Why Constants Matter Less

COMS10017 - (Object-Oriented Programming and) Algorithms

Dr Christian Konrad

Runtime of Algorithms

Runtime of an Algorithm

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in Peak Finding represents the number of unit operations very well

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in PEAK FINDING represents the number of unit operations very well

Which runtime is better?

- $4(n-1)$ (simple peak finding algorithm)
- $5 \log n$ (fast peak finding algorithm)

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in $\textsc{Peak Finding}$ represents the number of unit operations very well

Which runtime is better?

- $4(n-1)$ (simple peak finding algorithm)
- $5 \log n$ (fast peak finding algorithm)
- $0.1n^2$
- $n \log(0.5n)$
- $0.01 \cdot 2^n$

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in PEAK FINDING represents the number of unit operations very well

Which runtime is better?

- $4(n-1)$ (simple peak finding algorithm)
- $5 \log n$ (fast peak finding algorithm)
- $0.1n^2$
- $n \log(0.5n)$
- $0.01 \cdot 2^n$

Answer:

Runtime of Algorithms

Runtime of an Algorithm

- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in PEAK FINDING represents the number of unit operations very well

Which runtime is better?

- $4(n-1)$ (simple peak finding algorithm)
- $5 \log n$ (fast peak finding algorithm)
- $0.1n^2$
- $n \log(0.5n)$
- $0.01 \cdot 2^n$

Answer: It depends...

Runtime of Algorithms

Runtime of an Algorithm

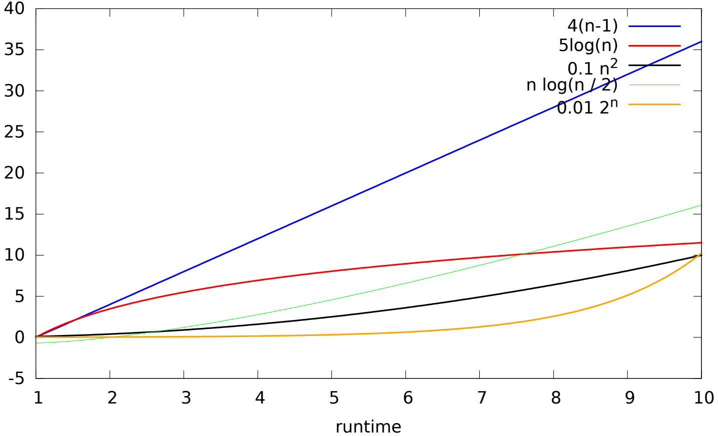
- Function $f : \mathbb{N} \rightarrow \mathbb{N}$ that maps the input length $n \in \mathbb{N}$ to the number of *simple/unit/elementary* operations (worst case, best case, average case, runtime on a specific input, ...)
- The number of array accesses in PEAK FINDING represents the number of unit operations very well

Which runtime is better?

- $4(n-1)$ (simple peak finding algorithm)
- $5 \log n$ (fast peak finding algorithm)
- $0.1n^2$
- $n \log(0.5n)$
- $0.01 \cdot 2^n$

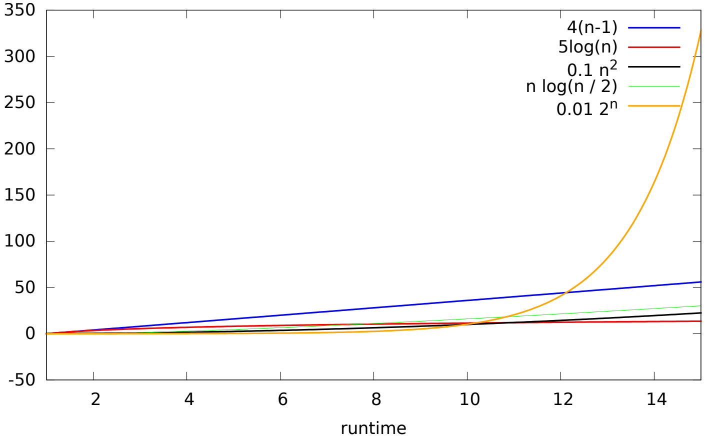
Answer: It depends... But there is a favourite

Runtime Comparisons



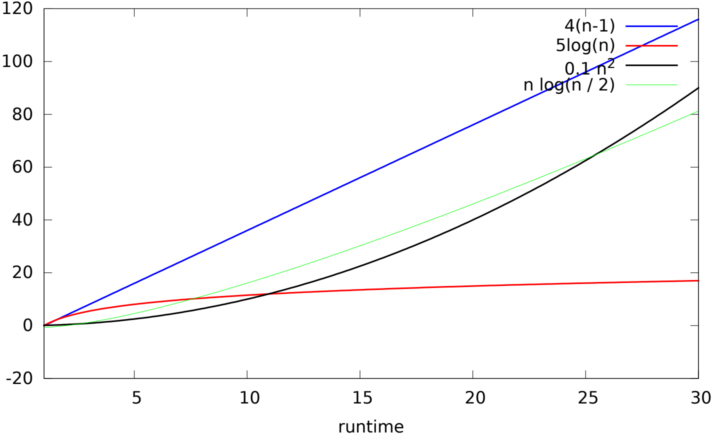
$$0.1n^2 \leq 0.01 \cdot 2^n \leq 5 \log n \leq n \log(n/2) \leq 4(n-1)$$
$$(n = 10)$$

Runtime Comparisons



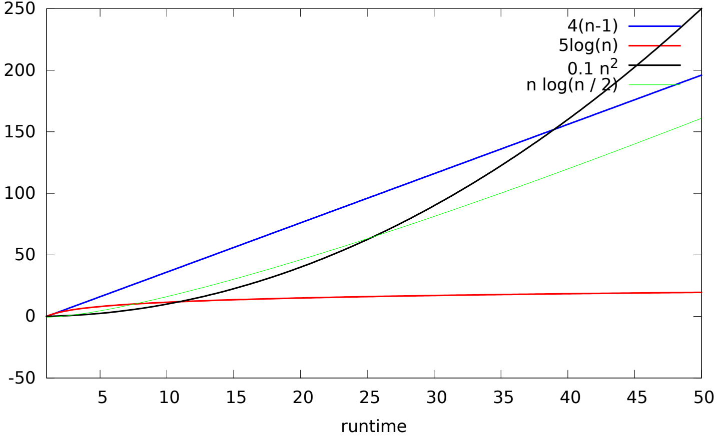
$$5 \log n \le 0.1 n^2 \le n \log(n/2) \le 4(n-1) \le 0.01 \cdot 2^n$$
$$(n = 15)$$

Runtime Comparisons



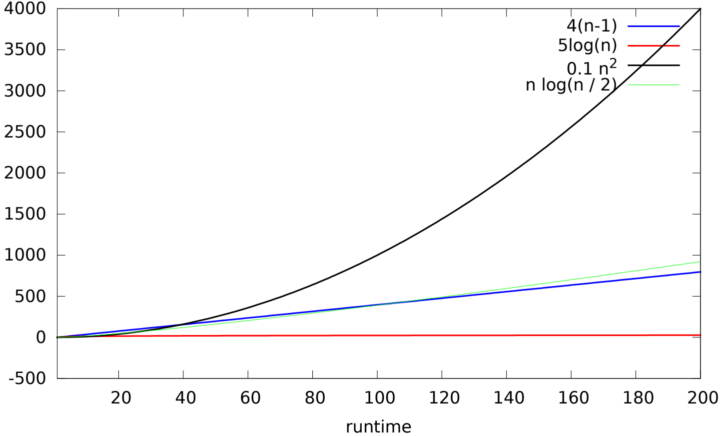
$$5 \log n \leq n \log(n/2) \leq 0.1n^2 \leq 4(n-1)$$
$$(n = 30)$$

Runtime Comparisons



$$5 \log n \leq n \log(n/2) \leq 4(n-1) \leq 0.1n^2$$
$$(n = 50)$$

Runtime Comparisons



$$5 \log n \leq 4(n-1) \leq n \log(n/2) \leq 0.1 n^2$$
$$(n = 200)$$

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

- For large enough n, constants seem to matter less

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

- For large enough n, constants seem to matter less
- For small values of n, most algorithms are fast anyway

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

- For large enough n, constants seem to matter less
- For small values of n, most algorithms are fast anyway (not always true!)

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

- For large enough n, constants seem to matter less
- For small values of n, most algorithms are fast anyway (not always true!)

Solution: Consider asymptotic behavior of functions

Order Functions Disregarding Constants

Aim: We would like to sort algorithms according to their runtime

Is algorithm A faster than algorithm B?

Asymptotic Complexity

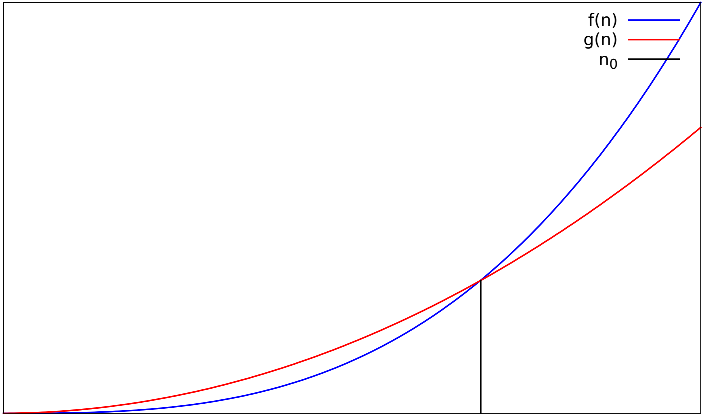
- For large enough n, constants seem to matter less
- For small values of n, most algorithms are fast anyway (not always true!)

Solution: Consider asymptotic behavior of functions

An increasing function $f : \mathbb{N} \rightarrow \mathbb{N}$ grows *asymptotically at least as fast as* an increasing function $g : \mathbb{N} \rightarrow \mathbb{N}$ if there exists an $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$ it holds:

$$f(n) \geq g(n) .$$

Example: f grows at least as fast as g



Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Then $f(n)$ grows asymptotically at least as fast as $g(n)$ since for every $n \geq n_0 = 6$ we have $f(n) \geq g(n)$

Proof:

Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Then $f(n)$ grows asymptotically at least as fast as $g(n)$ since for every $n \geq n_0 = 6$ we have $f(n) \geq g(n)$

Proof: Find values of n for which the following holds:

Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Then $f(n)$ grows asymptotically at least as fast as $g(n)$ since for every $n \geq n_0 = 6$ we have $f(n) \geq g(n)$

Proof: Find values of n for which the following holds:

$$\frac{1}{2}n^2 \quad \geq \quad 3n$$

Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Then $f(n)$ grows asymptotically at least as fast as $g(n)$ since for every $n \geq n_0 = 6$ we have $f(n) \geq g(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2}n^2 &\geq 3n \Rightarrow \\ n &\geq 6 \ . \end{aligned}$$

Example 1

Example: $f(n) = \frac{1}{2}n^2$, $g(n) = 3n$

Then $f(n)$ grows asymptotically at least as fast as $g(n)$ since for every $n \geq n_0 = 6$ we have $f(n) \geq g(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2}n^2 &\geq 3n \Rightarrow \\ n &\geq 6 \ . \end{aligned}$$

Thus, we can chose any $n_0 \geq 6$. □

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof:

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

$$\frac{1}{2} \cdot 2^n \quad \geq \quad 2n^3$$

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2} \cdot 2^n &\geq 2n^3 \\ 2^{n-1} &\geq 2^{3\log n + 1} \quad (\text{using } n = 2^{\log n}) \end{aligned}$$

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2} \cdot 2^n &\geq 2n^3 \\ 2^{n-1} &\geq 2^{3 \log n + 1} \quad (\text{using } n = 2^{\log n}) \\ n - 1 &\geq 3 \log n + 1 \end{aligned}$$

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2} \cdot 2^n &\geq 2n^3 \\ 2^{n-1} &\geq 2^{3 \log n + 1} \quad (\text{using } n = 2^{\log n}) \\ n - 1 &\geq 3 \log n + 1 \\ n &\geq 3 \log n + 2 \end{aligned}$$

Example 2

Example: $f(n) = 2n^3$, $g(n) = \frac{1}{2} \cdot 2^n$

Then $g(n)$ grows asymptotically at least as fast as $f(n)$ since for every $n \geq 16$ we have $g(n) \geq f(n)$

Proof: Find values of n for which the following holds:

$$\begin{aligned} \frac{1}{2} \cdot 2^n &\geq 2n^3 \\ 2^{n-1} &\geq 2^{3 \log n + 1} \quad (\text{using } n = 2^{\log n}) \\ n - 1 &\geq 3 \log n + 1 \\ n &\geq 3 \log n + 2 \end{aligned}$$

This holds for every $n \geq 16$ (which follows from the *racetrack principle*). Thus, we chose any $n_0 \geq 16$. □

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$
- We have: $(n)' =$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$
- We have: $(n)' = 1$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$
- We have: $(n)' = 1$ and $(3 \log n + 2)' =$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$
- We have: $(n)' = 1$ and $(3 \log n + 2)' = \frac{3}{n \ln 2}$

The Racetrack Principle

Racetrack Principle: Let f, g be functions, k an integer and suppose that the following holds:

1. $f(k) \geq g(k)$ and
2. $f'(n) \geq g'(n)$ for every $n \geq k$.

Then for every $n \geq k$, it holds that $f(n) \geq g(n)$.

Example: $n \geq 3 \log n + 2$ holds for every $n \geq 16$

- $n \geq 3 \log n + 2$ holds for $n = 16$
- We have: $(n)' = 1$ and $(3 \log n + 2)' = \frac{3}{n \ln 2} < \frac{1}{2}$ for every $n \geq 16$. The result follows.

Order Functions by Asymptotic Growth

If $\leq$ means *grows asymptotically at least as fast as* then we get:

$$5 \log n \leq 4(n-1) \leq n \log(n/2) \leq 0.1n^2 \leq 0.01 \cdot 2^n$$