# 5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of $n$. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

# 5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of $n$. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

# 5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of $n$. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

# 5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of $n$. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

# 5 Asymptotic Notation

We are usually not interested in exact running times, but only in an asymptotic classification of the running time, that ignores constant factors and constant additive offsets.

- We are usually interested in the running times for large values of $n$. Then constant additive terms do not play an important role.
- An exact analysis (e.g. *exactly* counting the number of operations in a RAM) may be hard, but wouldn't lead to more precise results as the computational model is already quite a distance from reality.
- A linear speed-up (i.e., by a constant factor) is always possible by e.g. implementing the algorithm on a faster machine.
- Running time should be expressed by simple functions.

# Asymptotic Notation

## Formal Definition

Let $f$ denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than $f$)

# Asymptotic Notation

## Formal Definition

Let $f$ denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than $f$)
- $\Omega(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not slower than $f$)

# Asymptotic Notation

## Formal Definition

Let $f$ denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than $f$)
- $\Omega(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not slower than $f$)
- $\Theta(f) = \Omega(f) \cap \mathcal{O}(f)$
  (functions that asymptotically have the same growth as $f$)

# Asymptotic Notation

## Formal Definition

Let $f$ denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than $f$)
- $\Omega(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not slower than $f$)
- $\Theta(f) = \Omega(f) \cap \mathcal{O}(f)$
  (functions that asymptotically have the same growth as $f$)
- $o(f) = \{g \mid \forall c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow slower than $f$)

# Asymptotic Notation

## Formal Definition

Let $f$ denote functions from $\mathbb{N}$ to $\mathbb{R}^+$.

- $\mathcal{O}(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not faster than $f$)
- $\Omega(f) = \{g \mid \exists c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow not slower than $f$)
- $\Theta(f) = \Omega(f) \cap \mathcal{O}(f)$
  (functions that asymptotically have the same growth as $f$)
- $o(f) = \{g \mid \forall c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \leq c \cdot f(n)]\}$
  (set of functions that asymptotically grow slower than $f$)
- $\omega(f) = \{g \mid \forall c > 0\ \exists n_0 \in \mathbb{N}_0\ \forall n \geq n_0 : [g(n) \geq c \cdot f(n)]\}$
  (set of functions that asymptotically grow faster than $f$)

# Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). $f$ and $g$ are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \leq \lim_{n \to \infty} \frac{g(n)}{f(n)} < \infty$

# Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). $f$ and $g$ are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \leq \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in \Omega(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} \leq \infty$

# Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). $f$ and $g$ are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \le \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in \Omega(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} \le \infty$
- $g \in \Theta(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$

# Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). $f$ and $g$ are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \leq \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in \Omega(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} \leq \infty$
- $g \in \Theta(f)$: $\quad 0 < \lim_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in o(f)$: $\quad \lim_{n\to\infty} \frac{g(n)}{f(n)} = 0$

# Asymptotic Notation

There is an equivalent definition using limes notation (assuming that the respective limes exists). $f$ and $g$ are functions from $\mathbb{N}_0$ to $\mathbb{R}_0^+$.

- $g \in \mathcal{O}(f)$: $\quad 0 \leq \lim\limits_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in \Omega(f)$: $\quad 0 < \lim\limits_{n\to\infty} \frac{g(n)}{f(n)} \leq \infty$
- $g \in \Theta(f)$: $\quad 0 < \lim\limits_{n\to\infty} \frac{g(n)}{f(n)} < \infty$
- $g \in o(f)$: $\quad \lim\limits_{n\to\infty} \frac{g(n)}{f(n)} = 0$
- $g \in \omega(f)$: $\quad \lim\limits_{n\to\infty} \frac{g(n)}{f(n)} = \infty$

# Asymptotic Notation

**Abuse of notation**

1. People write $f = \mathcal{O}(g)$, when they mean $f \in \mathcal{O}(g)$. This is **not** an equality (how could a function be equal to a set of functions).
2. People write $f(n) = \mathcal{O}(g(n))$, when they mean $f \in \mathcal{O}(g)$, with $f : \mathbb{N} \to \mathbb{R}^+, n \mapsto f(n)$, and $g : \mathbb{N} \to \mathbb{R}^+, n \mapsto g(n)$.
3. People write e.g. $h(n) = f(n) + o(g(n))$ when they mean that there exists a function $z : \mathbb{N} \to \mathbb{R}^+, n \mapsto z(n), z \in o(g)$ such that $h(n) = f(n) + z(n)$.
4. People write $\mathcal{O}(f(n)) = \mathcal{O}(g(n))$, when they mean $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$. Again this is not an equality.

# Asymptotic Notation

**Abuse of notation**

1. People write $f = \mathcal{O}(g)$, when they mean $f \in \mathcal{O}(g)$. This is **not** an equality (how could a function be equal to a set of functions).
2. People write $f(n) = \mathcal{O}(g(n))$, when they mean $f \in \mathcal{O}(g)$, with $f : \mathbb{N} \to \mathbb{R}^+, n \mapsto f(n)$, and $g : \mathbb{N} \to \mathbb{R}^+, n \mapsto g(n)$.
3. People write e.g. $h(n) = f(n) + o(g(n))$ when they mean that there exists a function $z : \mathbb{N} \to \mathbb{R}^+, n \mapsto z(n), z \in o(g)$ such that $h(n) = f(n) + z(n)$.
4. People write $\mathcal{O}(f(n)) = \mathcal{O}(g(n))$, when they mean $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$. Again this is not an equality.

# Asymptotic Notation

## Abuse of notation

1. People write $f = \mathcal{O}(g)$, when they mean $f \in \mathcal{O}(g)$. This is **not** an equality (how could a function be equal to a set of functions).
2. People write $f(n) = \mathcal{O}(g(n))$, when they mean $f \in \mathcal{O}(g)$, with $f : \mathbb{N} \to \mathbb{R}^+, n \mapsto f(n)$, and $g : \mathbb{N} \to \mathbb{R}^+, n \mapsto g(n)$.
3. People write e.g. $h(n) = f(n) + o(g(n))$ when they mean that there exists a function $z : \mathbb{N} \to \mathbb{R}^+, n \mapsto z(n), z \in o(g)$ such that $h(n) = f(n) + z(n)$.
4. People write $\mathcal{O}(f(n)) = \mathcal{O}(g(n))$, when they mean $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$. Again this is not an equality.

# Asymptotic Notation

**Abuse of notation**

1. People write $f = \mathcal{O}(g)$, when they mean $f \in \mathcal{O}(g)$. This is **not** an equality (how could a function be equal to a set of functions).
2. People write $f(n) = \mathcal{O}(g(n))$, when they mean $f \in \mathcal{O}(g)$, with $f : \mathbb{N} \to \mathbb{R}^+, n \mapsto f(n)$, and $g : \mathbb{N} \to \mathbb{R}^+, n \mapsto g(n)$.
3. People write e.g. $h(n) = f(n) + o(g(n))$ when they mean that there exists a function $z : \mathbb{N} \to \mathbb{R}^+, n \mapsto z(n), z \in o(g)$ such that $h(n) = f(n) + z(n)$.
4. People write $\mathcal{O}(f(n)) = \mathcal{O}(g(n))$, when they mean $\mathcal{O}(f(n)) \subseteq \mathcal{O}(g(n))$. Again this is not an equality.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

Here, $\Theta(n)$ stands for an anonymous function in the set $\Theta(n)$ that makes the expression true.

Note that $\Theta(n)$ is on the right hand side, otw. this interpretation is wrong.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

Here, $\Theta(n)$ stands for an anonymous function in the set $\Theta(n)$ that makes the expression true.

Note that $\Theta(n)$ is on the right hand side, otw. this interpretation is wrong.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

Here, $\Theta(n)$ stands for an anonymous function in the set $\Theta(n)$ that makes the expression true.

Note that $\Theta(n)$ is on the right hand side, otw. this interpretation is wrong.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + \mathcal{O}(n) = \Theta(n^2)$$

Regardless of how we choose the anonymous function $f(n) \in \mathcal{O}(n)$ there is an anonymous function $g(n) \in \Theta(n^2)$ that makes the expression true.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$2n^2 + \mathcal{O}(n) = \Theta(n^2)$$

Regardless of how we choose the anonymous function $f(n) \in \mathcal{O}(n)$ there is an anonymous function $g(n) \in \Theta(n^2)$ that makes the expression true.

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$

**Careful!**

"It is understood" that every occurence of an $\mathcal{O}$-symbol (or $\Theta, \Omega, o, \omega$) on the left represents one anonymous function.

Hence, the left side is **not** equal to

$$\Theta(1) + \Theta(2) + \cdots + \Theta(n-1) + \Theta(n)$$

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$

**Careful!**

"It is understood" that every occurence of an $\mathcal{O}$-symbol (or $\Theta, \Omega, o, \omega$) on the left represents one anonymous function.

Hence, the left side is **not** equal to

$$\Theta(1) + \Theta(2) + \cdots + \Theta(n-1) + \Theta(n)$$

# Asymptotic Notation in Equations

How do we interpret an expression like:

$$\sum_{i=1}^{n} \Theta(i) = \Theta(n^2)$$

**Careful!**

"It is understood" that every occurence of an $\mathcal{O}$-symbol (or $\Theta, \Omega, o, \omega$) on the left represents one anonymous function.

Hence, the left side is **not** equal to

$$\Theta(1) + \Theta(2) + \cdots + \Theta(n-1) + \Theta(n)$$

# Asymptotic Notation in Equations

We can view an expression containing asymptotic notation as generating a set:

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n)$$

represents

$$\Big\{f : \mathbb{N} \to \mathbb{R}^+ \mid f(n) = n^2 \cdot g(n) + h(n)$$
$$\text{with } g(n) \in \mathcal{O}(n) \text{ and } h(n) \in \mathcal{O}(\log n)\Big\}$$

# Asymptotic Notation in Equations

Then an asymptotic equation can be interpreted as containement btw. two sets:

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n) = \Theta(n^2)$$

represents

$$n^2 \cdot \mathcal{O}(n) + \mathcal{O}(\log n) \subseteq \Theta(n^2)$$

# Asymptotic Notation

**Lemma 1**

*Let $f, g$ be functions with the property $\exists n_0 > 0\, \forall n \geq n_0 : f(n) > 0$ (the same for $g$). Then*

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant* $c$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

*The expressions also hold for $\Omega$. Note that this means that $f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.*

# Asymptotic Notation

**Lemma 1**

*Let $f, g$ be functions with the property*
$\exists n_0 > 0\ \forall n \geq n_0 : f(n) > 0$ *(the same for $g$). Then*

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant $c$*
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

*The expressions also hold for $\Omega$. Note that this means that*
$f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.

# Asymptotic Notation

**Lemma 1**

*Let $f, g$ be functions with the property $\exists n_0 > 0\ \forall n \geq n_0 : f(n) > 0$ (the same for $g$). Then*

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant* $c$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

*The expressions also hold for $\Omega$. Note that this means that $f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.*

# Asymptotic Notation

**Lemma 1**
*Let $f, g$ be functions with the property $\exists n_0 > 0\, \forall n \geq n_0 : f(n) > 0$ (the same for $g$). Then*

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant* $c$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

*The expressions also hold for $\Omega$. Note that this means that $f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.*

# Asymptotic Notation

### Lemma 1

*Let $f, g$ be functions with the property $\exists n_0 > 0\ \forall n \geq n_0 : f(n) > 0$ (the same for $g$). Then*

- $c \cdot f(n) \in \Theta(f(n))$ *for any constant* $c$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(f(n) + g(n))$
- $\mathcal{O}(f(n)) \cdot \mathcal{O}(g(n)) = \mathcal{O}(f(n) \cdot g(n))$
- $\mathcal{O}(f(n)) + \mathcal{O}(g(n)) = \mathcal{O}(\max\{f(n), g(n)\})$

*The expressions also hold for $\Omega$. Note that this means that $f(n) + g(n) \in \Theta(\max\{f(n), g(n)\})$.*

# Asymptotic Notation

**Comments**

- Do not use asymptotic notation within induction proofs.
- For any constants $a, b$ we have $\log_a n = \Theta(\log_b n)$. Therefore, we will usually ignore the base of a logarithm within asymptotic notation.
- In general $\log n = \log_2 n$, i.e., we use $2$ as the default base for the logarithm.

# Asymptotic Notation

**Comments**

- Do not use asymptotic notation within induction proofs.
- For any constants $a, b$ we have $\log_a n = \Theta(\log_b n)$. Therefore, we will usually ignore the base of a logarithm within asymptotic notation.
- In general $\log n = \log_2 n$, i.e., we use $2$ as the default base for the logarithm.

# Asymptotic Notation

**Comments**

- Do not use asymptotic notation within induction proofs.
- For any constants $a, b$ we have $\log_a n = \Theta(\log_b n)$. Therefore, we will usually ignore the base of a logarithm within asymptotic notation.
- In general $\log n = \log_2 n$, i.e., we use $2$ as the default base for the logarithm.

# Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of $n$.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000\log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

  Clearly $f = o(g)$. However, as long as $\log n \le 1000$ Algorithm B will be more efficient.

# Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of $n$.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000 \log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

  Clearly $f = o(g)$. However, as long as $\log n \le 1000$ Algorithm B will be more efficient.

# Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of $n$.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000 \log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

Clearly $f = o(g)$. However, as long as $\log n \leq 1000$ Algorithm B will be more efficient.

# Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of $n$.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000 \log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

  Clearly $f = o(g)$. However, as long as $\log n \leq 1000$ Algorithm B will be more efficient.

# Asymptotic Notation

In general asymptotic classification of running times is a good measure for comparing algorithms:

- If the running time analysis is tight and actually occurs in practise (i.e., the asymptotic bound is not a purely theoretical worst-case bound), then the algorithm that has better asymptotic running time will always outperform a weaker algorithm for large enough values of $n$.
- However, suppose that I have two algorithms:
  - Algorithm A. Running time $f(n) = 1000 \log n = \mathcal{O}(\log n)$.
  - Algorithm B. Running time $g(n) = \log^2 n$.

  Clearly $f = o(g)$. However, as long as $\log n \leq 1000$ Algorithm B will be more efficient.