# Unit #2: Complexity Theory and Asymptotic Analysis

## CPSC 221: Algorithms and Data Structures

Lars Kotthoff[1]
larsko@cs.ubc.ca

[1]With material from Will Evans, Steve Wolfman, Alan Hu, Ed Knorr, and Kim Voll.

# Types of analysis

```
if random() == 0
    return
if random() == 1
    for i = 1 to n do
        for j = 1 to n do
            k = 1
    return
else
    for i = 1 to n do
        k = 1
return
```

# Types of analysis

```
if random() == 0
    return
if random() == 1
    for i = 1 to n do
        for j = 1 to n do
            k = 1
    return
else
    for i = 1 to n do
        k = 1
return
```

Best case: $T(n) = 1$

# Types of analysis

```
if random() == 0
    return
if random() == 1
    for i = 1 to n do
        for j = 1 to n do
            k = 1
    return
else
    for i = 1 to n do
        k = 1
return
```

Best case: $T(n) = 1$

Worst case: $T(n) = n^2$

# Types of analysis

```
if random() == 0
    return
if random() == 1
    for i = 1 to n do
        for j = 1 to n do
            k = 1
    return
else
    for i = 1 to n do
        k = 1
return
```

Best case: $T(n) = 1$

Worst case: $T(n) = n^2$

Average case: $T(n) = n$

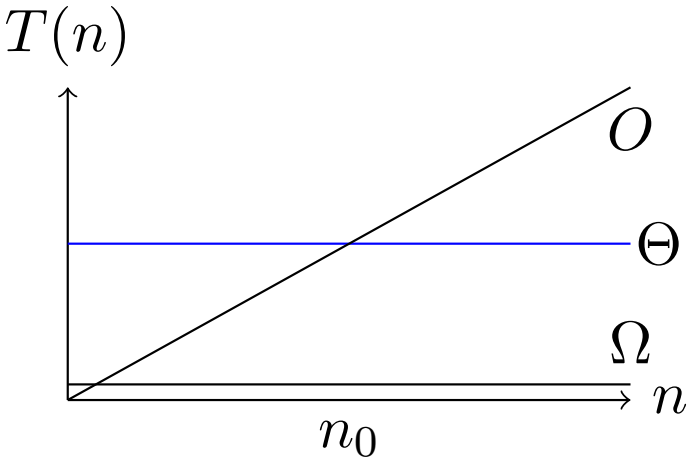
# Types of analysis

Best case: $T(n) = 1$

Worst case: $T(n) = n^2$

Average case: $T(n) = n$

# Types of analysis



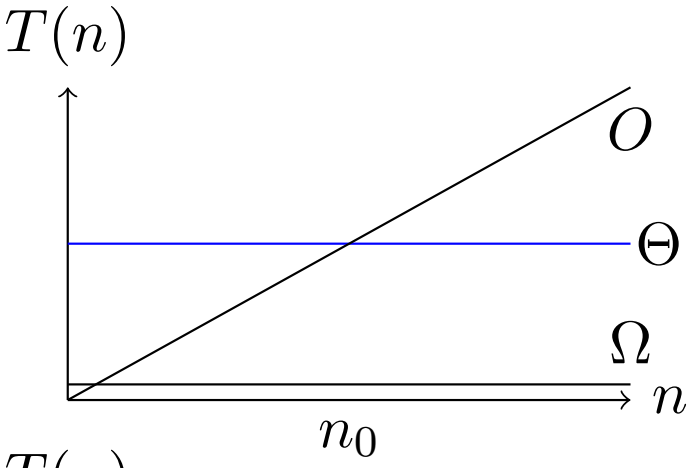

Best case: $T(n) = 1$

Worst case: $T(n) = n^2$

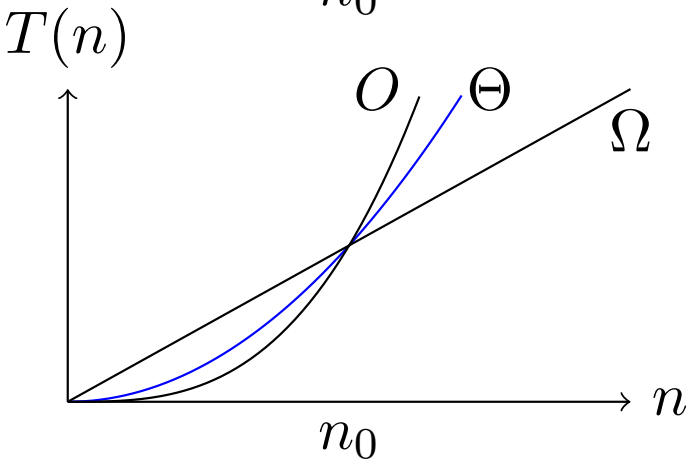
Average case: $T(n) = n$

# Types of analysis

Best case: $T(n) = 1$



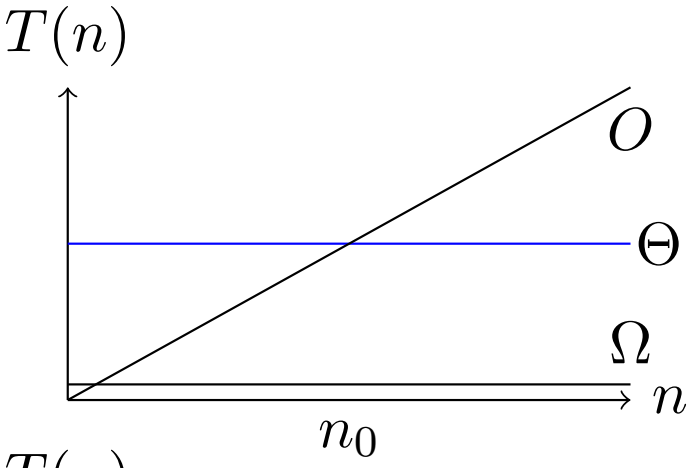

Worst case: $T(n) = n^2$



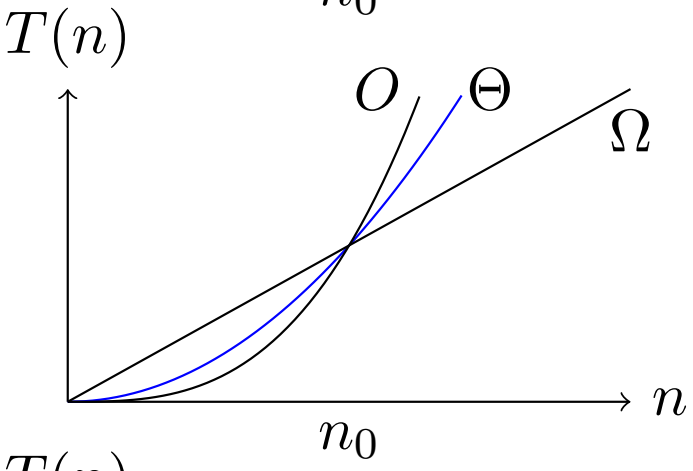

Average case: $T(n) = n$

# Types of analysis

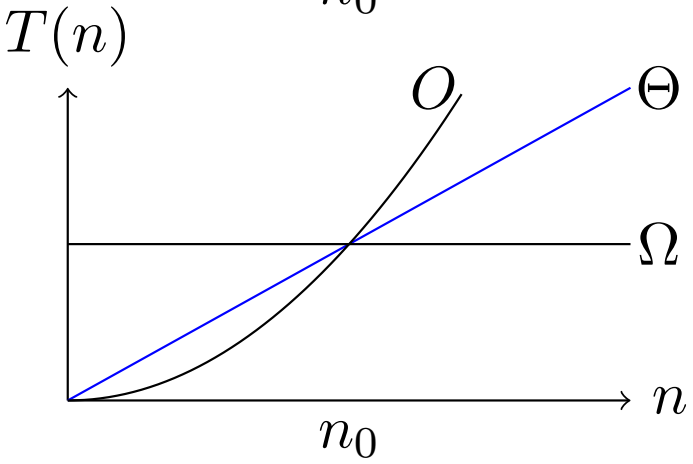
Best case: $T(n) = 1$




Worst case: $T(n) = n^2$




Average case: $T(n) = n$



# “Tight” bounds

- informally: want to have “good” bounds
- no better reasonable bound which is asymptotically different
- rigid definition: $\Theta$

# Runtime example #3

```
i = 1
while i < n do
  for j = 1 to i do
    sum = sum + 1
  i += i
```

# Runtime example #4

```
int max(A, n)
  if (n == 1) return A[0]
  return larger of A[n-1] and max(A, n-1)
```

Recursion almost always yields a recurrence relation:

$$
\begin{aligned}
T(1) &\leq b \\
T(n) &\leq c + T(n-1) \qquad \text{if } n > 1
\end{aligned}
$$

Solving recurrence:

$$
\begin{aligned}
T(n) &\leq c + c + T(n-2) && \text{(substitution)} \\
&\leq c + c + c + T(n-3) && \text{(substitution)} \\
&\leq kc + T(n-k) && \text{(extrapolating } k > 0\text{)} \\
&= (n-1)c + T(1) && \text{(for } k = n-1\text{)} \\
&\leq (n-1)c + b
\end{aligned}
$$

$T(n) \in$

# Runtime example #5: Mergesort

Mergesort algorithm:

Split list in half, sort first half, sort second half, merge together

Recurrence relation:

$$
\begin{aligned}
T(1) &\leq b \\
T(n) &\leq 2T(n/2) + cn \qquad \text{if } n > 1
\end{aligned}
$$

Solving recurrence:

$$
\begin{aligned}
T(n) &\leq 2T(n/2) + cn \\
&\leq 2(2T(n/4) + cn/2) + cn && \text{(substitution)} \\
&= 4T(n/4) + 2cn \\
&\leq 4(2T(n/8) + cn/4) + 2cn && \text{(substitution)} \\
&= 8T(n/8) + 3cn \\
&\leq 2^k T(n/2^k) + kcn && \text{(extrapolating } k > 0\text{)} \\
&= nT(1) + cn \lg n && \text{(for } 2^k = n\text{)}
\end{aligned}
$$

$T(n) \in$