CSE 250: Asymptotic Analysis

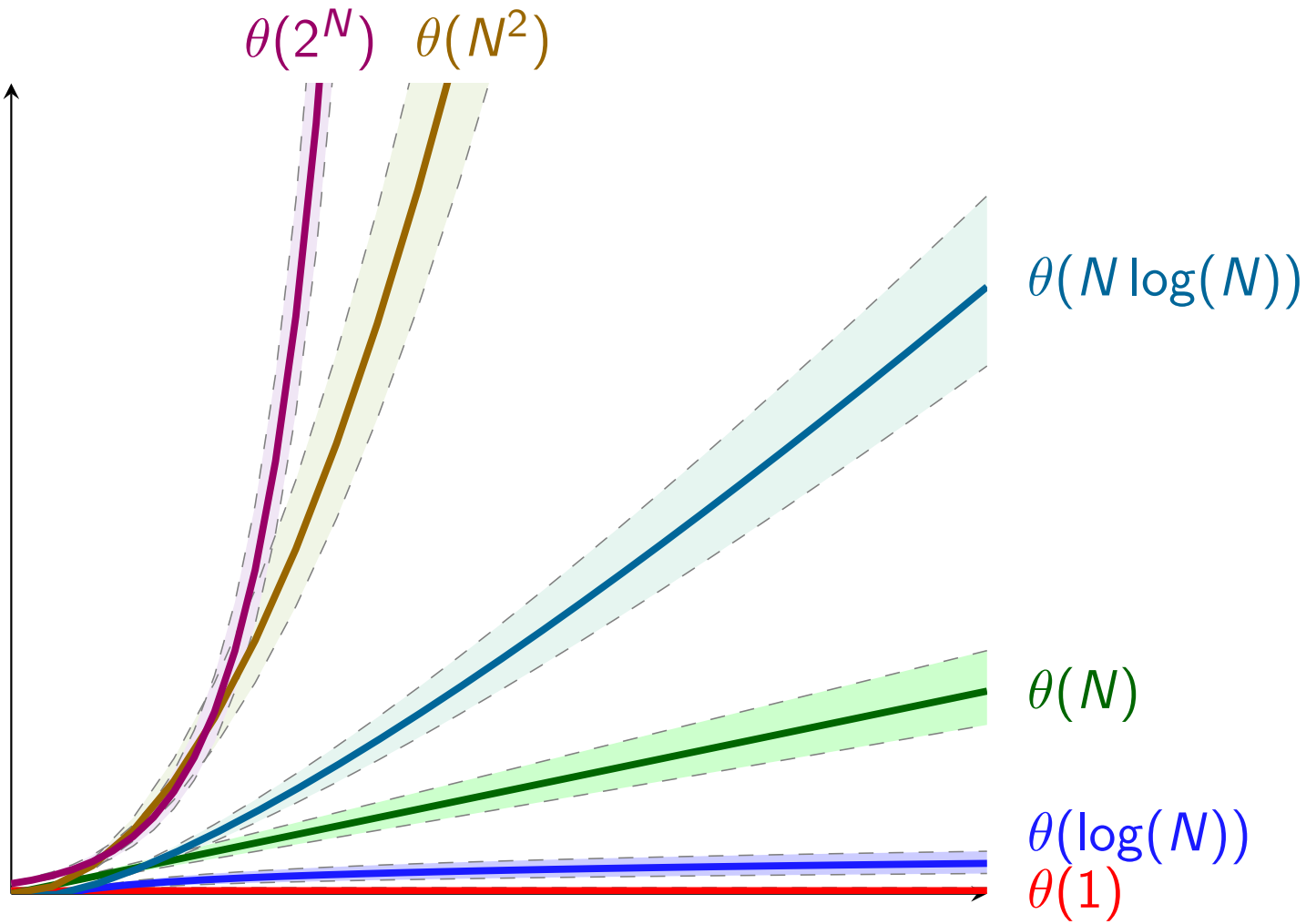
Lecture 7

Sept 13, 2023

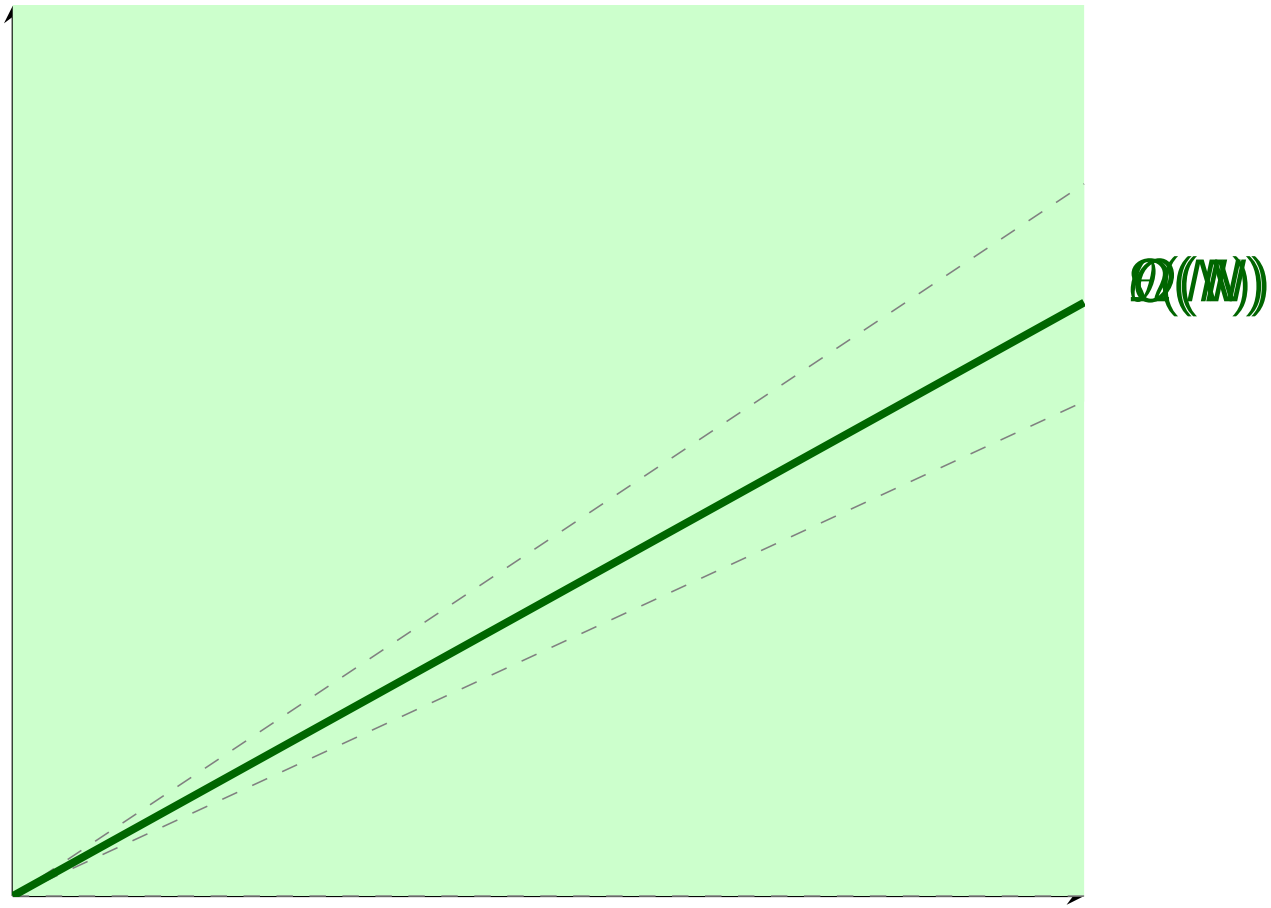
Reminders

- PA1 Tests due Sun, Sept 17 at 11:59 PM
  - Recitations will cover writing good test cases.
- PA1 Implementation due Sun, Sept 24 at 11:59 PM
  - Implement a Sorted Linked List

Complexity Classes



Complexity Bounds



Complexity Bounds

$g(N) \in O(f(N))$ (f is an upper bound for g) if and only if:

- You can pick an N_0
- You can pick a c
- For all $N > N_0$: $g(N) \leq c \cdot f(N)$

$g(N) \in \Omega(f(N))$ (f is a lower bound for g) if and only if:

- You can pick an N_0
- You can pick a c
- For all $N > N_0$: $g(N) \geq c \cdot f(N)$

$g(N) \in \theta(f(N))$ if and only if:

- $g(N) \in \Omega(f(N))$
- $g(N) \in O(f(N))$

Multi-Class Functions

If...

- $g(N) \in O(f(N))$ is a **tight** upper bound
- $g(N) \in \Omega(f'(N))$ is a **tight** lower bound
- $f'(N) \notin \theta(f(N))$

... then there is no θ bound for $g(N)$ (g is multi class)

Remember: Addition does *not* make a function multi-class.

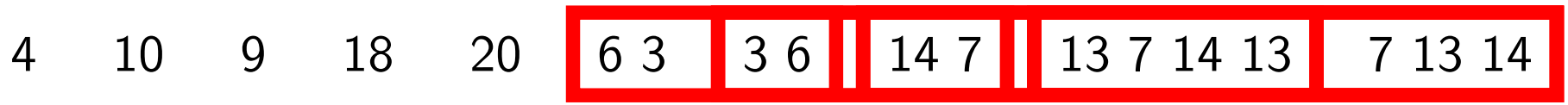
(A tight $\Omega(f(N))$ is the dominant (biggest) term being summed)

In practice...

Most documentation uses Big-O (upper, 'worst-case') bounds.

- There's always a Big-O bound.
- The best case usually doesn't bring down production servers.

Bubblesort



Bubblesort

```
1   public void bubblesort(int[] data)
2   {
3     int N = data.length;
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data[j+1] < data[j])
9         {
10          swap data[j] and data[j+1]
11        }
12      }
13    }
14  }
```

Bubblesort

```
1            if(data[j+1] < data[j])
2            {
3              swap data[j] and data[j+1]
4            }
```

$\theta(1)$

Bubblesort

```
1     public void bubblesort(int[] data)
2     {
3       int N = data.length;
4       for(int i = N - 2; i >= 0; i--)
5       {
6         for(int j = i; j <= N - 1; j++)
7         {
8           θ(1)
9         }
10      }
11    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        ∑_{j=i}^{N-1} θ(1)
7      }
8    }
```

Wait, what...?

$\theta(1)$ is a set of functions (or complexity class).

$\sum_{j=i}^{N-1} \theta(1) = \sum_{j=i}^{N-1} f(N)$ where we know $f(N) \in \theta(1)$

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      int N = data.length;
4      for(int i = N - 2; i >= 0; i--)
5      {
6        ∑_{j=i}^{N-1} θ(1)(N − 1 + 1 − i) · θ(1)(N − i) · θ(1)
7      }
8    }
```

Bubblesort

```
1     public void bubblesort(int[] data)
2     {
3       int N = data.length;
4       ∑_{i=0}^{N-2}(N − i) · θ(1)(∑_{i=0}^{N-2} N · θ(1)) − (∑_{i=0}^{N-2} i · θ(1))((N − 2 + 1 − 0) · N ·
5     }
```

Algebra with θ

Let $f'(N) = c \cdot f(N)$ where $f(N) \in \theta(g(N))$
(c is a constant)

What complexity class is $f'(N)$ in?

So $c \cdot \theta(g(N)) = \theta(g(N))$

Algebra with θ

$$
\begin{aligned}
c \cdot \theta(f(N)) &= \theta(f(N)) \\
N \cdot \theta(f(N)) &= \theta(N \cdot f(N)) \\
g(N) \cdot \theta(f(N)) &= \theta(g(N) \cdot f(N)) \quad (\text{if } \theta(g(N)) \text{ exists}) \\
\theta(g(N)) + \theta(f(N)) &= \theta(g(N) + f(N)) \\
&= \text{The greater of } \theta(f(N)) \textbf{ or } \theta(g(N))
\end{aligned}
$$

Bubblesort

```
1     public void bubblesort(int[] data)
2     {
3       int N = data.length;
4       (½N² + N − 2) · θ(1)θ((½N² + N − 2) · 1)θ(½N² + N − 2)θ(½N²)θ(N²)
5     }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      θ(1)
4      θ(N^2)
5    }
```

Bubblesort

```
1    public void bubblesort(int[] data)
2    {
3      θ(N^2)
4    }
```

Bubblesort on an array is $\theta(N^2)$

Rules of Thumb

- **Lines of Code**: Add Complexities
- **Loops**: Multiply Complexities
- **If/Then**: Cases block '{'

Bubblesort

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          int temp = data.get(j);
11          data.set(j, data.get(j+1));
12          data.set(j+1, temp);
13        }
14      }
15    }
16  }
```

Lists

A java `List` can be a:

- **Linked List**: `LinkedList`
  - `data.get(x)`, `data.set(x)` are $O(N)$[1]
- **Vector Array**: `ArrayList`
  - `data.get(x)`, `data.set(x)` are $\theta(1)$ (this implies $O(1)$)

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = \begin{cases} O(1) & \textbf{if } \texttt{data} \text{ is a } \texttt{ArrayList} \\ O(N) & \textbf{if } \texttt{data} \text{ is a } \texttt{LinkedList} \end{cases}$$

$$T_{\texttt{get}}(N) = T_{\texttt{set}}(N) = ???O(N)$$

[1] $\theta(x)$ would be more precise, but there's no better bound in terms of N.

Algebra with O

$$c \cdot O(f(N)) = O(f(N))$$

$$N \cdot O(f(N)) = O(N \cdot f(N))$$

$$g(N) \cdot O(f(N)) = O(g(N) \cdot f(N))$$

$$O(g(N)) + O(f(N)) = O(g(N) + f(N))$$
$$= \text{the greater of } O(f(N)) \textbf{ or } O(g(N))$$

$$\begin{cases} O(g(N)) & \textbf{if one thing} \\ O(f(N)) & \textbf{otherwise} \end{cases} = \text{the greater of } O(f(N)) \textbf{ or } O(g(N))$$
$$= O(g(N) + f(N))$$

Algebra with Ω

$$c \cdot \Omega(f(N)) = \Omega(f(N))$$

$$N \cdot \Omega(f(N)) = \Omega(N \cdot f(N))$$

$$g(N) \cdot \Omega(f(N)) = \Omega(g(N) \cdot f(N))$$

$$\Omega(g(N)) + \Omega(f(N)) = \Omega(g(N) + f(N))$$
$$= \text{the greater of } \Omega(f(N)) \textbf{ or } \Omega(g(N))$$

$$\begin{cases} \Omega(g(N)) & \textbf{if one thing} \\ \Omega(f(N)) & \textbf{otherwise} \end{cases} = \text{the lesser of } \Omega(f(N)) \textbf{ or } \Omega(g(N))$$

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          int temp = data.get(j);
11          data.set(j, data.get(j+1));
12          data.set(j+1, temp);
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11          O(N + N)
12          O(N)
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11        }
12      }
13    }
14  }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     int N = data.size();
4     for(int i = N - 2; i >= 0; i--)
5     {
6       for(int j = i; j <= N - 1; j++)
7       {
8         if(data.get(j+1) < data.get(j))
9         {
10          O(N)
11        } else {
12          O(1)
13        }
14      }
15    }
16  }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int N = data.size();
4       for(int i = N - 2; i >= 0; i--)
5       {
6         for(int j = i; j <= N - 1; j++)
7         {
8           O(2N) + O(N OR 1)
9         }
10      }
11    }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int N = data.size();
4       for(int i = N - 2; i >= 0; i--)
5       {
6         for(int j = i; j <= N - 1; j++)
7         {
8           O(2N) + O(N + 1)
9         }
10      }
11    }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int N = data.size();
4       for(int i = N - 2; i >= 0; i--)
5       {
6         for(int j = i; j <= N - 1; j++)
7         {
8           O(2N) + O(N)
9         }
10      }
11    }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int N = data.size();
4       for(int i = N - 2; i >= 0; i--)
5       {
6         for(int j = i; j <= N - 1; j++)
7         {
8           O(N)
9         }
10      }
11    }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int N = data.size();
4       for(int i = N - 2; i >= 0; i--)
5       {
6         ∑_{j=i}^{N-1} O(N)(N − 1 + 1 − i)O(N)(N − i)O(N)
7       }
8     }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      int N = data.size();
4      ∑_{i=0}^{N-2}(N − i) · O(N)(∑_{i=0}^{N-2} N · O(N)) − (∑_{i=0}^{N-2} i · O(N))((N − 2 + 1 − 0) ·
5    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(1)
4       O(N^3)
5    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      O(N^3)
4    }
```

Bubblesort on Lists

Can we do better?

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       int[] array = data.toArray()
4       bubblesort(array) // Use the array implementation
5       data.clear()
6       data.addAll(Arrays.toList(array))
7     }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       O(N)
4       bubblesort(array) // Use the array implementation
5       data.clear()
6       data.addAll(Arrays.toList(array))
7     }
```

Bubblesort on Lists

```
1     public void bubblesort(List[Integer] data)
2     {
3       O(N)
4       O(N^2)
5       data.clear()
6       data.addAll(Arrays.toList(array))
7     }
```

Bubblesort on Lists

```
1   public void bubblesort(List[Integer] data)
2   {
3     O(N)
4     O(N^2)
5     O(N)
6     data.addAll(Arrays.toList(array))
7   }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3      O(N)
4      O(N^2)
5      O(N)
6      O(N)
7    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(N + N^2 + N + N)
4    }
```

Bubblesort on Lists

```
1    public void bubblesort(List[Integer] data)
2    {
3       O(N^2)
4    }
```

Organizing data first can make code faster