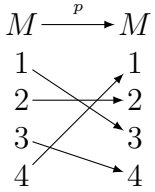
Algorithmische Erzeugung von Permutationen

Permutationen

Eine *Permutation* von n Objekten einer Menge M ist eine bijektive (umkehrbar eindeutige) Abbildung der Elemente von M auf sich selbst. *Beispiel:* $M = \{1, 2, 3, 4\}$



$$p(1) = 3$$
$$p(2) = 2$$
$$p(3) = 4$$
$$p(4) = 1$$

M kann eine beliebige Menge (z. B. $M = \{P, Q, R, S\}$) sein. Wenn die spezielle Bezeichung der Elemente keine Rolle spielt, verwenden wir jedoch wie im Beispiel oben die ersten n natürlichen Zahlen.

Die Zweizeilenform

Um Permutationen platzsparender darzustellen, verwendet man die *Matrixschreibweise* (Zweizeilenform). Die Permutation p aus dem obigen Beispiel kann dann so dargestellt werden:

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$$

Das in der oberen Zeile stehende Element wird jeweils auf das darunter liegende Element abgebildet. Es ist üblich (aber nicht zwingend), dass die Elemente in der ersten Zeile aufsteigend sortiert sind. Daher ist auch

$$p = \begin{pmatrix} 3 & 1 & 4 & 2 \\ 4 & 3 & 1 & 2 \end{pmatrix}$$

eine gültige Zweizeilendarstellung der Permutation p.

Die „Einzeilenform“

Wir denken uns die Elemente in der oberen Zeile aufsteigend sortiert. Dann ist die Zweizeilenform eindeutig und es genügt, nur die Elemente der unteren Zeile aufzuschreiben.

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix} \quad \Leftrightarrow \quad p = \begin{pmatrix} 3 & 2 & 4 & 1 \end{pmatrix}$$

Die Zyklendarstellung

$$q = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 4 & 3 & 1 & 5 \end{pmatrix} \quad \Leftrightarrow \quad \begin{aligned} q &= \begin{pmatrix} 1 & 6 & 5 \end{pmatrix} \begin{pmatrix} 2 \end{pmatrix} \begin{pmatrix} 3 & 4 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 6 & 5 \end{pmatrix} \begin{pmatrix} 3 & 4 \end{pmatrix} \end{aligned}$$

„Folge“ dem kleinsten Wert: $1 \to 6 \to 5 \to 1 \Rightarrow$ Zyklus (1 6 5)

„Folge“ dem nächstkleinsten unverbrauchten Wert: $2 \to 2 \Rightarrow$ Zyklus (2)

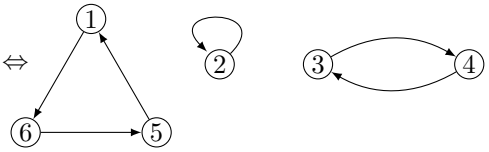
„Folge“ dem nächstkleinsten unverbrauchten Wert: $3 \to 4 \to 3 \Rightarrow$ Zyklus (3 4)

Fixpunkte wie (2) werden weggelassen. Sortierung $\Rightarrow$ Eindeutigkeit der Darstellung

Die Graphendarstellung

Hierbei wird die Zyklendarstellung als *gerichteter Graph* dargestellt.

$$q = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 2 & 4 & 3 & 1 & 5 \end{pmatrix} \Leftrightarrow$$



Die Komposition von Permutationen

Sind p und q Permutationen von $M = \{1, 2, \ldots, n\}$, so bezeichnet $(q \circ p)$ die Komposition (Verknüpfung) der Permutation p mit der Permutation q, die wie folgt definiert wird:

$(q \circ p)(1) = q(p(1))$

$(q \circ p)(2) = q(p(2))$

$\ldots$

$(q \circ p)(n) = q(p(n))$

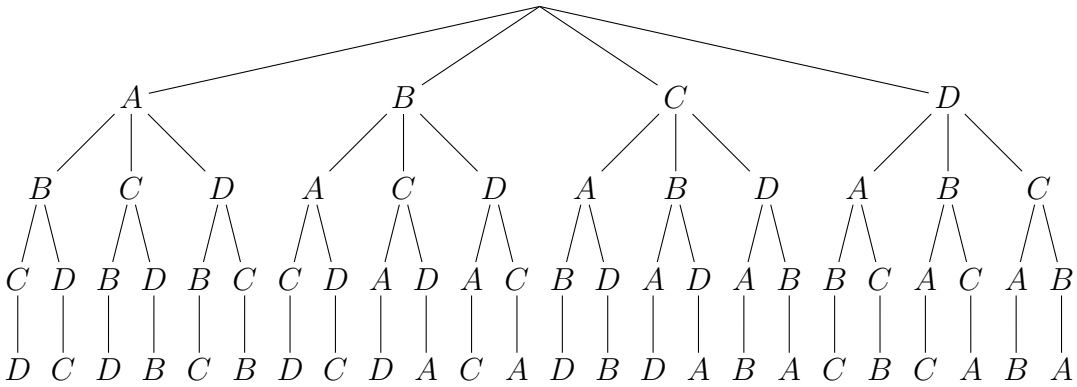
Beachte: Die Komposition wird von rechts nach links ausgeführt.

Beispiel: $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 2 & 3 \end{pmatrix}$

Man vergewissere sich, dass die Komposition von zwei Permutationen der Menge M „wohldefiniert“ ist, d. h. wieder eine gültige Permutation der Elemente von M erzeugt.

Anzahl Permutationen einer Menge mit 4 Elementen

Die Menge aller möglichen Permutationen der Elemente aus $M = \{A, B, C, D\}$ lassen sich in einem Baumdiagramm darstellen.



Da in einer Permutation jedes Element genau einmal verwendet werden muss, gilt hier: Eines von vier Elementen kann an der ersten Position, eines der drei übrigen Elemente kann an der zweiten Position, eines der übrigen zwei Elemente kann an der dritten Position und das verbleibende Element muss an der vierten Position stehen. Nach der Produktregel der Kombinatorik ergibt das $4 \cdot 3 \cdot 2 \cdot 1 = 24$ Möglichkeiten.

Anzahl Permutationen einer Menge mit n Elementen

Eine analoge Überlegung für eine Menge M mit n Elementen führt zur Formel für die Berechnung der Anzahl Permutationen der Elemente aus M.

$n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1 \stackrel{\text{def}}{=} n!$ [lies: n *Faktultät*]

Die folgende Tabelle enthält die Anzahl der Permutationen $n!$ für eine Menge mit n Elementen.

n	0	1	2	3	4	5	6	7
$n!$	1	1	2	6	24	120	720	5040
		$\overset{\cdot 1}{\rightarrow}$	$\overset{\cdot 2}{\rightarrow}$	$\overset{\cdot 3}{\rightarrow}$	$\overset{\cdot 4}{\rightarrow}$	$\overset{\cdot 5}{\rightarrow}$	$\overset{\cdot 6}{\rightarrow}$	$\overset{\cdot 7}{\rightarrow}$

Unter den Pfeilen stehen die Faktoren, mit denen $(n+1)!$ rekursiv aus $n!$ berechnet werden kann. Damit wird der Wert $0! = 1$ plausibel.

Fehlstände einer Permuation

Um die „Unordnung“ einer Permutation, d. h. den Grad der Abweichung einer Permuation p von der identischen Permutation

$$\text{id} = \begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ 1 & 2 & 3 & \ldots & n \end{pmatrix}$$

zu messen, wird die Zahl der Paare (i, j) gezählt, für die $i < j$ aber $p(i) > p(j)$ gilt; d. h. deren permutierte Positionen absteigend sortiert sind. Diese Zahl wird *Fehlstands-* oder *Inversionszahl* genannt.

Beispiele: (in Einzeilenform)

- $p = (2\ 3\ 1) \quad \Rightarrow \quad \text{inv}(p) = 2$ Fehlstände: (2 1), (3 1)
- $p = (2\ 4\ 1\ 3\ 5) \quad \Rightarrow \quad \text{inv}(p) = 3$ Fehlstände: (2 1), (4 1), (4 3)
- $p = (4\ 3\ 2\ 1) \quad \Rightarrow \quad \text{inv}(p) = 6$ Fehlstände: (4 3), (4 2), (4 1), (3 2), (3 1), (2 1)

Permutationen in der Informatik

Permutationen spielen in vielen Gebieten der Informatik eine wichtige Rolle.

- zur Analyse von Sortierverfahren und anderer Algorithmen,
- für die Verschlüsselung von Nachrichten,
- bei Algorithmen, die alle Permutationen berücksichtigen.

Algorithmische Erzeugung aller Permutationen

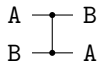
Wir werden nun ein Verfahren beschreiben, welches alle Permutationen von n Elementen ($M = \{1, 2, \ldots, n\}$) durch ein System geeigneter Vertauschungen von jeweils zwei Elementen erzeugt.

Zunächst tasten wir uns für $n = 2, 3, 4$ „naiv“ mit Hilfe sogenannter Permutationsnetzwerke an die Aufgabe heran.

Anschliessend geben wir ein System von Vertauschungsregeln an, das sich algorithmisch leichter implementieren lässt.

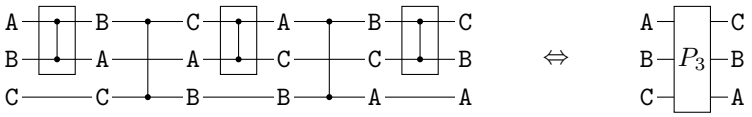
Netzwerkdarstellung der Permutationen für $n = 2$

Für zwei Elemente gibt es nur eine Vertauschungsart. Wir stellen sie in Form eines Permutationsnetzwerks dar, das die beiden zu tauschenden Elemente durch eine vertikalen Linie verbindet, deren Enden durch einen Punkt hervorgehoben werden.



Eine Netzwerkdarstellung der Permutationen für $n = 3$

Wir erzeugen nun das Permutationsnetzwerk für $n = 3$, indem wir jeweils das oben angegebene Permutationsnetzwerk für $n = 2$ verwenden, um die ersten beiden Elemente zu vertauschen und danach das Element an der dritten Position so mit einem der „darunterliegenden“ Elemente vertauschen, so dass an der dritten Postion (im Bild ganz unten) die Elemente in absteigender Reihenfolge erscheinen. Dieses Schema wurde willkürlich gewählt, denn es gibt mehrere Netzwerke, welche alle Permutationen für drei Elemente durch paarweise Vertauschungen erzeugen ($\rightarrow$ Übungen).

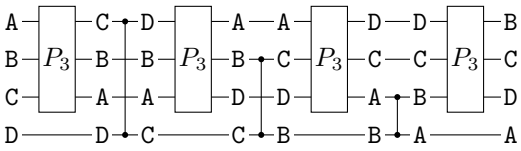


Die Vertauschungen für $n = 2$ sind jeweils durch einen Kasten gekennzeichnet. Um im nächsten Schritt die Permutationen für $n = 3$ als Subnetzwerk wiederzuverwenden, gebrauchen wir das rechts angegebene Symbol.

Eine Netzwerkdarstellung der Permutationen für $n = 4$

Wir haben gesehen, dass das vorangehende Netztwerk alle Permutationen von $n = 3$ Elementen generiert. Also können wir es dazu verwenden, die Permutationen der ersten drei Elemente zu erzeugen und dann durch systematisches Tauschen mit dem vierten Element zu kombineren.

Aus Platzgründen notieren wir jeweils nur den Schritt, der das letzte (unterste) Element vertauscht. Diesmal müssen wir ein andere Vertauschungssystem anwenden, damit das letzte Element absteigend sortiert wird.



Fortsetzung des Verfahrens

Wir erzeugen alle Permutationen von n Elementen, indem nach der i-ten Anwendung von P_{n-1} die Elemente in absteigeder Reihenfolge ihrer ursprünglichen Indizes an der Position n stehen. Die folgende Tabelle enthält diese Positionen $T(n, i)$.

n	$T(n,i)$								
2	1								
3	1	1							
4	1	2	3						
5	3	1	3	1					
6	3	4	3	2	1				
7	5	3	1	5	3	1			
8	5	2	7	2	1	2	3		
9	7	1	5	3	1	9	2	3	
10	7	8	1	6	5	4	9	2	1

Beispiel für $n = 5$: Führe P_4 vier Mal auf den ersten vier Positionen aus. Dazwischen tausche $P[5] \leftrightarrow P[3]$, $P[5] \leftrightarrow P[1]$, $P[5] \leftrightarrow P[3]$, $P[5] \leftrightarrow P[1]$.

Verallgemeinerung?

Auch wenn die obige Tabelle einige Regelmässigkeiten aufweist, ist es nicht unmittelbar klar, welcher Algorithmus diese Zahlen erzeugt.

Lassen wir die Forderung fallen, dass die Elemente an der letzten Position absteigend sortiert sein müssen, so gibt es mindestens ein Verfahren, das die Vertauschungen mit der letzten Position einfacher beschreibt.

Ein solches Verfahren wurde 1963 von B. R. Heap in der Fachzeitschift *The Computer Journal* veröffentlicht. Es besagt, dass das Element an der Position n bei

- ungeradem n immer mit dem Element an der Position 1 vertauscht wird,
- geradem n mit dem Element an der Position $i = 1, 2, \ldots, (n-1)$ vertauscht wird.

Der Algorithmus von Heap (Pseudocode)

```
function heap(A, k):
    if k = 1 then
        output(A)
    else
        heap(A, k-1)
        for(i←0; i<k-1; i←i+1)
            if k % 2 = 1 then
                swap(A, 0, k)
            else
                swap(A, i, k)
            end if
            heap(A, k-1)
        end for
    end if
```

Die Hilfsfunktion `swap(A, i, j)` (Pseudocode)

Diese Funktion vertauscht in der Liste `A` die Elemente an den Positionen `i` und `j`.

```
function swap(A, i, j):
    tmp ← A[i]
    A[i] ← A[j]
    A[j] ← tmp
```