 2 + 1$

$\quad = 1 + 2 + \ldots + (N - 2) + (N - 1)$

$\Theta(N^2)$

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

useful sums

$1 + 2 + 3 + \ldots + N = \Theta(N^2)$ -> "arithmetic" sum

$1 + 2 + 4 + 8 + \ldots + N = \Theta(N)$ -> "geometric" sum

[inspired by josh hug 2019, lec 13]

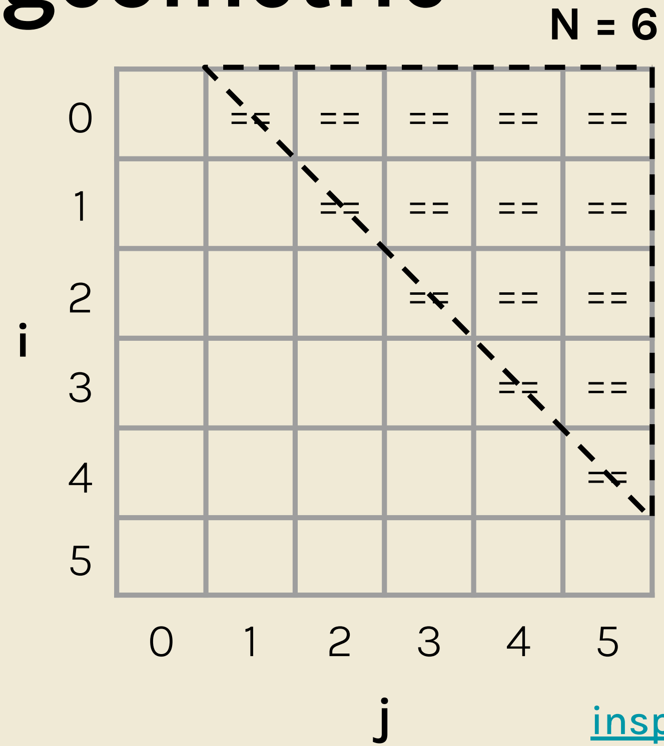
dup(...) geometric

N = 6

i \ j	0	1	2	3	4	5
0		==	==	==	==	==
1			==	==	==	==
2				==	==	==
3					==	==
4						==
5						

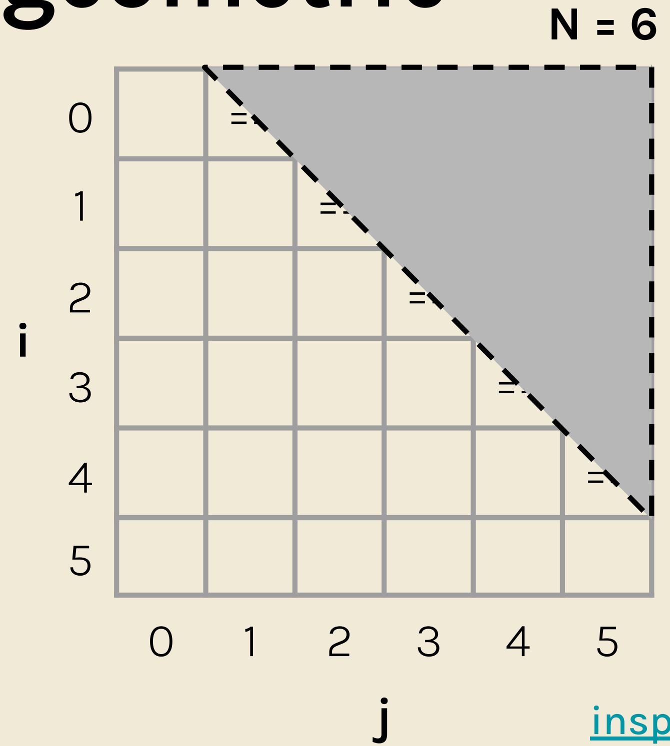
inspired by josh hug 2019, lec 13

dup(...) geometric



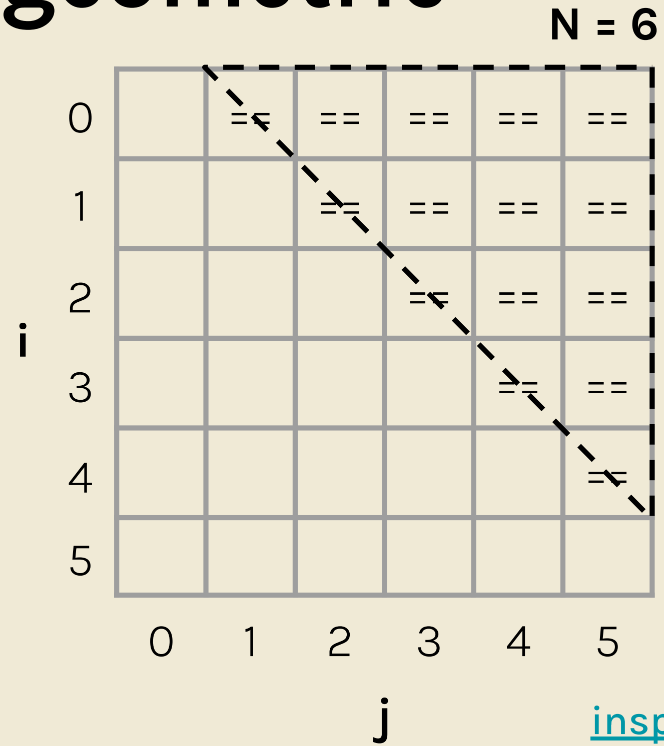
[inspired by josh hug 2019, lec 13]()

dup(...) geometric



[inspired by josh hug 2019, lec 13]

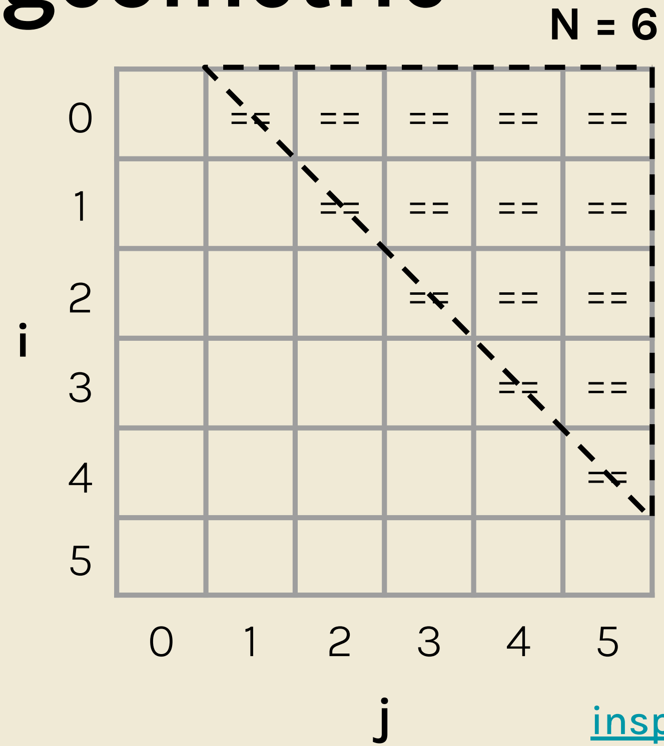
dup(…) geometric



$A = \frac{1}{2}bh$

[inspired by josh hug 2019, lec 13]

dup(...) geometric

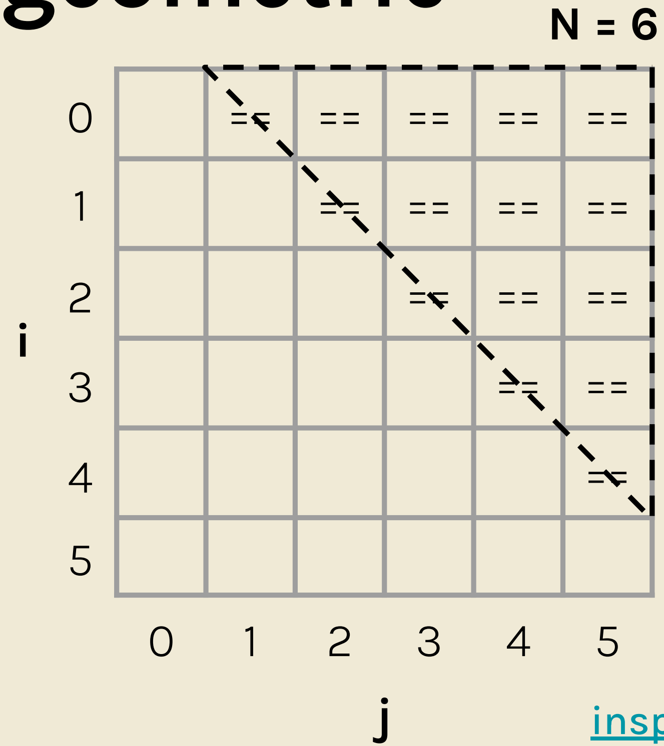


$A = \frac{1}{2}bh$

$A = \frac{1}{2}(N - 1)(N - 1)$

[inspired by josh hug 2019, lec 13]

dup(...) geometric



$A = \frac{1}{2}bh$

$A = \frac{1}{2}(N - 1)(N - 1)$

'==' grows on the scale of $\Theta(N^2)$

[inspired by josh hug 2019, lec 13]

worksheet

(on 61B website)

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$
B. $\Omega(N)$ vs. $\Omega(N^2)$
C. $O(N)$ vs. $O(N^2)$
D. $\Theta(N^2)$ vs. $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$
C. $O(N)$ vs. $O(N^2)$
D. $\Theta(N^2)$ vs. $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$ → Neither
C. $O(N)$ vs. $O(N^2)$
D. $\Theta(N^2)$ vs. $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$ → Neither
C. $O(N)$ vs. $O(N^2)$ → Neither
D. $\Theta(N^2)$ vs. $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$ → Neither
C. $O(N)$ vs. $O(N^2)$ → Neither
D. $\Theta(N^2)$ vs. $O(\log N)$ → $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$ → Neither
C. $O(N)$ vs. $O(N^2)$ → Neither
D. $\Theta(N^2)$ vs. $O(\log N)$ → $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$ → Neither

Why do we need to assume that N is large?

1 What's Faster?

Which is faster?

A. $\Theta(N)$ vs. $\Theta(N^2)$ → $\Theta(N)$
B. $\Omega(N)$ vs. $\Omega(N^2)$ → Neither
C. $O(N)$ vs. $O(N^2)$ → Neither
D. $\Theta(N^2)$ vs. $O(\log N)$ → $O(\log N)$
E. $O(N\log N)$ vs. $\Omega(N\log N)$ → Neither

Why do we need to assume that N is large?

Asymptotic bounds only make sense as N gets large because it allows us to disregard constant factors and lower order terms.

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \_(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in \_(g(x) = x^5)$

$f(x) = \log(x) \in \_(g(x) = 5x)$

$f(x) = e^x \in \_(g(x) = x^5)$

$f(x) = \log(5^x) \in \_(g(x) = x)$

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \Theta(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in \_(g(x) = x^5)$

$f(x) = \log(x) \in \_(g(x) = 5x)$

$f(x) = e^x \in \_(g(x) = x^5)$

$f(x) = \log(5^x) \in \_(g(x) = x)$

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \Theta(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in O(g(x) = x^5)$

$f(x) = \log(x) \in \_(g(x) = 5x)$

$f(x) = e^x \in \_(g(x) = x^5)$

$f(x) = \log(5^x) \in \_(g(x) = x)$

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \Theta(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in O(g(x) = x^5)$

$f(x) = \log(x) \in O(g(x) = 5x)$

$f(x) = e^x \in \_(g(x) = x^5)$

$f(x) = \log(5^x) \in \_(g(x) = x)$

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \Theta(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in O(g(x) = x^5)$

$f(x) = \log(x) \in O(g(x) = 5x)$

$f(x) = e^x \in \Omega(g(x) = x^5)$

$f(x) = \log(5^x) \in \_(g(x) = x)$

2 Basic Algorithmic Analysis

$f(x) = x^2 \in \Theta(g(x) = x^2 + x)$

$f(x) = 5000000x^3 \in O(g(x) = x^5)$

$f(x) = \log(x) \in O(g(x) = 5x)$

$f(x) = e^x \in \Omega(g(x) = x^5)$

$f(x) = \log(5^x) \in \Theta(g(x) = x)$

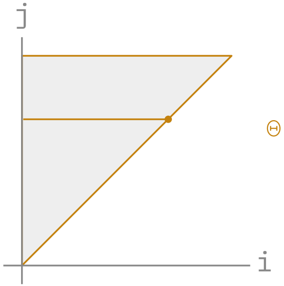
3A Practice with Runtime

```
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) {
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) {
        constant(k);
    }
}
```

3A Practice with Runtime

```
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) {
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) {
        constant(k);
    }
}
```

when i = x, the inner loop does x work



Θ(1) + Θ(2) + ... + Θ(N)
= Θ(N * (N + 1) / 2)
= Θ(N^2)

3A Practice with Runtime

```java
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) { // Θ(N^2)
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) {
        constant(k);
    }
}
```

3A Practice with Runtime

```
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) { // Θ(N^2)
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) {
        constant(k);
    }
}
```

1 work for each of the N steps

3A Practice with Runtime

```
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) { // Θ(N^2)
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) { // Θ(N)
        constant(k);
    }
}
```

3A Practice with Runtime

```
public static void bars(int n) {
    for (int i = 0; i < n; i += 1) { // Θ(N^2)
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) { // Θ(N)
        constant(k);
    }
}
```

Θ(N^2 + N) = Θ(N^2)

3A Practice with Runtime

```
public static void bars(int n) { // Θ(N^2)
    for (int i = 0; i < n; i += 1) { // Θ(N^2)
        for (int j = 0; j < i; j += 1) {
            System.out.println(i + j);
        }
    }

    for (int k = 0; k < n; k += 1) { // Θ(N)
        constant(k);
    }
}
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) {
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println(“moove”);
            }
        }
    }
}

public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) { <- This whole thing is independent of N!
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println("moove");
            }
        }
    }
}

public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) { // Θ(1)
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println(“moove”);
            }
        }
    }
}

public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) { // Θ(1)
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println(“moove”);
            }
        }
    }
}

public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j); // We know this is constant time now
        }
    }
}
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) { // Θ(1)
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println("moove");
            }
        }
    }
}

public static void barsRearranged(int n) {
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

```
Θ(1) + Θ(2) + ... + Θ(N/2) + Θ(N)
    = sum_(i=0)^(log N) 2^i
    = Θ(2^(log N + 1) - 1)
    = Θ(N)
```

3B Practice with Runtime

```
public static void cowsGo(int n) {
    for (int i = 0; i < 100; i += 1) { // Θ(1)
        for (int j = 0; j < i; j += 1) {
            for (int k = 0; k < j; k += 1) {
                System.out.println(“moove”);
            }
        }
    }
}

public static void barsRearranged(int n) { // Θ(N)
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < i; j += 1) {
            cowsGo(j);
        }
    }
}
```

attendance

bit.ly/abhi-attendance

feedback

bit.ly/abhi-feedback

slides: bit.ly/abhi-disc