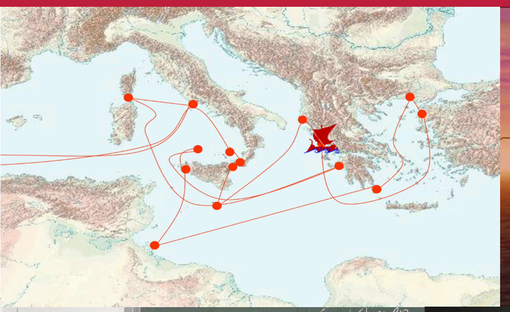
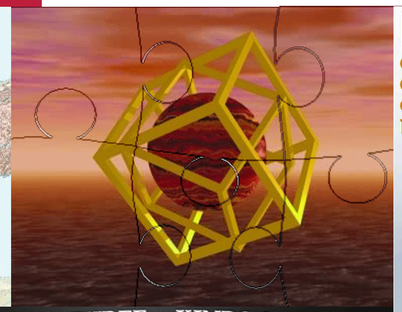
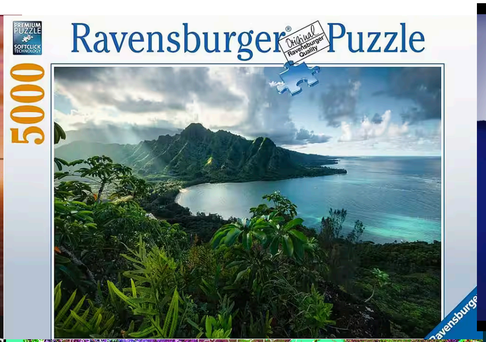
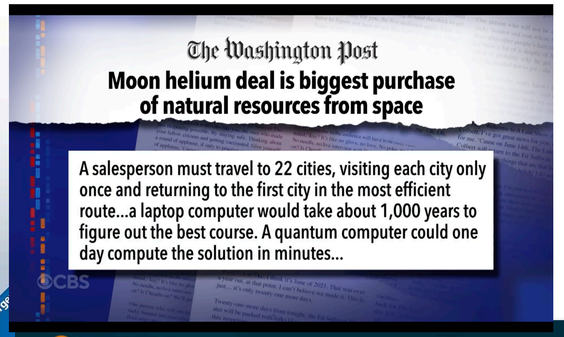
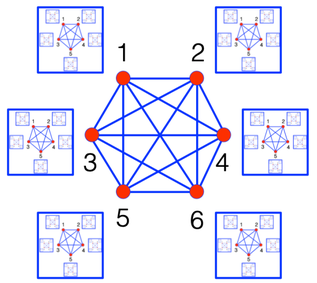
Technische Universität Braunschweig

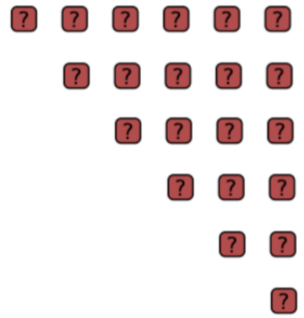


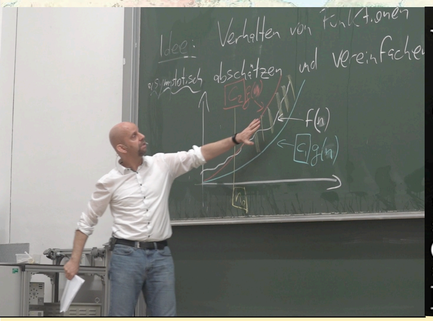




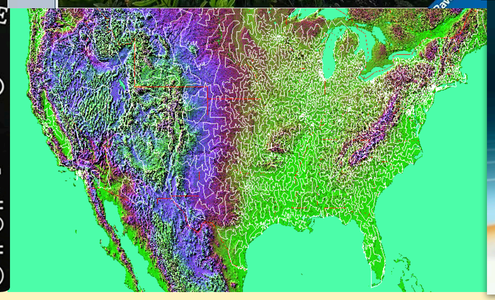


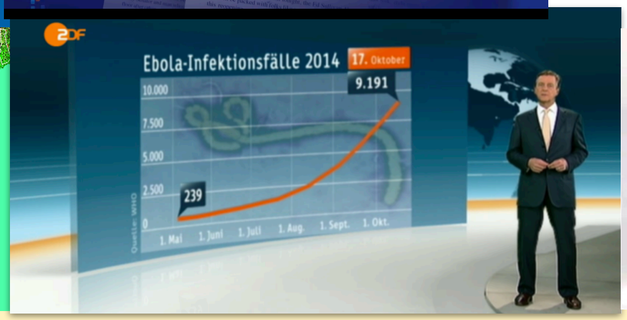


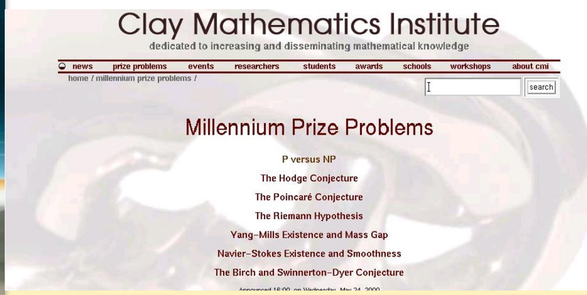












Geht's nicht noch etwas schneller?

Eine Rundreise durch gelöste und ungelöste Probleme des algorithmischen Alltags

Sándor P. Fekete

Department of Computer Science

TU Braunschweig

Die wichtigste Person in dieser Vorlesung?



Menti

Interaktive Umfragen und Quiz-Wettbewerbe in Echtzeit mit Mentimeter



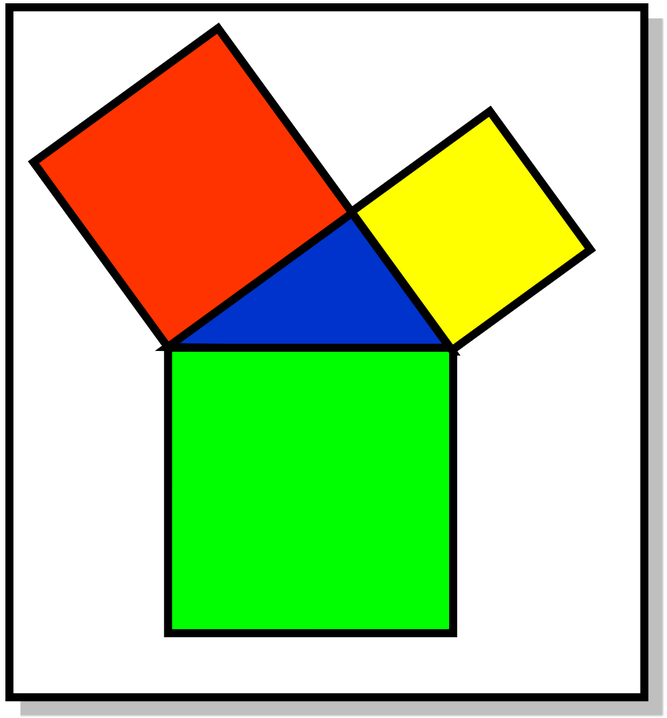
Join at menti.com 7362 6840

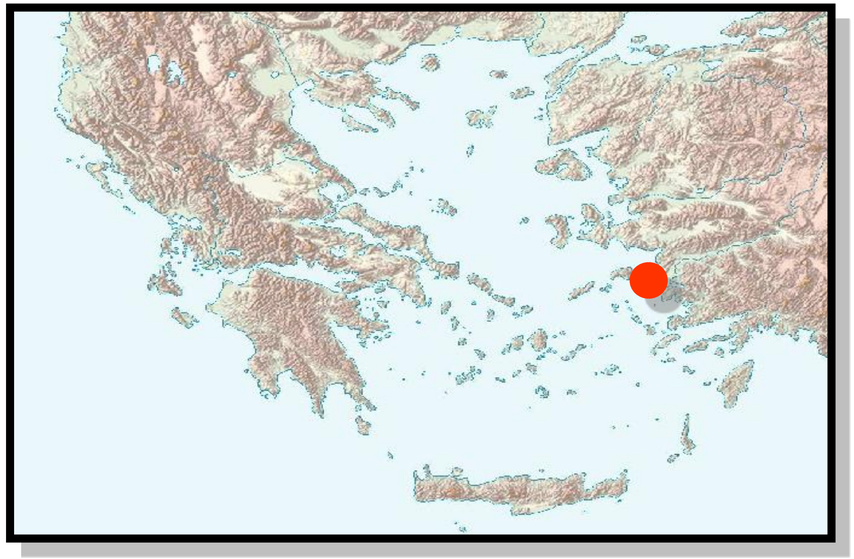
0

1 Die Laufzeit der Geschichte

Die bekannteste mathematische Formel alles Zeiten

$$a^2 + b^2 = c^2$$





Pythagoras (ca. 500 v.Chr.)

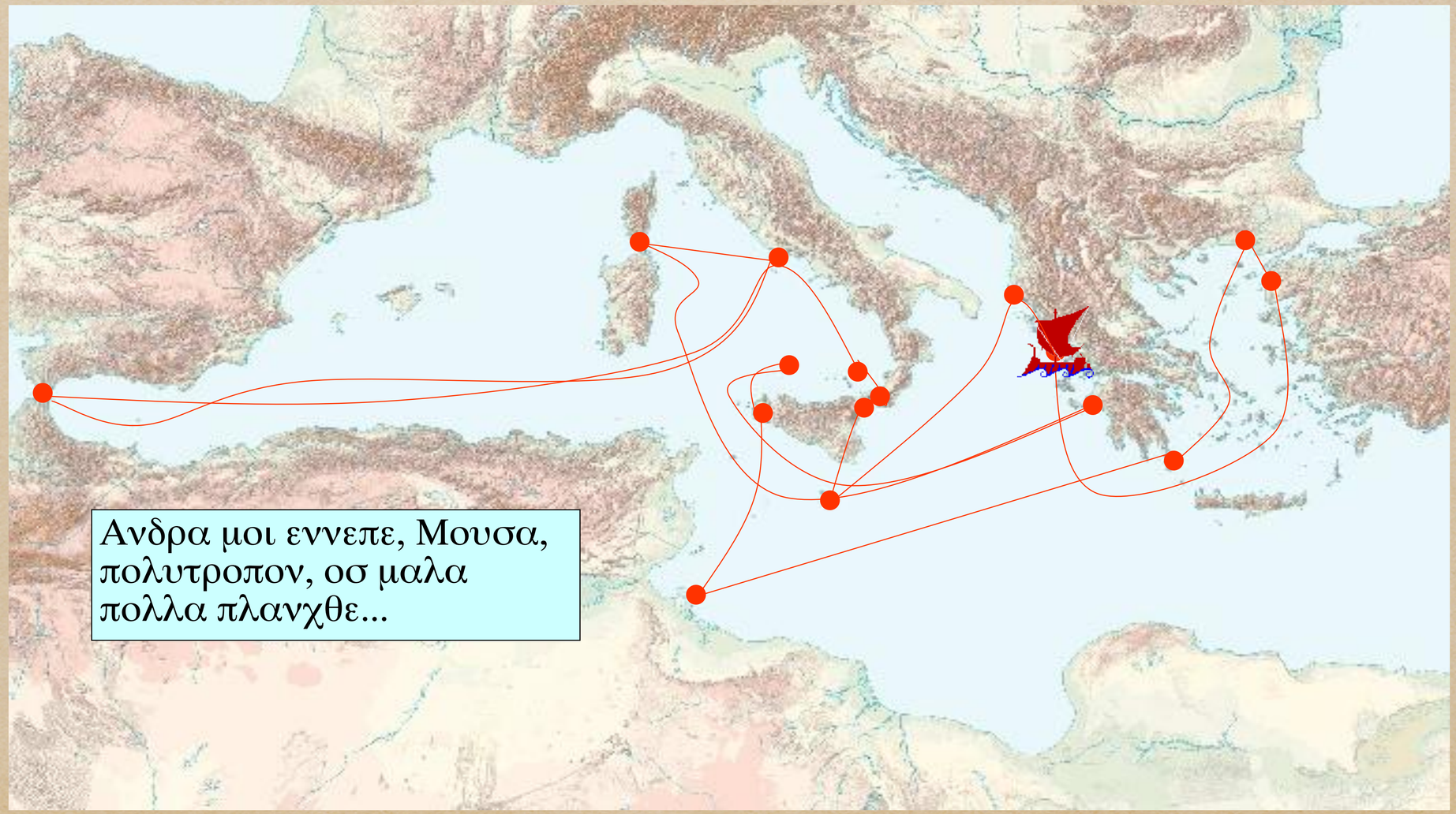
EINIGE JAHRE VOR EUKLID



EINIGE JAHRE VOR EUKLID



Einige Jahre vor Euklid

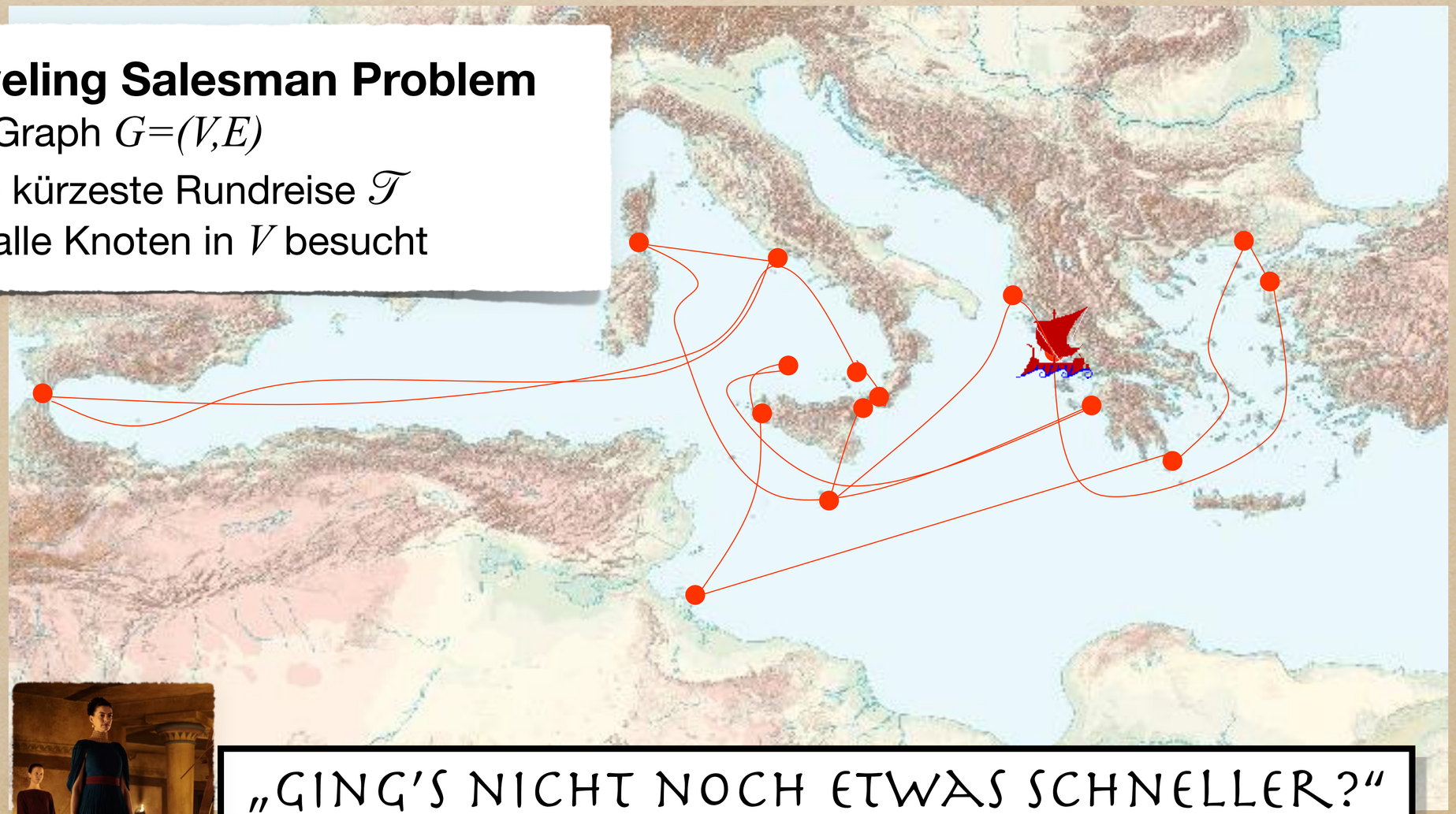


20 Jahre später!

Das Traveling Salesman Problem

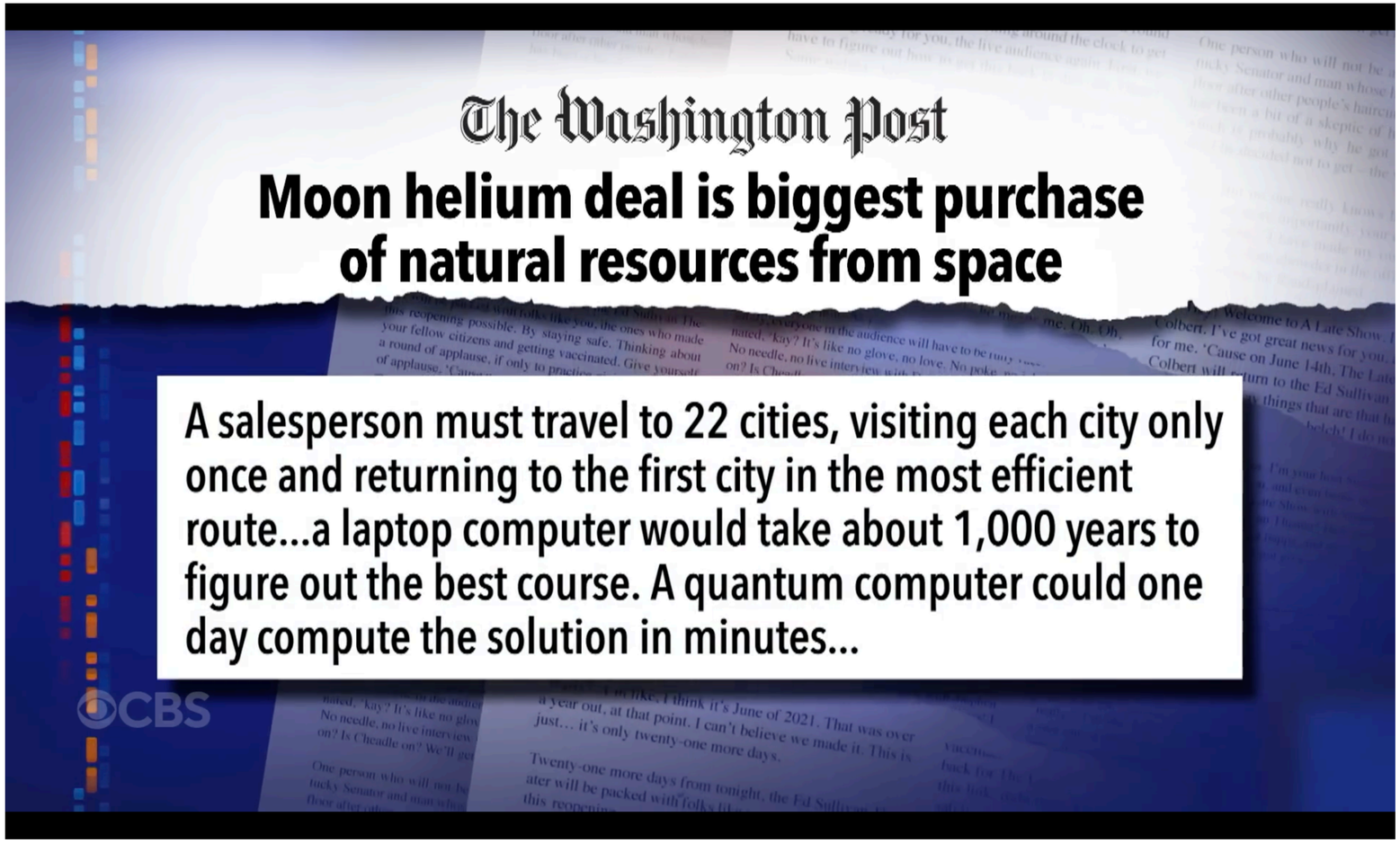
Input: Ein Graph $G=(V,E)$

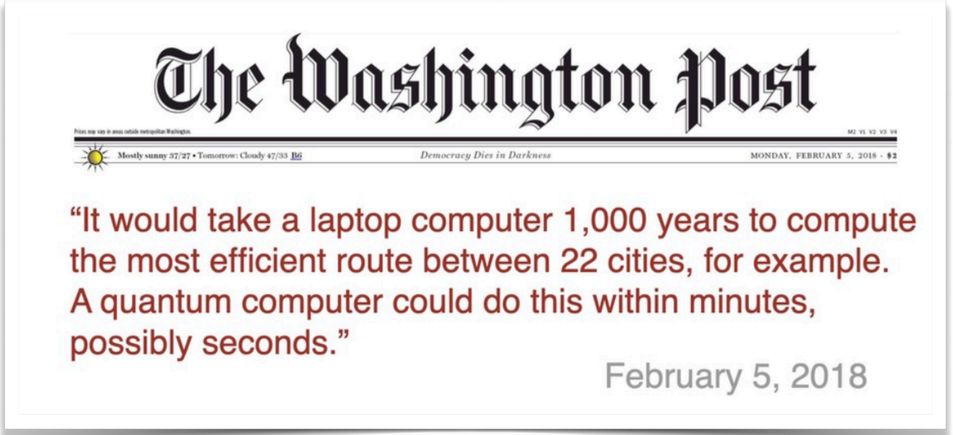
Output: Eine kürzeste Rundreise $\mathcal{T}$
die alle Knoten in V besucht



„Ging's nicht noch etwas schneller?“

Geschäftsmodell





(1) 300 Mio. einsammeln
(2) Zum Mond fliegen
(3) Helium-3 abbauen
(4) Zur Erde bringen
(5) Quantencomputer bauen
(6) Beste Tour berechnen

Geschäftsmodell



BC **Bill Cook <bico@uwaterloo.ca>** CG:SHOP 18. September 2025 um 14:23
Aw: Next Level TSP
An: Fekete, Sandor <s.fekete@tu-braunschweig.de>,
Umgeleitet von: s.fekete@tu-braunschweig.de

Wheeeee! Thanks for the link! I'll add this to a TSP talk
I'm preparing for Korea in October.

Bico

The Washington Post

Democracy Dies in Darkness

"It would take a laptop computer 1,000 years to compute the most efficient route between 22 cities, for example. A quantum computer could do this within minutes, possibly seconds."

February 5, 2018

(1) 300 Mio. einsammeln
(2) Zum Mond fliegen
(3) Helium-3 abbauen
(4) Zur Erde bringen
(5) Quantencomputer bauen
(6) Beste Tour berechnen



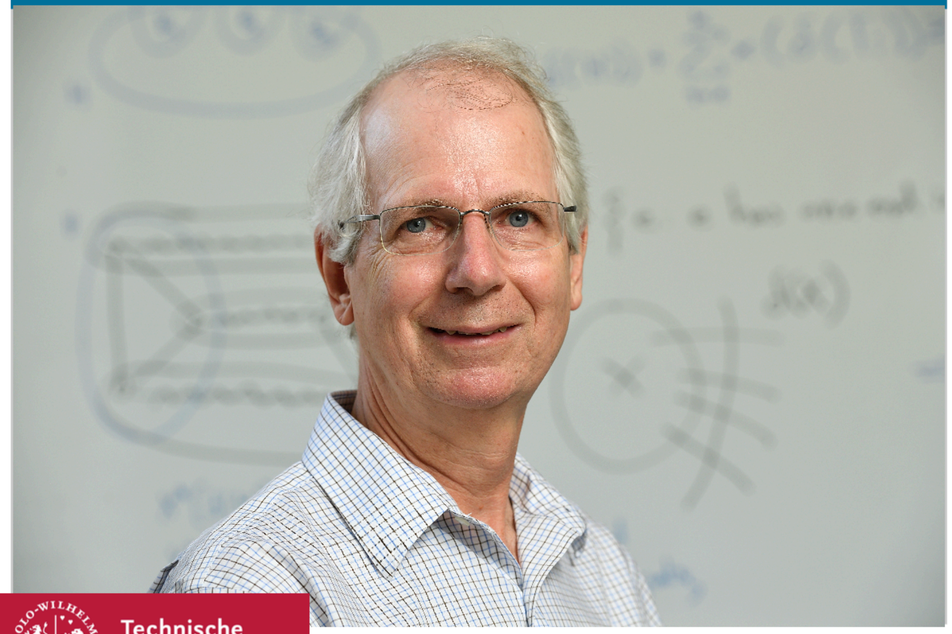
Carl-Friedrich-Gauß-Vorlesung 2026

Prof. Dr. William J. Cook

University of Waterloo

The Traveling Salesman Problem: Package Deliveries, Pub Walks, and Astro Tours

Donnerstag, 07.05.2026, 17:00
Aula, Haus der Wissenschaft, Pockelsstr. 11

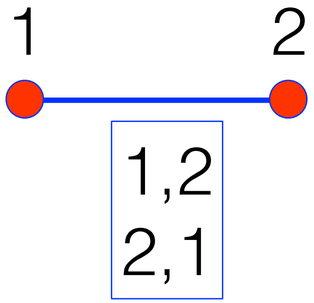


Technische Universität Braunschweig

Carl-Friedrich-Gauß-Fakultät

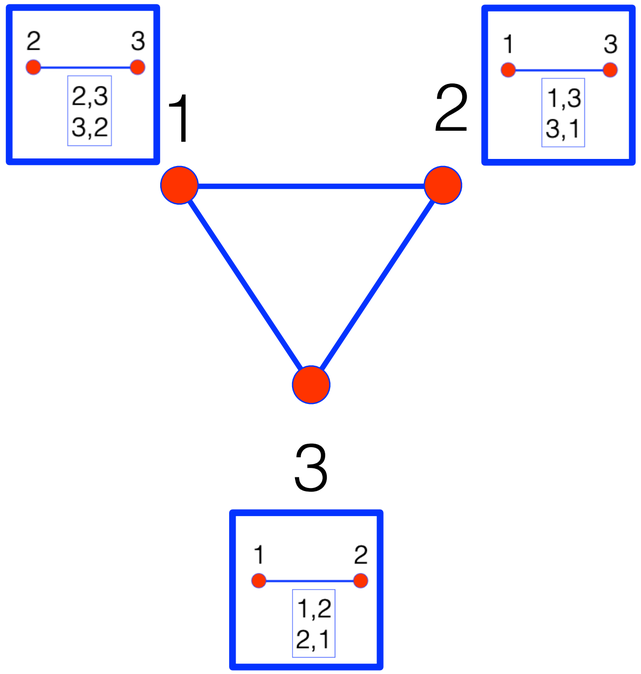
Wie viele mögliche Touren gibt es?

n=2



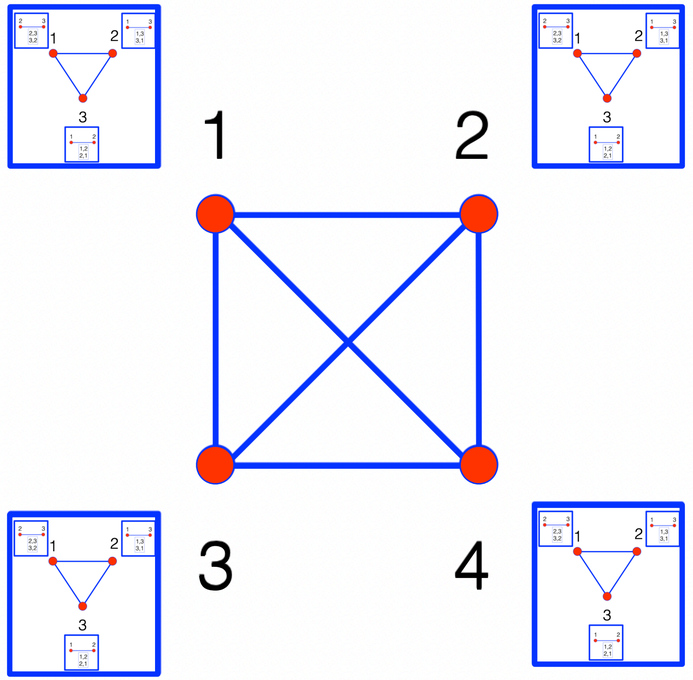
2 Touren

n=3



3*2 = 6 Touren

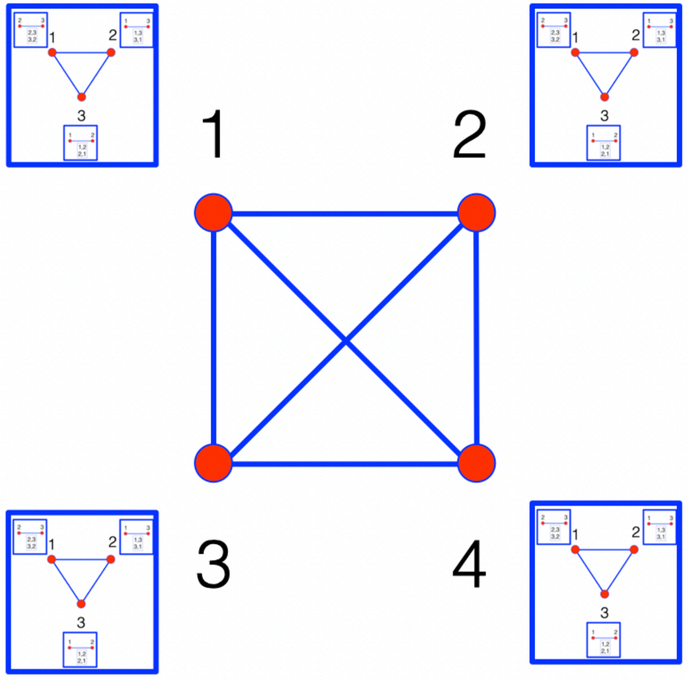
n=4



4*6 = 24 Touren

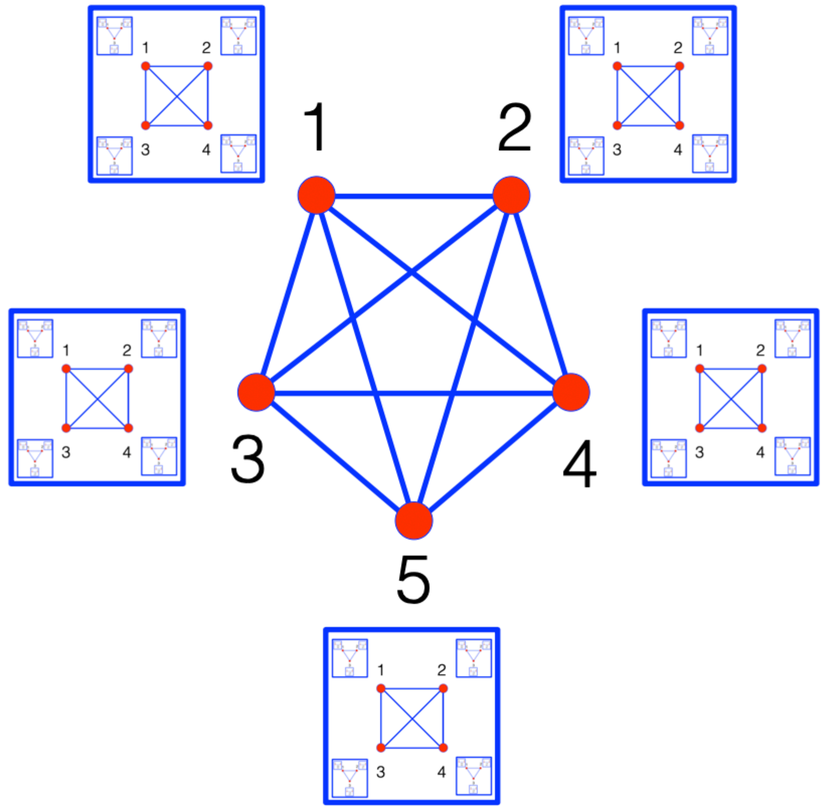
Wie viele mögliche Touren gibt es?

n=4



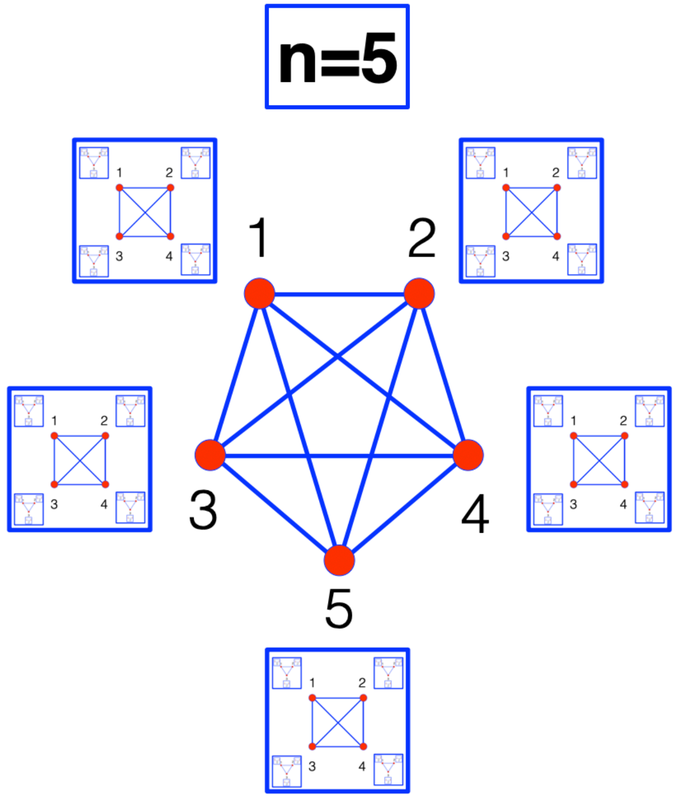
4*6 = 24 Touren

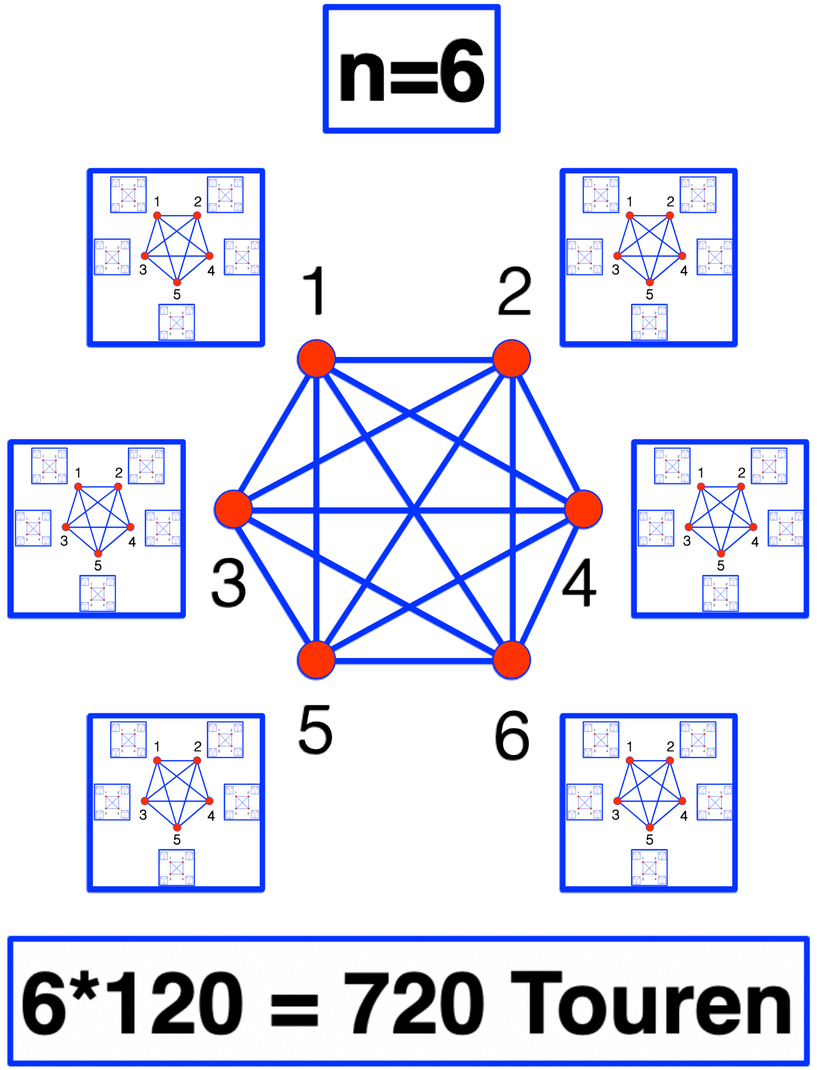
n=5



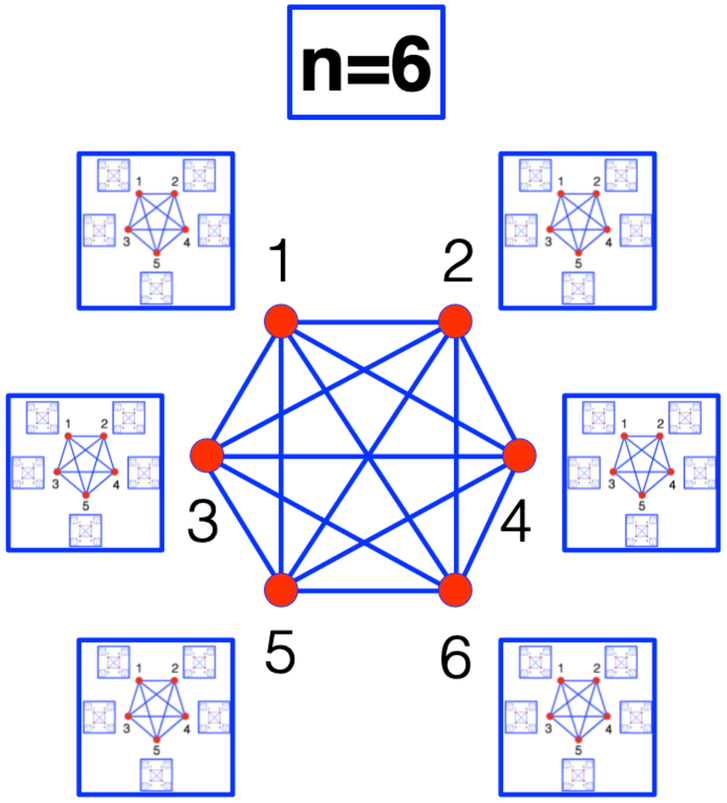
5*24 = 120 Touren

Wie viele mögliche Touren gibt es?





Wie viele mögliche Touren gibt es?



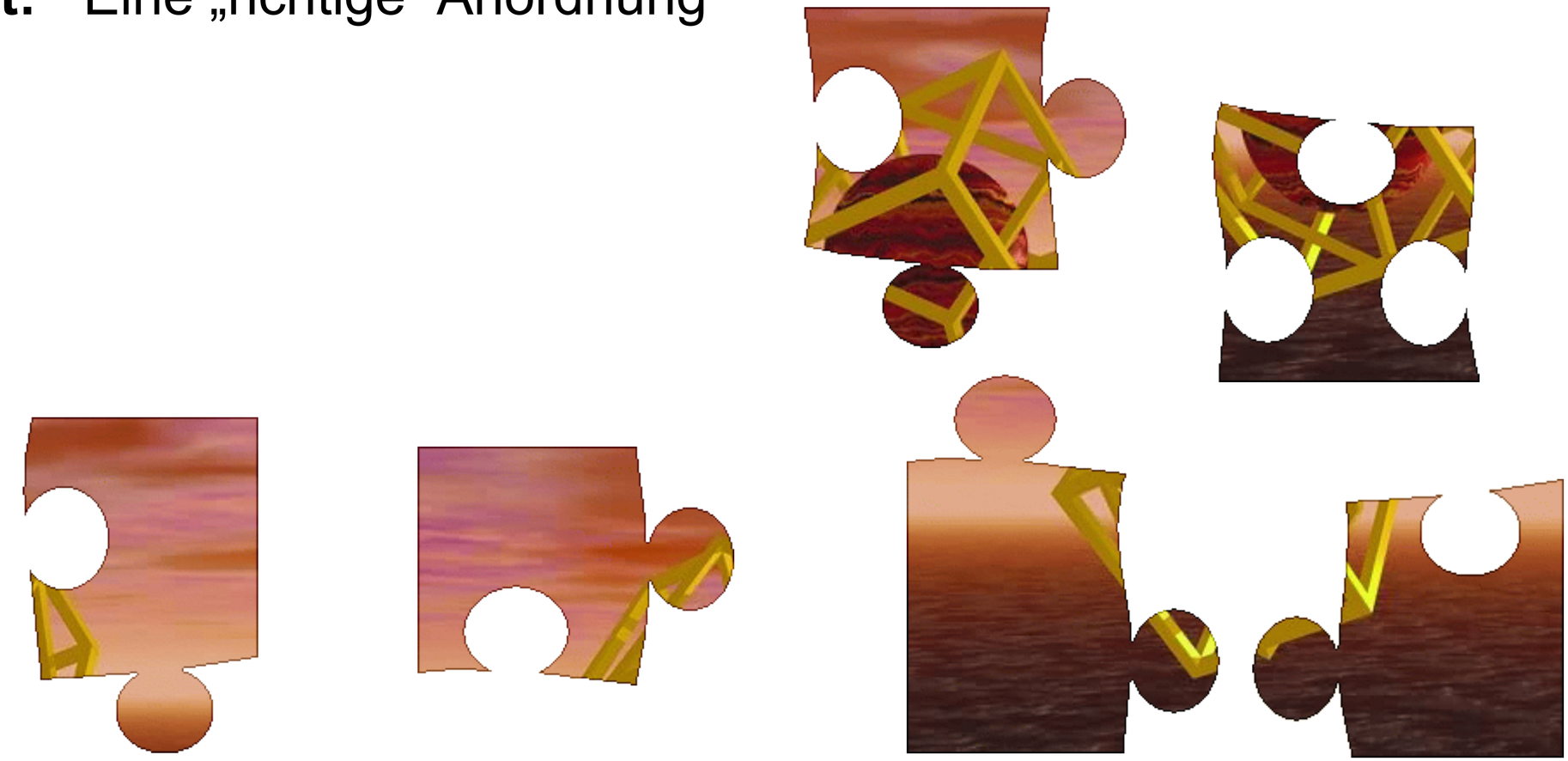
n=2	2 * 1 = 2
n=3	3 * 2 = 3*2*1= 6
n=4	4 * 6 = 4*3*2*1= 24
n=5	5 * 24 = 5*4*3*2*1= 120
n=6	6 * 120 = 6*5*4*3*2*1= 720
n=22	22*21*…*2*1 = 1.124.000.727.777.607.680.000

Bei 3,5 GHz:
31.557.600.000 Sekunden
~ 10.000 Jahre

Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung



Wie lange dauert es, ein Puzzle zu lösen?

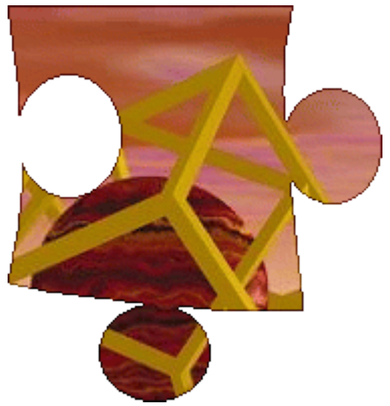
Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung













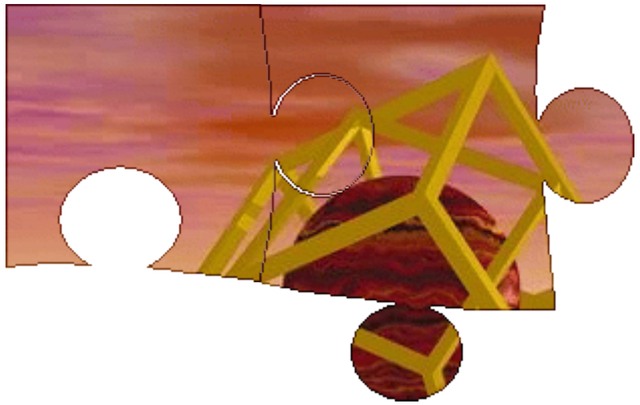


Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

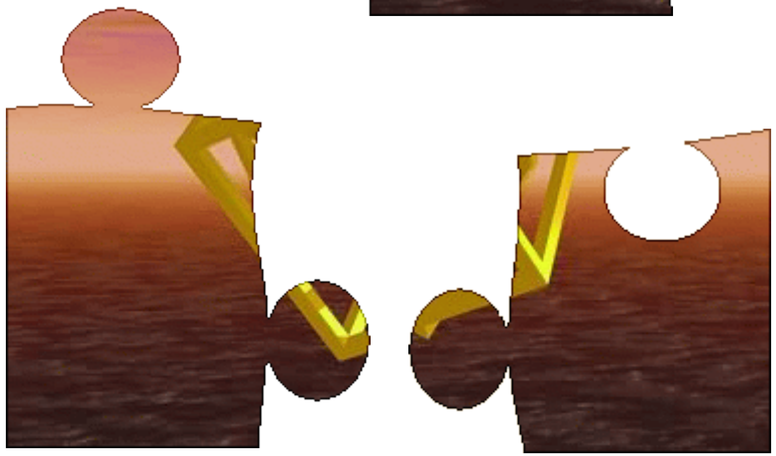
Gesucht: Eine „richtige“ Anordnung







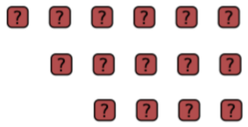


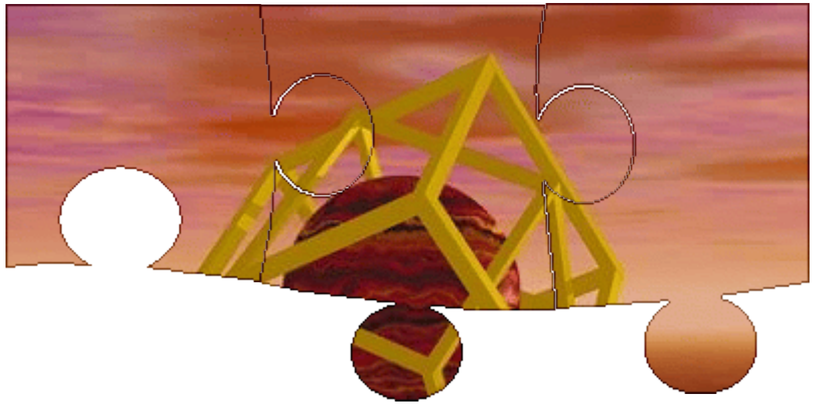


Wie lange dauert es, ein Puzzle zu lösen?

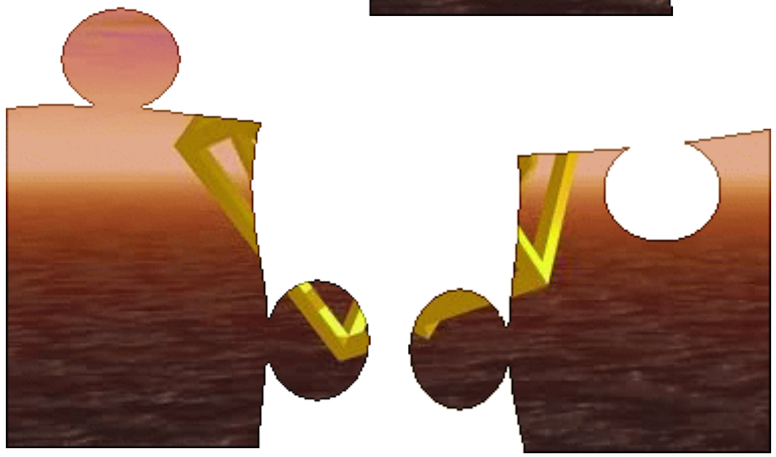
Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung







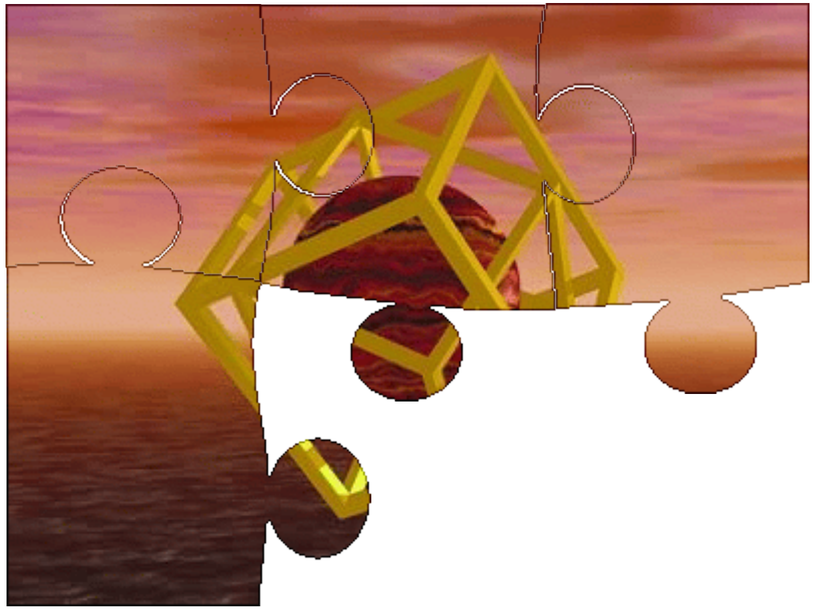


Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung





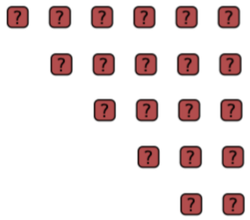


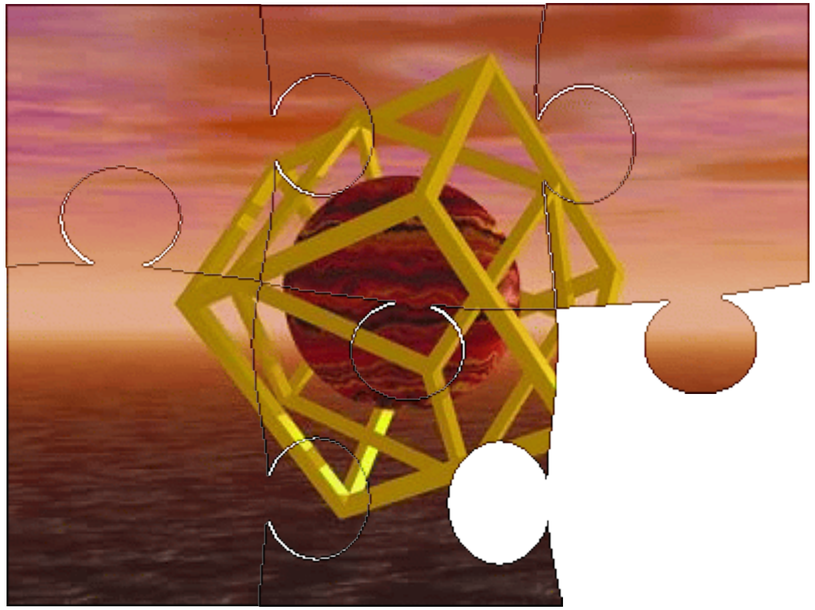


Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung



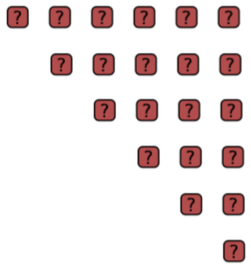


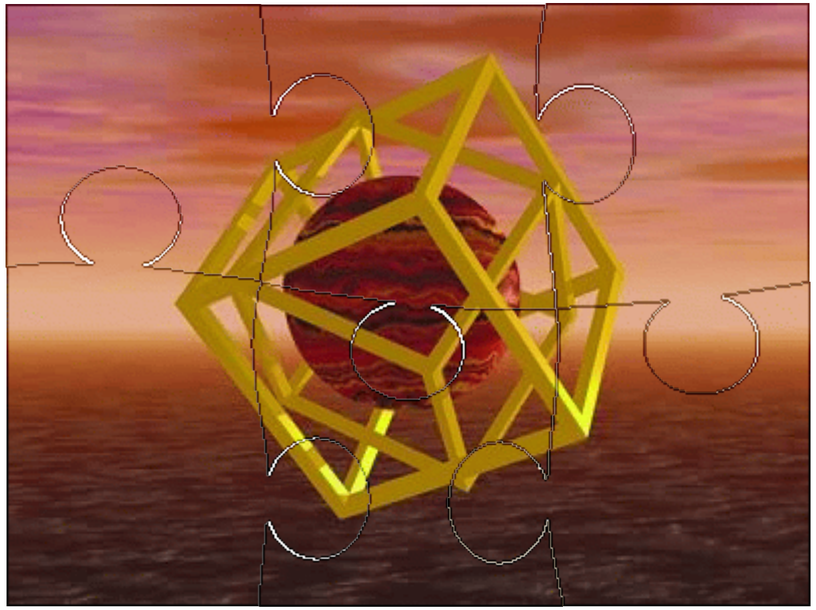


Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung



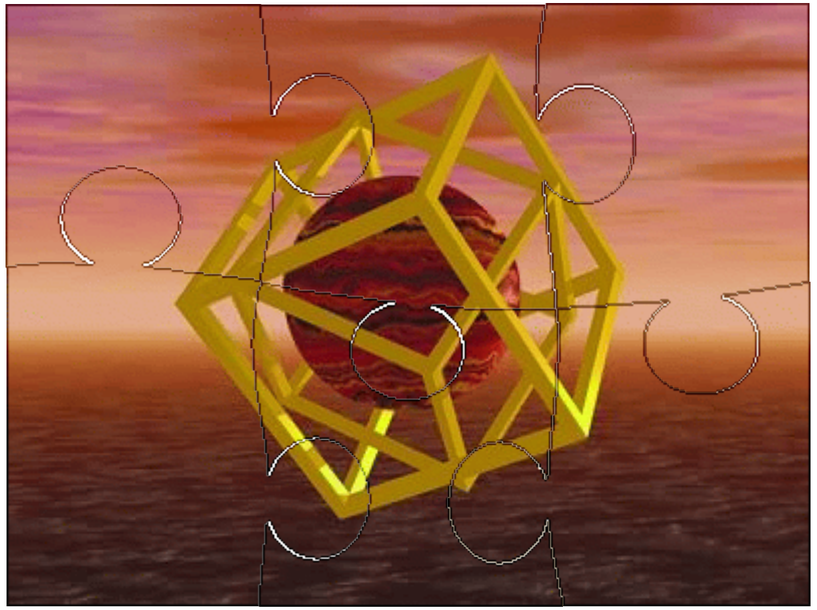


Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine systematische Lösungsmethode





Einfach so:

6+5+4+3+2+1 = 21

1+2+3+4+5+6 = 21

7+7+7+7+7+7 = 42

Wie lange braucht man für *n* Teile?

$$\boxed{n(n+1)/2}$$

n = 6: 21 Schritte

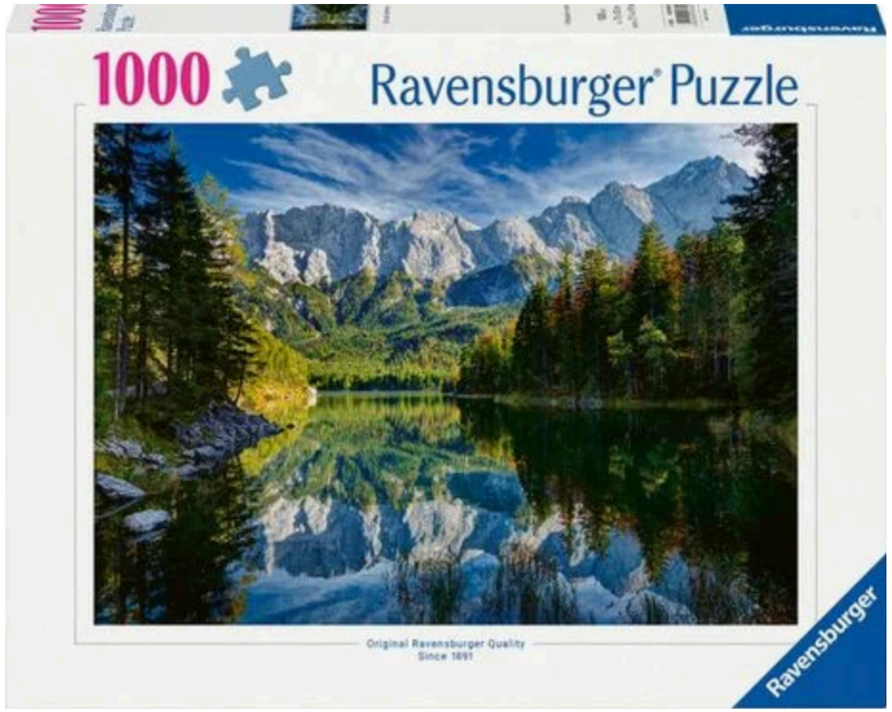
n = 100: 5050 Schritte

Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine systematische Lösungsmethode





Einfach so:

6+5+4+3+2+1 = 21

1+2+3+4+5+6 = 21

7+7+7+7+7+7 = 42

Wie lange braucht man für *n* Teile?

$$\boxed{n(n+1)/2}$$

n =	6:	21 Schritte
n =	100:	5050 Schritte
n =	1000:	500.500 Schritte

Wie lange dauert es, ein Puzzle zu lösen?

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine systematische Lösungsmethode





Einfach so:

6+5+4+3+2+1 = 21

1+2+3+4+5+6 = 21

7+7+7+7+7+7 = 42

Wie lange braucht man für *n* Teile?

n(n+1)/2 $\Theta(n^2)$

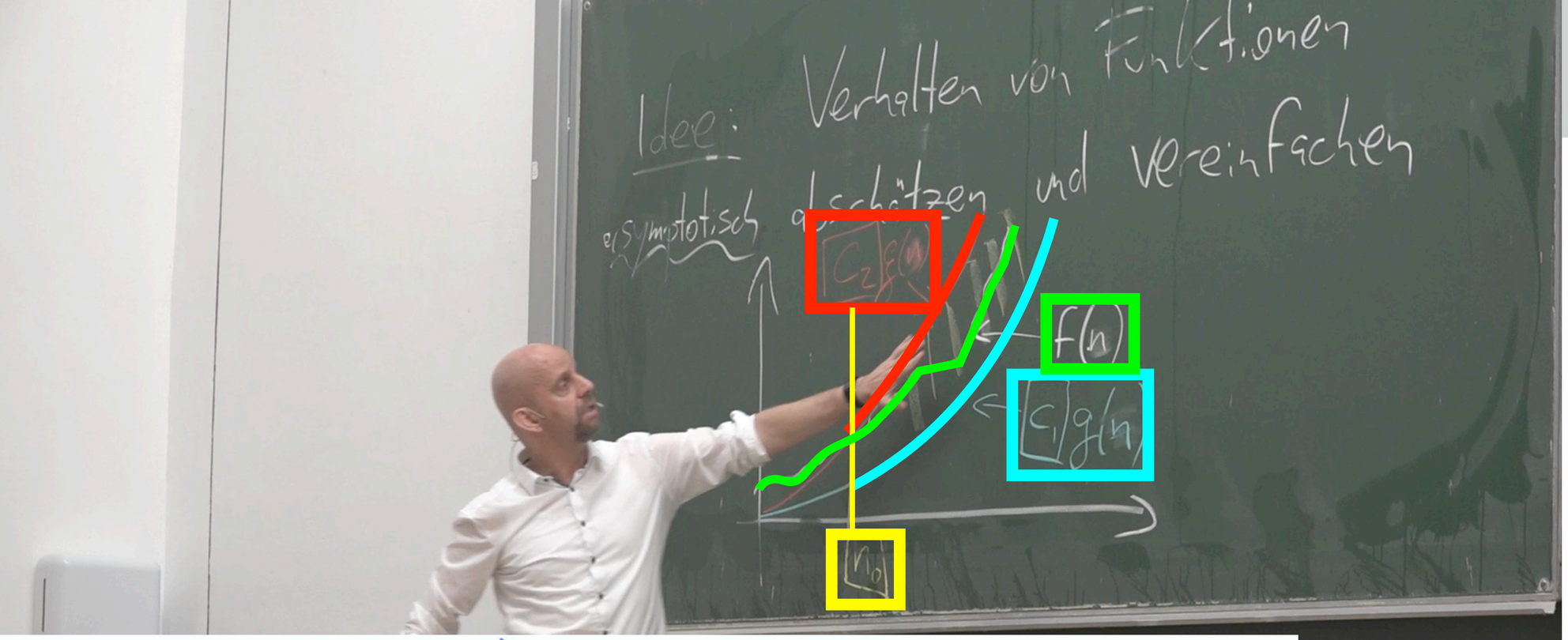
n = 6: 21 Schritte

n = 100: 5050 Schritte

n =1000: 500.500 Schritte

n= 5000: 12.502.500 Schritte

„Geht's nicht noch etwas schneller?“



DEFINITION 3.9 (Θ-Notation)

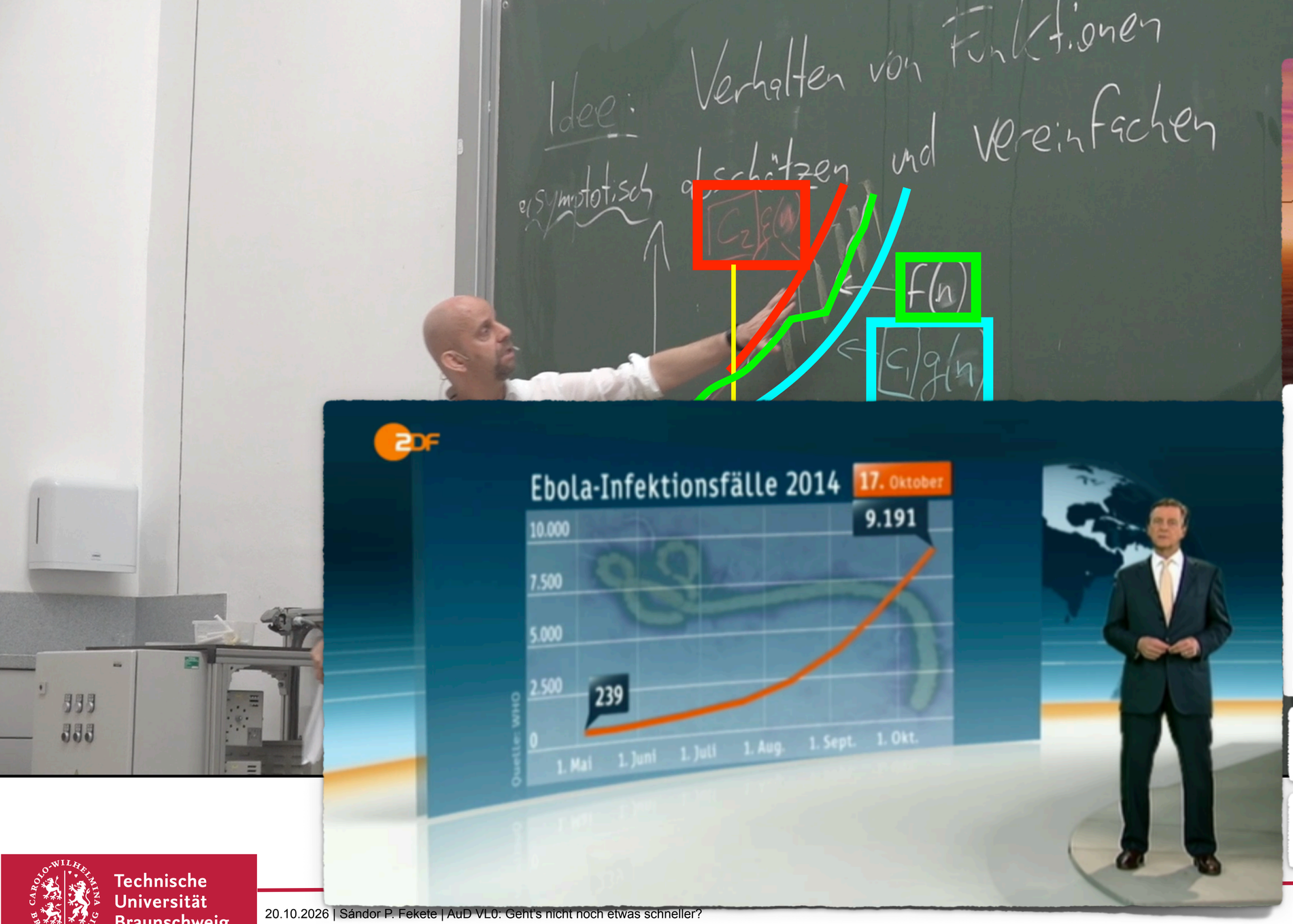
Seien $f, g : \mathbb{N} \to \mathbb{R}$ Funktionen.

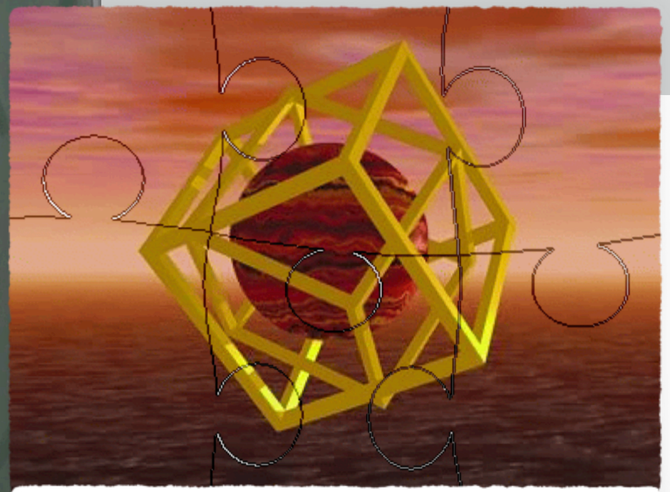
Dann gilt

$f \in \Theta(g) \Leftrightarrow$ Es gibt positive Konstanten c_1 c_2 n_0 mit

$$0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{für alle} \quad n \geq n_0$$

Man sagt: f wächst asymptotisch in derselben Größenordnung wie g.





Wie lange braucht man für *n* Teile?

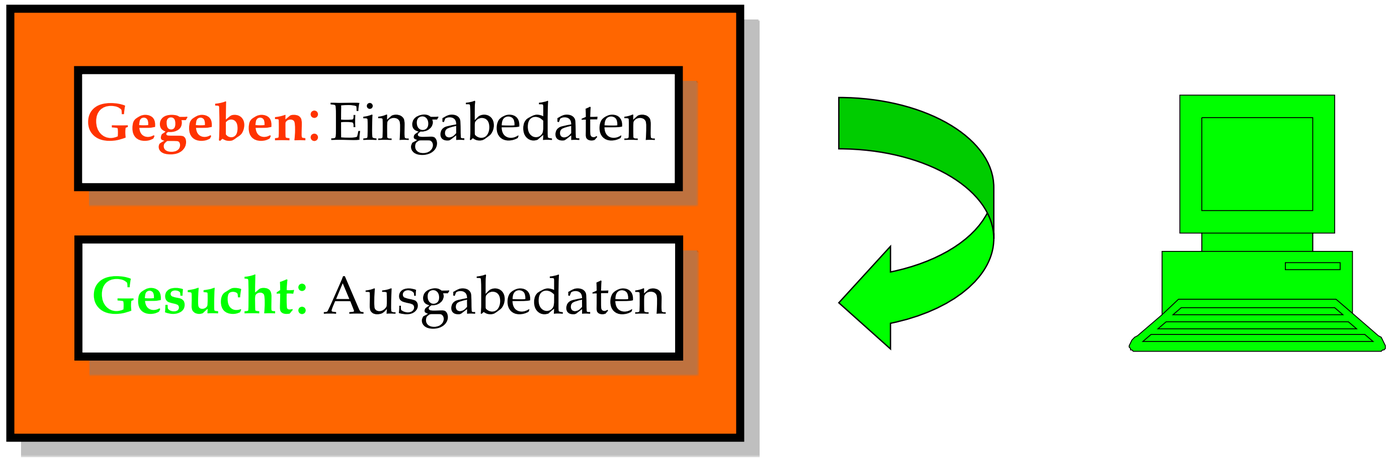
	n(n+1)/2	$\Theta(n^2)$
n = 6:	21	Schritte
n = 100:	5050	Schritte
n =1000:	500.500	Schritte
n= 5000:	12.502.500	Schritte

$\Theta(n^2)$ $(2n)^2 = 4 \cdot n^2$

$\Theta(2^n)$ $2^{2n} = 2^n \cdot 2^n$

Konstruktives Lösen von Problemen

Gegeben: Eine Problembeschreibung



Gesucht: Ein Algorithmus, der das Problem löst

„Geht‘s nicht noch etwas schneller?“

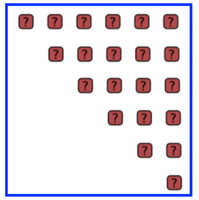
Die Laufzeit von Algorithmen

Gegeben: Eine Problembeschreibung der Größe O(n)

Gegeben: O(n) Zahlen

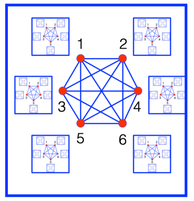
Gesucht: O(n) Zahlen

Die Klasse P



$\Theta(n^2)$ ✓

Gesucht: Ein polynomialer Algorithmus



$\Theta(2^n)$

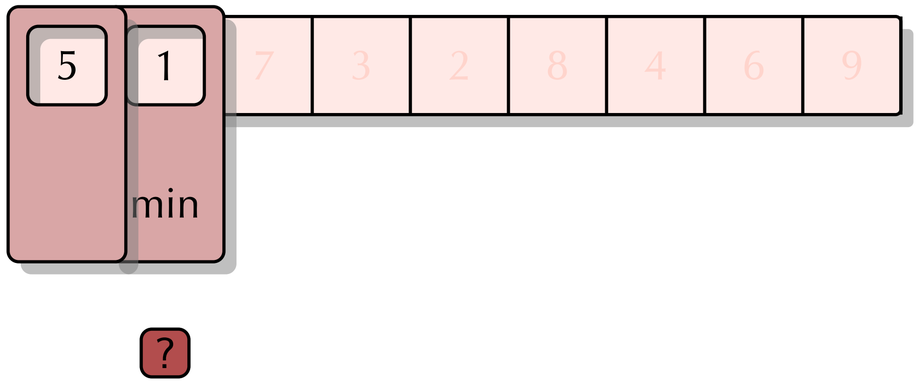
$\Theta(n!)$ ✗

2 Sortieren geht über Probieren

Zwei Sortieralgorithmen

Selection Sort:

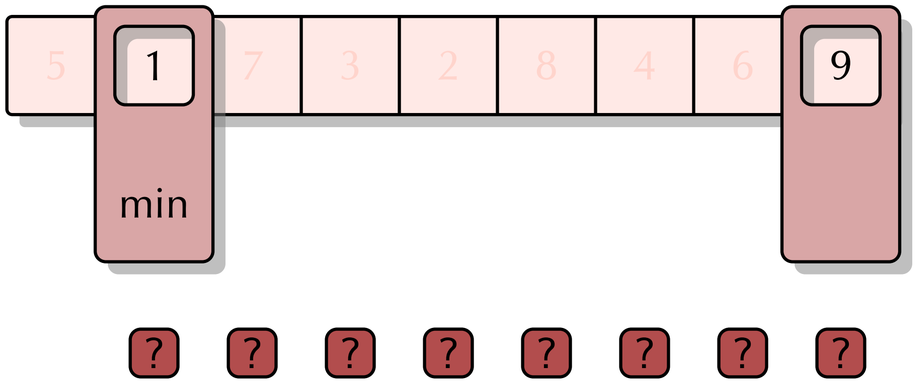
- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle



Zwei Sortieralgorithmen

Selection Sort:

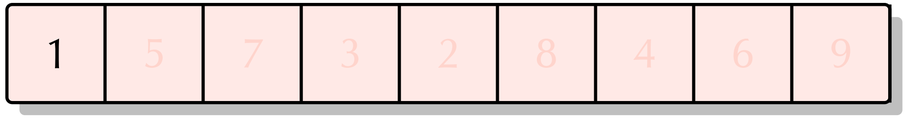
- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle



Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

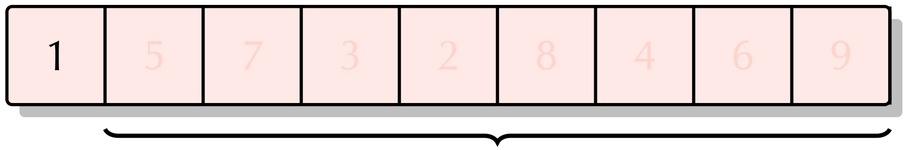




Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

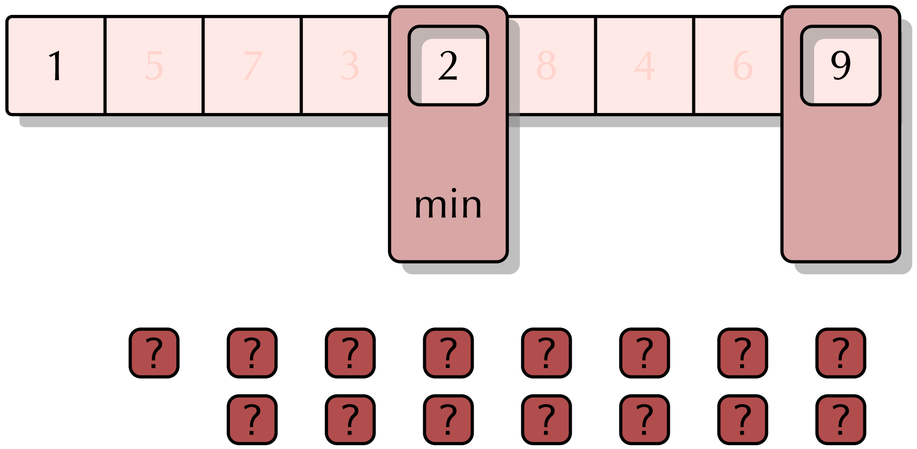




Zwei Sortieralgorithmen

Selection Sort:

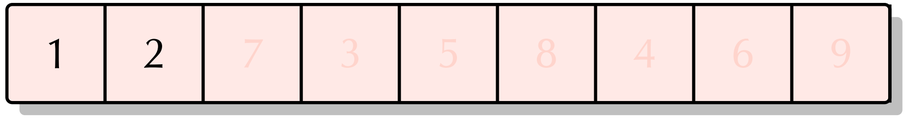
- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

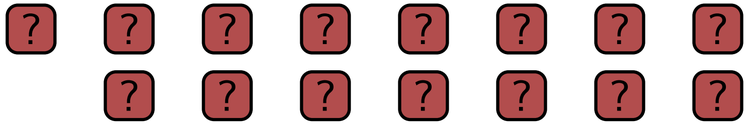


Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

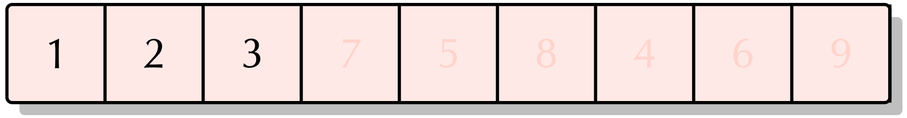


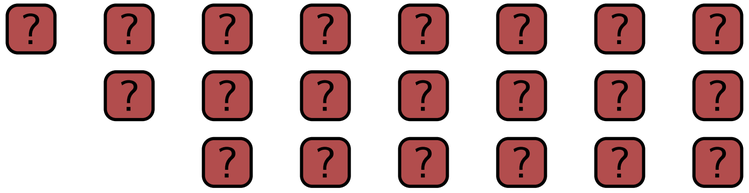


Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

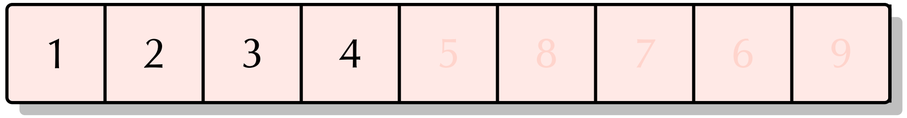


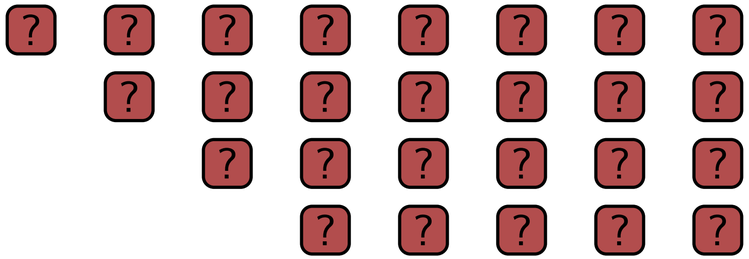


Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

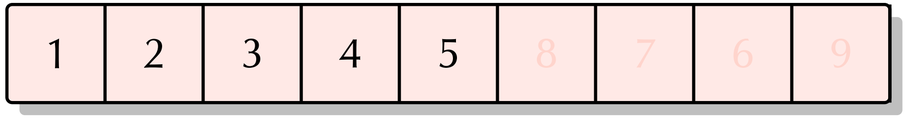


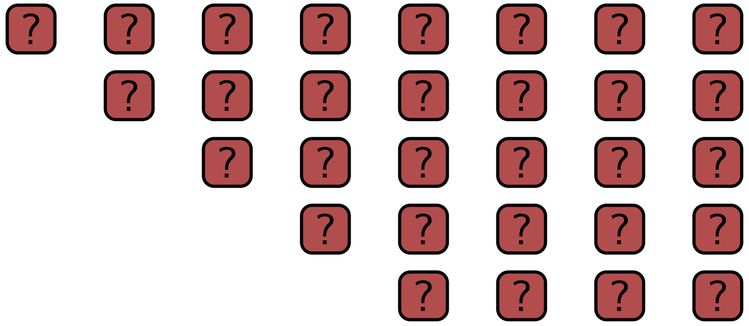


Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

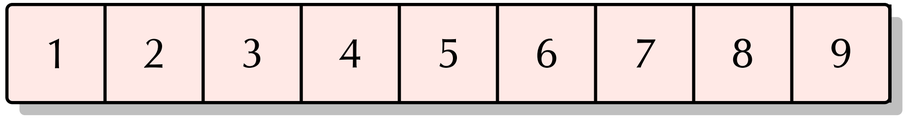




Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle



36 Vergleiche

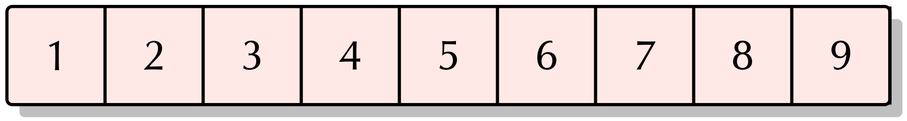


„Geht‘s nicht noch etwas schneller?“

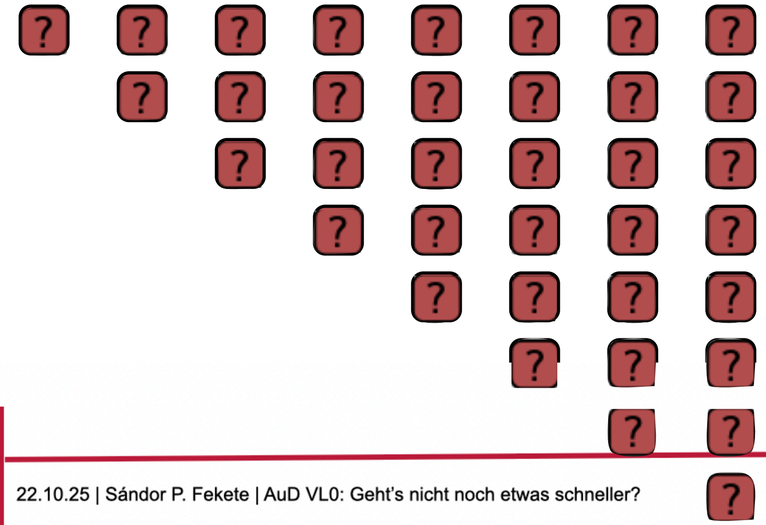
Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

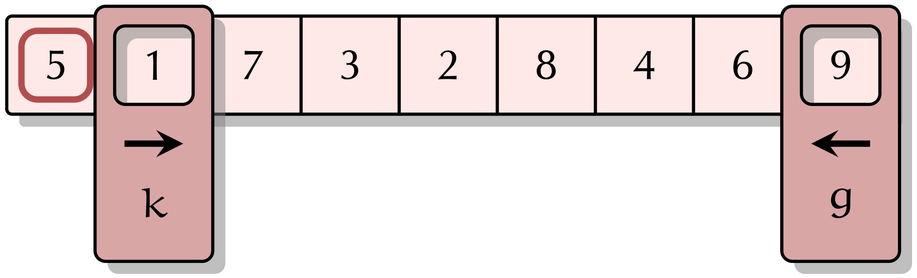


36 Vergleiche



Quicksort:

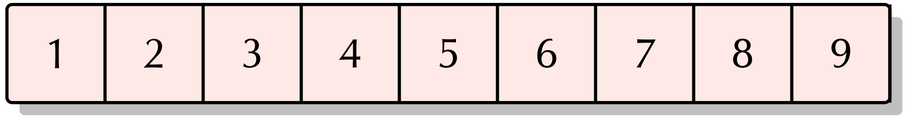
- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts



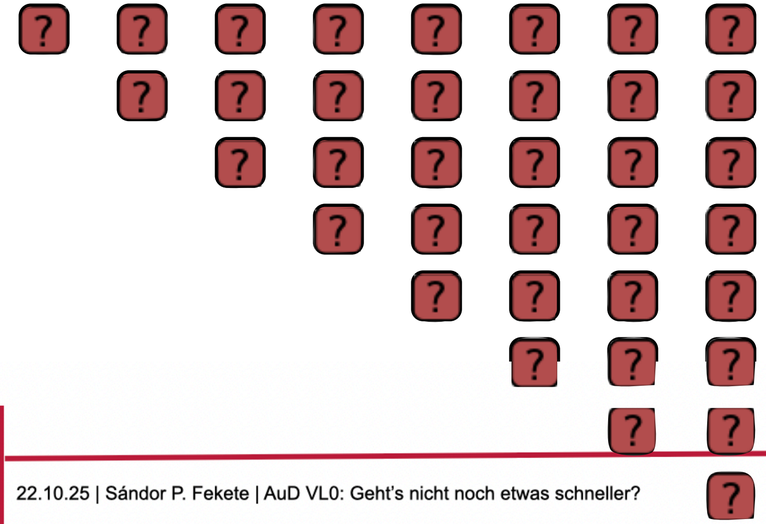
Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

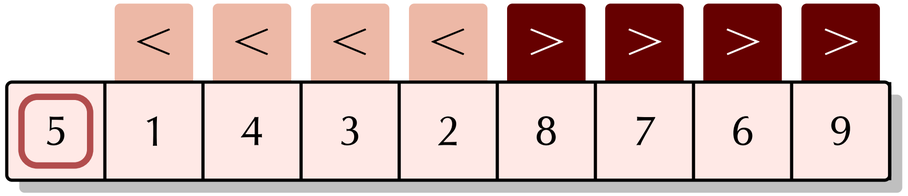


36 Vergleiche



Quicksort:

- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts

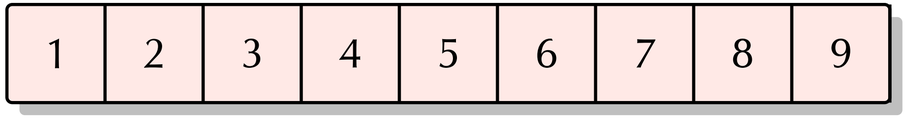




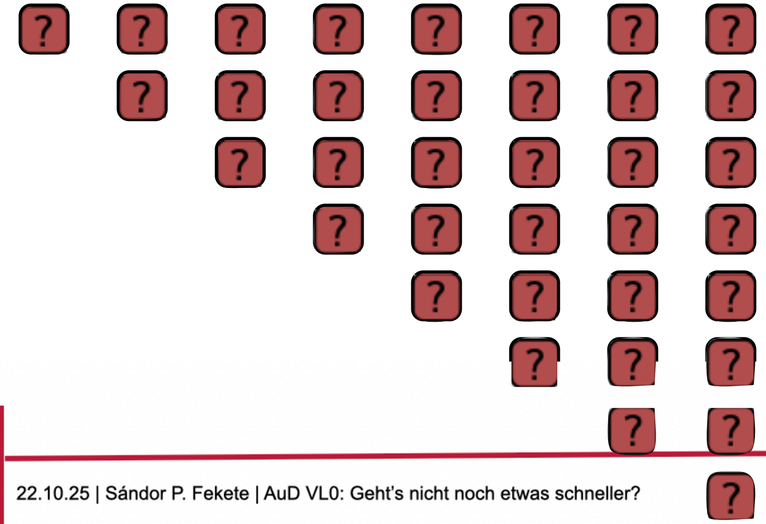
Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

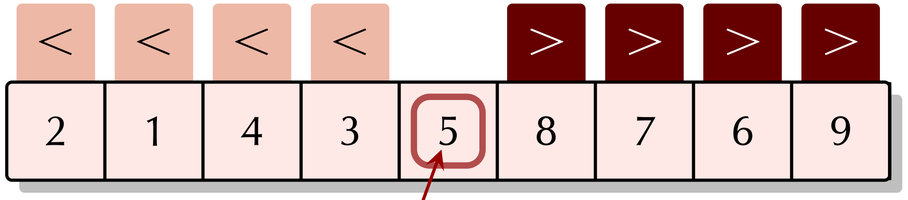


36 Vergleiche



Quicksort:

- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts



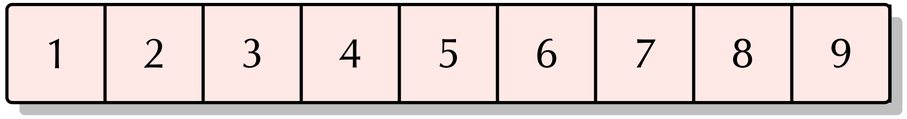
reached final place!



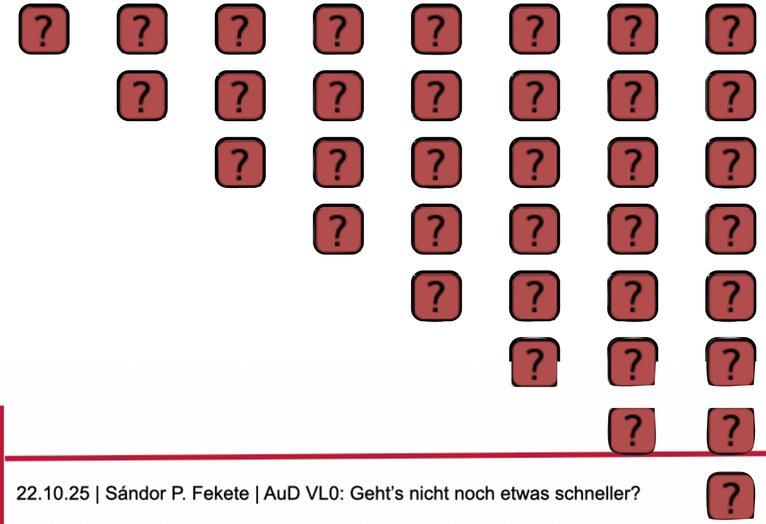
Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

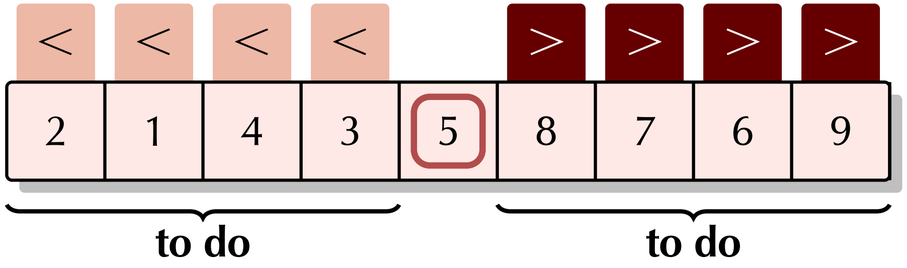


36 Vergleiche



Quicksort:

- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts

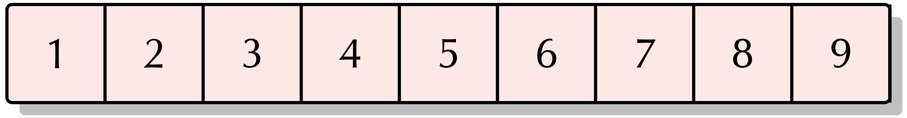




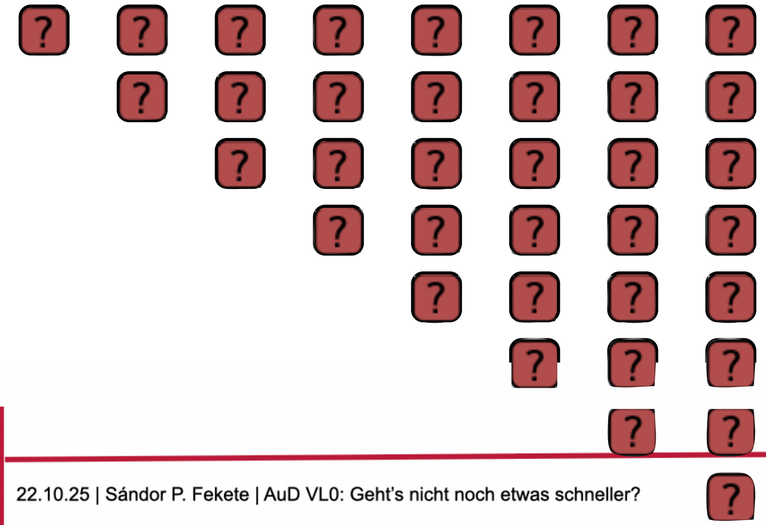
Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

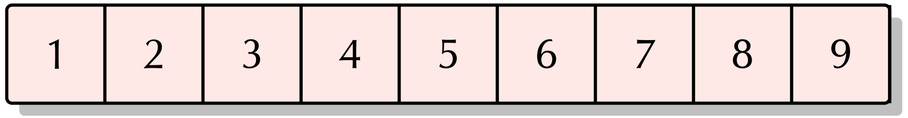


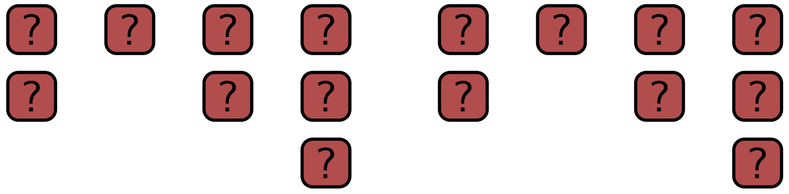
36 Vergleiche



Quicksort:

- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts

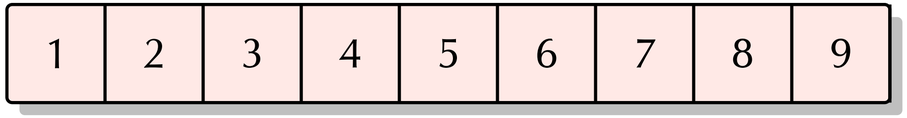




Zwei Sortieralgorithmen

Selection Sort:

- Finde das kleinste verbleibende Objekt
- Bringe es an die richtige Stelle

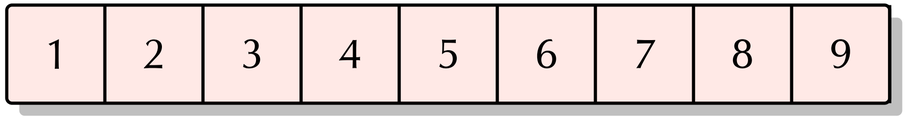


36 Vergleiche

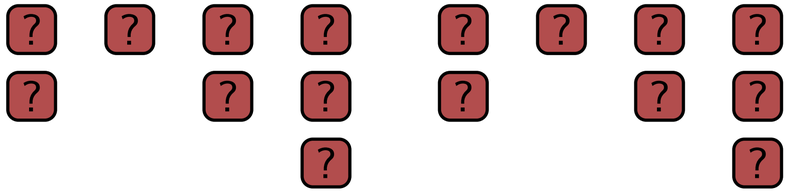


Quicksort:

- Bringe ein Objekt an die richtige Stelle
- Bringe die anderen nach links und rechts



16 Vergleiche



„Geht‘s nicht noch etwas schneller?“

3 P versus NP

Noch ein Puzzle

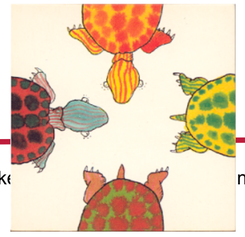
Gegeben: Eine Menge von *n* Puzzleteilen

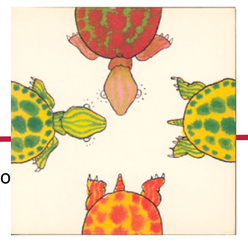
Gesucht: Eine „richtige“ Anordnung

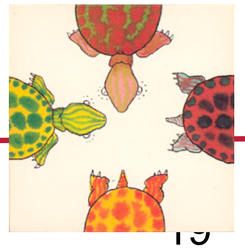


















Noch ein Puzzle

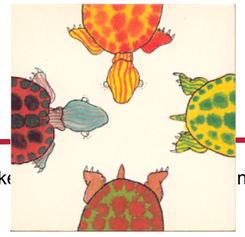
Gegeben: Eine Menge von *n* Puzzleteilen

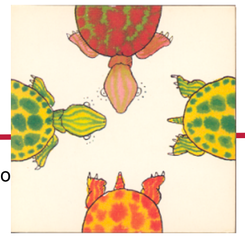
Gesucht: Eine „richtige“ Anordnung

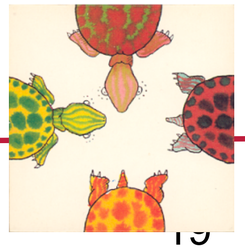


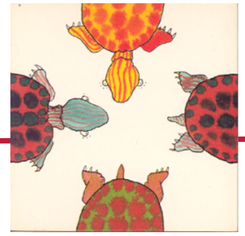


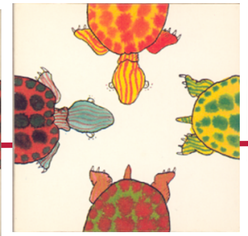














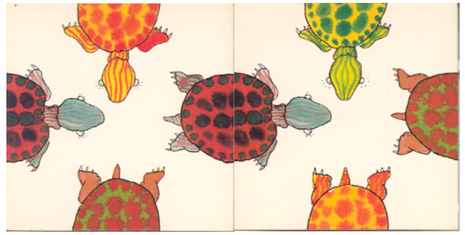
Noch ein Puzzle

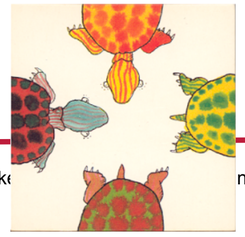
Gegeben: Eine Menge von *n* Puzzleteilen

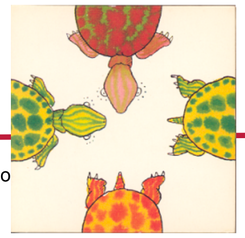
Gesucht: Eine „richtige“ Anordnung

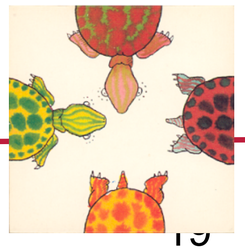


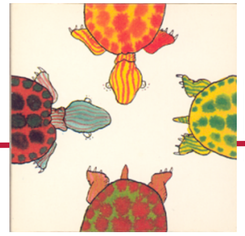














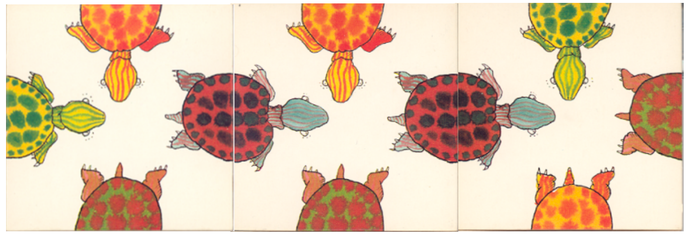
Noch ein Puzzle

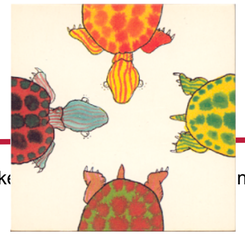
Gegeben: Eine Menge von *n* Puzzleteilen

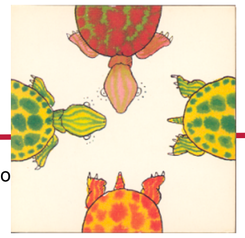
Gesucht: Eine „richtige“ Anordnung

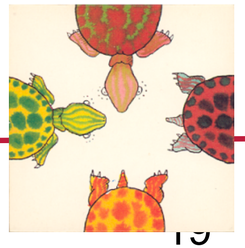


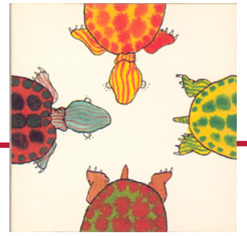








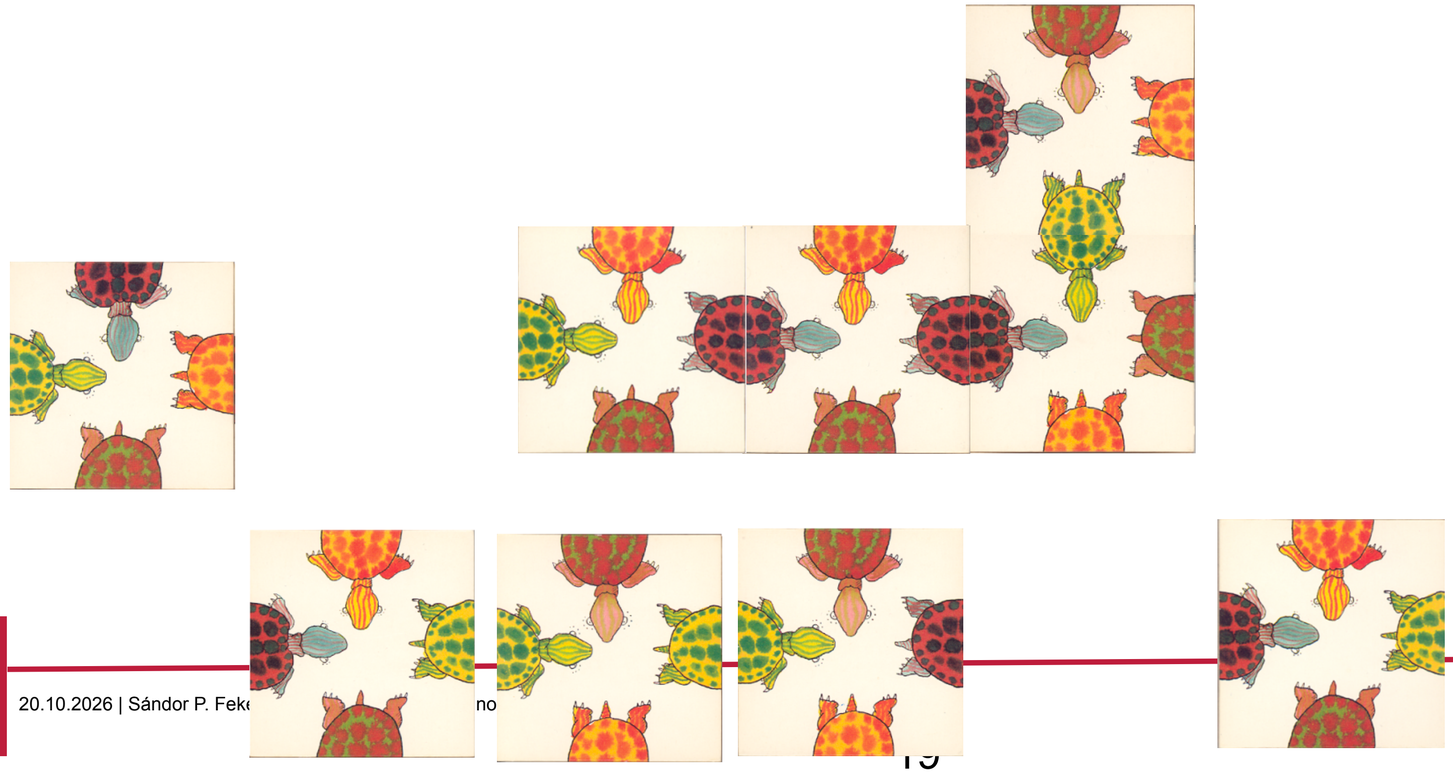




Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung

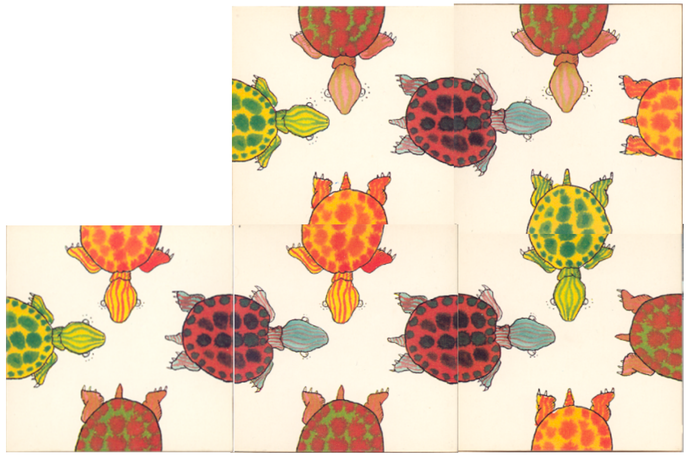


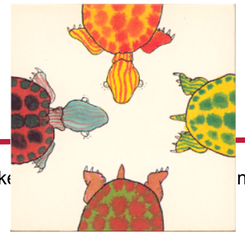
Noch ein Puzzle

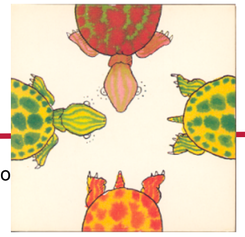
Gegeben: Eine Menge von *n* Puzzleteilen

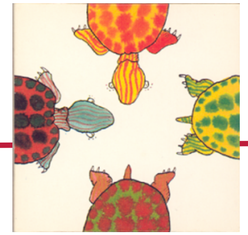
Gesucht: Eine „richtige“ Anordnung









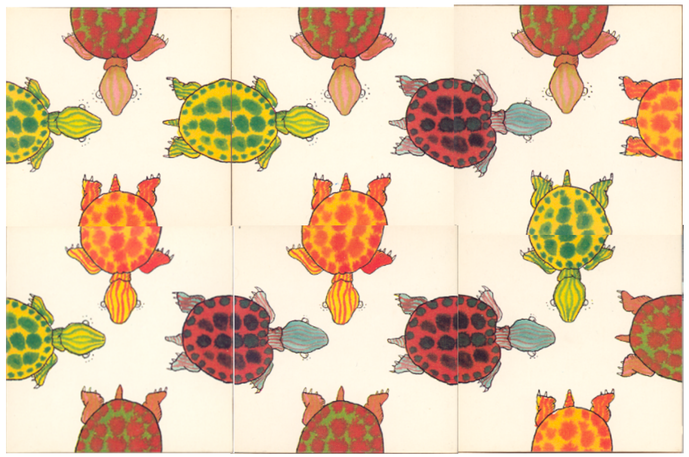


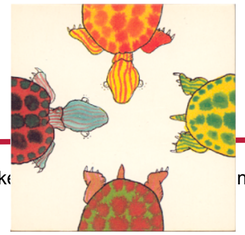
Noch ein Puzzle

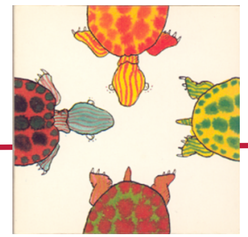
Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung





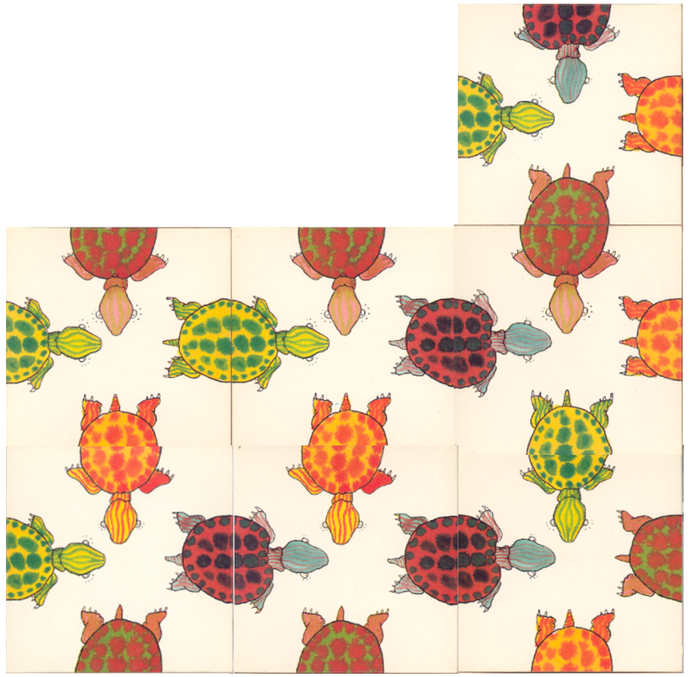


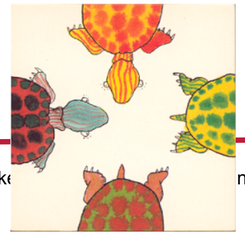


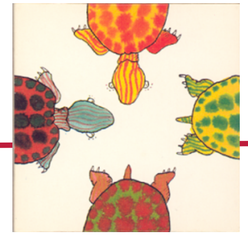
Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung



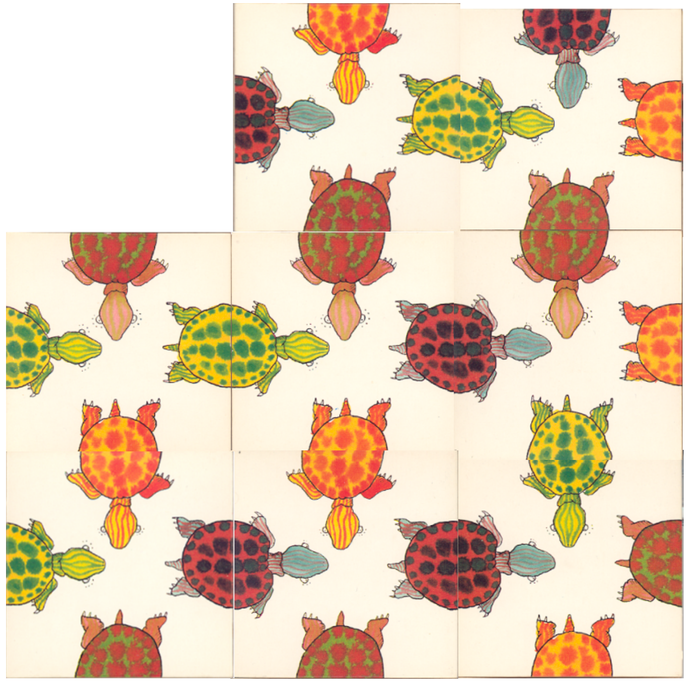


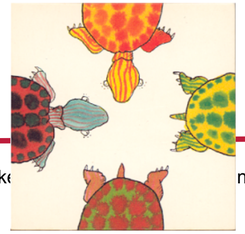


Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine „richtige“ Anordnung

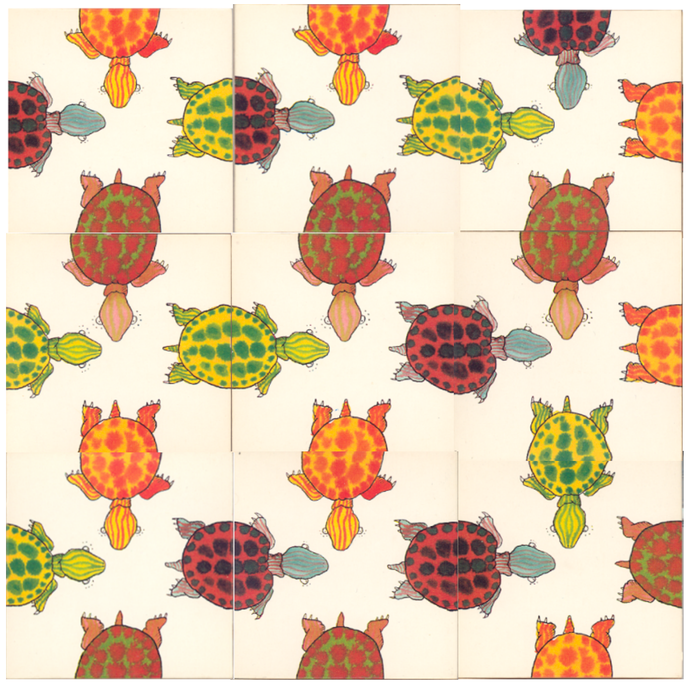




Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

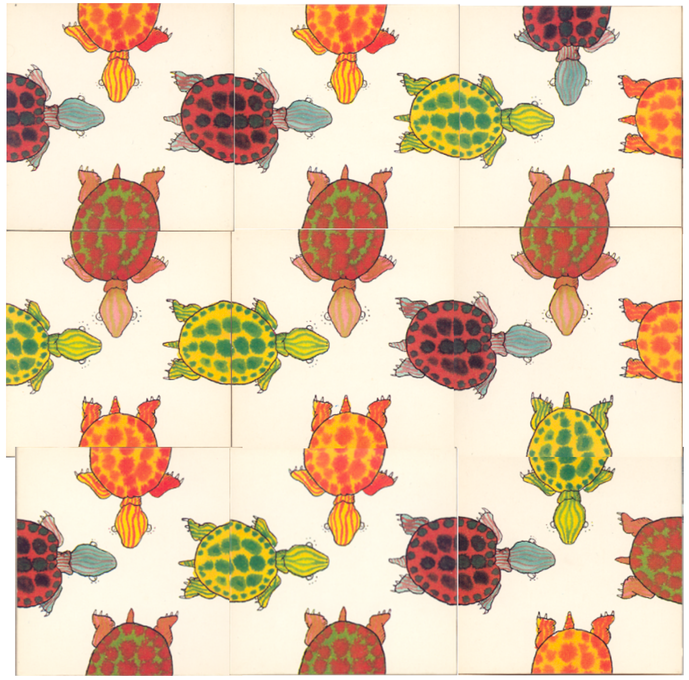
Gesucht: Eine „richtige“ Anordnung



Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

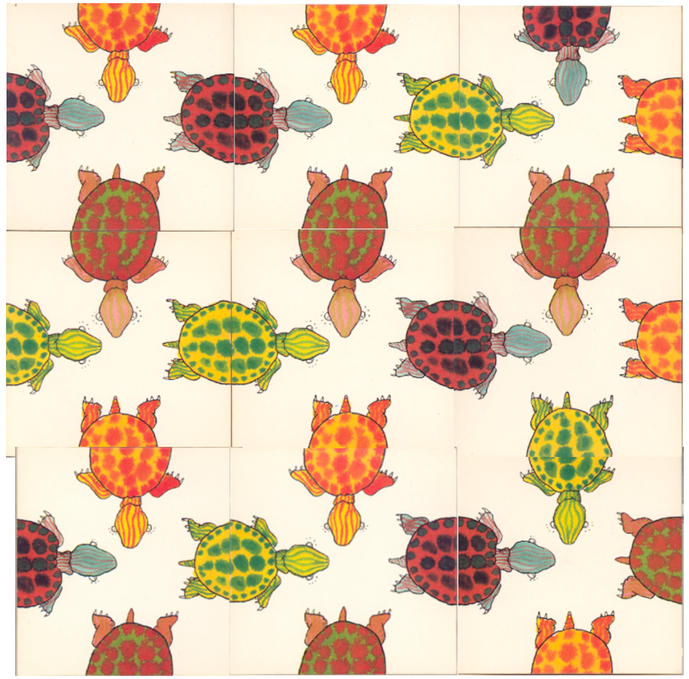
Gesucht: Eine „richtige“ Anordnung



Noch ein Puzzle

Gegeben: Eine Menge von *n* Puzzleteilen

Gesucht: Eine systematische Lösungsmethode

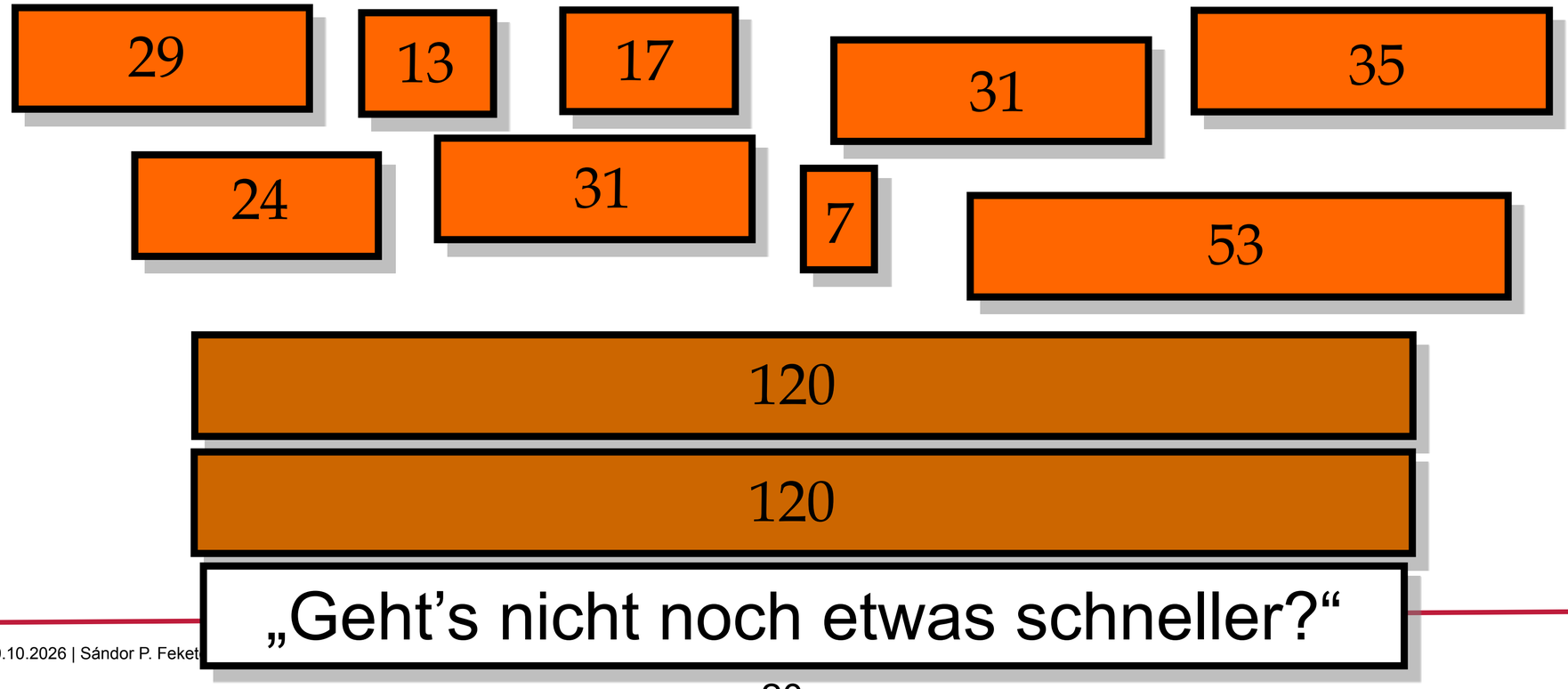


Problem: „Sackgassen“

Für n=9:
9! * 4^8=
23.781.703.680
mögliche Anordnungen

Kofferpacken für die Reise

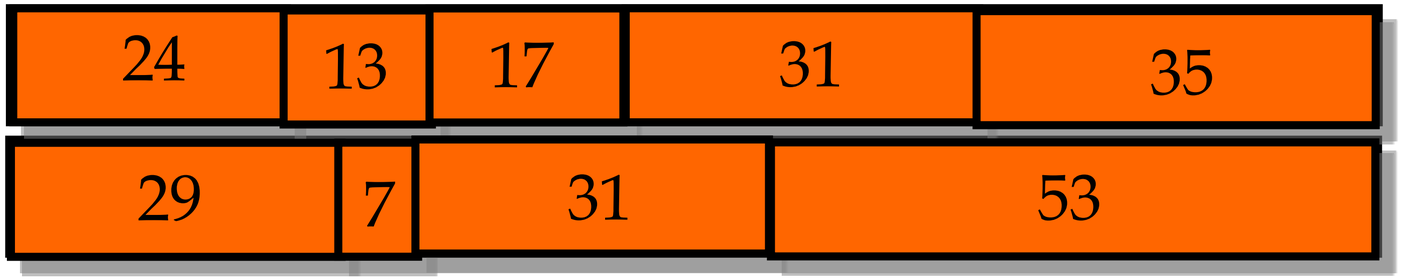
Gegeben: Zahlen {7, 13, 17, 24, 29, 31, 31, 35, 53}
Gesucht: Eine Aufteilung in zwei Teilmengen mit jeweils Summe 120



Kofferpacken für die Reise

Gegeben: Zahlen {7, 13, 17, 24, 29, 31, 31, 35, 53}
Gesucht: Eine Aufteilung in zwei Teilmengen mit jeweils Summe 120

Existenz: Eine existierende Lösung kann man einfach **N**ach**P**rüfen
Konstruktion: Wie **P**findet man eine Lösung in kurzer Zeit?
Nichtexistenz: Wie beweist man, dass es gar keine Lösung gibt?



Eigenschaften des Problems

(1) Wenn es eine Lösung gibt, so lässt sie sich ziemlich schnell *NachPrüfen*.

Die Klasse NP

(2) Wenn es *keine* Lösung gibt, lässt sich das anscheinend nur schlecht *beweisen*.

Probleme, für die man pfiffig „schnell“ eine Lösung *Pfinden* kann:

Die Klasse P

(Echt pfundig - gepfundes Pfressen für Algorithmiker!)

$\Theta(n^2)$ ✓

$\Theta(2^n)$ ✗

Wo ist mein Schlüssel??

Ein wiedergefundener Schlüssel lässt sich relativ leicht identifizieren.



Wo ist mein Schlüssel??

Ein wiedergefundener Schlüssel lässt sich relativ leicht identifizieren.

Aber wie weist man nach, dass ein Schlüssel NICHT in einem Raum ist?

Wo ist mein Schlüssel??

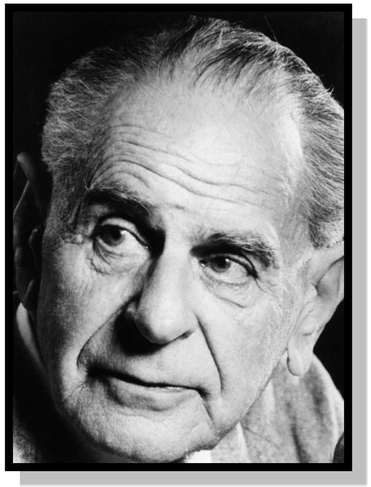
Alltagserfahrungen:

Verlieren geht schneller als Wiederpfinden.

Ja sagen geht schneller als Nein begründen.

Fragen stellen geht schneller als Fragen beantworten.

Wo ist mein Schlüssel??



Karl Popper (1902-1994)

Universalaussagen lassen sich nicht verifizieren, sondern nur falsifizieren.

Wo ist mein Schlüssel??

Ein komplexitätstheoretischer Glaubenssatz

$$P \neq NP$$

Also: NachPrüfen allein ist noch lange nicht Pfindig!

(Oder Pfiffig)

Schlechte Nachrichten

Satz von Cook (1973):

Es gibt schwerste Probleme in NP, sogenannte „NP-vollständige Probleme“

Schlechte Nachrichten

Noch schlechtere Nachrichten

Ein polynomieller Algorithmus für

Rundreiseproblem

Partitionsproblem

die meisten interessanten Optimierungsprobleme

bedeutet auch schon

P = NP

Schlechte Nachrichten

Noch schlechtere Nachrichten

Ein polynomieller Algorithmus für

The danger if $P = NP$

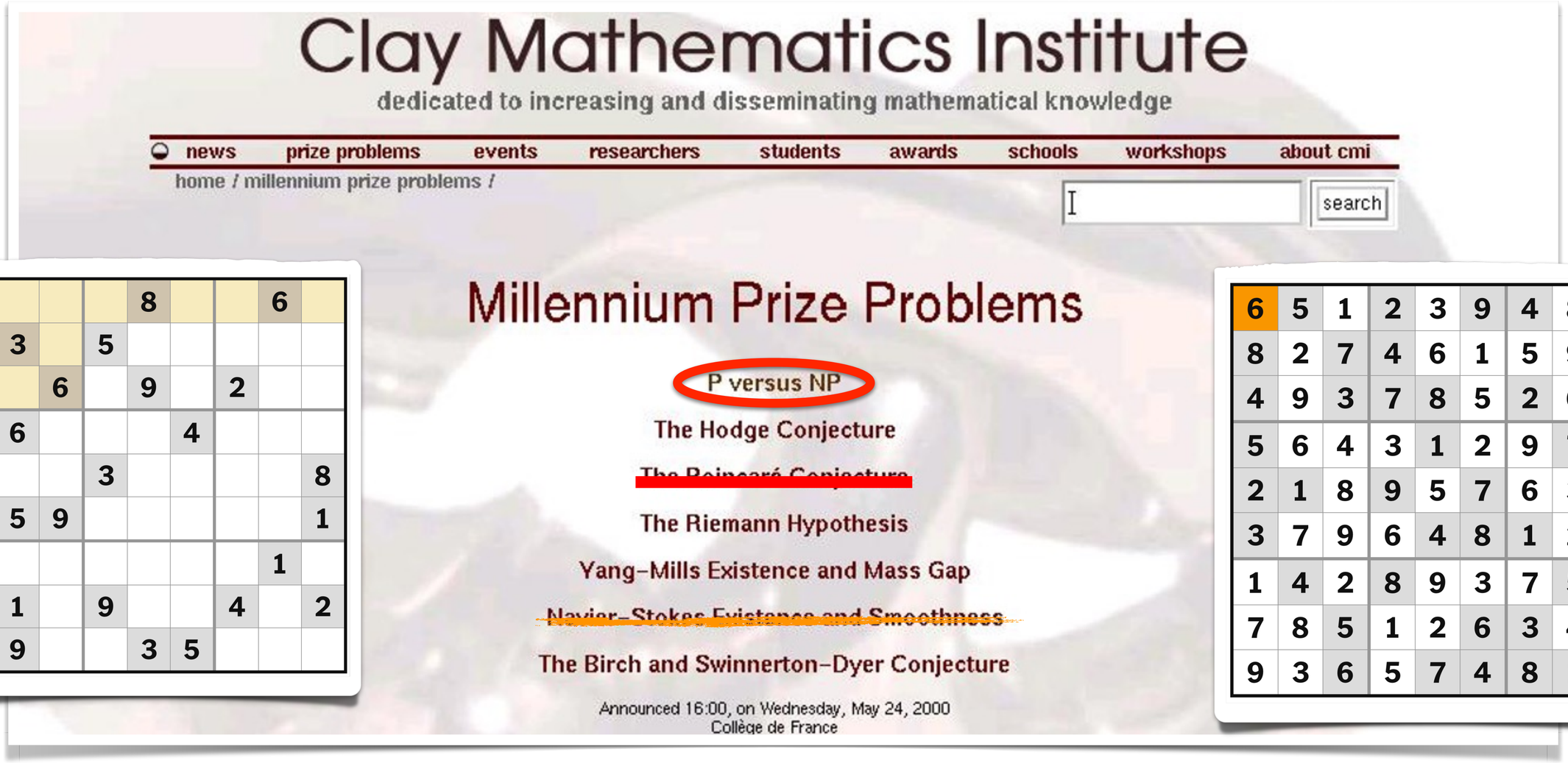
- **Breakdown of cryptography:** Modern encryption, which relies on the difficulty of problems like factoring large numbers (an NP problem), would become obsolete. This would compromise the security of online communication, financial transactions, and data storage.

Optimierungsprobleme

bedeutet auch schon

P = NP

Ein Millenium-Problem



				8			6	
	3		5					
		6		9		2		
	6				4			
			3					8
3	5	9						1
							1	
	1		9			4		2
	9			3	5			

6	5	1	2	3	9	4	8	7
8	2	7	4	6	1	5	9	3
4	9	3	7	8	5	2	6	1
5	6	4	3	1	2	9	7	8
2	1	8	9	5	7	6	3	4
3	7	9	6	4	8	1	2	5
1	4	2	8	9	3	7	5	6
7	8	5	1	2	6	3	4	9
9	3	6	5	7	4	8	1	2

Geht’s nicht noch etwas schneller?



BC **Bill Cook <bico@uwaterloo.ca>** CG:SHOP 18. September 2025 um 14:23

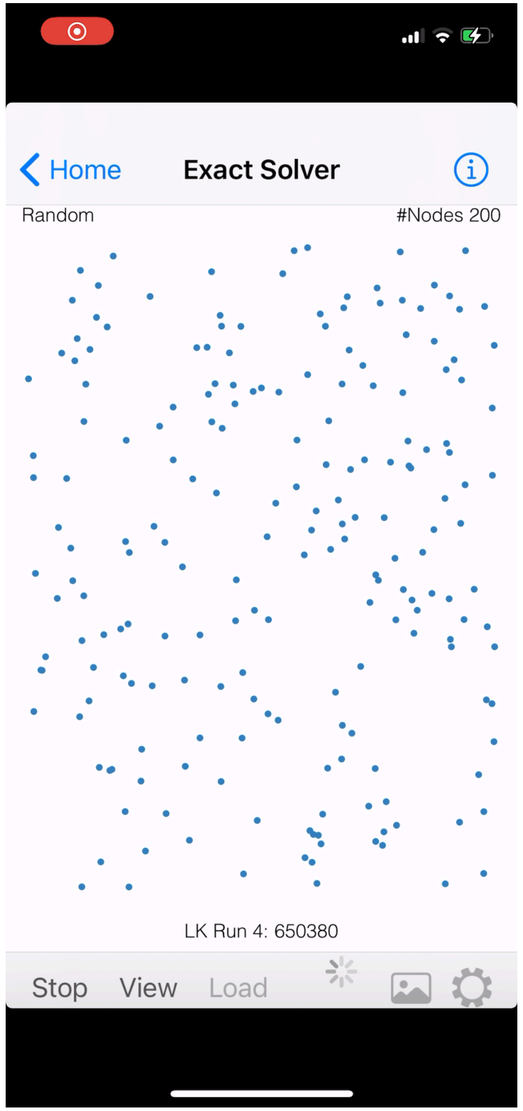
Aw: Next Level TSP

An: Fekete, Sandor <s.fekete@tu-braunschweig.de>,

Umgeleitet von: s.fekete@tu-braunschweig.de

Wheeeee! Thanks for the link! I’ll add this to a TSP talk
I’m preparing for Korea in October.

Bico



$200! \approx 10^{370}$

Geht’s nicht noch etwas schneller?



BC **Bill Cook <bico@uwaterloo.ca>** CG:SHOP 18. September 2025

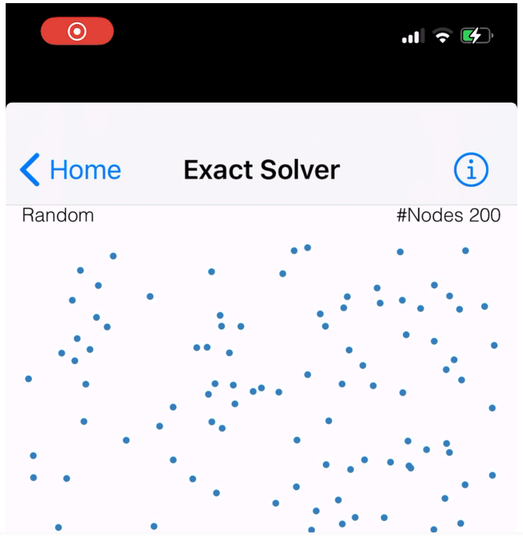
Aw: Next Level TSP

An: Fekete, Sandor <s.fekete@tu-braunschweig.de>,

Umgeleitet von: s.fekete@tu-braunschweig.de

Wheeeee! Thanks for the link! I’ll add this to a TSP talk
I’m preparing for Korea in October.

Bico

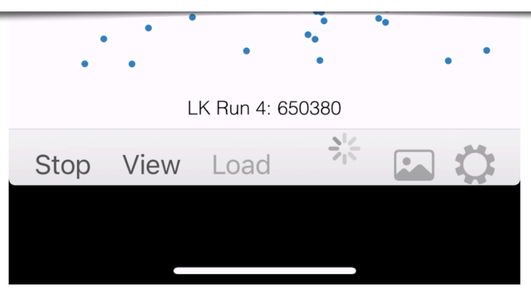




Bill Cook @wjcook · Feb 12

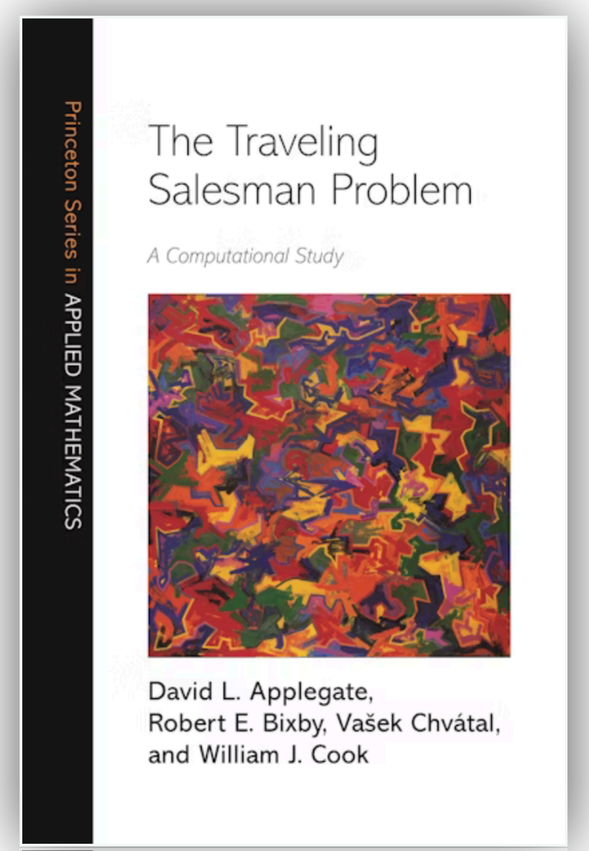
To clarify, computing an optimal TSP route on 22 points takes 0.005 seconds on my iMac, not the 1000 years reported in @washingtonpost An error factor of 6 trillion. Like reporting the US National Debt is $4.

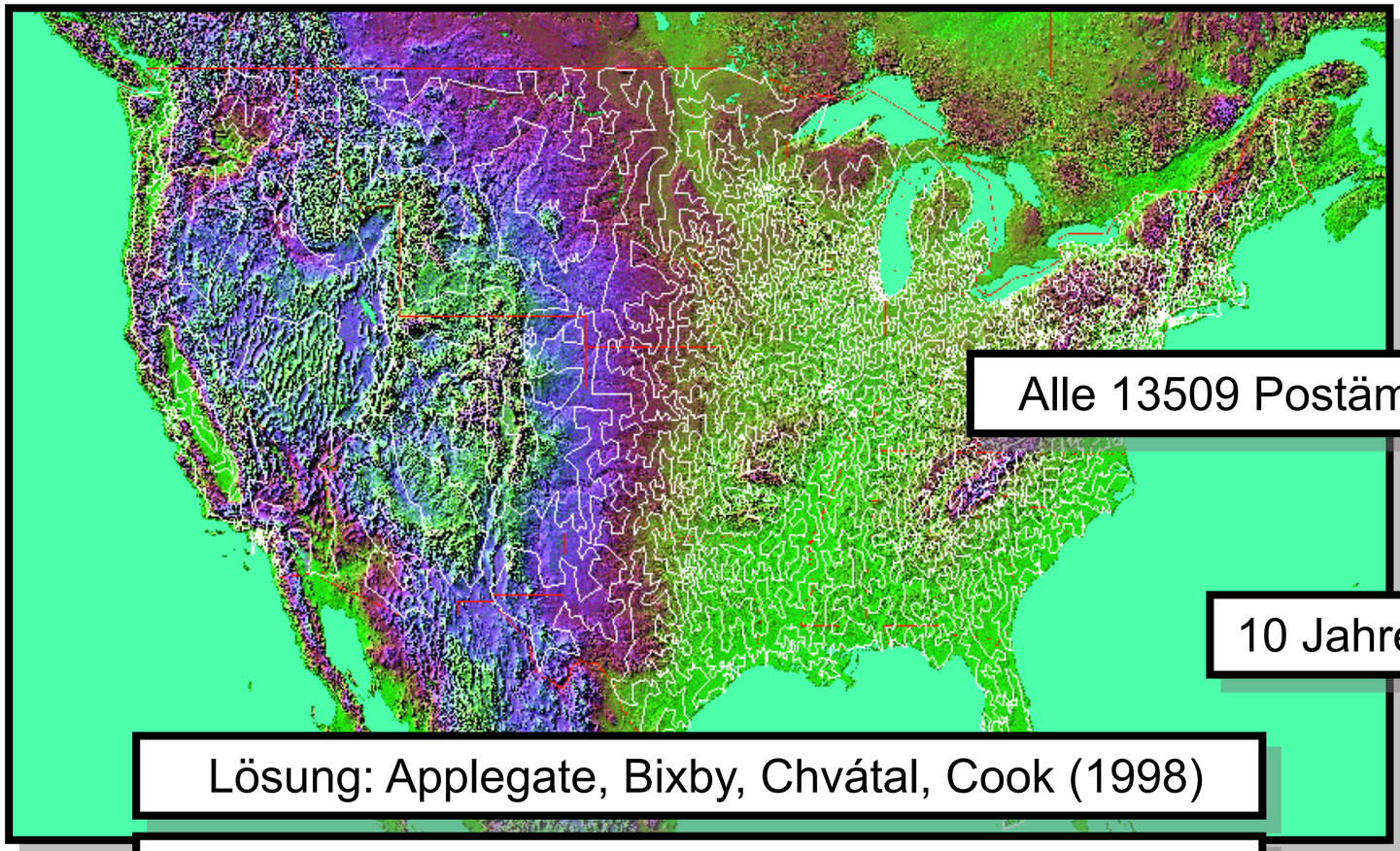
washingtonpost.com/news/innovatio…



$$200! \approx 10^{370}$$

Eine optimale Rundreise in der Neuen Welt





Alle 13509 Postämter der USA

10 Jahre (!) CPU-Zeit

Lösung: Applegate, Bixby, Chvátal, Cook (1998)

Beale-Orchard-Hays-Preis 2000

„Geht‘s nicht noch etwas schneller?“

Ausblicke

Algorithmen und Datenstrukturen

Semester Wintersemester 2025/2026

Studiengänge Informations-Systemtechnik Bachelor, Informatik Bachelor, Wirtschaftsinformatik Bachelor

Algorithmen und Datenstrukturen 2

Semester Sommersemester 2025

Modulnummer INF-ALG-23

Veranstaltungsnummer INF-ALG-042, INF-ALG-043, INF-ALG-044

Studiengänge Wirtschaftsinformatik Bachelor, Informations-Systemtechnik Bachelor, Informatik Bachelor



Algorithmik-Praktikum: *Solving NP-hard Problems in Practice*

Semester Wintersemester 2025/2026

Modulnummer INF-ALG-10

Studiengänge Informatik Bachelor, Wirtschaftsinformatik Bachelor

IBR Gruppe ALG (Prof. Fekete)

Art Praktikum

Mathematische Methoden der Algorithmik

Semester Wintersemester 2025/2026

Studiengänge Wirtschaftsinformatik Master, Informations-Systemtechnik Master, Informatik Master

Vielen Dank!

Algorithmen und Datenstrukturen

Semester Wintersemester 2025/2026

Studiengänge Informations-Systemtechnik Bachelor, Informatik Bachelor, Wirtschaftsinformatik Bachelor

Algorithmen und Datenstrukturen 2

Semester Sommersemester 2025

Modulnummer INF-ALG-23

Veranstaltungsnummer INF-ALG-042, INF-ALG-043, INF-ALG-044

Studiengänge Wirtschaftsinformatik Bachelor, Informations-Systemtechnik Bachelor, Informatik Bachelor



Algorithmik-Praktikum: *Solving NP-hard Problems in Practice*

Semester Wintersemester 2025/2026

Modulnummer INF-ALG-10

Studiengänge Informatik Bachelor, Wirtschaftsinformatik Bachelor

IBR Gruppe ALG (Prof. Fekete)

Art Praktikum

Mathematische Methoden der Algorithmik

Semester Wintersemester 2025/2026

Studiengänge Wirtschaftsinformatik Master, Informations-Systemtechnik Master, Informatik Master