 Broken recordings:

B2. Sequences.

Notes originally prepared by Pierre Portal
Editing and expansion by Malcolm Brooks.

Text Reference (Epp)

3ed:	Sections	4.1-4, 8.1-3 (Sequences and induction), 9.3,5 (Sorting)
4ed:	Sections	5.1-4,6-8, (Sequences and induction), 11.3,5 (Sorting)
5ed:	Sections	5.1-4,6-7, (Sequences and induction), 11.3,5 (Sorting)

Sequences

Let S be a set and $I \subseteq \overset{\mathbb{Z}}{\cancel{\mathbb{N}^*}}$. A function $a : \overset{\text{index}}{I} \to S$ is called a **sequence in S**.

Sequences

Let S be a set and $I \subseteq \cancel{\mathbb{N}^{\star}}$ $\mathbb{Z}$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$ $n \mapsto a(n).$	$(a_n)_{n \in I} \subseteq S$

Sequences

Let S be a set and $I \subseteq \mathbb{N}^{\star}$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$, $n \mapsto a(n)$.	$(a_n)_{n \in I} \subseteq S$

The notation $(a_n)_{n \in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.

Sequences

Let S be a set and $I \subseteq \mathbb{N}^\star$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$ $n \mapsto a(n).$	$(a_n)_{n \in I} \subseteq S$

The notation $(a_n)_{n \in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.
(Unlike a *set*, a list has an order, and can have repeated entries.)

Sequences

Let S be a set and $I \subseteq \mathbb{N}^\star$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$ $n \mapsto a(n).$	$(a_n)_{n\in I} \subseteq S$

The notation $(a_n)_{n\in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.
(Unlike a *set*, a list has an order, and can have repeated entries.)
Examples:

- $I = \{1, 2, 3\}$: $(a_n)_{n\in I} = (a_1, a_2, a_3)$.

Sequences

Let S be a set and $I \subseteq \mathbb{N}^{\star}$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$\begin{aligned} a : I &\to S \\ n &\mapsto a(n). \end{aligned}$	$(a_n)_{n\in I} \subseteq S$

The notation $(a_n)_{n\in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.
(Unlike a *set*, a list has an order, and can have repeated entries.)
Examples:

- $I = \{1, 2, 3\}$: $(a_n)_{n\in I} = (a_1, a_2, a_3)$.
- $I = \mathbb{N}^{\star}$: $(a_n)_{n\in I} = (a_0, a_1, a_2, \ldots\ldots)$.

Sequences

Let S be a set and $I \subseteq \mathbb{N}^\star$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$, $n \mapsto a(n)$.	$(a_n)_{n\in I} \subseteq S$

The notation $(a_n)_{n\in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.
(Unlike a *set*, a list has an order, and can have repeated entries.)
Examples:

- $I = \{1, 2, 3\}$: $(a_n)_{n\in I} = (a_1, a_2, a_3)$.
- $I = \mathbb{N}^\star$: $(a_n)_{n\in I} = (a_0, a_1, a_2, \ldots\ldots)$.

In practice we usually leave out the parentheses and speak of "the sequence a_1, a_2, a_3"

Sequences

Let S be a set and $I \subseteq \mathbb{N}^\star$. A function $a : I \to S$ is called a **sequence in S**. Special **sequence notation** is often used:

Function notation	Sequence notation
$a : I \to S$ $n \mapsto a(n).$	$(a_n)_{n\in I} \subseteq S$

The notation $(a_n)_{n\in I}$ indicates that the function can be represented as an *ordered -tuple* or, more simply, as a *list*.
(Unlike a *set*, a list has an order, and can have repeated entries.)
Examples:

- $I = \{1, 2, 3\}$: $(a_n)_{n\in I} = (a_1, a_2, a_3)$.
- $I = \mathbb{N}^\star$: $(a_n)_{n\in I} = (a_0, a_1, a_2, \ldots\ldots)$.

In practice we usually leave out the parentheses and speak of "the sequence a_1, a_2, a_3" or "the sequence $a_0, a_1, a_2, \ldots\ldots$."

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S;

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

The sequence *itself* is **not** a subset of S, since it is not a *set*.

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n\in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

The sequence *itself* is **not** a subset of S, since it is not a *set*.

Examples:

1. $(a_n)_{n\in\mathbb{N}} \subseteq \mathbb{Q}$ sequence of interests rates. a_n is an interest rate at time n.

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n \in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

The sequence *itself* is **not** a subset of S, since it is not a *set*.

Examples:

1. $(a_n)_{n \in \mathbb{N}} \subseteq \mathbb{Q}$ sequence of interests rates. a_n is an interest rate at time n.
2. $(p_n)_{n \in \mathbb{N}} \subseteq \mathbb{N}^\star \times \mathbb{N}^\star \times \mathbb{N}^\star$ sequence of states of an ecosystem. $p_n = (r_n, s_n, t_n)$: population of species r, s, t at time n.

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n\in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

The sequence *itself* is **not** a subset of S, since it is not a *set*.

Examples:

1. $(a_n)_{n\in\mathbb{N}} \subseteq \mathbb{Q}$ sequence of interests rates. a_n is an interest rate at time n.
2. $(p_n)_{n\in\mathbb{N}} \subseteq \mathbb{N}^\star \times \mathbb{N}^\star \times \mathbb{N}^\star$ sequence of states of an ecosystem. $p_n = (r_n, s_n, t_n)$: population of species r, s, t at time n.
3. $(a_n)_{n\in\mathbb{N}} \subseteq \mathbb{N}^\star$ sequence of amplitudes. a_n: amplitude of the harmonic of frequency $n \times f$ (f fundamental frequency).

Examples of Sequences

The "$\subseteq S$" part of the sequence notation $(a_n)_{n\in I} \subseteq S$ indicates that the sequence members belong to S; *i.e.* that the range of the sequence function $a : I \to S$ is a subset of its codomain S.

The sequence *itself* is **not** a subset of S, since it is not a *set*.

Examples:

1. $(a_n)_{n\in\mathbb{N}} \subseteq \overset{\mathbb{R}}{\mathbb{Q}}$ sequence of interests rates. a_n is an interest rate at time n.
2. $(p_n)_{n\in\mathbb{N}} \subseteq \mathbb{N}^\star \times \mathbb{N}^\star \times \mathbb{N}^\star$ sequence of states of an ecosystem. $p_n = (r_n, s_n, t_n)$: population of species r, s, t at time n.
3. $(a_n)_{n\in\mathbb{N}} \subseteq \mathbb{N}^\star$ sequence of amplitudes. a_n: amplitude of the harmonic of frequency $n \times f$ (f fundamental frequency).
4. U set of users. $(u_n)_{n\in\{1,2,3,4,5\}} \subseteq U$: a list of 5 users.

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.

Examples:

1. $\forall n \in \mathbb{N} \quad a_n = 2^n$.

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.

Examples:

1. $\forall n \in \mathbb{N} \ \ a_n = 2^n$. $\quad (a_n)_{n\in\mathbb{N}} \ = \ 2, 4, 8, 16, \ldots,$

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \;\; a_n = 2^n$. $\;\; (a_n)_{n\in\mathbb{N}} \; = \; 2, 4, 8, 16, \ldots,$
2. a_1 = Pierre, a_2 = Julie, a_3 = Paul.
   $(a_n)_{n\in\{1,2,3\}}$ = Pierre, Julie, Paul.

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \ \ a_n = 2^n$. $\ (a_n)_{n\in\mathbb{N}} \ = \ 2, 4, 8, 16, \ldots,$
2. $a_1 =$ Pierre, $a_2 =$ Julie, $a_3 =$ Paul.
   $(a_n)_{n\in\{1,2,3\}} \ =$ Pierre, Julie, Paul.

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \;\; a_n = 2^n$. $\;(a_n)_{n\in\mathbb{N}} \; = \; 2, 4, 8, 16, \ldots,$
2. $a_1 =$ Pierre, $a_2 =$ Julie, $a_3 =$ Paul.
   $(a_n)_{n\in\{1,2,3\}} \; =$ Pierre, Julie, Paul.

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.
Examples:

1. $\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \;\; a_n = 2^n$. $\;\; (a_n)_{n\in\mathbb{N}} = 2, 4, 8, 16, \ldots,$
2. $a_1 =$ Pierre, $a_2 =$ Julie, $a_3 =$ Paul.
   $(a_n)_{n\in\{1,2,3\}} =$ Pierre, Julie, Paul.

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.
Examples:

1. $\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$ $\qquad \begin{aligned} a_1 &= 2 \\ a_2 &= 2(2) = 4 \\ a_3 &= 2(4) = 8 \\ &\ldots\ldots \end{aligned}$

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \;\; a_n = 2^n. \quad (a_n)_{n\in\mathbb{N}} \; = \; 2, 4, 8, 16, \ldots,$
2. a_1 = Pierre, a_2 = Julie, a_3 = Paul.
   $(a_n)_{n\in\{1,2,3\}}$ = Pierre, Julie, Paul.

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.
Examples:

1. $\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$ $\quad \begin{array}{l} a_1{=}2 \\ a_2{=}2(2) = 4 \\ a_3{=}2(4) = 8 \\ \ldots\ldots \end{array} \quad (a_n)_{n\in\mathbb{N}} \; = \; 2, 4, 8, 16, \ldots,$

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \quad a_n = 2^n. \quad (a_n)_{n\in\mathbb{N}} = 2, 4, 8, 16, \ldots,$
2. $a_1 =$ Pierre, $a_2 =$ Julie, $a_3 =$ Paul.
   $(a_n)_{n\in\{1,2,3\}} =$ Pierre, Julie, Paul.

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.
Examples:

1. $\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$ $\quad \begin{array}{l} a_1{=}2 \\ a_2{=}2(2) = 4 \\ a_3{=}2(4) = 8 \\ \ldots\ldots \end{array}$ $\quad (a_n)_{n\in\mathbb{N}} = 2, 4, 8, 16, \ldots,$

2. $\begin{cases} a_{n+1} = -a_n + a_{n-1}, \\ a_2 = 1, \\ a_1 = 0. \end{cases}$

Describing sequences

An **explicit definition** of a sequence is a formula for a_n.
Examples:

1. $\forall n \in \mathbb{N} \quad a_n = 2^n. \quad (a_n)_{n\in\mathbb{N}} \;=\; 2, 4, 8, 16, \ldots,$
2. $a_1 = \text{Pierre},\ a_2 = \text{Julie},\ a_3 = \text{Paul}.$
   $(a_n)_{n\in\{1,2,3\}} \;=\; \text{Pierre, Julie, Paul}.$

An **implicit definition** of a sequence comprises starting value(s) and a relationship between the a_n's.
Examples:

1. $\begin{cases} a_{n+1} = 2a_n, \\ a_1 = 2. \end{cases}$ $\quad \begin{array}{l} a_1{=}2 \\ a_2{=}2(2) = 4 \\ a_3{=}2(4) = 8 \\ \ldots\ldots \end{array} \quad (a_n)_{n\in\mathbb{N}} \;=\; 2, 4, 8, 16, \ldots,$

2. $\begin{cases} a_{n+1} = -a_n + a_{n-1}, \\ a_2 = 1, \\ a_1 = 0. \end{cases}$ $\quad \begin{array}{l} a_1{=}0 \\ a_2{=}1 \end{array} \quad \begin{array}{l} a_3{=}{-1}{+}0{=}{-1} \\ a_4{=}{-}({-1}){+}1{=}2 \\ a_5{=}{-2}{+}({-1}){=}{-3} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{array}$

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N}$ $P(n)$ by mathematical induction:

- Prove $P(1)$.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \; P(n)$?

Proof of $\forall n \in \mathbb{N}$ $P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \ \ P(n)$?

Proof of $\forall n \in \mathbb{N}$ $P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N} \; P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge ... \wedge P(n)$.
- Prove $P(n+1)$.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N}\ \ P(n)$?

Proof of $\forall n \in \mathbb{N}\ P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N}$ $P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then: Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

This works because:

$P(1)$ is true.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N}\ \ P(n)$?

Proof of $\forall n \in \mathbb{N}\ P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then: Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

This works because:

$P(1)$ is true.

$P(1) \implies P(2)$, so $P(2)$ is true.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N} \;\; P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

This works because:
$P(1)$ is true.
$P(1) \implies P(2)$, so $P(2)$ is true.
$P(1) \wedge P(2) \implies P(3)$, so $P(3)$ is true.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N} \;\; P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

This works because:
$P(1)$ is true.
$P(1) \implies P(2)$, so $P(2)$ is true.
$P(1) \wedge P(2) \implies P(3)$, so $P(3)$ is true.
$P(1) \wedge P(2) \wedge P(3) \implies P(4)$, so $P(4)$ is true too.

Mathematical induction

Let $P(n)$ be a predicate with variable $n \in \mathbb{N}$.
How to prove that $\forall n \in \mathbb{N} \;\; P(n)$?

Proof of $\forall n \in \mathbb{N}\;\; P(n)$ by mathematical induction:

- Prove $P(1)$.
- Let $n \in \mathbb{N}$. Treat n as temporarily fixed and then:
  Assume $P(1) \wedge P(2) \wedge P(3) \wedge \ldots \wedge P(n)$.
- Prove $P(n+1)$.

(This is also known as *strong* mathematical induction.)

This works because:

$P(1)$ is true.
$P(1) \implies P(2)$, so $P(2)$ is true.
$P(1) \wedge P(2) \implies P(3)$, so $P(3)$ is true.
$P(1) \wedge P(2) \wedge P(3) \implies P(4)$, so $P(4)$ is true too.
Continuing to argue in this manner gives $P(n)$ for all $n \in \mathbb{N}$.

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$

Can we get an explicit definition?

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3$, $a_2 =$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3$, $a_2 = 9$,

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 =$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 =$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$ **Claim:** $\forall n \in \mathbb{N}$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$

Claim: $\forall n \in \mathbb{N}\ \underbrace{a_n = 3^n}_{P(n)}$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$

Claim: $\forall n \in \mathbb{N}\ a_n = 3^n$.

Proof by mathematical induction that the claim is correct:

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3$, $a_2 = 9$, $a_3 = 27$, $a_4 = 81$,.. **Claim:** $\forall n \in \mathbb{N}$ $a_n = 3^n$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3$, $a_2 = 9$, $a_3 = 27$, $a_4 = 81$,.. **Claim:** $\forall n \in \mathbb{N}$ $\boldsymbol{a_n = 3^n}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n.

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{a_n = 3^n}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$a_{n+1} = 3a_n \quad \text{(from the implicit definition)}$$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81,..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{a_n = 3^n}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} a_{n+1} &= 3a_n \quad \text{(from the implicit definition)} \\ &= 3(3^n) \quad \text{(by the inductive assumption)} \end{aligned}$$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} a_{n+1} = 3a_n & \forall n \in \mathbb{N}, \\ a_1 = 3. \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$ **Claim:** $\forall n \in \mathbb{N}\ a_n = 3^n$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} a_{n+1} &= 3a_n \quad \text{(from the implicit definition)} \\ &= 3(3^n) \quad \text{(by the inductive assumption)} \\ &= 3^{n+1} \end{aligned}$$

From implicit to explicit definitions; Example 1

A sequence is defined implicitly by $\begin{cases} \boldsymbol{a_{n+1} = 3a_n} & \forall n \in \mathbb{N}, \\ \boldsymbol{a_1 = 3.} \end{cases}$

Can we get an explicit definition? First generate some values:

$a_1 = 3,\ a_2 = 9,\ a_3 = 27,\ a_4 = 81, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{a_n = 3^n}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $a_1 = 3^1 = 3$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} a_{n+1} &= 3a_n \quad \text{(from the implicit definition)} \\ &= 3(3^n) \quad \text{(by the inductive assumption)} \\ &= 3^{n+1} \end{aligned}$$

and so the formula is also correct for $n+1$.

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \; \ldots.$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

$$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k,$$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

$$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k, \qquad \prod_{n=1}^{k} a_n = a_1 \times a_2 \times a_3 \times \ldots \times a_k.$$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

$$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k, \qquad \prod_{n=1}^{k} a_n = a_1 \times a_2 \times a_3 \times \ldots \times a_k.$$

Examples: (1) $\sum_{n=1}^{10} n = 1 + 2 + 3 + 4 + \ldots + 9 + 10 = 55.$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

$$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k, \qquad \prod_{n=1}^{k} a_n = a_1 \times a_2 \times a_3 \times \ldots \times a_k.$$

Examples:

(1) $\displaystyle\sum_{n=1}^{10} n = 1 + 2 + 3 + 4 + \ldots + 9 + 10 = 55.$

(2) $\displaystyle\sum_{n=0}^{7} 2^n = 1 + 2 + 4 + 8 + \ldots + 128 = 255.$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

Sigma $$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k,$$ Pi $$\prod_{n=1}^{k} a_n = a_1 \times a_2 \times a_3 \times \ldots \times a_k.$$

Examples: (1) $\sum_{n=1}^{10} n = 1 + 2 + 3 + 4 + \ldots + 9 + 10 = 55.$

(2) $\sum_{n=0}^{7} 2^n = 1 + 2 + 4 + 8 + \ldots + 128 = 255.$

(3) $\prod_{n=1}^{5} n = 1 \times 2 \times 3 \times 4 \times 5 = 5! = 120.$

Sum and products of terms

Terms of a sequence can be summed: $a_1 + a_2 + a_3 + \ldots$ or multiplied: $a_1 \times a_2 \times a_3 \times \ldots$. We use the special notation

$$\sum_{n=1}^{k} a_n = a_1 + a_2 + a_3 + \ldots + a_k, \qquad \prod_{n=1}^{k} a_n = a_1 \times a_2 \times a_3 \times \ldots \times a_k.$$

Examples: (1) $\sum_{n=1}^{10} n = 1 + 2 + 3 + 4 + \ldots + 9 + 10 = 55.$

(2) $\sum_{n=0}^{7} 2^n = 1 + 2 + 4 + 8 + \ldots + 128 = 255.$

(3) $\prod_{n=1}^{5} n = 1 \times 2 \times 3 \times 4 \times 5 = 5! = 120.$

(4) $\prod_{n=1}^{8} n^2 = 4 \times 9 \times 16 \times \ldots \times 64 = 1\,625\,702\,400.$

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n}$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k}}$

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$a_k = a$ (a is the **first term**) $a_{n+1} = ra_n$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $a_n = ar^{n-k}$

When terms or a sequence are summed, we get a **series**. In particular

Geometric Series	
Series of N terms	Sum of N terms
$b_N := \displaystyle\sum_{n=k}^{k+(N-1)} ar^{n-k} = a + ar + \cdots + ar^{N-1}$ [Usually $k=0$ or $k=1$.]	$\begin{cases} \dfrac{a(1-r^N)}{(1-r)} & \text{if } r \neq 1 \\ Na & \text{if } r = 1 \end{cases}$

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n}$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k}}$

When terms or a sequence are summed, we get a **series**. In particular

Geometric Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{n=k}^{k+(N-1)} ar^{n-k} = a + ar + \cdots + ar^{N-1}$ [Usually $k=0$ or $k=1$.]	$\begin{cases} \dfrac{a(1-r^N)}{(1-r)} & \text{if } r \neq 1 \\ Na & \text{if } r = 1 \end{cases}$

Example: $k = 0$, $a = 6$, $r = \frac{1}{2}$, $N = 5$:

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n}$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k}}$

When terms or a sequence are summed, we get a **series**. In particular

Geometric Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{\boldsymbol{n=k}}^{\boldsymbol{k+(N-1)}} \boldsymbol{ar^{n-k}} = a+ar+\cdots+ar^{N-1}$ [Usually $k=0$ or $k=1$.]	$\begin{cases} \dfrac{a(1-r^N)}{(1-r)} & \text{if } r \neq 1 \\ Na & \text{if } r = 1 \end{cases}$

Example: $k = 0,\ a = 6,\ r = \frac{1}{2},\ N = 5$:

$$6+3+\frac{3}{2}+\frac{3}{4}+\frac{3}{8} = \sum_{n=0}^{4} 6\left(\frac{1}{2}\right)^n$$

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n}$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k}}$

When terms or a sequence are summed, we get a **series**. In particular

Geometric Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{\boldsymbol{n=k}}^{\boldsymbol{k+(N-1)}} \boldsymbol{ar^{n-k}} = a+ar+\cdots+ar^{N-1}$ [Usually $k=0$ or $k=1$.]	$\begin{cases} \dfrac{a(1-r^N)}{(1-r)} & \text{if } r \neq 1 \\ Na & \text{if } r = 1 \end{cases}$

Example: $k = 0$, $a = 6$, $r = \frac{1}{2}$, $N = 5$:

$$6+3+\frac{3}{2}+\frac{3}{4}+\frac{3}{8} = \sum_{n=0}^{4} 6\left(\frac{1}{2}\right)^n = \frac{6(1-2^{-5})}{1-2^{-1}}$$

Geometric Sequences and Series

Example 1 (Slide 6) is a special case of

Geometric Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n}$, $\forall n \geq k$ (r is the **common ratio**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k}}$

When terms or a sequence are summed, we get a **series**. In particular

Geometric Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{\boldsymbol{n=k}}^{\boldsymbol{k+(N-1)}} \boldsymbol{ar^{n-k}} = a+ar+\cdots+ar^{N-1}$ [Usually $k=0$ or $k=1$.]	$\begin{cases} \dfrac{a(1-r^N)}{(1-r)} & \text{if } r \neq 1 \\ Na & \text{if } r = 1 \end{cases}$

Example: $k=0,\ a=6,\ r=\frac{1}{2},\ N=5$:

$$6+3+\frac{3}{2}+\frac{3}{4}+\frac{3}{8}=\sum_{n=0}^{4}6\left(\frac{1}{2}\right)^n=\frac{6(1-2^{-5})}{1-2^{-1}}=\frac{6(32-1)}{32}\cdot\frac{2}{1}=\frac{93}{8}=11\frac{5}{8}$$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition?

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n =}$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} b_{n+1} = b_n + 5 & \forall n \in \mathbb{N}, \\ b_1 = 0. \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ b_n = 5(n-1)$.

Proof by mathematical induction that the claim is correct:

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} b_{n+1} = b_n + 5 & \forall n \in \mathbb{N}, \\ b_1 = 0. \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ b_n = 5(n-1)$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n.

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$b_{n+1} = b_n + 5 \quad \text{(from the implicit definition)}$$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} b_{n+1} &= b_n + 5 \quad \text{(from the implicit definition)} \\ &= 5(n-1) + 5 \quad \text{(by the inductive assumption)} \end{aligned}$$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}$ $\boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} b_{n+1} &= b_n + 5 \quad \text{(from the implicit definition)} \\ &= 5(n-1) + 5 \quad \text{(by the inductive assumption)} \\ &= 5n \end{aligned}$$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} b_{n+1} &= b_n + 5 \quad \text{(from the implicit definition)} \\ &= 5(n-1) + 5 \quad \text{(by the inductive assumption)} \\ &= 5n \ = \ 5((n+1) - 1) \end{aligned}$$

From implicit to explicit definitions; Example 2

A sequence is defined implicitly by $\begin{cases} \boldsymbol{b_{n+1} = b_n + 5} & \forall n \in \mathbb{N}, \\ \boldsymbol{b_1 = 0.} \end{cases}$

Can we get an explicit definition? First generate some values:

$b_1 = 0,\ b_2 = 5,\ b_3 = 10,\ b_4 = 15, ..$ **Claim:** $\forall n \in \mathbb{N}\ \boldsymbol{b_n = 5(n-1)}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 1$, formula gives $b_1 = 0$, agreeing with the definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} b_{n+1} &= b_n + 5 \quad \text{(from the implicit definition)} \\ &= 5(n-1) + 5 \quad \text{(by the inductive assumption)} \\ &= 5n = 5((n+1) - 1) \end{aligned}$$

and so the formula is also correct for $n+1$.

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

When the terms or an arithmetic sequence are summed, we get:

Arithmetic Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{n=k}^{k+(N-1)} [\boldsymbol{a+(n-k)d}] = a+(a+d)+\cdots+(a+(N-1)d)$ [Usually $k=0$ or $k=1$.]	$\frac{N}{2}[2a+(N-1)d]$ $= N\left[\frac{a \;+\; (a+(N-1)d)}{2}\right]$

Exercise: prove this by induction.

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

When the terms or an arithmetic sequence are summed, we get:

Arithmetic Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{\boldsymbol{n=k}}^{\boldsymbol{k+(N-1)}} [\boldsymbol{a+(n-k)d}] = a+(a+d)+\cdots+(a+(N-1)d)$ [Usually $k=0$ or $k=1$.]	$\frac{N}{2}[2a+(N-1)d]$ $= N\left[\frac{a\ +\ (a+(N-1)d)}{2}\right]$

The second sum formula can be expressed as

sum = number of terms times average of first and last

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

When the terms or an arithmetic sequence are summed, we get:

Arithmetic Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{n=k}^{k+(N-1)} [\boldsymbol{a+(n-k)d}] = a+(a+d)+\cdots+(a+(N-1)d)$ [Usually $k=0$ or $k=1$.]	$\frac{N}{2}\left[2a+(N-1)d\right]$ $= N\left[\frac{a\ +\ (a+(N-1)d)}{2}\right]$

The second sum formula can be expressed as

sum = number of terms times average of first and last

Example: $1+3+5+7+\cdots+99$

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

When the terms or an arithmetic sequence are summed, we get:

Arithmetic Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{n=k}^{k+(N-1)} [\boldsymbol{a+(n-k)d}] = a+(a+d)+\cdots+(a+(N-1)d)$ [Usually $k=0$ or $k=1$.]	$\frac{N}{2}[2a+(N-1)d]$ $= N\left[\frac{a + (a+(N-1)d)}{2}\right]$

The second sum formula can be expressed as

sum = number of terms times average of first and last

Example: $1+3+5+7+\cdots+99 = 50\left(\frac{1+99}{2}\right)$

Arithmetic Sequences and Series

Example 2 (Slide 9) is a special case of

Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = a_n + d}$, $\forall n \geq k$ (d is the **common difference**)	$\forall n \geq k$ $\boldsymbol{a_n = a + (n-k)d}$

When the terms or an arithmetic sequence are summed, we get:

Arithmetic Series	
Series of $\boldsymbol{N}$ terms	Sum of $\boldsymbol{N}$ terms
$\displaystyle\sum_{n=k}^{k+(N-1)} [\boldsymbol{a+(n-k)d}] = a+(a+d)+\cdots+(a+(N-1)d)$ [Usually $k=0$ or $k=1$.]	$\frac{N}{2}[2a+(N-1)d]$ $= N\left[\frac{a + (a+(N-1)d)}{2}\right]$

The second sum formula can be expressed as

sum = number of terms times average of first and last

Example: $1+3+5+7+\cdots+99 = 50\left(\dfrac{1+99}{2}\right) = 2500.$

Simplest Malthusian model in population dynamics

(Exponential growth)


T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of bacteria ~~individuals~~ in a population after n years.

Assume the population doubles every year,

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star\ p_{n+1} = 2p_n$.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \; p_{n+1} = 2p_n$.

Assume the current population is five million.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \ p_{n+1} = 2p_n$.

Assume the current population is five million. *i.e.* $p_0 = 5 \times 10^6$.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \; p_{n+1} = 2p_n$.

Assume the current population is five million. *i.e.* $p_0 = 5 \times 10^6$.

Questions: 1. What will the population size be in 10 years time?

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \; p_{n+1} = 2p_n$.

Assume the current population is five million. *i.e.* $p_0 = 5 \times 10^6$.

Questions:
1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \; p_{n+1} = 2p_n$.

Assume the current population is five million. *i.e.* $p_0 = 5 \times 10^6$.

Questions:
1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall \boldsymbol{n} \in \mathbb{N}^\star \; \boldsymbol{p_{n+1} = 2p_n}$.

Assume the current population is five million. *i.e.* $\boldsymbol{p_0 = 5 \times 10^6}$.

Questions: 1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers: 1. $a_{10} = 5 \times 10^6 \times 2^{10} = \mathbf{5{\cdot}12 \times 10^9}$

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star \; \boldsymbol{p_{n+1} = 2p_n}$.

Assume the current population is five million. *i.e.* $\boldsymbol{p_0 = 5 \times 10^6}$.

Questions:
1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers:
1. $a_{10} = 5 \times 10^6 \times 2^{10} = \boldsymbol{5.12 \times 10^9}$
2. We need N so that $a_N \geq 10^9 > a_{\cancel{n}}$. $N-1$

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\forall n \in \mathbb{N}^\star\ p_{n+1} = 2p_n$.

Assume the current population is five million. *i.e.* $p_0 = 5 \times 10^6$.

Questions:
1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers:
1. $a_{10} = 5 \times 10^6 \times 2^{10} = \mathbf{5{\cdot}12 \times 10^9}$
2. We need N so that $a_N \geq 10^9 > a_{n_1}$.
   From 1. $a_8 = \frac{1}{4}(5{\cdot}12 \times 10^9) = 1{\cdot}28 \times 10^9$

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\boldsymbol{\forall n \in \mathbb{N}^\star \ p_{n+1} = 2p_n}$.

Assume the current population is five million. *i.e.* $\boldsymbol{p_0 = 5 \times 10^6}$.

Questions: 1. What will the population size be in 10 years time?
2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers: 1. $a_{10} = 5 \times 10^6 \times 2^{10} = \mathbf{5{\cdot}12 \times 10^9}$

2. We need N so that $a_N \geq 10^9 > a_{n_1}$.
From 1. $a_8 = \frac{1}{4}(5{\cdot}12 \times 10^9) = 1{\cdot}28 \times 10^9$
and $a_7 = \frac{1}{8}(5{\cdot}12 \times 10^9) = 0{\cdot}64 \times 10^9$.

Simplest Malthusian model in population dynamics

(Exponential growth)

T. Robert Malthus 1766 - 1834

Let p_n be number of individuals in a population after n years.

Assume the population doubles every year, *i.e.* $\boldsymbol{\forall n \in \mathbb{N}^\star \; p_{n+1} = 2p_n}$.

Assume the current population is five million. *i.e.* $\boldsymbol{p_0 = 5 \times 10^6}$.

Questions: 1. What will the population size be in 10 years time?

2. When will the population size reach 10^9 ?

We have a geometric sequence with $k = 0$, $a = 5 \times 10^6$, $r = 2$. Hence:

Answers: 1. $a_{10} = 5 \times 10^6 \times 2^{10} = \mathbf{5{\cdot}12 \times 10^9}$

2. We need N so that $a_N \geq 10^9 > a_{n_1}$.

From 1. $a_8 = \frac{1}{4}(5{\cdot}12 \times 10^9) = 1{\cdot}28 \times 10^9$

and $a_7 = \frac{1}{8}(5{\cdot}12 \times 10^9) = 0{\cdot}64 \times 10^9$.

So $\boldsymbol{N = 8}$ *i.e.* eight years.

Compound Interest

Let c_n be the capital (in \$) after n years.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year,

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall \boldsymbol{n} \in \mathbb{N}^\star\ \boldsymbol{c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall \boldsymbol{n} \in \mathbb{N}^\star\ \boldsymbol{c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Assume the current capital is \$20K, *i.e.* $\boldsymbol{c_0 = 2 \times 10^4}$.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall n \in \mathbb{N}^\star \; c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n$.

Assume the current capital is \$20K, *i.e.* $c_0 = 2 \times 10^4$.

Question: What will the capital be in 10 years?

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall n \in \mathbb{N}^\star \; c_{n+1} = c_n + 0.03c_n = 1.03c_n$.

Assume the current capital is \$20K, *i.e.* $c_0 = 2 \times 10^4$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star \; c_n = 2 \times 10^4 \times 1.03^n$.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall n \in \mathbb{N}^\star \; c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n$.

Assume the current capital is \$20K, *i.e.* $c_0 = 2 \times 10^4$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star \; c_n = 2 \times 10^4 \times 1.03^n$.

$$c_{10} = 2 \times 10^4 \times 1.03^{10} = \$26\,878:33.$$

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.* $\forall n \in \mathbb{N}^\star$ $c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n$.

Assume the current capital is \$20K, *i.e.* $c_0 = 2 \times 10^4$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star$ $c_n = 2 \times 10^4 \times 1.03^n$.

$$c_{10} = 2 \times 10^4 \times 1.03^{10} = \mathbf{\$26\,878:33}.$$

Monthly compounding:

Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\boldsymbol{\forall n \in \mathbb{N}^\star\ c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Assume the current capital is \$20K, *i.e.* $\boldsymbol{c_0 = 2 \times 10^4}$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star\ c_n = 2\times 10^4 \times 1.03^n$.
$c_{10} = 2\times 10^4 \times 1.03^{10} = \mathbf{\$26\,878:33}$.

Monthly compounding:
Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Question: Is it the same as 3% per year?

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\boldsymbol{\forall n \in \mathbb{N}^\star\ c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Assume the current capital is \$20K, *i.e.* $\boldsymbol{c_0 = 2 \times 10^4}$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star\ c_n = 2 \times 10^4 \times 1.03^n$.
$c_{10} = 2 \times 10^4 \times 1.03^{10} = \mathbf{\$26\,878 : 33}$.

Monthly compounding:
Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Question: Is it the same as 3% per year?

Answer: Now n is time in months and d_n capital at time n.

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall \boldsymbol{n} \in \mathbb{N}^\star\ \boldsymbol{c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Assume the current capital is \$20K, *i.e.* $\boldsymbol{c_0 = 2 \times 10^4}$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star\ c_n = 2 \times 10^4 \times 1.03^n$.
$$c_{10} = 2 \times 10^4 \times 1.03^{10} = \boldsymbol{\$26\,878:33}.$$

Monthly compounding:
Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Question: Is it the same as 3% per year?

Answer: Now n is time in months and d_n capital at time n.
$$\boldsymbol{d_0 = 2 \times 10^4,\quad \forall n \in \mathbb{N}\ d_{n+1} = d_n + 0.0025d_n = 1.0025d_n}.$$

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.*
$\forall n \in \mathbb{N}^\star \; c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n$.

Assume the current capital is \$20K, *i.e.* $c_0 = 2 \times 10^4$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star \; c_n = 2 \times 10^4 \times 1.03^n$.

$$c_{10} = 2 \times 10^4 \times 1.03^{10} = \mathbf{\$26\,878.33}.$$

Monthly compounding:

Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Question: Is it the same as 3% per year?

Answer: Now n is time in months and d_n capital at time n.

$$d_0 = 2 \times 10^4, \quad \forall n \in \mathbb{N} \; d_{n+1} = d_n + 0.0025d_n = 1.0025d_n.$$

$$\underbrace{d_{120}}_{\text{10 years}} = 2 \times 10^4 \times 1.0025^{120} = \underline{\mathbf{\$26\,987.71}}.$$

Compound Interest

Let c_n be the capital (in \$) after n years.

Assume the capital yields 3% interest every year, *i.e.* $\boldsymbol{\forall n \in \mathbb{N}^\star \; c_{n+1} = c_n + 0.03c_n = 1{\cdot}03c_n}$.

Assume the current capital is \$20K, *i.e.* $\boldsymbol{c_0 = 2\times 10^4}$.

Question: What will the capital be in 10 years?

Answer: $\forall n \in \mathbb{N}^\star \; c_n = 2\times 10^4 \times 1.03^n$.

$$c_{10} = 2\times 10^4 \times 1.03^{10} = \mathbf{\$26\,878:33}.$$

Monthly compounding:

Now suppose the capital yields $\frac{3}{12} = 0.25\%$ interest per *month*.

Question: Is it the same as 3% per year?

Answer: Now n is time in months and d_n capital at time n.

$$\boldsymbol{d_0 = 2\times 10^4, \quad \forall n \in \mathbb{N} \; d_{n+1} = d_n + 0.0025d_n = 1.0025d_n}.$$

$$d_{120} = 2\times 10^4 \times 1.0025^{120} = \mathbf{\$26\,987:71}. \quad \textbf{Slightly better!}$$

Better models

Compound Interest: If the bank is charging a fee of \$10 per year, the compound interest model becomes

$$\begin{cases} c_{n+1} = 1.03c_n - 10 \quad \forall n \in \mathbb{N}^\star, \\ c_0 = 2 \times 10^4. \end{cases}$$

Better models

Compound Interest: If the bank is charging a fee of \$10 per year, the compound interest model becomes

$$\begin{cases} c_{n+1} = 1.03c_n - 10 \quad \forall n \in \mathbb{N}^\star, \\ c_0 = 2 \times 10^4. \end{cases}$$

Population growth: If there is some immigration, bringing 10^3 new individuals to the population each year, the population dynamics model becomes

$$\begin{cases} p_{n+1} = 2p_n + 10^3 \quad \forall n \in \mathbb{N}^\star, \\ p_0 = 5 \times 10^6. \end{cases}$$

Better models

Compound Interest: If the bank is charging a fee of \$10 per year, the compound interest model becomes

$$\begin{cases} c_{n+1} = 1.03c_n - 10 \quad \forall n \in \mathbb{N}^\star, \\ c_0 = 2 \times 10^4. \end{cases}$$

Population growth: If there is some immigration, bringing 10^3 new individuals to the population each year, the population dynamics model becomes

$$\begin{cases} p_{n+1} = 2p_n + 10^3 \quad \forall n \in \mathbb{N}^\star, \\ p_0 = 5 \times 10^6. \end{cases}$$

The sequences defined by these models are neither geometric nor arithmetic, but are a generalisation of both.

Better models

Compound Interest: If the bank is charging a fee of \$10 per year, the compound interest model becomes

$$\begin{cases} c_{n+1} = 1.03c_n - 10 \quad \forall n \in \mathbb{N}^\star, \\ c_0 = 2 \times 10^4. \end{cases}$$

Population growth: If there is some immigration, bringing 10^3 new individuals to the population each year, the population dynamics model becomes

$$\begin{cases} p_{n+1} = 2p_n + 10^3 \quad \forall n \in \mathbb{N}^\star, \\ p_0 = 5 \times 10^6. \end{cases}$$

The sequences defined by these models are neither geometric nor arithmetic, but are a generalisation of both.
We need to start again from scratch.

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} p_{n+1} = 2p_n + d \ \forall n \in \mathbb{N}^\star, \\ p_0 = p. \end{cases}$$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$,

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p + d,$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p+d, \quad p_2 = 2(2p+d)+d =$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p+d, \quad p_2 = 2(2p+d)+d = 4p+3d,$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p + d, \quad p_2 = 2(2p+d) + d = 4p + 3d,$

$p_3 =$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^{\star}, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p + d, \quad p_2 = 2(2p + d) + d = 4p + 3d,$

$p_3 = 8p + 7d,$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} p_{n+1} = 2p_n + d \ \forall n \in \mathbb{N}^\star, \\ p_0 = p. \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $\quad p_1 = 2p + d$, $\quad p_2 = 2(2p+d) + d = 4p + 3d$,
$p_3 = 8p + 7d$,

Claim: $\forall n \in \mathbb{N}^\star \quad p_n = \underbrace{2^n p} + \underbrace{(2^n - 1)d}$.

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p}. \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $\quad p_1 = 2p + d$, $\quad p_2 = 2(2p+d) + d = 4p + 3d$,

$p_3 = 8p + 7d$,

Claim: $\forall n \in \mathbb{N}^\star \ \ \boldsymbol{p_n = 2^n p + (2^n - 1)d}$.

Proof by mathematical induction that the claim is correct:

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $p_1 = 2p + d$, $p_2 = 2(2p+d)+d = 4p+3d$,

$p_3 = 8p + 7d$,

Claim: $\forall n \in \mathbb{N}^\star$ $\boldsymbol{p_n = 2^n p + (2^n - 1)d}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p + d, \quad p_2 = 2(2p + d) + d = 4p + 3d,$

$p_3 = 8p + 7d,$ **Claim:** $\forall n \in \mathbb{N}^\star \ \ \boldsymbol{p_n = 2^n p + (2^n - 1)d}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n.

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p}. \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $\quad p_1 = 2p + d$, $\quad p_2 = 2(2p+d)+d = 4p+3d$,

$p_3 = 8p + 7d$, $\qquad$ **Claim:** $\forall n \in \mathbb{N}^\star \ \ \boldsymbol{p_n = 2^n p + (2^n - 1)d}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$p_{n+1} = 2p_n + d \quad \text{(from the implicit definition)}$$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $\quad p_1 = 2p + d$, $\quad p_2 = 2(2p+d) + d = 4p + 3d$,

$p_3 = 8p + 7d$,

Claim: $\forall n \in \mathbb{N}^\star \ \ \boldsymbol{p_n = 2^n p + (2^n - 1)d}$.

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} p_{n+1} &= 2p_n + d \quad \text{(from the implicit definition)} \\ &= 2(\underline{2^n p + (2^n - 1)d}) + d \quad \text{(by the inductive assumption)} \end{aligned}$$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p, \quad p_1 = 2p + d, \quad p_2 = 2(2p+d) + d = 4p + 3d,$
$p_3 = 8p + 7d,$

Claim: $\forall n \in \mathbb{N}^\star \ \ \boldsymbol{p_n = 2^n p + (2^n - 1)d}.$

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} p_{n+1} &= 2p_n + d \quad \text{(from the implicit definition)} \\ &= 2(2^n p + (2^n - 1)d) + d \quad \text{(by the inductive assumption)} \\ &= \underbrace{2^{n+1}}\, p + (2^{n+1} - 1)d \quad \text{(by algebraic simplification)} \end{aligned}$$

From implicit to explicit definitions; Example 3

We seek an explicit formula for the population given by the implicit formula at right, where d and p are shorthand for 10^3 and 5×10^6.

$$\begin{cases} \boldsymbol{p_{n+1} = 2p_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{p_0 = p.} \end{cases}$$

We start by generating the first few terms of the sequence:

$p_0 = p$, $\quad p_1 = 2p + d$, $\quad p_2 = 2(2p+d)+d = 4p+3d$,

$p_3 = 8p + 7d$,

Claim: $\forall n \in \mathbb{N}^\star$ $\left(\boldsymbol{p_n = 2^n p + (2^n - 1)d.}\right) = P(n)$

Proof by mathematical induction that the claim is correct:

Basis step: For $n = 0$, formula gives $p_0 = 2^0 p + (2^0 - 1)d = p$, agreeing with the implicit definition.

Inductive step: Assume the formula is correct for up to and including some fixed n. Then

$$\begin{aligned} p_{n+1} &= 2p_n + d \quad \text{(from the implicit definition)} \\ &= 2(2^n p + (2^n - 1)d) + d \quad \text{(by the inductive assumption)} \\ &= 2^{n+1} p + (2^{n+1} - 1)d \quad \text{(by algebraic simplification)} \end{aligned}$$

and so the formula is also correct for $n+1$.

So by mathematical induction, $\forall n$, $P(n)$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} c_{n+1} = rc_n + d \ \forall n \in \mathbb{N}^\star, \\ c_0 = c. \end{cases}$$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2\times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c$,

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c, \quad c_1 = rc + d,$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc + d) + d =$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc+d) + d = r^2 c + (r+1)d,$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc + d) + d = r^2c + (r+1)d,$

$c_3 = r(r^2c + (r+1)d) + d$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc + d) + d = r^2 c + (r+1)d,$

$c_3 = r(r^2 c + (r+1)d) + d = r^3 c + (r^2 + r + 1)d.$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:
$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc + d) + d = r^2 c + (r+1)d,$
$c_3 = r(r^2 c + (r+1)d) + d = r^3 c + (r^2 + r + 1)d.$
So we guess that $c_n = r^n c + (1 + r + r^2 + \cdots + r^{n-1})d.$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c}. \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c$, $\quad c_1 = rc + d$, $\quad c_2 = r(rc+d) + d = r^2c + (r+1)d$,

$c_3 = r(r^2c + (r+1)d) + d = r^3c + (r^2 + r + 1)d$.

So we guess that $c_n = r^n c + (\underbrace{1 + r + r^2 + \cdots + r^{n-1}}_{\text{sum of geom. sequence}})d$.

Using the formula for the sum of a geometric series (Slide 8), this simplifies to

Claim: $\forall n \in \mathbb{N}^\star \quad \boldsymbol{c_n = r^n c + \underline{\left(\frac{1-r^n}{1-r}\right)} d}$.

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2 \times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:
$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc+d) + d = r^2c + (r+1)d,$
$c_3 = r(r^2c + (r+1)d) + d = r^3c + (r^2 + r + 1)d.$
So we guess that $c_n = r^n c + (1 + r + r^2 + \cdots + r^{n-1})d$.

Using the formula for the sum of a geometric series (Slide 8), this simplifies to

Claim: $\forall n \in \mathbb{N}^\star \quad \boldsymbol{c_n = r^n c + \left(\frac{1 - r^n}{1 - r}\right) d}.$

As with the previous examples, this claim can be verified using proof by **mathematical induction**. Try it!

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2\times 10^4$.

$$\begin{cases} c_{n+1} = rc_n + d \ \forall n \in \mathbb{N}^\star, \\ c_0 = c. \end{cases}$$

We start by generating the first few terms of the sequence:

$c_0 = c$, $\quad c_1 = rc + d$, $\quad c_2 = r(rc+d)+d = r^2c + (r+1)d$,
$c_3 = r(r^2c + (r+1)d) + d = r^3c + (r^2+r+1)d$.
So we guess that $c_n = r^n c + (1+r+r^2+\cdots+r^{n-1})d$.

Using the formula for the sum of a geometric series (Slide 8), this simplifies to

Claim: $\forall n \in \mathbb{N}^\star \quad c_n = r^n c + \left(\dfrac{1-r^n}{1-r}\right) d.$

As with the previous examples, this claim can be verified using proof by **mathematical induction**. Try it!

Applying the formula gives

$$c_{10} = (1.03)^{10}(2\times 10^4) - \left(\frac{1-(1.03)^{10}}{1-1.03}\right)10 = 26\,878.33 - 114.64.$$

From implicit to explicit definitions; Example 4

We seek an explicit formula for the investment capital given by the implicit formula at right, where $r = 1.03$, $d = -10$ and $c = 2\times 10^4$.

$$\begin{cases} \boldsymbol{c_{n+1} = rc_n + d} \ \forall n \in \mathbb{N}^\star, \\ \boldsymbol{c_0 = c.} \end{cases}$$

We start by generating the first few terms of the sequence:
$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc+d)+d = r^2c+(r+1)d,$
$c_3 = r(r^2c+(r+1)d)+d = r^3c+(r^2+r+1)d.$
So we guess that $c_n = r^nc + (1+r+r^2+\cdots+r^{n-1})d$.

Using the formula for the sum of a geometric series (Slide 8), this simplifies to

Claim: $\forall n \in \mathbb{N}^\star \quad \boldsymbol{c_n = r^nc + \left(\frac{1-r^n}{1-r}\right)d}.$

As with the previous examples, this claim can be verified using proof by **mathematical induction**. Try it!

Applying the formula gives
$c_{10} = (1.03)^{10}(2\times 10^4) - \left(\frac{1-(1.03)^{10}}{1-1.03}\right)10 = 26\,878.33 - 114.64.$
So the \$10 annual fee over 10 years costs the investment \$114.64.

Mixed Sequences

It takes very little extra analysis to generalise the previous example:

Mixed Geometric-Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n + d}$, $\forall n \geq k$ ($r \neq 1$ is the **multiplier** and d is the **offset**)	$\forall n \geq k$ $$\boldsymbol{a_n = ar^{n-k} + \left(\frac{1 - r^{n-k}}{1 - r}\right) d}$$

Mixed Sequences

It takes very little extra analysis to generalise the previous example:

Mixed Geometric-Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n + d}$, $\forall n \geq k$ ($r \neq 1$ is the **multiplier** and d is the **offset**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k} + \left(\frac{1-r^{n-k}}{1-r}\right)d}$

Example: A mixed geometric-arithmetic sequence $(a_n)_{n\in\mathbb{N}}$ has multiplier $\frac{1}{2}$, offset 2 and first term 1. What is the 10-th term?

Mixed Sequences

It takes very little extra analysis to generalise the previous example:

Mixed Geometric-Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n + d}$, $\forall n \geq k$ ($r \neq 1$ is the **multiplier** and d is the **offset**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k} + \left(\frac{1-r^{n-k}}{1-r}\right)d}$

Example: A mixed geometric-arithmetic sequence $(a_n)_{n\in\mathbb{N}}$ has multiplier $\frac{1}{2}$, offset 2 and first term 1. What is the 10-th term?

Answer: As $k=1$, $a_{10} = 1(\frac{1}{2})^9 + \left(\frac{1-(\frac{1}{2})^9}{1-\frac{1}{2}}\right)2 = 4-3(\frac{1}{2})^9 \approx 3.99.$

Mixed Sequences

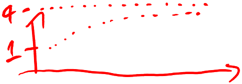
It takes very little extra analysis to generalise the previous example:

Mixed Geometric-Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n + d}$, $\forall n \geq k$ ($r \neq 1$ is the **multiplier** and d is the **offset**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k} + \left(\frac{1-r^{n-k}}{1-r}\right)d}$

Example: A mixed geometric-arithmetic sequence $(a_n)_{n\in\mathbb{N}}$ has multiplier $\frac{1}{2}$, offset 2 and first term 1. What is the 10-th term?

Answer: As $k=1$, $a_{10} = 1(\frac{1}{2})^9 + \left(\frac{1-(\frac{1}{2})^9}{1-\frac{1}{2}}\right)2 = 4-3(\frac{1}{2})^9 \approx 3.99.$

Remark: For this sequence, as n increases a_n approaches 4 ever more closely.



Mixed Sequences

It takes very little extra analysis to generalise the previous example:

Mixed Geometric-Arithmetic Sequence	
Implicit Definition	Explicit Definition
$\boldsymbol{a_k = a}$ (a is the **first term**) $\boldsymbol{a_{n+1} = ra_n + d}$, $\forall n \geq k$ ($r \neq 1$ is the **multiplier** and d is the **offset**)	$\forall n \geq k$ $\boldsymbol{a_n = ar^{n-k} + \left(\frac{1-r^{n-k}}{1-r}\right)d}$

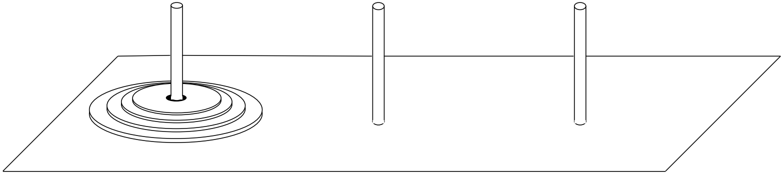
Example: A mixed geometric-arithmetic sequence $(a_n)_{n\in\mathbb{N}}$ has multiplier $\frac{1}{2}$, offset 2 and first term 1. What is the 10-th term?

Answer: As $k = 1$, $a_{10} = 1(\frac{1}{2})^9 + \left(\frac{1-(\frac{1}{2})^9}{1-\frac{1}{2}}\right)2 = 4 - 3(\frac{1}{2})^9 \approx 3.99.$

Remark: For this sequence, as n increases a_n approaches 4 ever more closely. In fact the value 4 is called the **steady state** of the sequence, because if $a_n = 4$ then from the implicit definition $a_{n+1} = (\frac{1}{2})4 + 2 = 4$, so the sequence values remain at 4 for ever.

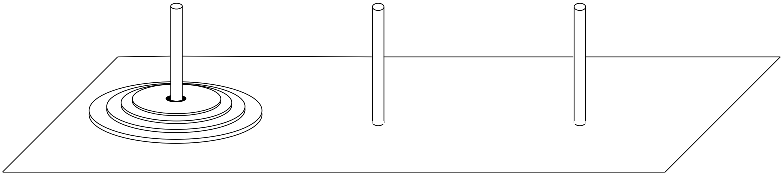
Counting moves: Towers of Hanoi

"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.



Counting moves: Towers of Hanoi

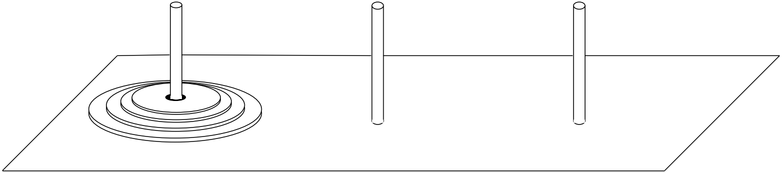
“The Towers of Hanoi” is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.



Aim: transfer all discs to the rightmost peg according to the following rules.

Counting moves: Towers of Hanoi

"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.

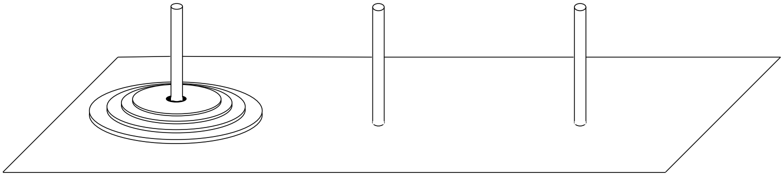


Aim: transfer all discs to the rightmost peg according to the following rules.

- You may move only one disc at a time to one of the other two pegs.

Counting moves: Towers of Hanoi

"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.

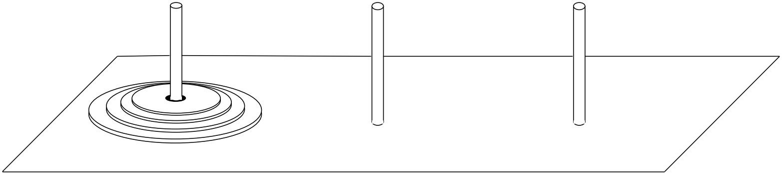


Aim: transfer all discs to the rightmost peg according to the following rules.

- You may move only one disc at a time to one of the other two pegs.
- You can only move the discs that are at the top of one of the piles.

Counting moves: Towers of Hanoi

"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.

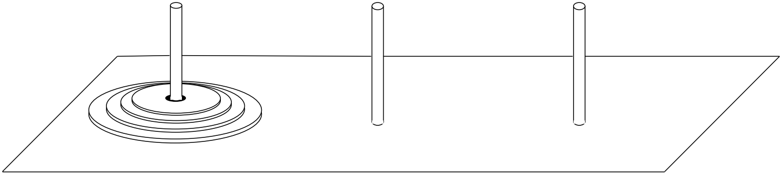


Aim: transfer all discs to the rightmost peg according to the following rules.

- You may move only one disc at a time to one of the other two pegs.
- You can only move the discs that are at the top of one of the piles.
- No disc may sit on top of a smaller disc.

Counting moves: Towers of Hanoi

"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.



Aim: transfer all discs to the rightmost peg according to the following rules.

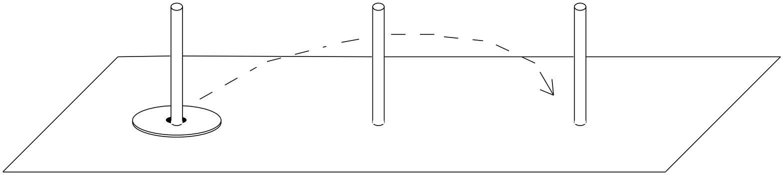
- You may move only one disc at a time to one of the other two pegs.
- You can only move the discs that are at the top of one of the piles.
- No disc may sit on top of a smaller disc.

At one move per second, how fast can you solve a puzzle with 64 discs?

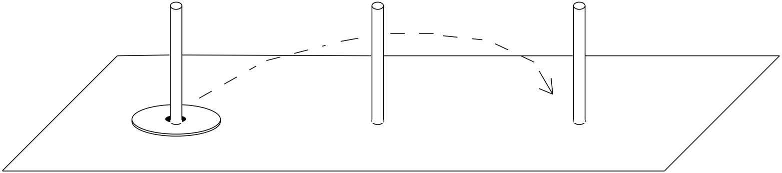
Assume you have n discs (we are ultimately interested in $n = 64$).

Assume you have n discs (we are ultimately interested in $n = 64$).
Let x_n be the number of moves (or seconds) needed.

Assume you have n discs (we are ultimately interested in $n = 64$).
Let x_n be the number of moves (or seconds) needed.

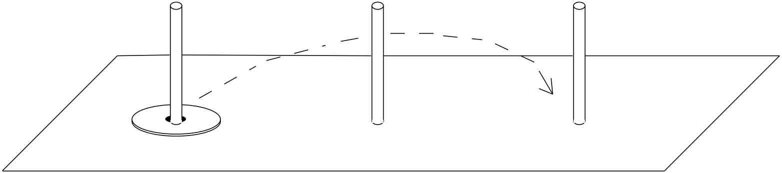


Assume you have n discs (we are ultimately interested in $n = 64$).
Let x_n be the number of moves (or seconds) needed.

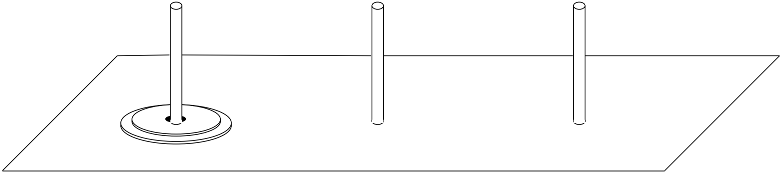


$x_1 = 1$.

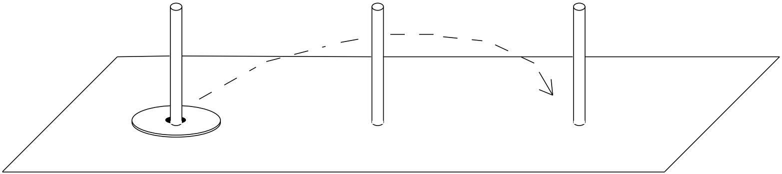
Assume you have n discs (we are ultimately interested in $n = 64$). Let x_n be the number of moves (or seconds) needed.



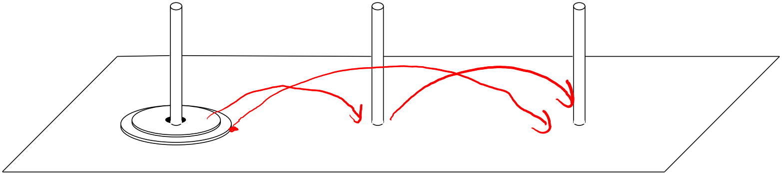
$x_1 = 1$.



Assume you have n discs (we are ultimately interested in $n = 64$). Let x_n be the number of moves (or seconds) needed.



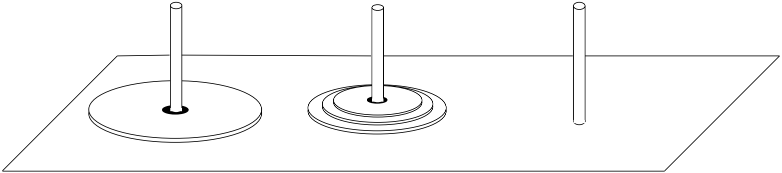
$x_1 = 1.$



$x_2 = 3.$

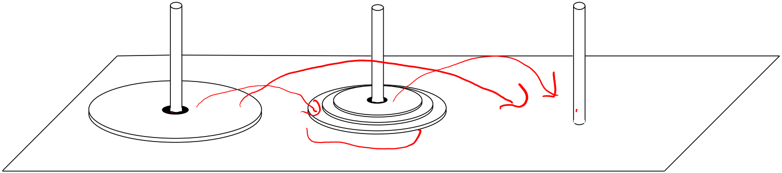
Towers of Hanoi: Solution

To move $n+1$ discs first move top n discs to central peg (x_n mvs):



Towers of Hanoi: Solution

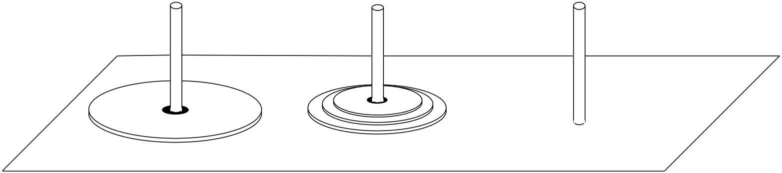
To move $n+1$ discs first move top n discs to central peg (x_n mvs):



Next move base disc (1mv). Then remaining n discs (x_n mvs).

Towers of Hanoi: Solution

To move $n+1$ discs first move top n discs to central peg (x_n mvs):

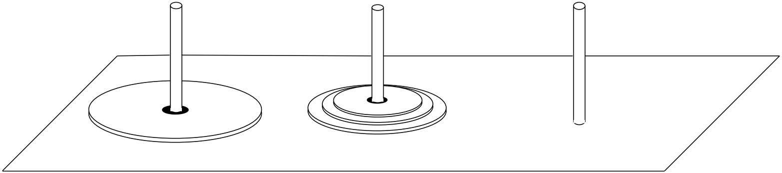


Next move base disc (1mv). Then remaining n discs (x_n mvs).

$$\underbrace{x_{n+1}} = \underbrace{x_n} + \underbrace{1} + \underbrace{x_n} = \underbrace{2x_n + 1} \quad \forall n \in \mathbb{N}$$

Towers of Hanoi: Solution

To move $n+1$ discs first move top n discs to central peg (x_n mvs):



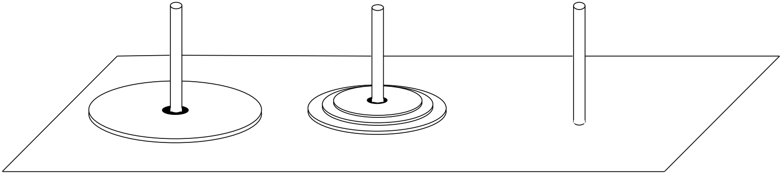
Next move base disc (1mv). Then remaining n discs (x_n mvs).

$$x_{n+1} = x_n + 1 + x_n = 2x_n + 1 \quad \forall n \in \mathbb{N}$$

So we have an implicit definition of a mixed geometric-arithmetic sequence $(x_n)_{n\in\mathbb{N}}$ with multiplier 2, offset 1 and first term 1.

Towers of Hanoi: Solution

To move $n+1$ discs first move top n discs to central peg (x_n mvs):



Next move base disc (1mv). Then remaining n discs (x_n mvs).

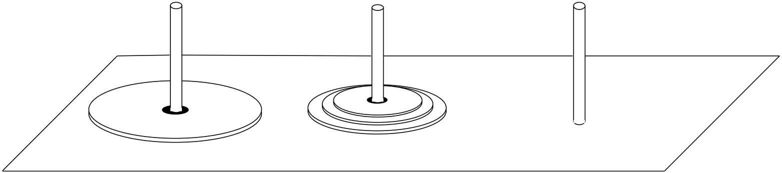
$$x_{n+1} = x_n + 1 + x_n = 2x_n + 1 \quad \forall n \in \mathbb{N}$$

So we have an implicit definition of a mixed geometric-arithmetic sequence $(x_n)_{n\in\mathbb{N}}$ with multiplier 2, offset 1 and first term 1. Using the explicit formula gives

$$x_n = 1(2^{n-1}) + \left(\frac{1-2^{n-1}}{1-2}\right)1 = 2^n - 1 \quad \forall n \in \mathbb{N}.$$

Towers of Hanoi: Solution

To move $n+1$ discs first move top n discs to central peg (x_n mvs):



Next move base disc (1mv). Then remaining n discs (x_n mvs).

$$x_{n+1} = x_n + 1 + x_n = 2x_n + 1 \quad \forall n \in \mathbb{N}$$

So we have an implicit definition of a mixed geometric-arithmetic sequence $(x_n)_{n\in\mathbb{N}}$ with multiplier 2, offset 1 and first term 1.
Using the explicit formula gives

$$x_n = 1(2^{n-1}) + \left(\frac{1-2^{n-1}}{1-2}\right)1 = 2^n - 1 \quad \forall n \in \mathbb{N}.$$

In particular $x_{64} = \underline{(2^{64} - 1) \text{ seconds}} \sim 5.8 \times 10^{11}$ years.

Sorting algorithms

$(x_n) \in \underbrace{S \times \cdots \times S}_{N \text{ copies}}$

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n\in\{1,\ldots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \ldots, N\}$;

Sorting algorithms

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n\in\{1,\ldots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \ldots, N\}$;
it does not imply that all the x_n's are different.
So *sequences may contain some elements more than once.*

Sorting algorithms

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n\in\{1,\ldots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \ldots, N\}$; it does not imply that all the x_n's are different.
So *sequences may contain some elements more than once.*

A **sorting algorithm** is a procedure for sorting a sequence into increasing order according to some specified ordering rule (*e.g.* numerical, alphabetical, etc.)

Sorting algorithms

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n\in\{1,\ldots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \ldots, N\}$; it does not imply that all the x_n's are different.
So *sequences may contain some elements more than once.*

A **sorting algorithm** is a procedure for sorting a sequence into increasing order according to some specified ordering rule (*e.g.* numerical, alphabetical, etc.) *i.e.* it replaces $(x_n)_{n\in\{1,\ldots,N\}}$ by a rearrangement $(y_n)_{n\in\{1,\ldots,N\}}$ with

$$y_1 \leq y_2 \leq y_3 \cdots y_{N-1} \leq y_N$$

where "$\leq$" denotes the ordering rule.

Sorting algorithms

Let $N \in \mathbb{N}$, S be a set, and $(x_n)_{n\in\{1,\ldots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \ldots, N\}$; it does not imply that all the x_n's are different.
So *sequences may contain some elements more than once.*

A **sorting algorithm** is a procedure for sorting a sequence into increasing order according to some specified ordering rule (*e.g.* numerical, alphabetical, etc.) *i.e.* it replaces $(x_n)_{n\in\{1,\ldots,N\}}$ by a rearrangement $(y_n)_{n\in\{1,\ldots,N\}}$ with

$$y_1 \leq y_2 \leq y_3 \cdots y_{N-1} \leq y_N$$

where "$\leq$" denotes the ordering rule.

Example:
$(x_n)_{n\in\{1,\ldots,5\}}$ = Jane, Fred, Jo, Jane, Ann
$(y_n)_{n\in\{1,\ldots,5\}}$ = Ann, Fred, Jane, Jane, Jo (in alphabetical order)

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \overset{\mathbb{Z}_{\geq 0}}{\cancel{\mathbb{N}}}^\star \ : \ s \leq i \leq f\} = \{s, \ldots, f\}$$

where $s, f \in \mathbb{N}^\star, \ s \leq f$, are the **start index** and the **finish index**.

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \;:\; s \leq i \leq f\} = \{s, \ldots, f\}$$

where $s, f \in \mathbb{N}^\star$, $s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\}$ $\quad (s = 3,\; f = 6)$

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \ : \ s \leq i \leq f\} = \{s, \dots, f\}$$

where $s, f \in \mathbb{N}^\star$, $s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\}$ $\qquad (s = 3, \ f = 6)$

For $I = \{s, \dots, f\}$ we may denote the sequence $(a_n)_{n \in I}$ by $(a_n)_{s..f}$.

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \ : \ s \leq i \leq f\} = \{s, \ldots, f\}$$

where $s, f \in \mathbb{N}^\star, \ s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\}$ $(s = 3, \ f = 6)$

For $I = \{s, \ldots, f\}$ we may denote the sequence $(a_n)_{n\in I}$ by $\boldsymbol{(a_n)_{s..f}}$.

Example: Suppose $\forall n \in \mathbb{N} \ a_n = 2n + 1$. Then $(a_n)_{3..6}$ =7,9,11,13.

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \ : \ s \leq i \leq f\} = \{s, \ldots, f\}$$

where $s, f \in \mathbb{N}^\star$, $s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\} \qquad (s = 3,\ f = 6)$

For $I = \{s, \ldots, f\}$ we may denote the sequence $(a_n)_{n \in I}$ by $\boldsymbol{(a_n)_{s..f}}$.

Example: Suppose $\forall n \in \mathbb{N}\ a_n = 2n + 1$. Then $(a_n)_{3..6}$ =7,9,11,13.

An **index permutation** on an index set I is a bijection (one-to-one correspondence) $\pi : I \to I$.

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \ : \ s \leq i \leq f\} = \{s, \ldots, f\}$$

where $s, f \in \mathbb{N}^\star, \ s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\} \qquad (s = 3, \ f = 6)$

For $I = \{s, \ldots, f\}$ we may denote the sequence $(a_n)_{n \in I}$ by $\boldsymbol{(a_n)_{s..f}}$.

Example: Suppose $\forall n \in \mathbb{N} \ a_n = 2n + 1$. Then $(a_n)_{3..6}$ =7,9,11,13.

An **index permutation** on an index set I is a bijection (one-to-one correspondence) $\pi : I \to I$. For $I = \{s, \ldots, f\}$ the permutation can be specified using the notation

$$\pi = \begin{pmatrix} s & s+1 & \ldots & f \\ \pi(s) & \pi(s+1) & \ldots & \pi(f) \end{pmatrix}.$$

Sorting preliminaries

An **index set** I is a set of the form

$$I = \{i \in \mathbb{N}^\star \ : \ s \leq i \leq f\} = \{s, \dots, f\}$$

where $s, f \in \mathbb{N}^\star$, $s \leq f$, are the **start index** and the **finish index**.

Example: $I = \{3, 4, 5, 6\}$ $\quad (s = 3,\ f = 6)$

For $I = \{s, \dots, f\}$ we may denote the sequence $(a_n)_{n \in I}$ by $\boldsymbol{(a_n)_{s..f}}$.

Example: Suppose $\forall n \in \mathbb{N}\ a_n = 2n + 1$. Then $(a_n)_{3..6}$ =7,9,11,13.

An **index permutation** on an index set I is a bijection (one-to-one correspondence) $\pi : I \to I$. For $I = \{s, \dots, f\}$ the permutation can be specified using the notation

$$\pi = \begin{pmatrix} s & s+1 & \dots & f \\ \pi(s) & \pi(s+1) & \dots & \pi(f) \end{pmatrix}.$$

Example:

$\pi = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 6 & 4 & 3 & 5 \end{pmatrix}$ means $\begin{array}{l} I = \{3, 4, 5, 6\} \\ \pi(3) = 6,\ \pi(4) = 4,\ \pi(5) = 3,\ \pi(6) = 5 \end{array}$.

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

A **reordering** of a sequence $(x_n)_{s..t}$ is a sequence $(y_n)_{s..t}$ where $y_n = x_{\pi(n)}$ for some index permutation π.

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

A **reordering** of a sequence $(x_n)_{s..t}$ is a sequence $(y_n)_{s..t}$ where $y_n = x_{\pi(n)}$ for some index permutation π.
The reordering of a sequence $(a_n)_{s..t}$ can be denoted by $(a_{\pi(n)})_{s..t}$.

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

A **reordering** of a sequence $(x_n)_{s..t}$ is a sequence $(y_n)_{s..t}$ where $y_n = x_{\pi(n)}$ for some index permutation π.
The reordering of a sequence $(a_n)_{s..t}$ can be donoted by $(a_{\pi(n)})_{s..t}$.

3 4 5 6

Example: For $\pi = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 6 & 4 & 3 & 5 \end{pmatrix}$, if $(a_n)_{3..6} = 7, 9, 11, 13$. then $(a_{\pi(n)})_{3..6} = 13, 9, 7, 11$

More sorting preliminaries

Using index permutations for sorting has two benefits:

- it allows for more precise algorithm specification: and
- items being sorted do not get moved - only their indices are affected. This is valuable when the items have long and/or variable storage length.

A **reordering** of a sequence $(x_n)_{s..t}$ is a sequence $(y_n)_{s..t}$ where $y_n = x_{\pi(n)}$ for some index permutation π.
The reordering of a sequence $(a_n)_{s..t}$ can be donoted by $(a_{\pi(n)})_{s..t}$.

Example: For $\pi = \begin{pmatrix} 3 & 4 & 5 & 6 \\ 6 & 4 & 3 & 5 \end{pmatrix}$, if $(a_n)_{3..6}$= 7, 9, 11, 13 .
then $(a_{\pi(n)})_{3..6}$=13, 9, 7, 11

Example: (names example recast using an index permutation)
If $(x_n)_{n\in\{1,\ldots,5\}}$ = Jane, Fred, Jo, Jane, Ann
then $(x_{\pi(n)})_{n\in\{1,\ldots,5\}}$ = Ann, Fred, Jane, Jane, Jo
where $\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{pmatrix}$ sorts the sequence into alphabetical order.

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S
and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, ..., f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, ..., f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s = 1,\ f = 6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S
and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: ($s = 1$, $f = 6$)

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2
m	1
$x_{\pi(i)}$	D
$x_{\pi(m)}$	F

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s=1,\ f=6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3
m	1	2
$x_{\pi(i)}$	D	C
$x_{\pi(m)}$	F	D

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S
and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s = 1, f = 6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3	4
m	1	2	3
$x_{\pi(i)}$	D	C	E
$x_{\pi(m)}$	F	D	C

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S
and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, ..., f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: ($s=1$, $f=6$)

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3	4	5
m	1	2	3	3
$x_{\pi(i)}$	D	C	E	B
$x_{\pi(m)}$	F	D	C	C

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s = 1,\ f = 6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3	4	5	6
m	1	2	3	3	5
$x_{\pi(i)}$	D	C	E	B	C
$x_{\pi(m)}$	F	D	C	C	B

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S
and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, ..., f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} < x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s = 1,\ f = 6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3	4	5	6	7
m	1	2	3	3	5	5
$x_{\pi(i)}$	D	C	E	B	C	-
$x_{\pi(m)}$	F	D	C	C	B	B

Least element algorithm

In writing algorithms from now on we will use the notation $a \leftarrow b$ to mean "assign a the value b, leaving b unchanged". (Some authors use $a := b$ for this.)

Input: Sequence $(x_i)_{s..f} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{s, \ldots, f\}$.

Output: Modification to π so that $x_{\pi(s)} \leq x_{\pi(i)}$ for $i = s, \ldots, f$.

Method:

$i \leftarrow s + 1$. [Initialisation]

$m \leftarrow s$, [m is a marker; $x_{\pi(m)}$ is the least sequence member so far tested]

Loop: If $i = f + 1$ stop.

If $x_{\pi(i)} \leq x_{\pi(m)}$ then $m \leftarrow i$.

$i \leftarrow i + 1$

Repeat loop

Swap the values of $\pi(s)$ and $\pi(m)$.

Example: $(s = 1,\ f = 6)$

	i	1	2	3	4	5	6
BEFORE	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C

Trace:

i	2	3	4	5	6	7
m	1	2	3	3	5	5
$x_{\pi(i)}$	D	C	E	B	C	-
$x_{\pi(m)}$	F	D	C	C	B	B

	i	1	2	3	4	5	6
AFTER	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$,
an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

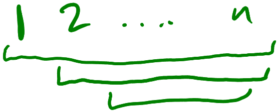
Output: Modification to π,
so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order
$x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.
Run least element algorithm on $(x_{\pi(i)})_{s..n}$
$s \leftarrow s + 1$

Repeat loop



Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input:	$\pi(i)$	1	2	3	4	5	6
$(n = 6)$	$x_{\pi(i)}$	F	D	C	E	B	C

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input: ($n = 6$)	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C
$s = 1$							
After 1st iteration	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input: ($n = 6$)	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C
$s = 1$							
After 1st iteration	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C
$s = 2$							
After 2nd iteration	$\pi(i)$	5	**3**	**2**	4	1	6
	$x_{\pi(i)}$	B	**C**	**D**	E	F	C

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input: ($n = 6$)	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C
$s = 1$							
After 1st iteration	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C
$s = 2$							
After 2nd iteration	$\pi(i)$	5	**3**	**2**	4	1	6
	$x_{\pi(i)}$	B	**C**	**D**	E	F	C
$s = 3$							
After 3rd iteration	$\pi(i)$	5	3	**6**	4	1	**2**
	$x_{\pi(i)}$	B	C	**C**	E	F	**D**

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input: ($n = 6$)	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C
$s = 1$							
After 1st iteration	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C
$s = 2$							
After 2nd iteration	$\pi(i)$	5	**3**	**2**	4	1	6
	$x_{\pi(i)}$	B	**C**	**D**	E	F	C
$s = 3$							
After 3rd iteration	$\pi(i)$	5	3	**6**	4	1	**2**
	$x_{\pi(i)}$	B	C	**C**	E	F	**D**
$s = 4$							
After 4th iteration	$\pi(i)$	5	3	6	**2**	1	**4**
	$x_{\pi(i)}$	B	C	C	**D**	F	**E**

Selection sort algorithm

Input: Sequence $(x_i)_{1..n} \subseteq S$, an ordering rule "$\leq$" for S and an index function π on $\{1, \ldots, n\}$.

Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in non-decreasing order $x_{\pi(1)} \leq x_{\pi(2)} \leq \cdots \leq x_{\pi(n)}$.

Method:

$s \leftarrow 1$ [Initialisation]

Loop: If $s = n$ stop.

Run least element algorithm on $(x_{\pi(i)})_{s..n}$

$s \leftarrow s + 1$

Repeat loop

Example:

	i:	1	2	3	4	5	6
Input: ($n = 6$)	$\pi(i)$	1	2	3	4	5	6
	$x_{\pi(i)}$	F	D	C	E	B	C
$s = 1$							
After 1st iteration	$\pi(i)$	**5**	2	3	4	**1**	6
	$x_{\pi(i)}$	**B**	D	C	E	**F**	C
$s = 2$							
After 2nd iteration	$\pi(i)$	5	**3**	**2**	4	1	6
	$x_{\pi(i)}$	B	**C**	**D**	E	F	C
$s = 3$							
After 3rd iteration	$\pi(i)$	5	3	**6**	4	1	**2**
	$x_{\pi(i)}$	B	C	**C**	E	F	**D**
$s = 4$							
After 4th iteration	$\pi(i)$	5	3	6	**2**	1	**4**
	$x_{\pi(i)}$	B	C	C	**D**	F	**E**
$s = 5$							
After final iteration	$\pi(i)$	5	3	6	2	**4**	**1**
	$x_{\pi(i)}$	B	C	C	D	**E**	**F**

Selection sort: number of operations

How many operations are required by Selection Sort?

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

$\underbrace{1\ 2\ \ldots\ n}$

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

So: 1st iteration uses $n-1$ comparisons

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

So:	1st	iteration uses	$n-1$	comparisons
	2nd	iteration uses	$n-2$	comparisons

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

So:	1st	iteration uses	$n-1$	comparisons
	2nd	iteration uses	$n-2$	comparisons
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	last	iteration uses	1	comparison

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

$$\text{So:} \quad \begin{array}{rcccl} \text{1st} & \text{iteration uses} & n-1 & \text{comparisons} \\ \text{2nd} & \text{iteration uses} & n-2 & \text{comparisons} \\ \vdots & \vdots & \vdots & \vdots \\ \text{last} & \text{iteration uses} & 1 & \text{comparison} \end{array}$$

Hence the total number of comparisons, T_n say, is given by

$$1 + 2 + \cdots + (n-1) = (n-1)\left(\frac{1+(n-1)}{2}\right) \text{ (sum of an arithmetic series).}$$

Selection sort: number of operations

How many operations are required by Selection Sort?

By *operation* here we mean any comparison step;

i.e. a step of the form "If $x_{\pi(i)} \leq x_{\pi(j)}$ then ..."

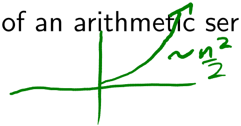
The loop of the Selection Sort algorithm is iterated $n-1$ times; once each for $s = 1, \ldots, n-1$.

Iteration s runs the Least Element algorithm on $(x_{\pi(i)})_{s..n}$ and so uses $n-s$ comparisons.

So:				
	1st	iteration uses	$n-1$	comparisons
	2nd	iteration uses	$n-2$	comparisons
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	last	iteration uses	1	comparison

Hence the total number of comparisons, T_n say, is given by

$$1 + 2 + \cdots + (n-1) = (n-1)\left(\frac{1+(n-1)}{2}\right) \text{ (sum of an arithmetic series).}$$



That is: $\quad \forall n \in \mathbb{N} \quad T_n = \dfrac{n(n-1)}{2}.$

Other sorting algorithms

There are many different sorting algorithms, with various pros and cons. A full study of the topic belongs in course on algorithms and data structures.

Other sorting algorithms

There are many different sorting algorithms, with various pros and cons. A full study of the topic belongs in course on algorithms and data structures.

We will look at just one more; "Merge Sort".
This will provide us with an opportunity to compare two algorithms designed to do the same job – what are their respective advantages and disadvantages?

Other sorting algorithms

There are many different sorting algorithms, with various pros and cons. A full study of the topic belongs in course on algorithms and data structures.

We will look at just one more; “Merge Sort”.
This will provide us with an opportunity to compare two algorithms designed to do the same job – what are their respective advantages and disadvantages?

In order to keep the description simple, I will not use an indexing function π in specifying the algorithm, though it is possible, and often preferable, to do so.

Other sorting algorithms

There are many different sorting algorithms, with various pros and cons. A full study of the topic belongs in course on algorithms and data structures.

We will look at just one more; “Merge Sort”.
This will provide us with an opportunity to compare two algorithms designed to do the same job – what are their respective advantages and disadvantages?

In order to keep the description simple, I will not use an indexing function π in specifying the algorithm, though it is possible, and often preferable, to do so.

As with Selection Sort, Merge Sort makes use of a sub-algorithm, which we treat first.

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1.$ [Initialize the indeces for the a-, b- and z- lists]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\dots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\dots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\dots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

If $i = n+1$ then $[\, z_k \leftarrow b_j,\ j \leftarrow j+1 \,]$ [a-list empty; take from b-list]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

If $i = n+1$ then [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [a-list empty; take from b-list]

Else if $j = p+1$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [b-list empty; take from a-list]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\dots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\dots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\le$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\dots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

If $i = n+1$ then [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [a-list empty; take from b-list]

Else if $j = p+1$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [b-list empty; take from a-list]

Else if $a_i < b_j$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [a-list item less; take it]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1.$ [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

If $i = n+1$ then [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [a-list empty; take from b-list]

Else if $j = p+1$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [b-list empty; take from a-list]

Else if $a_i < b_j$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [a-list item less; take it]

Else [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [else take item from b-list]

Merge algorithm

Input: Two lists (sequences) $(a_i)_{i\in\{1,\ldots,n\}} \subseteq S$ and $(b_j)_{j\in\{1,\ldots,p\}} \subseteq S$ pre-sorted according to an ordering rule "$\leq$" on S.

Output: In-order list (sorted sequence) $(z_k)_{k\in\{1,\ldots,n+p\}}$ that merges the two input lists.

Method:

$i, j, k \leftarrow 1$. [Initialize the indeces for the a-, b- and z- lists]

Loop: If $k = n+p+1$ stop. [there will be $n+p$ items in the z-list]

If $i = n+1$ then [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [a-list empty; take from b-list]

Else if $j = p+1$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [b-list empty; take from a-list]

Else if $a_i < b_j$ then [$z_k \leftarrow a_i$, $i \leftarrow i+1$] [a-list item less; take it]

Else [$z_k \leftarrow b_j$, $j \leftarrow j+1$] [else take item from b-list]

$k \leftarrow k+1$ [prepare to add next item to z-list]

Repeat loop.

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$

Merge algorithm: example of execution

Example: Merge $(1,3,7)$ and $(2,3,6,8,9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1,\ldots,z_{k-1})$
0	1	1	1	$(\mathbf{1},3,7)$	$(\mathbf{2},3,6,8,9)$	$()$
1	2	1	2	$(1,\mathbf{3},7)$	$(\mathbf{2},3,6,8,9)$	(1)
2	2	2	3	$(1,\mathbf{3},7)$	$(2,\mathbf{3},6,8,9)$	$(1,2)$
3	2	3	4	$(1,\mathbf{3},7)$	$(2,3,\mathbf{6},8,9)$	$(1,2,3)$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$
3	2	3	4	$(1, \mathbf{3}, 7)$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3)$
4	3	3	5	$(1, 3, \mathbf{7})$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3, 3)$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$
3	2	3	4	$(1, \mathbf{3}, 7)$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3)$
4	3	3	5	$(1, 3, \mathbf{7})$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3, 3)$
5	3	4	6	$(1, 3, \mathbf{7})$	$(2, 3, 6, \mathbf{8}, 9)$	$(1, 2, 3, 3, 6)$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$
3	2	3	4	$(1, \mathbf{3}, 7)$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3)$
4	3	3	5	$(1, 3, \mathbf{7})$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3, 3)$
5	3	4	6	$(1, 3, \mathbf{7})$	$(2, 3, 6, \mathbf{8}, 9)$	$(1, 2, 3, 3, 6)$
6	4	4	7	$(1, 3, 7)$	$(2, 3, 6, \mathbf{8}, 9,)$	$(1, 2, 3, 3, 6, 7)$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$
3	2	3	4	$(1, \mathbf{3}, 7)$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3)$
4	3	3	5	$(1, 3, \mathbf{7})$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3, 3)$
5	3	4	6	$(1, 3, \mathbf{7})$	$(2, 3, 6, \mathbf{8}, 9)$	$(1, 2, 3, 3, 6)$
6	4	4	7	$(1, 3, 7)$	$(2, 3, 6, \mathbf{8}, 9,)$	$(1, 2, 3, 3, 6, 7)$
7	4	5	8	$(1, 3, 7)$	$(2, 3, 6, 8, \mathbf{9})$	$(1, 2, 3, 3, 6, 7, 8)$

Merge algorithm: example of execution

Example: Merge $(1, 3, 7)$ and $(2, 3, 6, 8, 9)$.

After iteration	i	j	k	a_i [green]	b_j [green]	$(z_1, \ldots, z_{k-1})$
0	1	1	1	$(\mathbf{1}, 3, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	$()$
1	2	1	2	$(1, \mathbf{3}, 7)$	$(\mathbf{2}, 3, 6, 8, 9)$	(1)
2	2	2	3	$(1, \mathbf{3}, 7)$	$(2, \mathbf{3}, 6, 8, 9)$	$(1, 2)$
3	2	3	4	$(1, \mathbf{3}, 7)$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3)$
4	3	3	5	$(1, 3, \mathbf{7})$	$(2, 3, \mathbf{6}, 8, 9)$	$(1, 2, 3, 3)$
5	3	4	6	$(1, 3, \mathbf{7})$	$(2, 3, 6, \mathbf{8}, 9)$	$(1, 2, 3, 3, 6)$
6	4	4	7	$(1, 3, 7)$	$(2, 3, 6, \mathbf{8}, 9,)$	$(1, 2, 3, 3, 6, 7)$
7	4	5	8	$(1, 3, 7)$	$(2, 3, 6, 8, \mathbf{9})$	$(1, 2, 3, 3, 6, 7, 8)$
8	4	6	9	$(1, 3, 7)$	$(2, 3, 6, 8, 9)$	$(1, 2, 3, 3, 6, 7, 8, 9)$

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n\in\{1,\dots,2^r\}} \subseteq S$,and an ordering rule "$\leq$" for S.

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n \in \{1,\ldots,2^r\}} \subseteq S$,and an ordering rule "$\leq$" for S.
(For lists whose length is not a power of 2, see workshop question.)

Output: In-order list (sorted sequence) $(z_n)_{n \in \{1,\ldots,2^r\}}$ that is a rearrangement of the input list.

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n\in\{1,\ldots,2^r\}} \subseteq S$,and an ordering rule "$\leq$" for S.
(For lists whose length is not a power of 2, see workshop question.)

Output: In-order list (sorted sequence) $(z_n)_{n\in\{1,\ldots,2^r\}}$ that is a rearrangement of the input list.

Method: There are r steps.
If $r < 3$ adjust the description below accordingly.

- Step 1: Apply the Merge algorithm 2^{r-1} times with inputs $\{(x_1),(x_2)\}$, $\{(x_3),(x_4)\}$,...,$\{(x_{2^r-1}),(x_{2^r})\}$.
  This gives 2^{r-1} in-order lists of length 2.

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n\in\{1,\dots,2^r\}} \subseteq S$,and an ordering rule "$\leq$" for S.
(For lists whose length is not a power of 2, see workshop question.)

Output: In-order list (sorted sequence) $(z_n)_{n\in\{1,\dots,2^r\}}$ that is a rearrangement of the input list.

Method: There are r steps.
If $r < 3$ adjust the description below accordingly.

- Step 1: Apply the Merge algorithm 2^{r-1} times with inputs $\{(x_1),(x_2)\}$, $\{(x_3),(x_4)\},\dots,\{(x_{2^r-1}),(x_{2^r})\}$.
This gives 2^{r-1} in-order lists of length 2.
- Step 2: Apply the Merge algorithm 2^{r-2} times with pairs of these lists as input.
This gives 2^{r-2} in-order lists of length $2 \times 2 = 2^2$.

Merge Sort algorithm

Input: $r \in \mathbb{N}$, $(x_n)_{n\in\{1,\ldots,2^r\}} \subseteq S$,and an ordering rule "$\leq$" for S.

(For lists whose length is not a power of 2, see workshop question.)

Output: In-order list (sorted sequence) $(z_n)_{n\in\{1,\ldots,2^r\}}$ that is a rearrangement of the input list.

Method: There are r steps.

If $r < 3$ adjust the description below accordingly.

- Step 1: Apply the Merge algorithm 2^{r-1} times with inputs $\{(x_1),(x_2)\}$, $\{(x_3),(x_4)\}$,…,$\{(x_{2^r-1}),(x_{2^r})\}$.
  This gives 2^{r-1} in-order lists of length 2.
- Step 2: Apply the Merge algorithm 2^{r-2} times with pairs of these lists as input.
  This gives 2^{r-2} in-order lists of length $2 \times 2 = 2^2$.
- Steps 3 to r: Continue in this vein until you have just one $(= 2^{r-r})$ in-order list with 2^r elements.

Merge sort: example

Example: merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$.

Merge sort: example

Example: merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$.

Step 1:
Merging $\{(1), (2)\}$ gives $(1, 2)$.

Merge sort: example

Example: merge sort $(\underbrace{1,2},\underbrace{6,1},\underbrace{7,9},\underbrace{4,5})$.

Step 1:
Merging $\{(1),(2)\}$ gives $(1,2)$.
Merging $\{(6),(1)\}$ gives $(1,6)$.
Merging $\{(7),(9)\}$ gives $(7,9)$.
Merging $\{(4),(5)\}$ gives $(4,5)$.

Merge sort: example

Example: merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$.

Step 1:
Merging $\{(1), (2)\}$ gives $(1, 2)$.
Merging $\{(6), (1)\}$ gives $(1, 6)$.
Merging $\{(7), (9)\}$ gives $(7, 9)$.
Merging $\{(4), (5)\}$ gives $(4, 5)$.

Step 2:
Merging $\{(1, 2), (1, 6)\}$ gives $(1, 1, 2, 6)$.
Merging $\{(7, 9), (4, 5)\}$ gives $(4, 5, 7, 9)$.

Merge sort: example

Example: merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$.

Step 1:
Merging $\{(1), (2)\}$ gives $(1, 2)$.
Merging $\{(6), (1)\}$ gives $(1, 6)$.
Merging $\{(7), (9)\}$ gives $(7, 9)$.
Merging $\{(4), (5)\}$ gives $(4, 5)$.

Step 2:
Merging $\{(1, 2), (1, 6)\}$ gives $(1, 1, 2, 6)$.
Merging $\{(7, 9), (4, 5)\}$ gives $(4, 5, 7, 9)$.

Step 3:
Merging $\{(1, 1, 2, 6), (4, 5, 7, 9)\}$ gives $(1, 1, 2, 4, 5, 6, 7, 9)$.

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

$(1) \quad (2) \quad (6) \quad (1) \quad (7) \quad (9) \quad (4) \quad (5)$	
$(1, 2) \quad (1, 6) \quad (7, 9) \quad (4, 5)$	$1+1+1+1=4$ comps
$(1, 1, 2, 6) \quad (4, 5, 7, 9)$	$3+2=5$ comps
$(1, 1, 2, 4, 5, 6, 7, 9)$	6 comps
TOTAL:	15 comparisons

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

$$
\begin{array}{llr}
(1)\quad(2)\quad(6)\quad(1)\quad(7)\quad(9)\quad(4)\quad(5) & & \\
(1,2)\qquad(1,6)\qquad(7,9)\qquad(4,5) & & 1{+}1{+}1{+}1{=}4 \text{ comps} \\
(1,1,2,6)\qquad\qquad(4,5,7,9) & & 3{+}2{=}5 \text{ comps} \\
(1,1,2,4,5,6,7,9) & & 6 \text{ comps} \\
 & \text{TOTAL:} & 15 \text{ comparisons}
\end{array}
$$

Note for example that when merging $(7, 9)$ and $(4, 5)$ only 2 comparisons are used:

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

$$
\begin{array}{lr}
(1) \quad (2) \quad (6) \quad (1) \quad (7) \quad (9) \quad (4) \quad (5) & \\
(1,2) \qquad (1,6) \qquad (7,9) \qquad (4,5) & 1{+}1{+}1{+}1{=}4 \text{ comps} \\
(1,1,2,6) \qquad\qquad (4,5,7,9) & 3{+}2{=}5 \text{ comps} \\
(1,1,2,4,5,6,7,9) & 6 \text{ comps} \\
\text{TOTAL:} & 15 \text{ comparisons}
\end{array}
$$

Note for example that when merging $(7, 9)$ and $(4, 5)$ only 2 comparisons are used:

7 and 4 are compared; 4 is transferred

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

$$
\begin{array}{lr}
(1)\quad(2)\quad(6)\quad(1)\quad(7)\quad(9)\quad(4)\quad(5) & \\
(1,2)\qquad(1,6)\qquad(7,9)\qquad(4,5) & 1+1+1+1=4 \text{ comps} \\
(1,1,2,6)\qquad\qquad(4,5,7,9) & 3+2=5 \text{ comps} \\
(1,1,2,4,5,6,7,9) & 6 \text{ comps} \\
\text{TOTAL:} & 15 \text{ comparisons}
\end{array}
$$

Note for example that when merging $(7, 9)$ and $(4, 5)$ only 2 comparisons are used:

7 and 4 are compared; 4 is transferred

7 and 5 are compared; 5 is transferred

Merge sort: counting comparisons

As with Selection Sort, we can analyze the complexity of Merge sort by counting the number of comparisons involved.

Revisiting the previous example, merge sort $(1, 2, 6, 1, 7, 9, 4, 5)$:

$$
\begin{array}{ll}
(1)\quad (2)\quad (6)\quad (1)\quad (7)\quad (9)\quad (4)\quad (5) & \\
(1,2)\qquad (1,6)\qquad (7,9)\qquad (4,5) & 1+1+1+1=4 \text{ comps} \\
(1,1,2,6)\qquad\qquad (4,5,7,9) & 3+2=5 \text{ comps} \\
(1,1,2,4,5,6,7,9) & 6 \text{ comps} \\
\text{TOTAL:} & 15 \text{ comparisons}
\end{array}
$$

Note for example that when merging $(7, 9)$ and $(4, 5)$ only 2 comparisons are used:

7 and 4 are compared; 4 is transferred

7 and 5 are compared; 5 is transferred

7 and 9 are transferred without comparison (other list exhausted.)

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1},

Merge sort: number of operations

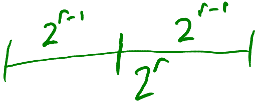
Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1}, so an implicit definition for T_r is

$$\begin{cases} T_r = 2^r + 2T_{r-1} & \forall r \in \mathbb{N}\backslash\{1\} \\ T_1 = 2. \end{cases}$$



Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1}, so an implicit definition for T_r is
$$\begin{cases} \boldsymbol{T_r = 2^r + 2T_{r-1}} & \boldsymbol{\forall r \in \mathbb{N}\backslash\{1\}} \\ \boldsymbol{T_1 = 2.} \end{cases}$$

So $T_1 = 2, \quad T_2 = 2^2 + 4 = 8, \quad T_3 = 2^3 + 2\times 8 = 3\times 2^3,$

$T_4 = 2^4 + 2 \times (3\times 2^3) = 4\times 2^4, \quad \ldots$

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1}, so an implicit definition for T_r is
$$\begin{cases} \boldsymbol{T_r = 2^r + 2T_{r-1}} & \forall \boldsymbol{r} \in \mathbb{N} \setminus \{\mathbf{1}\} \\ \boldsymbol{T_1 = 2.} \end{cases}$$

So $T_1 = 2, \quad T_2 = 2^2 + 4 = 8, \quad T_3 = 2^3 + 2{\times}8 = 3{\times}2^3,$

$$T_4 = 2^4 + 2 \times (3{\times}2^3) = 4{\times}2^4, \quad \ldots$$

Claim: $\forall \boldsymbol{r} \in \mathbb{N} \quad \boldsymbol{T_r = r2^r}.$

Merge sort: number of operations

Since the number of comparisons used to Merge Sort a list of length n depends to some extent on the nature of the list, there is no precise formula for this number as there is with Selection Sort.

However an upper bound is given by the number of *transfers*.

Let T_r denote the number of transfers required to sort a sequence of length 2^r using Merge sort.

Now $2^{r-1} + 2^{r-1} = 2^r$ transfers are required for the Merge algorithm to merge two sequences of length 2^{r-1}, so an implicit definition for T_r is
$$\begin{cases} T_r = 2^r + 2T_{r-1} & \forall r \in \mathbb{N}\backslash\{1\} \\ T_1 = 2. \end{cases}$$

So $T_1 = 2, \quad T_2 = 2^2 + 4 = 8, \quad T_3 = 2^3 + 2\times 8 = 3\times 2^3,$

$T_4 = 2^4 + 2 \times (3\times 2^3) = 4\times 2^4, \quad \ldots$

Claim: $\forall r \in \mathbb{N} \quad T_r = r2^r$. Verify by induction!

(The sequence $(T_r)_{r\in\mathbb{N}}$ is neither geometric, arithmetic nor mixed.)

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

For the same length, Selection Sort uses $\dfrac{2^r(2^r - 1)}{2}$ $\sim 4^r$ comparisons.

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

For the same length, Selection Sort uses $\dfrac{2^r(2^r-1)}{2}$ comparisons.

Merge sort is **much faster**, e.g. for $r = 10$ (so $N = 1024$):

$$\begin{array}{lcl} \text{Merge sort:} \qquad r2^r & = & 10\,240 \\ \text{Selection sort:} \qquad \dfrac{2^r(2^r-1)}{2} & = & 523\,776. \end{array}$$

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

For the same length, Selection Sort uses $\dfrac{2^r(2^r-1)}{2}$ comparisons.

Merge sort is **much faster**, e.g. for $r = 10$ (so $N = 1024$):

$$\begin{array}{lrcl} \text{Merge sort:} & r2^r & = & 10\,240 \\ \text{Selection sort:} & \dfrac{2^r(2^r-1)}{2} & = & 523\,776. \end{array}$$

Merge sort can be modified to work even faster on machines with parallel processing capabilities since then many of the uses of the Merge algorithm can be done simultaneously. There is no direct way to use parallel processing with Selection sort.

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

For the same length, Selection Sort uses $\dfrac{2^r(2^r-1)}{2}$ comparisons.

Merge sort is **much faster**, e.g. for $r = 10$ (so $N = 1024$):

$$\begin{array}{lrcl} \text{Merge sort:} & r2^r & = & 10\,240 \\ \text{Selection sort:} & \dfrac{2^r(2^r-1)}{2} & = & 523\,776. \end{array}$$

Merge sort can be modified to work even faster on machines with parallel processing capabilities since then many of the uses of the Merge algorithm can be done simultaneously. There is no direct way to use parallel processing with Selection sort.

Selection Sort may be simpler to program, especially if indexing is required to avoid transfers of large blocks of data.

Merge Sort and Selection Sort compared

Merge Sort sorts a sequence of length 2^r with less than $r2^r$ comparisons.

For the same length, Selection Sort uses $\dfrac{2^r(2^r-1)}{2}$ comparisons.

Merge sort is **much faster**, e.g. for $r = 10$ (so $N = 1024$):

$$\begin{array}{lcl} \text{Merge sort:} \quad r2^r & = & 10\,240 \\ \text{Selection sort:} \quad \dfrac{2^r(2^r-1)}{2} & = & 523\,776. \end{array}$$

Merge sort can be modified to work even faster on machines with parallel processing capabilities since then many of the uses of the Merge algorithm can be done simultaneously. There is no direct way to use parallel processing with Selection sort.

Selection Sort may be simpler to program, especially if indexing is required to avoid transfers of large blocks of data.
For short lists on high speed computers, slower speed may not matter.

END OF SECTION B2