Discrete Mathematical Models

Lecture 12

Kane Townsend

Semester 2, 2024

Section B: Digital Information (cont.)

Section B2: Sequences, Induction, Sorting (cont.)

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 $c = 2 \times 10^4$.

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

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

$c_0 = c$,

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:

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

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:

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

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:

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

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 $c = 2 \times 10^4$.

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

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

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

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

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 $c = 2 \times 10^4$.

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

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

$c_0 = c, \quad c_1 = rc + d, \quad c_2 = r(rc+d) + d = r^2 c + (r+1)d,$
$c_3 = r(r^2 c + (r+1)d) + d = r^3 c + (r^2 + r + 1)d.$

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 $c = 2 \times 10^4$.

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

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 $c_n = r^n c + (1 + r + r^2 + \cdots + r^{n-1})d$.

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

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

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

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

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

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

Applying the formula gives

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

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 $c = 2 \times 10^4$.

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

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

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

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

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

Mixed Sequences

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

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

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

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

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.

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

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

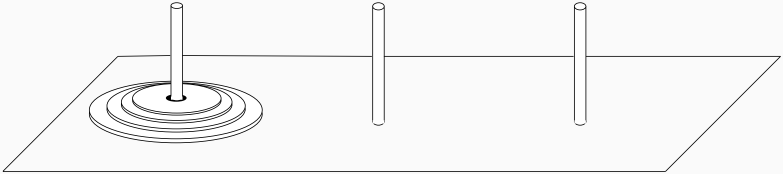
Example: A mixed geometric-arithmetic sequence $(a_n)_{n\in\mathbb{N}}$ has multiplier $\frac{1}{2}$, offset 2 and first term 1. What is the 10-th term?

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.

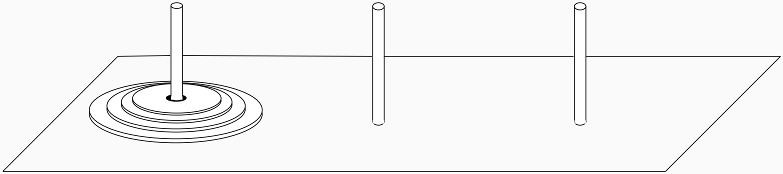
Counting moves: Towers of Hanoi

"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

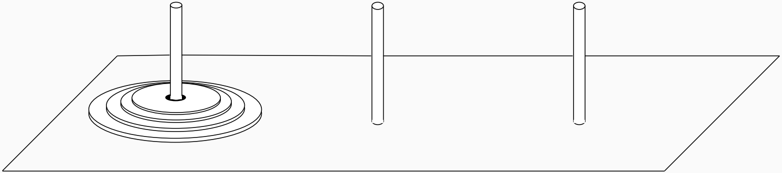
"The Towers of Hanoi" is a puzzle with 3 pegs and a number of punctured discs of decreasing sizes initially on the leftmost peg.



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.

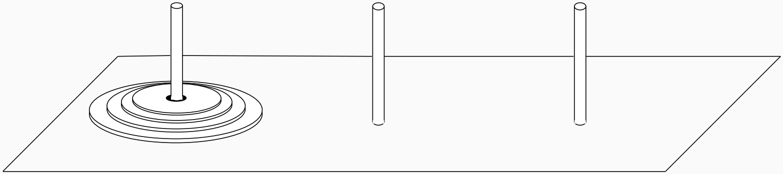


Aim: transfer all discs to the rightmost peg according to the following rules.

- 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.

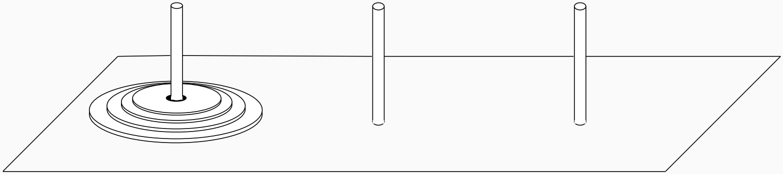


Aim: transfer all discs to the rightmost peg according to the following rules.

- 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.

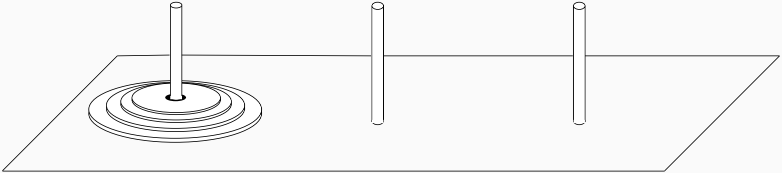


Aim: transfer all discs to the rightmost peg according to the following rules.

- 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.

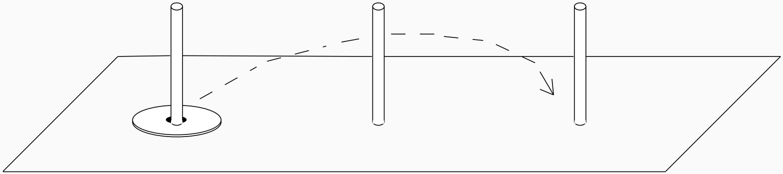
- You may move only one disc at a time to one of the other two pegs.
- You can only move the discs that are at the top of one of the piles.
- No disc may sit on top of a smaller disc.

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 $n = 64$).

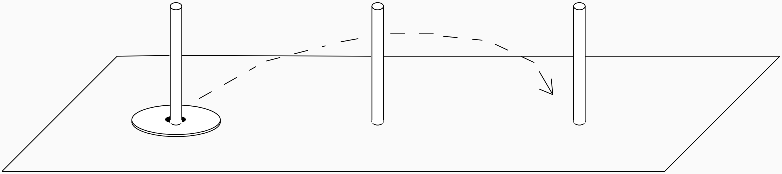
Assume you have n discs (we are ultimately interested in $n = 64$).
Let x_n be the number of moves (or seconds) needed.

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



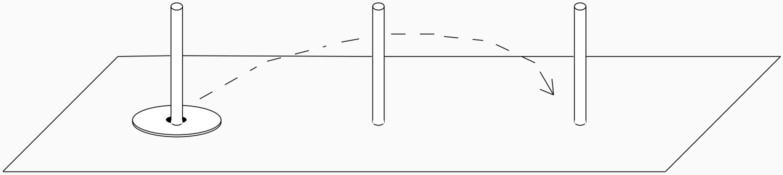
Assume you have n discs (we are ultimately interested in $n = 64$).

Let x_n be the number of moves (or seconds) needed.

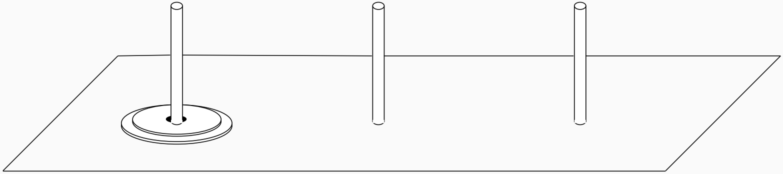


$x_1 = 1.$

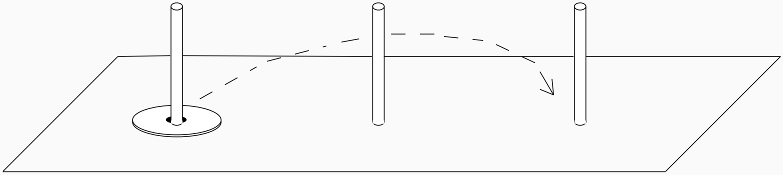
Assume you have n discs (we are ultimately interested in $n = 64$).
Let x_n be the number of moves (or seconds) needed.



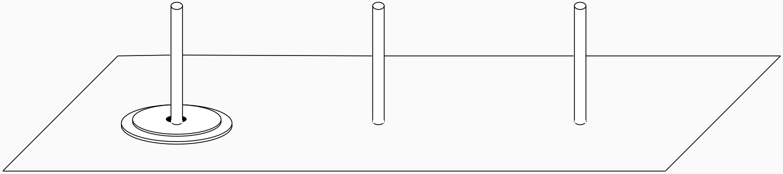
$x_1 = 1$.



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



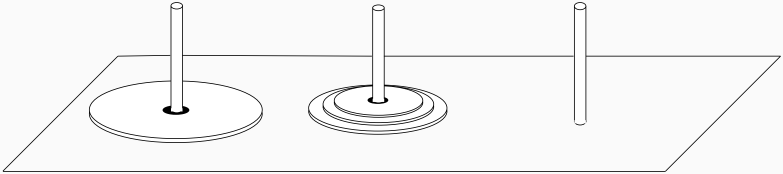
$x_1 = 1$.



$x_2 = 3$.

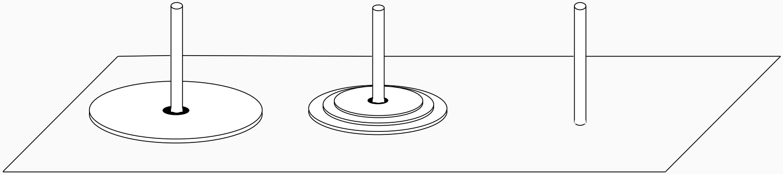
Towers of Hanoi: Solution

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



Towers of Hanoi: Solution

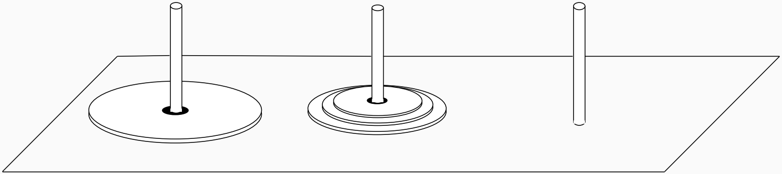
To move $n+1$ discs first move top n discs to central peg (x_n mvs):



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):

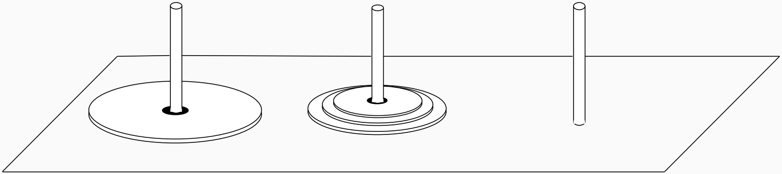


Next move base disc (1mv). Then remaining n discs (x_n mvs).

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

Towers of Hanoi: Solution

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



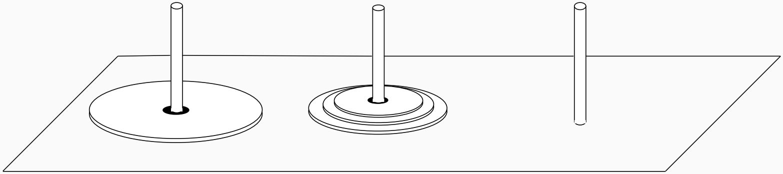
Next move base disc (1mv). Then remaining n discs (x_n mvs).

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

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).

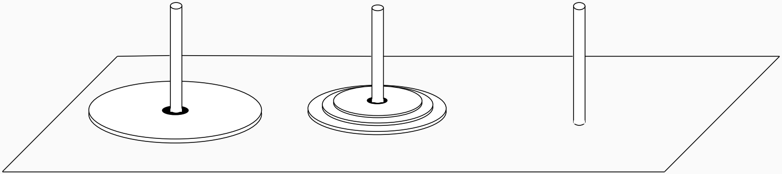
$$x_{n+1} = x_n + 1 + x_n = 2x_n + 1 \quad \forall n \in \mathbb{N}$$

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

Sorting

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 $n \in \{1, \ldots, N\}$;

Sorting algorithms

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

Remember that this just means that $x_n \in S$ for each $n \in \{1, \dots, N\}$; 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 $(x_n)_{n\in\{1,\dots,N\}} \subseteq S$.

Remember that this just means that $x_n \in S$ for each $n \in \{1, \dots, N\}$;
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 $(x_n)_{n\in\{1,\dots,N\}}$ by a rearrangement $(y_n)_{n\in\{1,\dots,N\}}$ 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

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

where "$\le$" 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

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

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

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

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

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

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

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

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

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

where $s, f \in \mathbb{N}^\star, \ s \le f$, 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 $(a_n)_{s..f}$.

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 $\pi : I \to I$.

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

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

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 $\pi : I \to I$. 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

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

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

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

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

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 $\pi : I \to I$. For $I = \{s, \ldots, f\}$ the permutation can be specified using the notation

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

Example:

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

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 denoted 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 $(a_{\pi(n)})_{3..6} = 13, 9,\ 7,\ 11$

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 $(a_{\pi(n)})_{s..t}$.

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

The first algorithm we will use to begin sorting is the **least element algorithm**.

Least element algorithm

The first algorithm we will use to begin sorting is the **least element algorithm**.

Example: $(x_i)_{1..5} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) so that the smallest letter with respect to alphabetical order can be first.

Least element algorithm

The first algorithm we will use to begin sorting is the **least element algorithm**.

Example: $(x_i)_{1..5} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) so that the smallest letter with respect to alphabetical order can be first.

We begin with our marker $m = 1$ and index $i = 1 + 1 = 2$, and we compare $x_1 = J$ and $x_{1+1} = x_2 = O$. Since J is before O in the alphabet we keep $m = 1$ as our marker (least in sequence tested so far). We increase our index by 1, $i = 2 + 1 = 3$.

Least element algorithm

The first algorithm we will use to begin sorting is the **least element algorithm**.

Example: $(x_i)_{1..5} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) so that the smallest letter with respect to alphabetical order can be first.

We begin with our marker $m = 1$ and index $i = 1 + 1 = 2$, and we compare $x_1 = J$ and $x_{1+1} = x_2 = O$. Since J is before O in the alphabet we keep $m = 1$ as our marker (least in sequence tested so far). We increase our index by 1, $i = 2 + 1 = 3$.

We then compare $x_1 = J$ with $x_{2+1} = x_3 = E$. Since E is before J in the alphabet we change our marker to $m = 3$. We cannot increase our index anymore so we stop the algorithm.

Least element algorithm

The first algorithm we will use to begin sorting is the **least element algorithm**.

Example: $(x_i)_{1..5} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) so that the smallest letter with respect to alphabetical order can be first.

We begin with our marker $m = 1$ and index $i = 1 + 1 = 2$, and we compare $x_1 = J$ and $x_{1+1} = x_2 = O$. Since J is before O in the alphabet we keep $m = 1$ as our marker (least in sequence tested so far). We increase our index by 1, $i = 2 + 1 = 3$.

We then compare $x_1 = J$ with $x_{2+1} = x_3 = E$. Since E is before J in the alphabet we change our marker to $m = 3$. We cannot increase our index anymore so we stop the algorithm.

We now put our marker in the first place by modifying π to $\pi(1) = 3$, $\pi(2) = 2$ and $\pi(3) = 1$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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 $i = s, ..., f$.

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
before	$x_{\pi(i)}$	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
after	$x_{\pi(i)}$	**B**	D	C	E	**F**	C

Selection sort algorithm

The **selection sort algorithm** algorithm is a sorting algorithm.

Selection sort algorithm

The **selection sort algorithm** algorithm is a sorting algorithm.

Example: $(x_i)_{1..3} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) into alphabetical order.

Selection sort algorithm

The **selection sort algorithm** algorithm is a sorting algorithm.

Example: $(x_i)_{1..3} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) into alphabetical order.

Apply the least element algorithm to $(x_{\pi(i)})_{1..3}$. Gaining a modification to π given by $\pi(1) = 3$, $\pi(2) = 2$ and $\pi(3) = 1$.

Selection sort algorithm

The **selection sort algorithm** algorithm is a sorting algorithm.

Example: $(x_i)_{1..3} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) into alphabetical order.

Apply the least element algorithm to $(x_{\pi(i)})_{1..3}$. Gaining a modification to π given by $\pi(1) = 3$, $\pi(2) = 2$ and $\pi(3) = 1$.

Apply the least element algorithm $(x_{\pi(i)})_{2..3}$. We gain a modification to π given by $\pi(1) = 3$, $\pi(2) = 1$ and $\pi(3) = 2$.

Selection sort algorithm

The **selection sort algorithm** algorithm is a sorting algorithm.

Example: $(x_i)_{1..3} = (J, O, E)$ with index set $\{1, 2, 3\}$. Our index function begins as $\pi(i) = i$. We want to know how to rearrange (J, O, E) into alphabetical order.

Apply the least element algorithm to $(x_{\pi(i)})_{1..3}$. Gaining a modification to π given by $\pi(1) = 3$, $\pi(2) = 2$ and $\pi(3) = 1$.

Apply the least element algorithm $(x_{\pi(i)})_{2..3}$. We gain a modification to π given by $\pi(1) = 3$, $\pi(2) = 1$ and $\pi(3) = 2$.

We complete stop the algorithm since we reached the end of our indexing.

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 $\{1, \dots, n\}$.
Output: Modification to π, so that $(x_{\pi(i)})_{1..n}$ is in increasing 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
increasing 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 increasing 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 increasing 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 increasing 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
$s = 1$							
After 1st	$\pi(i)$	**5**	2	3	4	**1**	6
iteration	$x_{\pi(i)}$	**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 increasing 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
$s = 1$							
After 1st	$\pi(i)$	**5**	2	3	4	**1**	6
iteration	$x_{\pi(i)}$	**B**	D	C	E	**F**	C
$s = 2$							
After 2nd	$\pi(i)$	5	**3**	**2**	4	1	6
iteration	$x_{\pi(i)}$	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 increasing 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 increasing 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 increasing 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.

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.

So:

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

$1 + 2 + \cdots + (n-1) = (n-1)\left(\frac{1+(n-1)}{2}\right)$ (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

$1 + 2 + \cdots + (n-1) = (n-1)\left(\frac{1+(n-1)}{2}\right)$ (sum of an arithmetic series).

That is: $\quad \forall \boldsymbol{n} \in \quad \boldsymbol{T_n} = \dfrac{\boldsymbol{n(n-1)}}{2}.$

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.