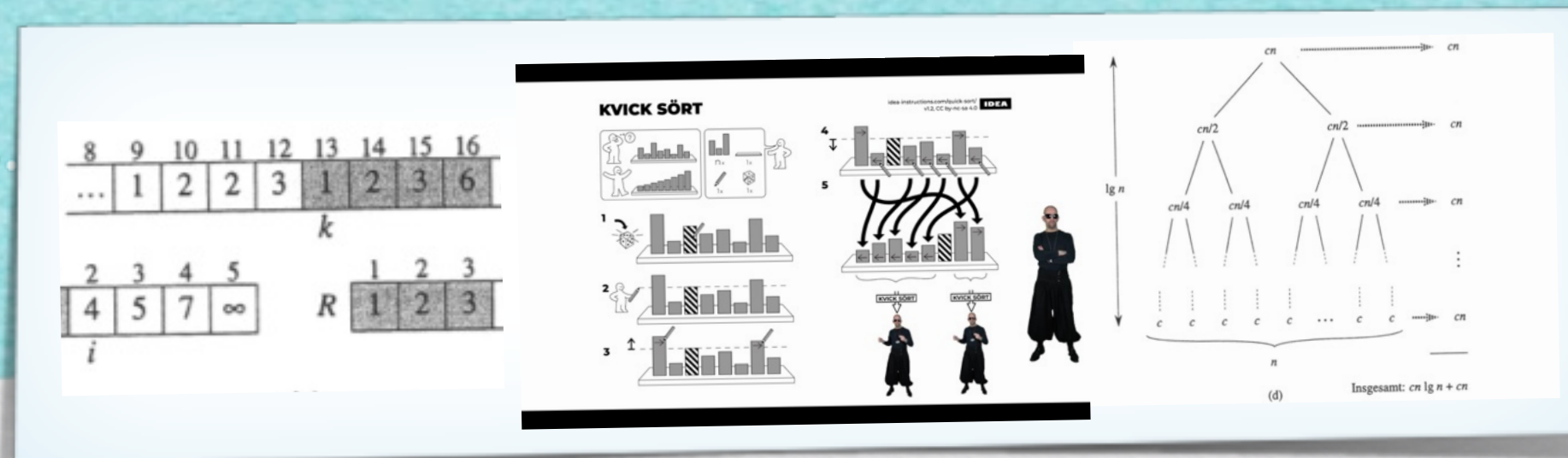




Kapitel 5.7: *Sortieren in Linearzeit* *Algorithmen und Datenstrukturen* *WS 2025/26*

Prof. Dr. Sándor Fekete

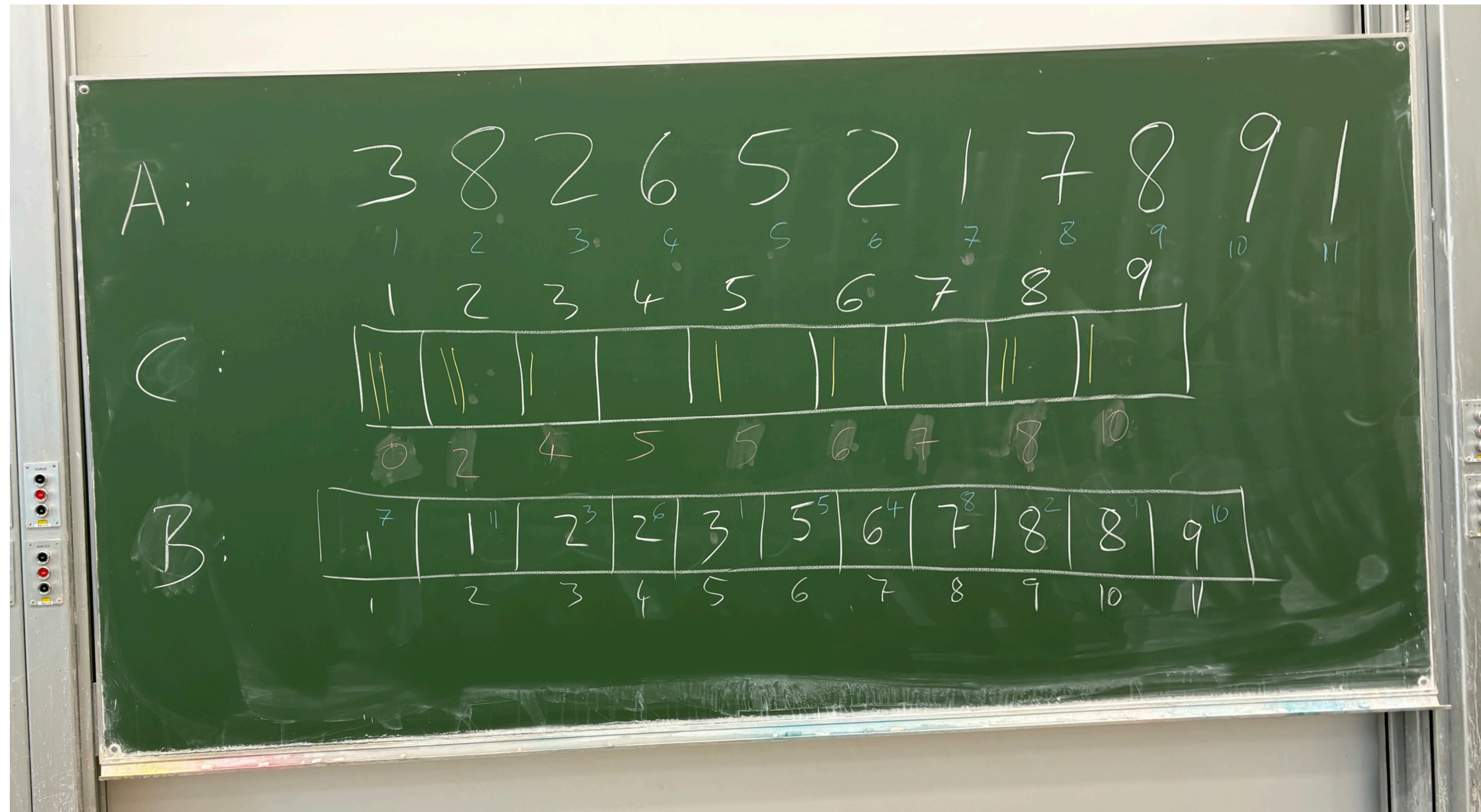
Laufzeitschranke fürs Sortieren

Satz 5.6. *Für n Objekte $x_1, \dots, x_n$*

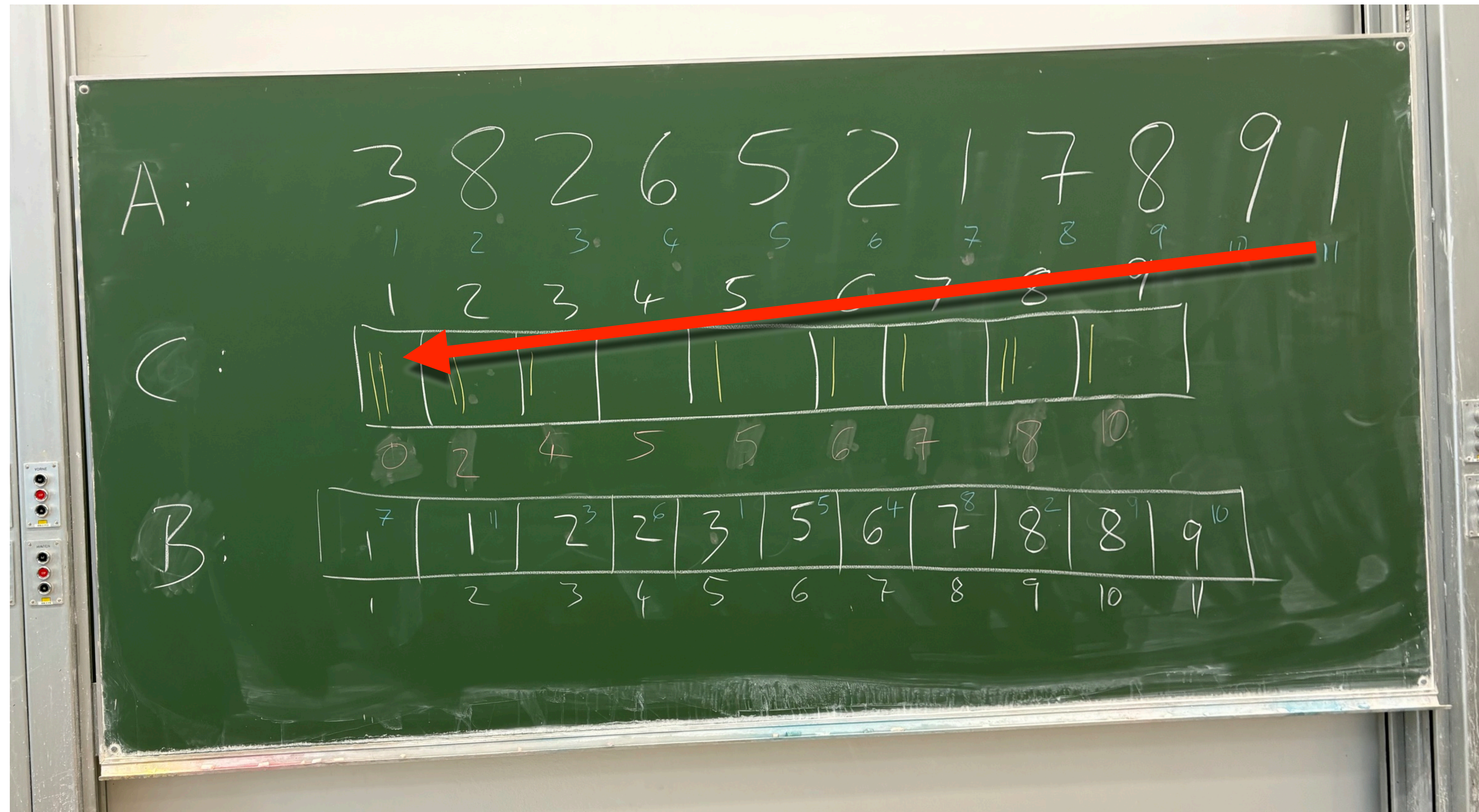
benötigt man zum Sortieren mindestens $\Omega(n \log(n))$,

wenn man die Objekte nur paarweise vergleichen kann.

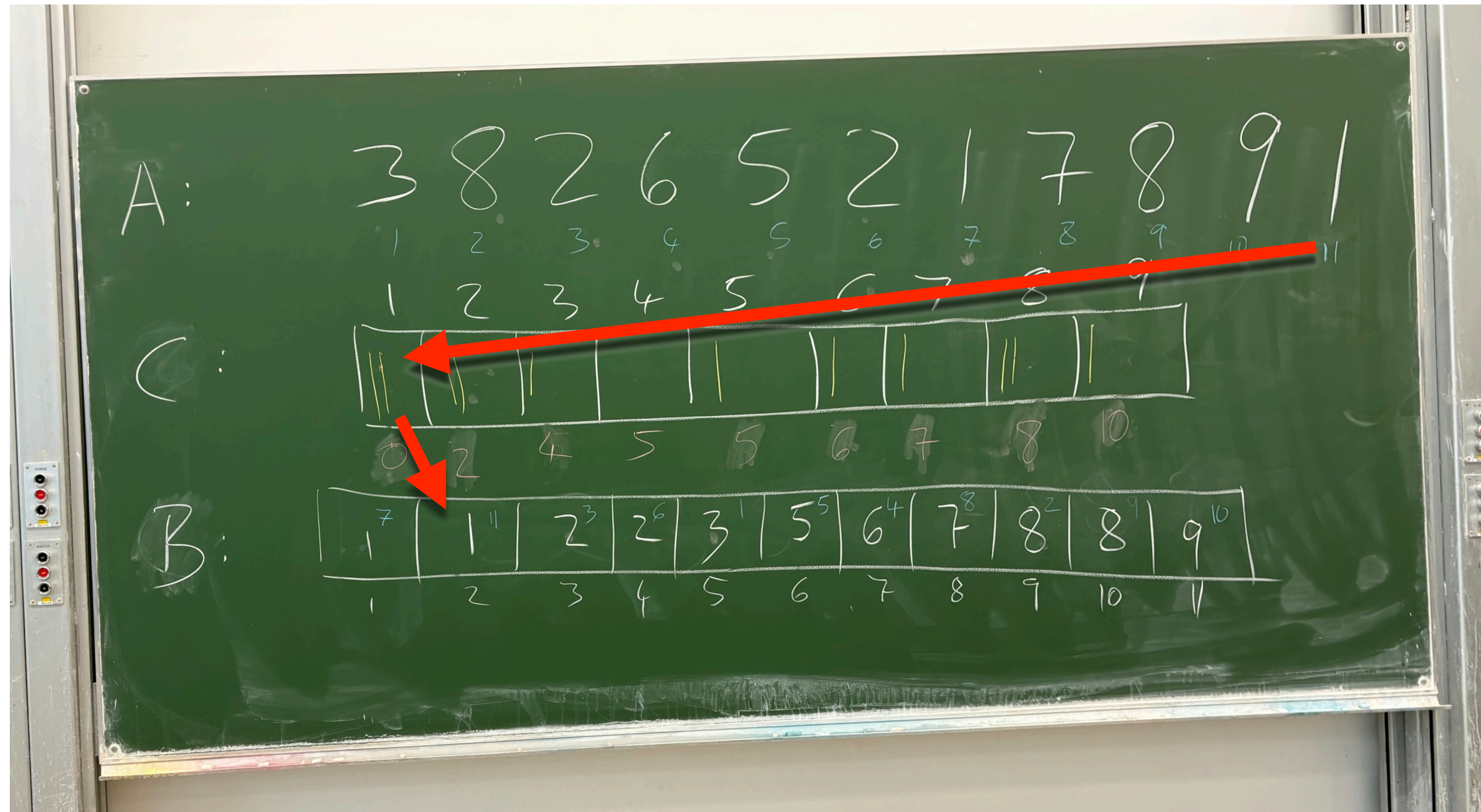
Countingsort



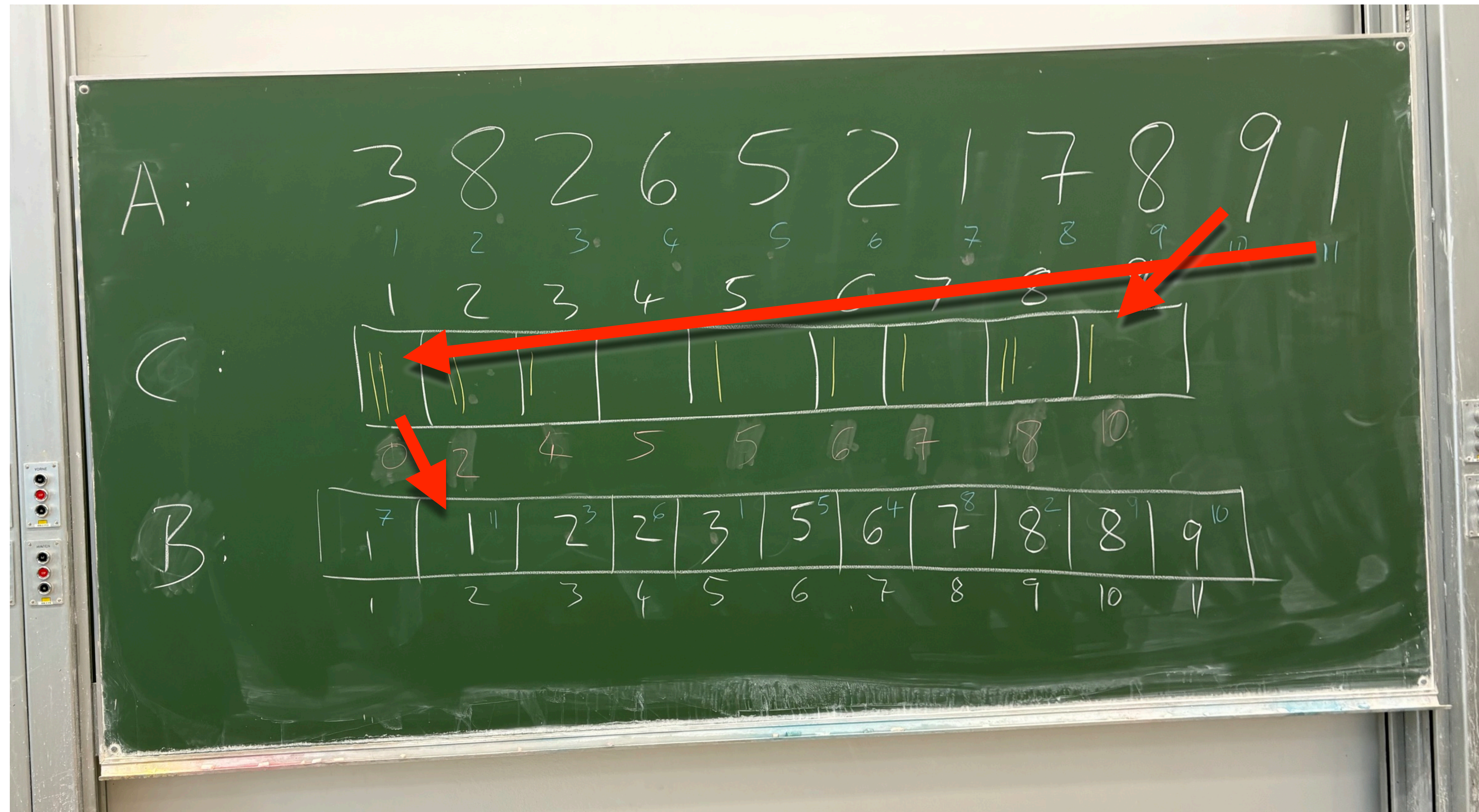
Countingsort



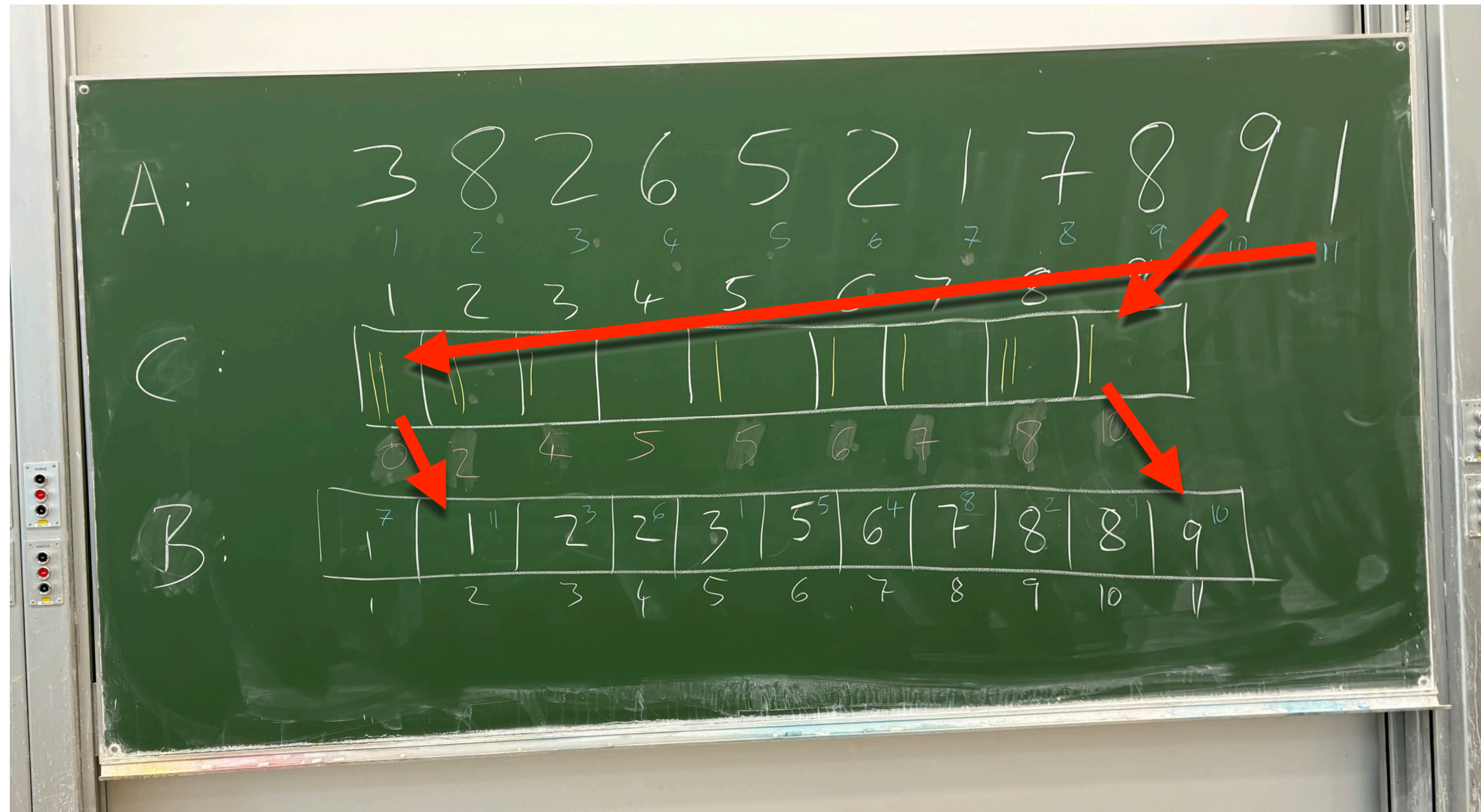
Countingsort



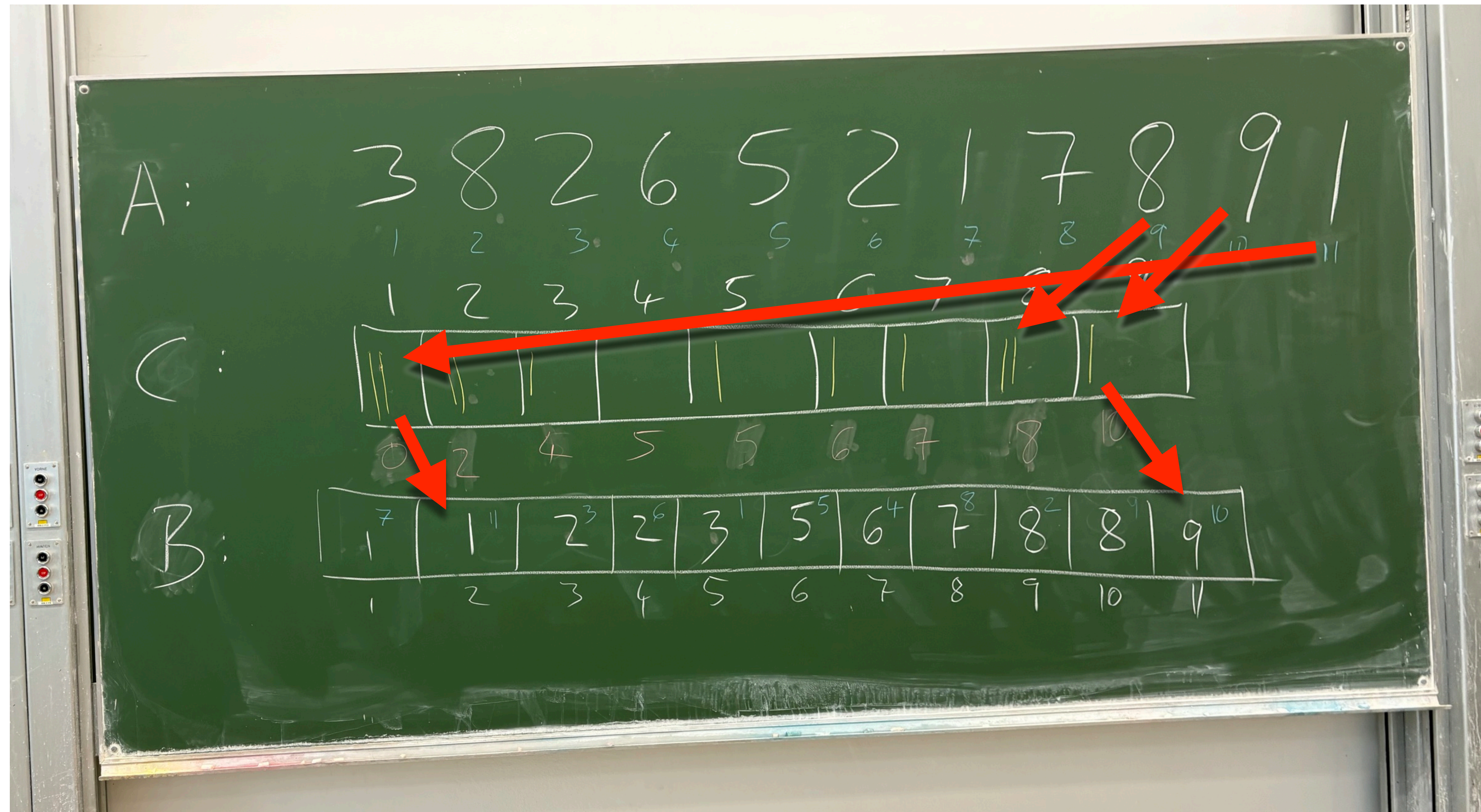
Countingsort



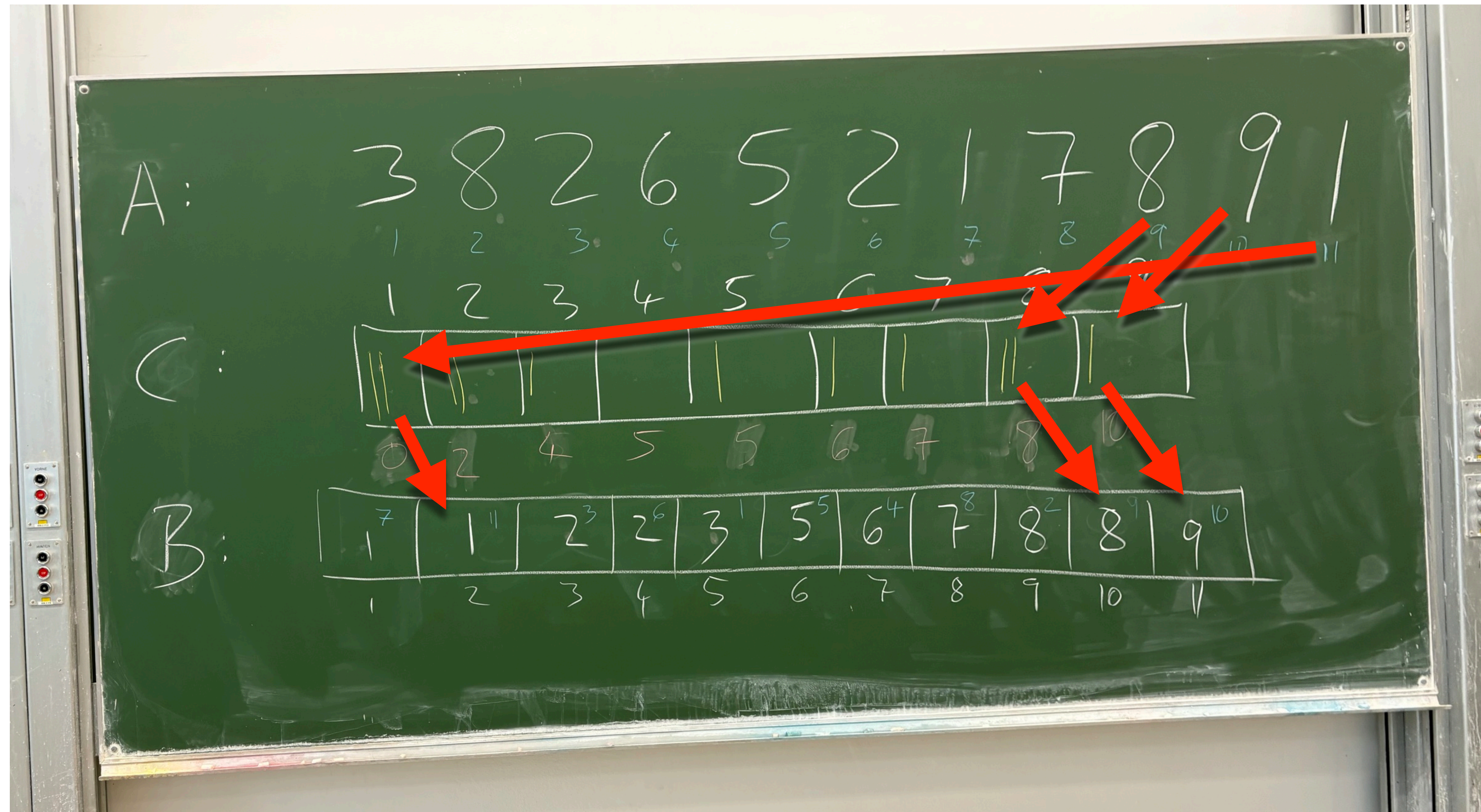
Countingsort



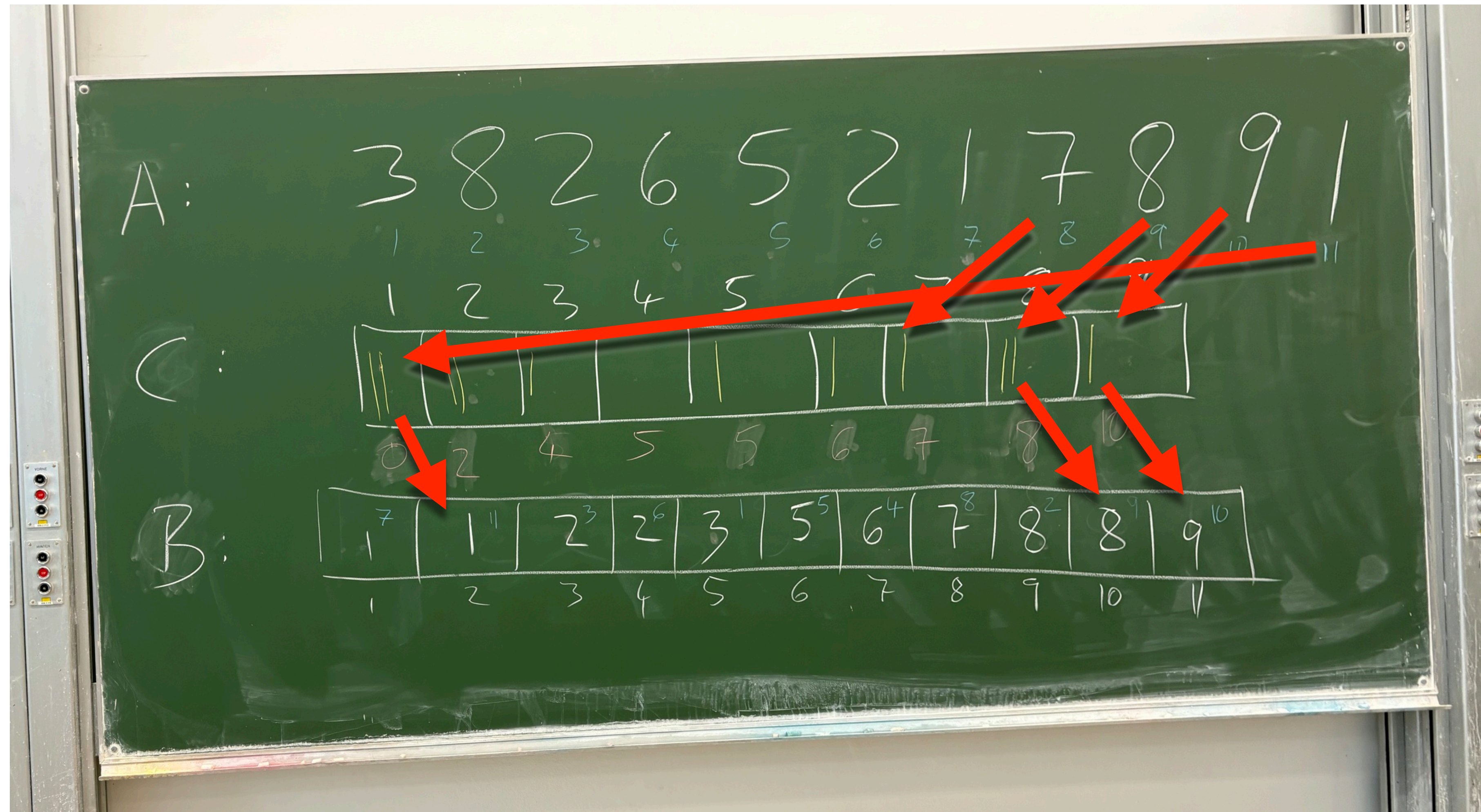
Countingsort



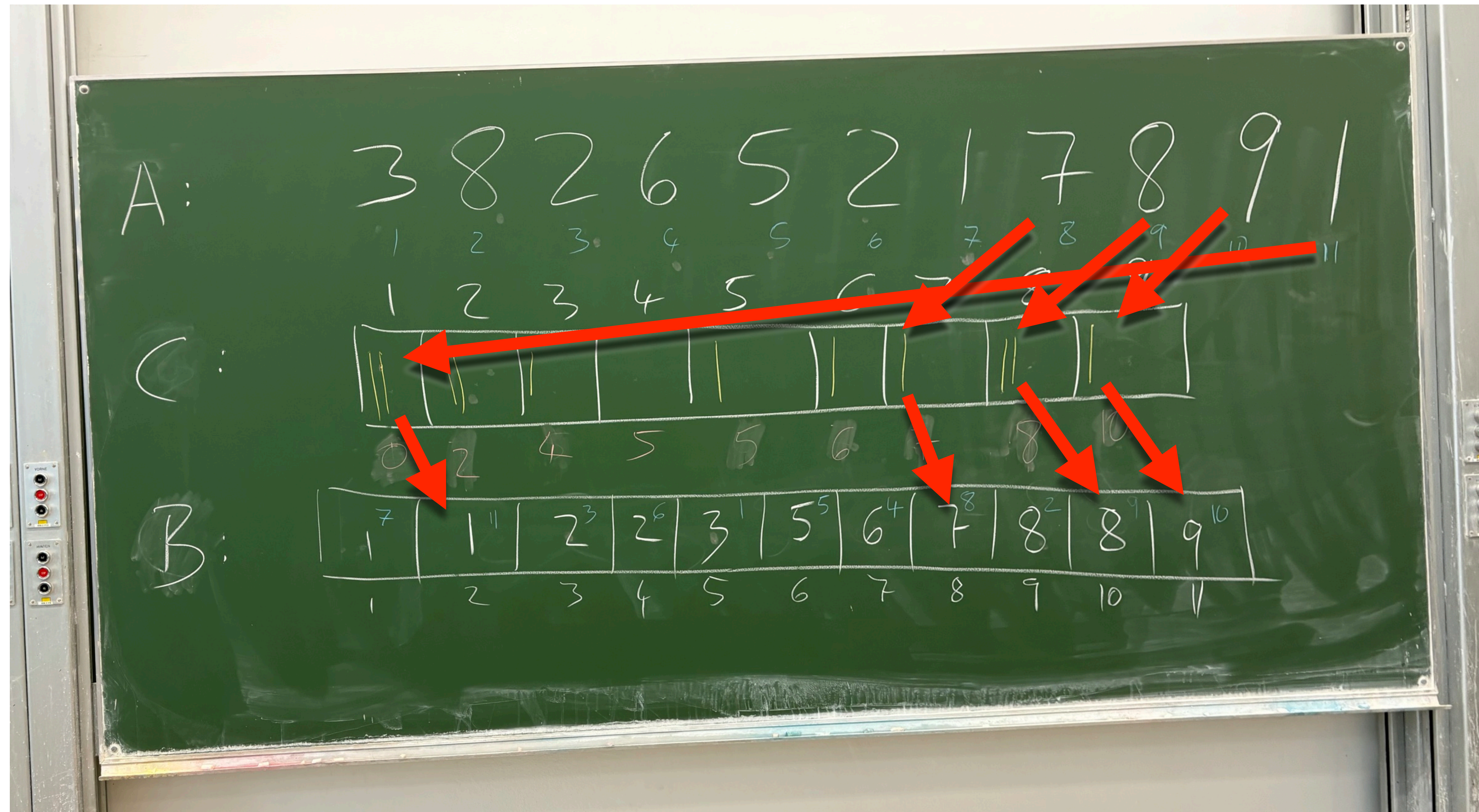
Countingsort



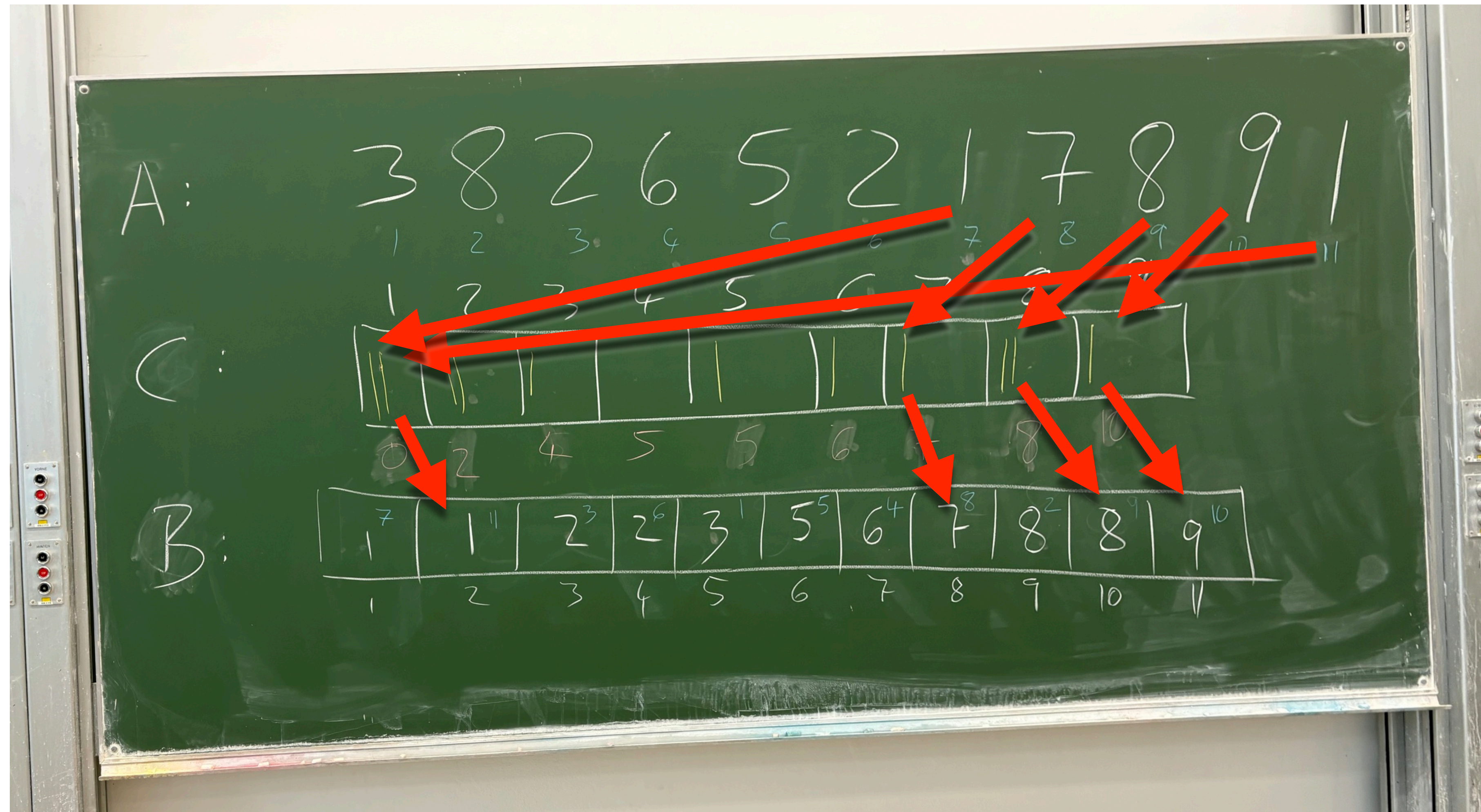
Countingsort



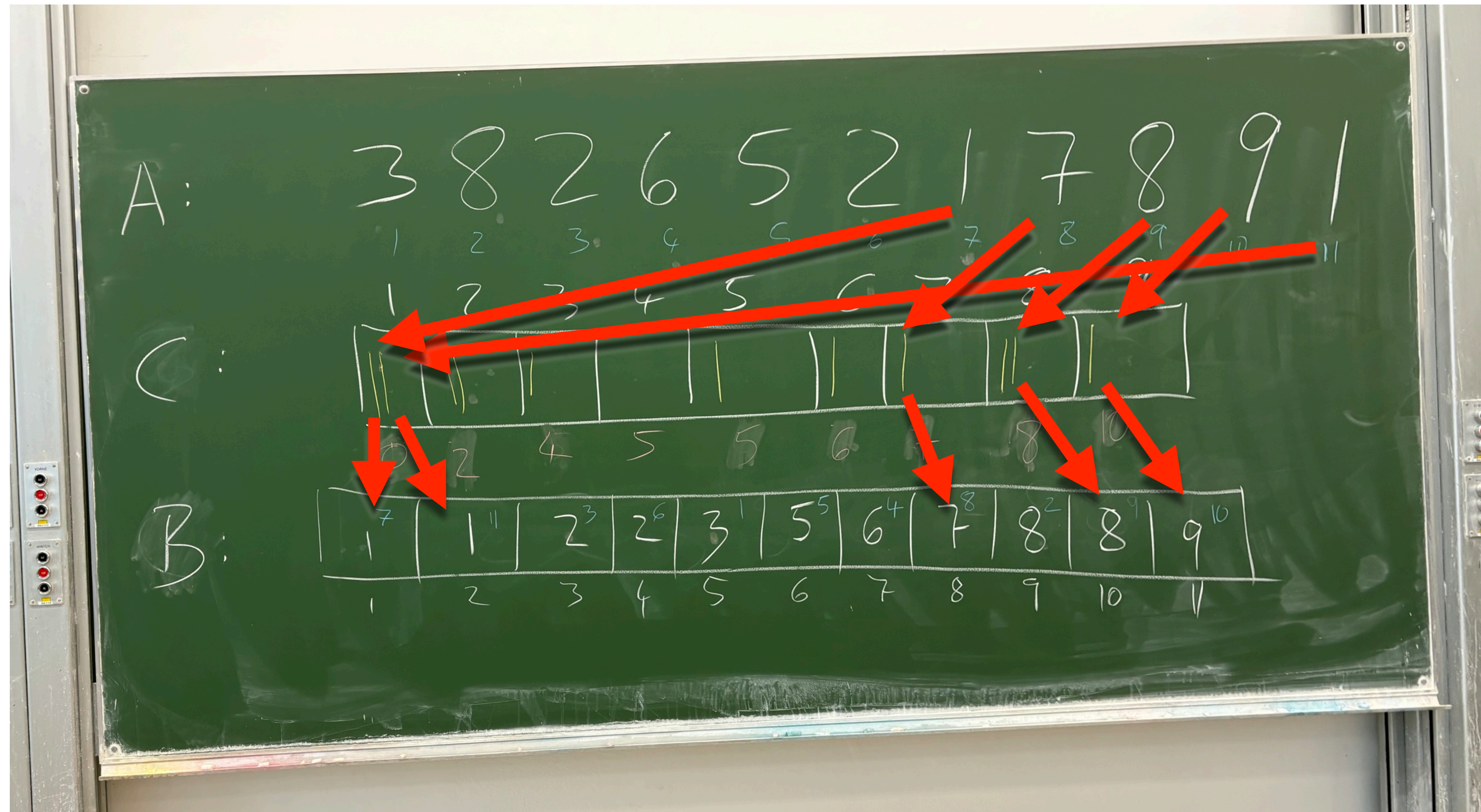
Countingsort



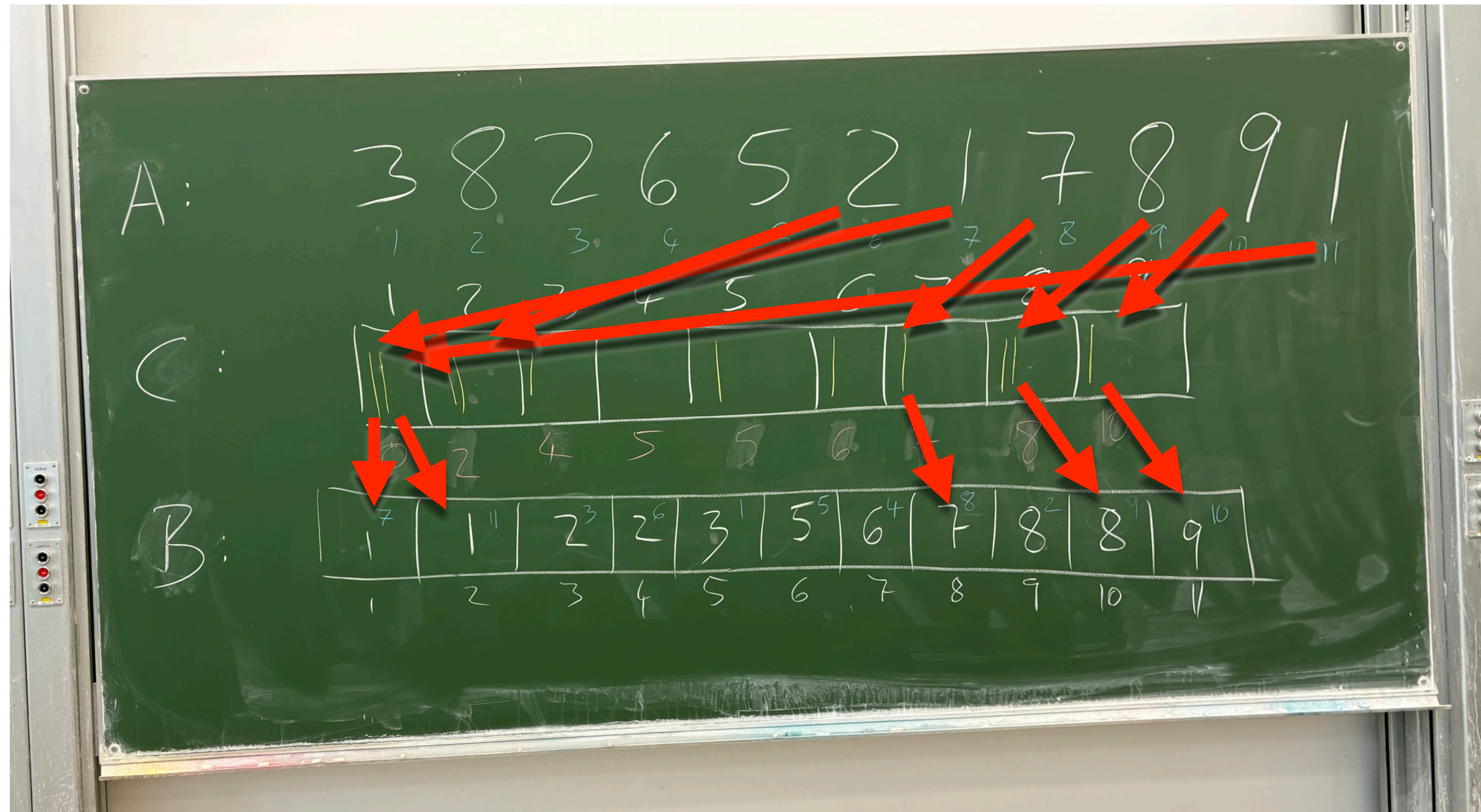
Countingsort



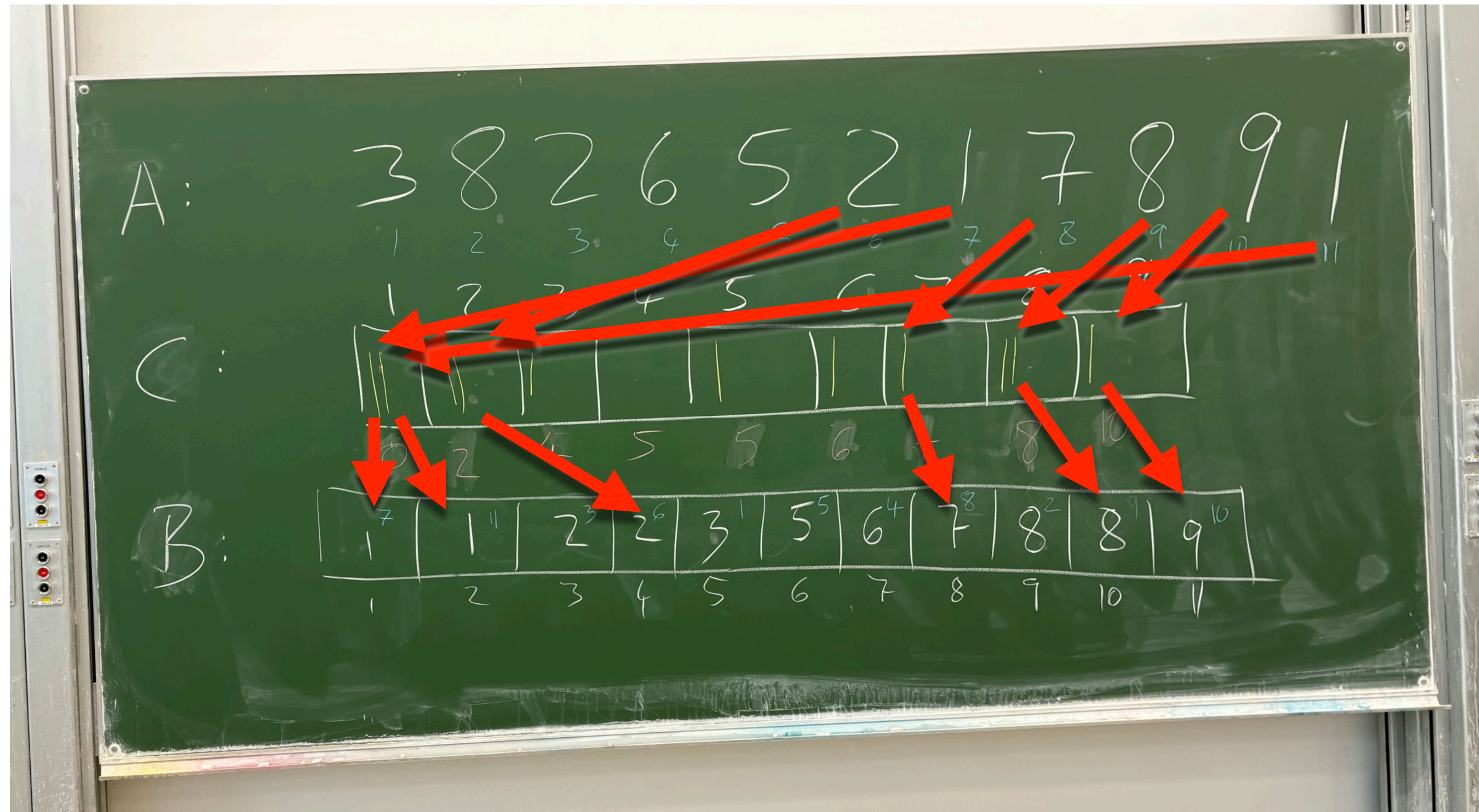
Countingsort



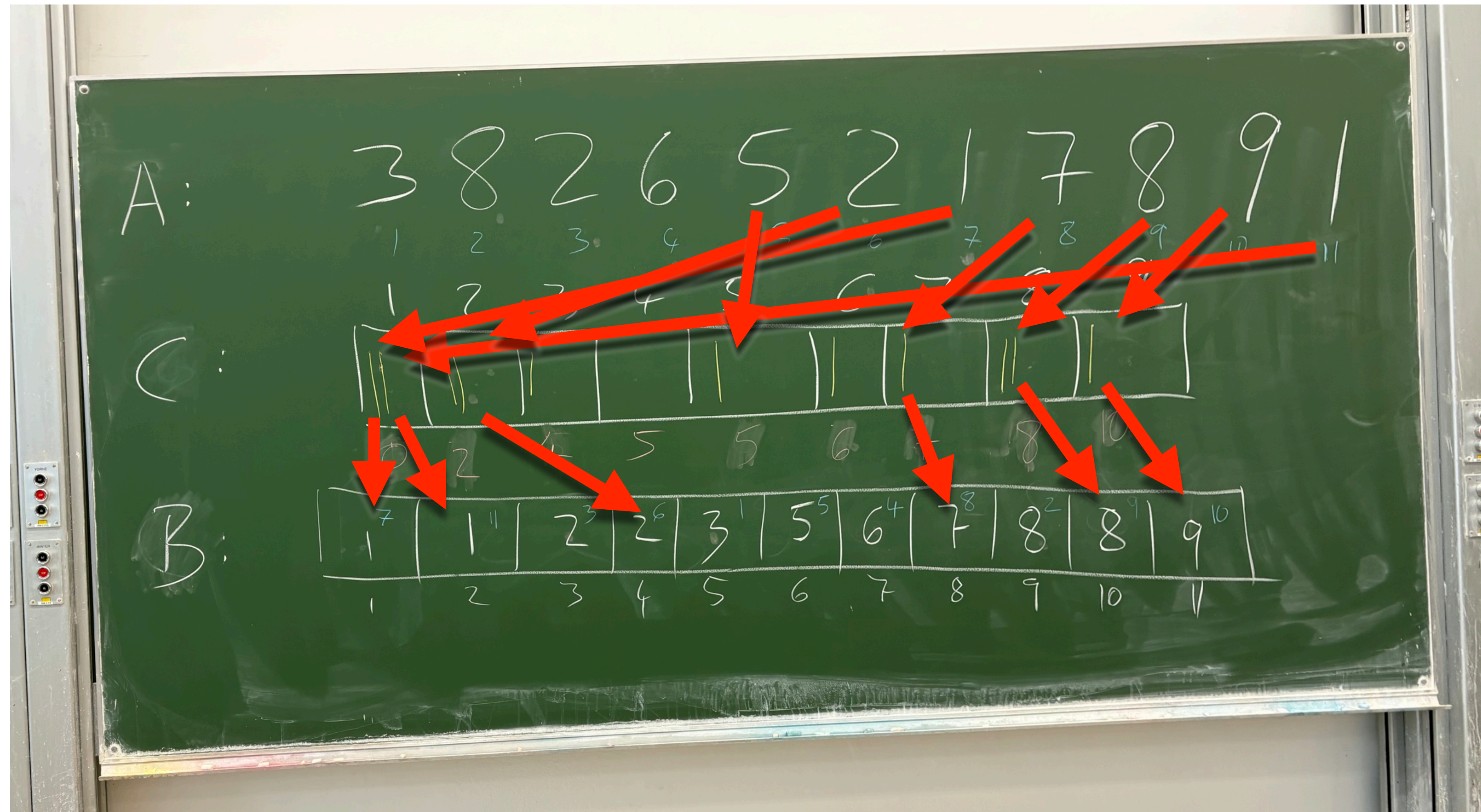
Countingsort



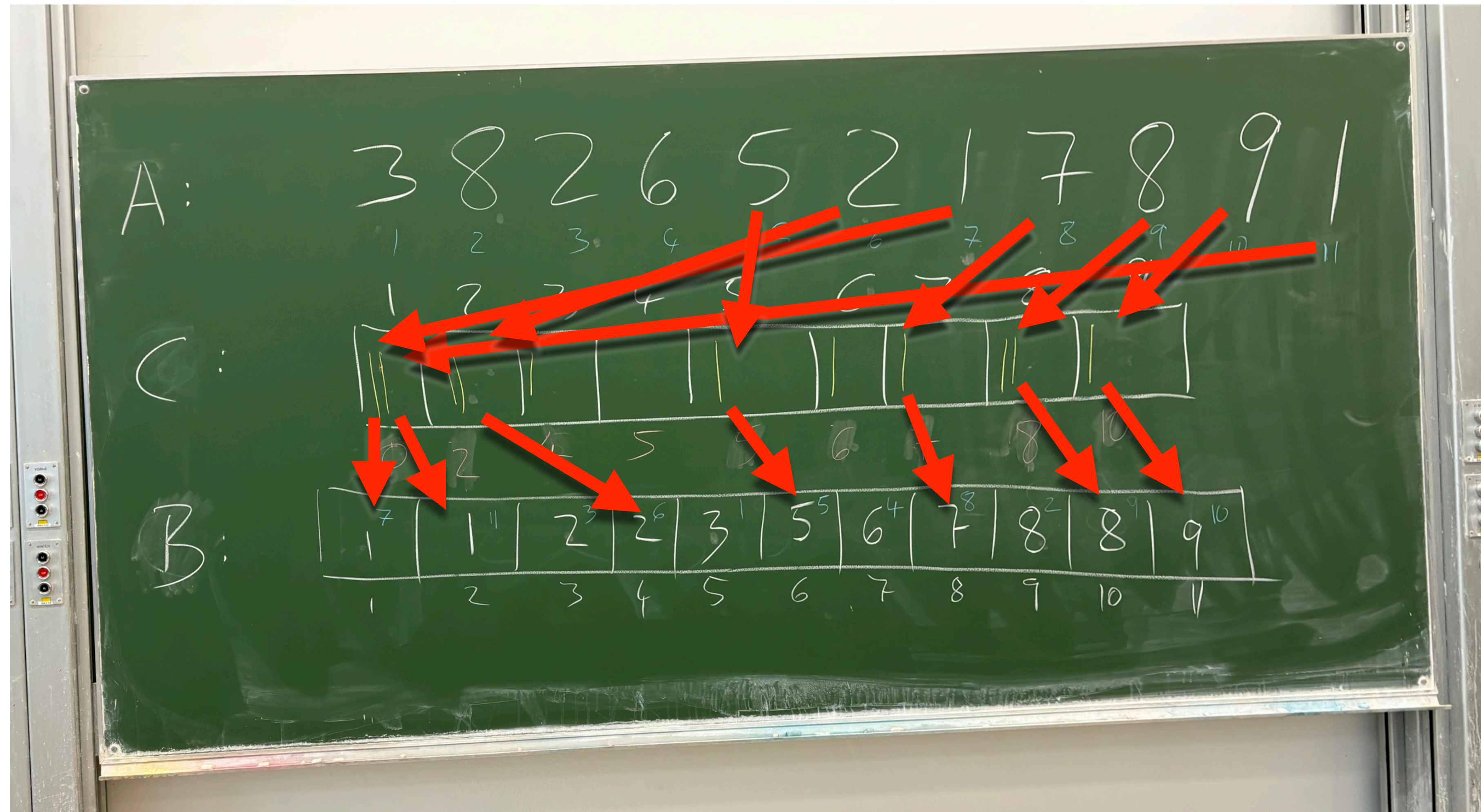
Countingsort



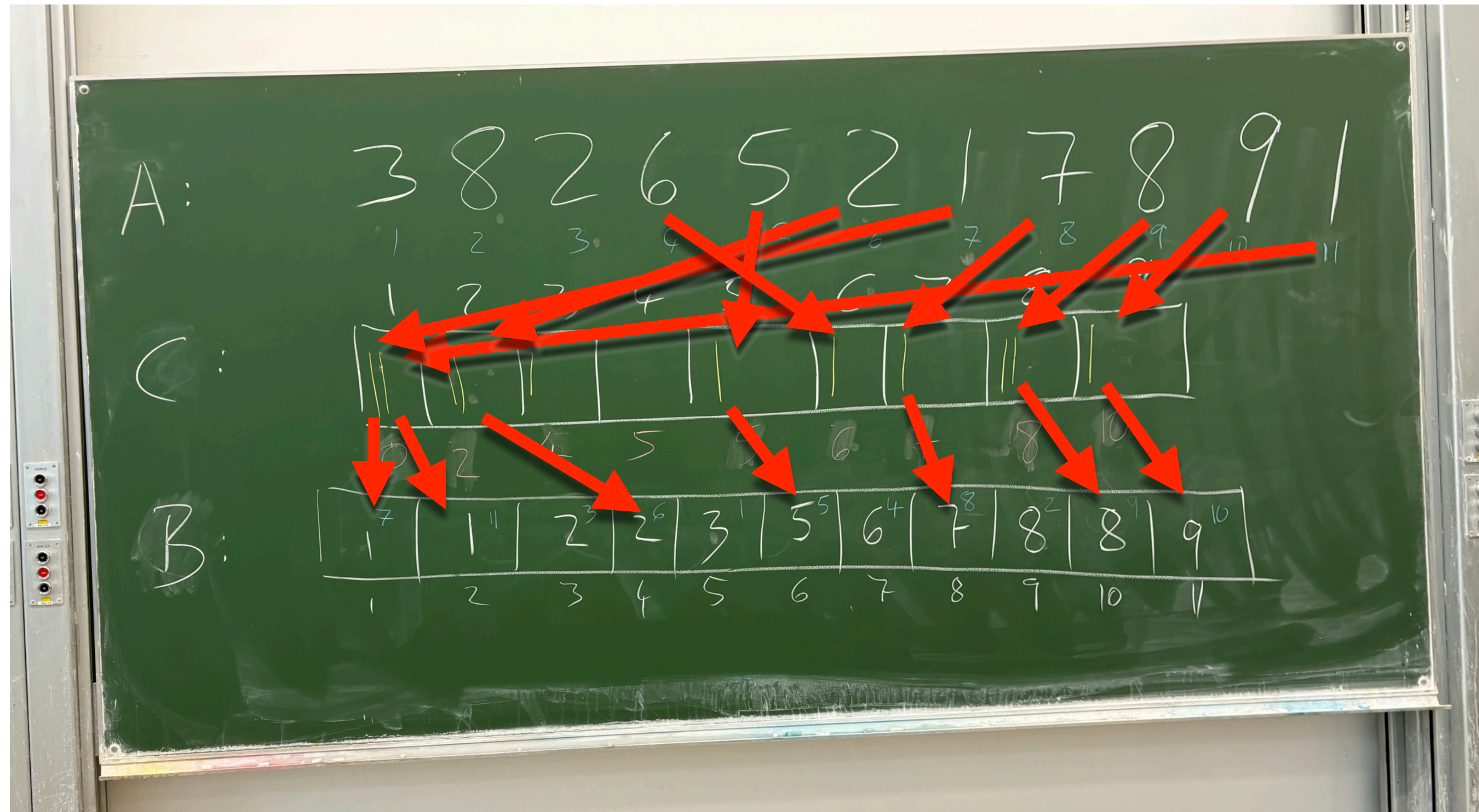
Countingsort



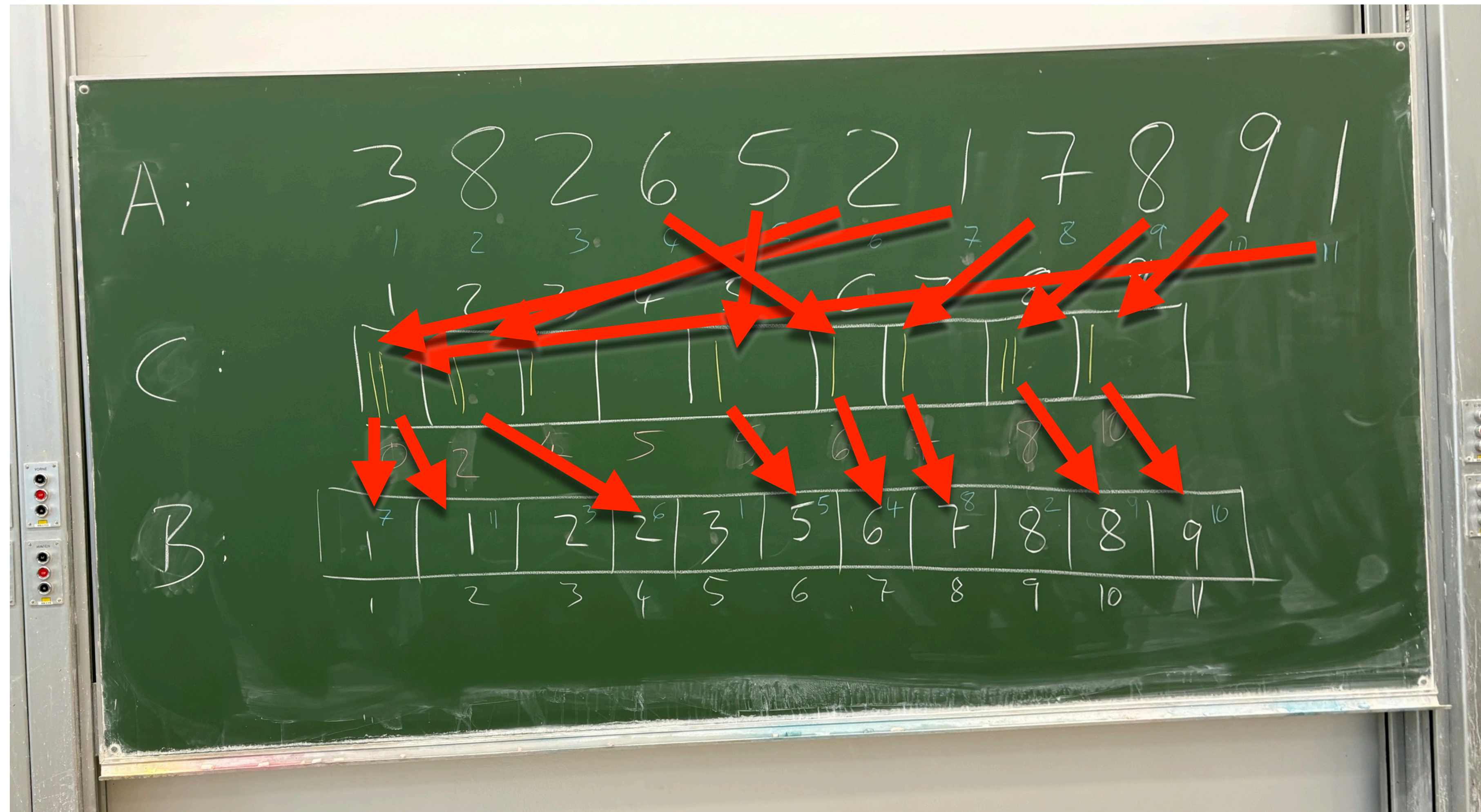
Countingsort



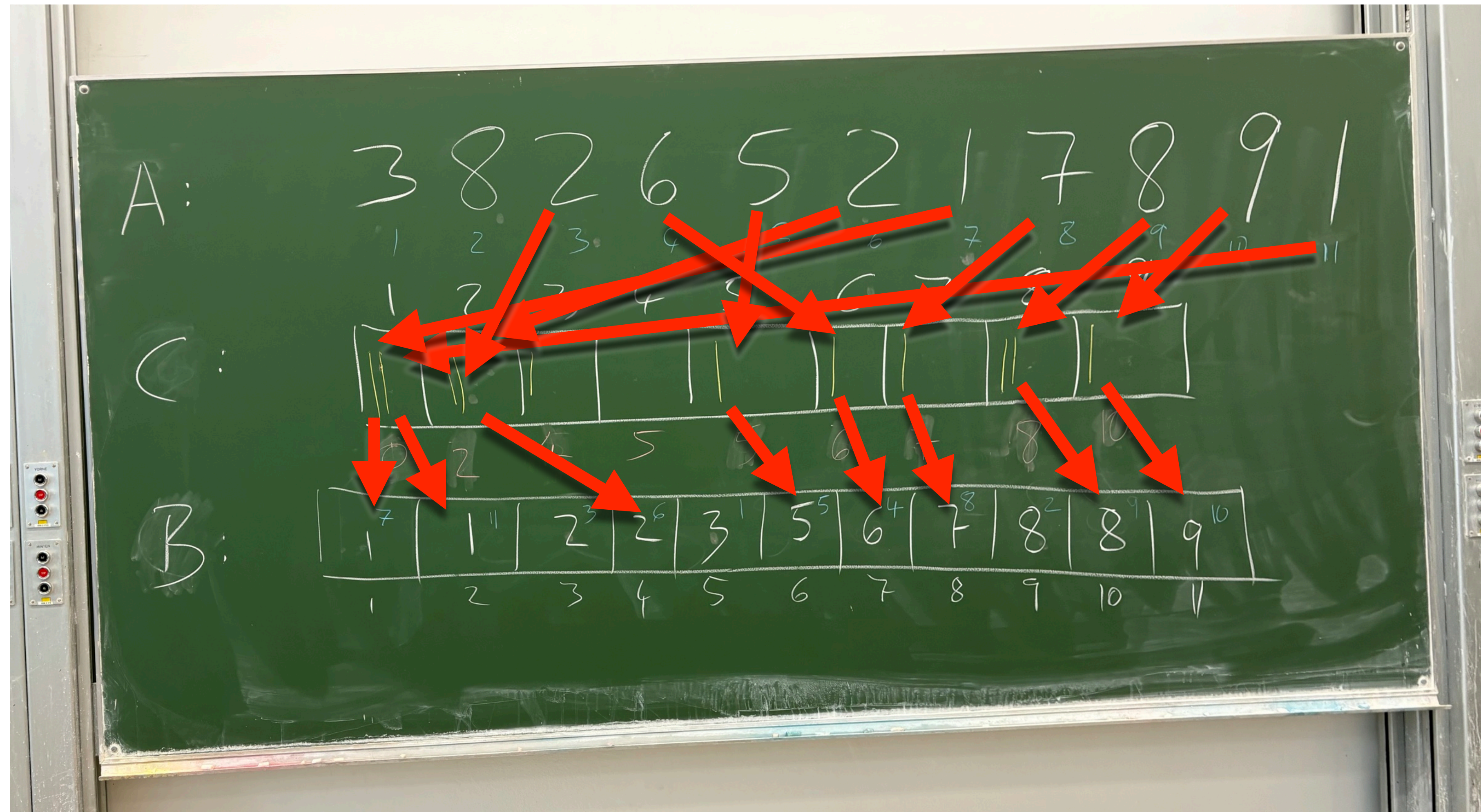
Countingsort



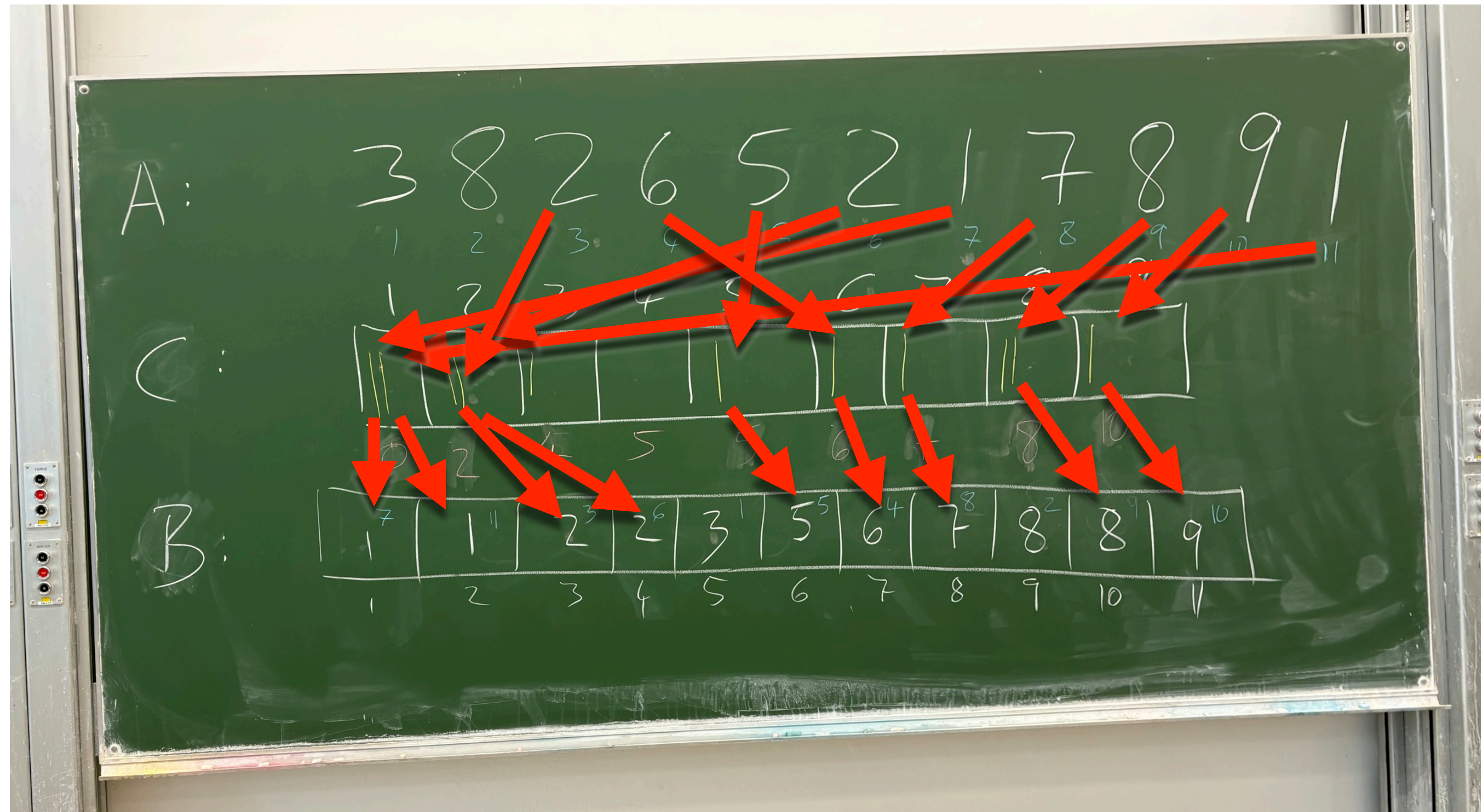
Countingsort



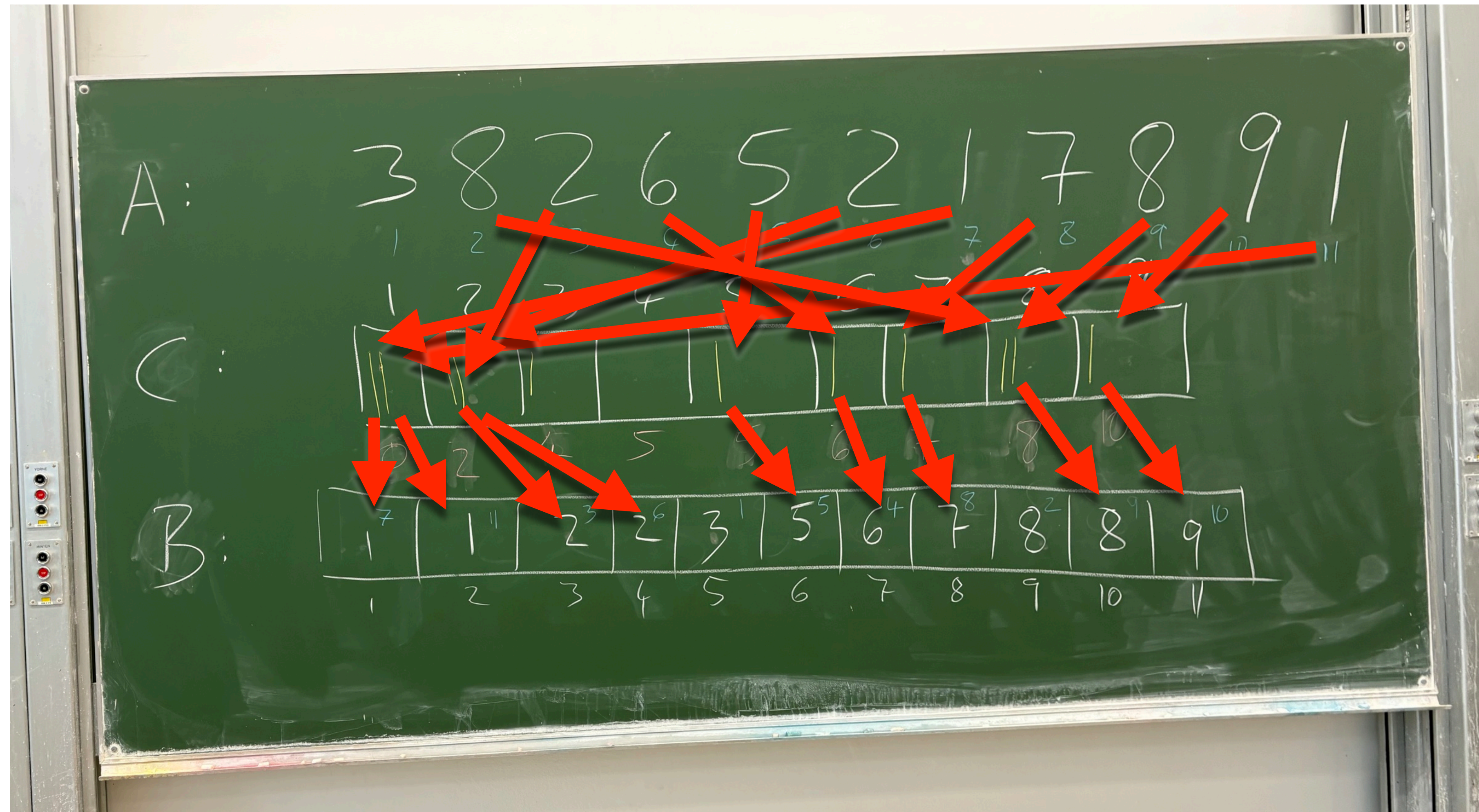
Countingsort



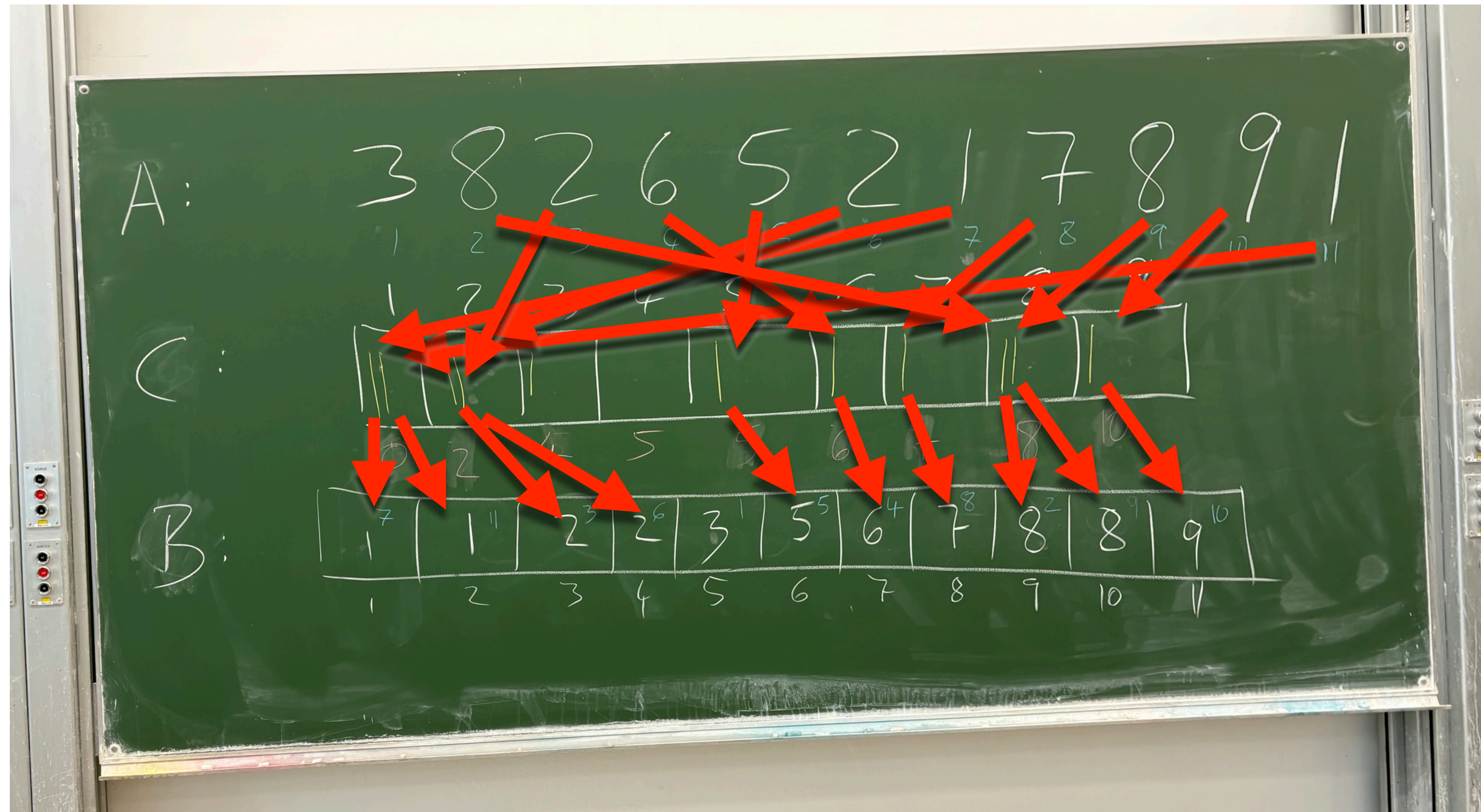
Countingsort



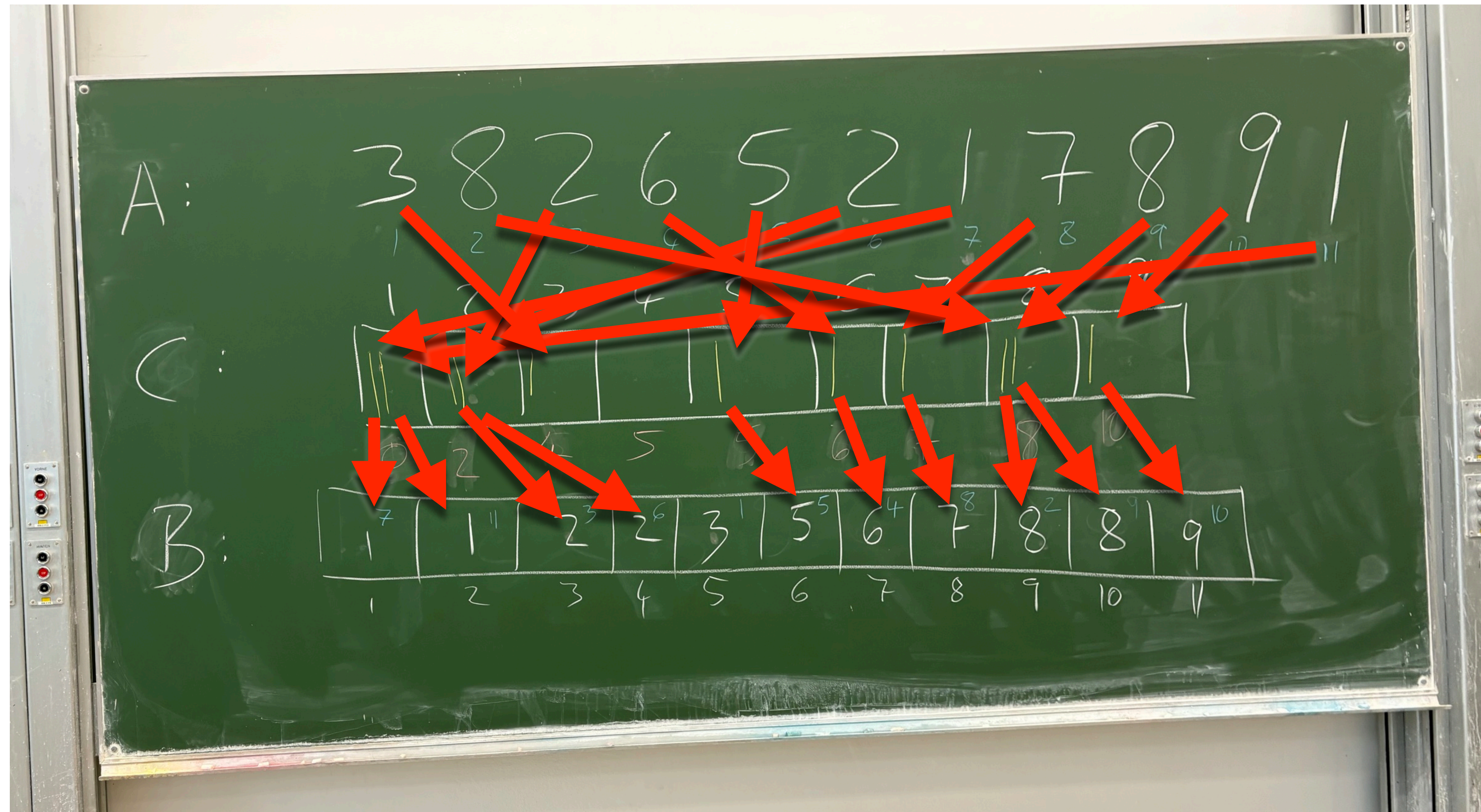
Countingsort



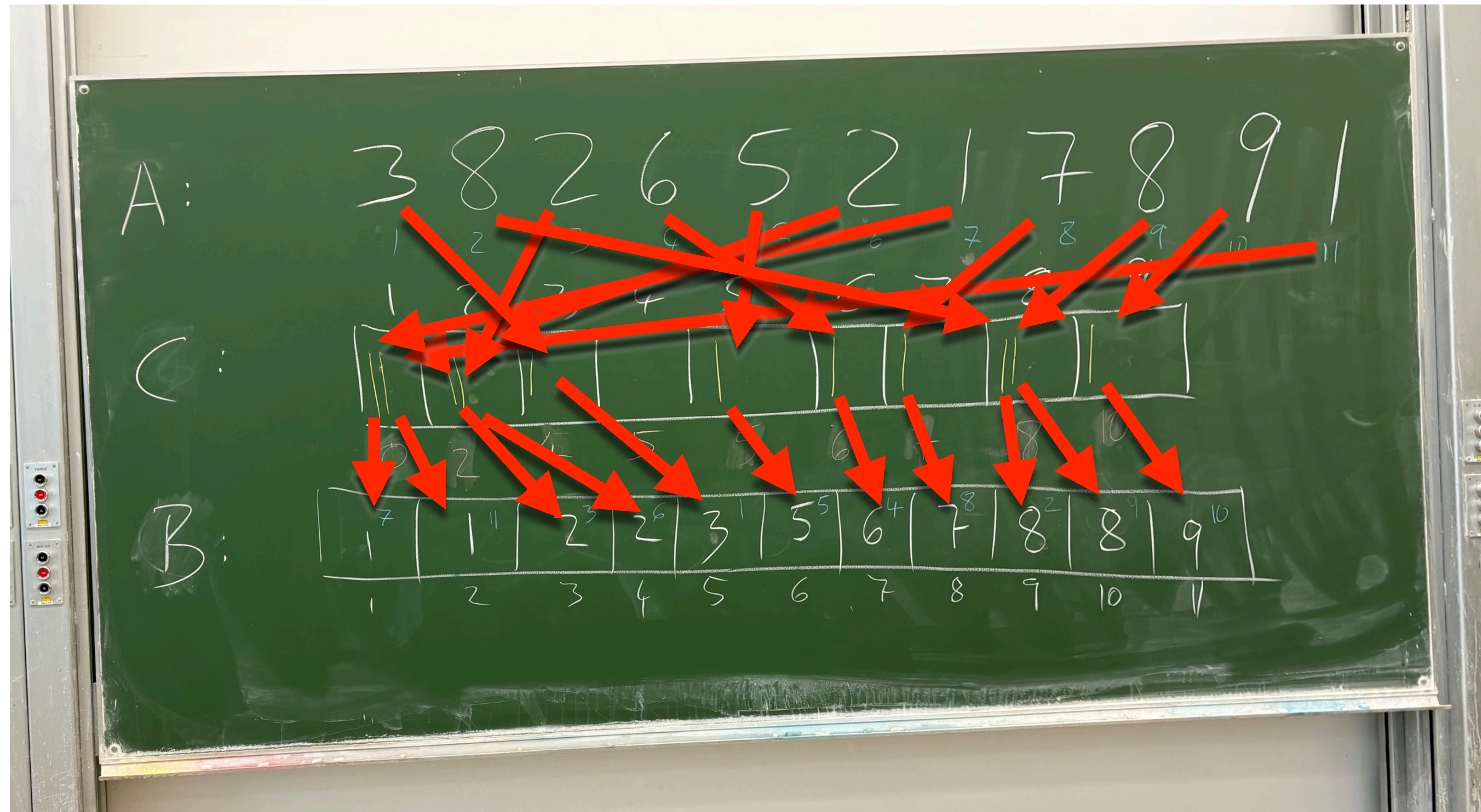
Countingsort



Countingsort



Countingsort

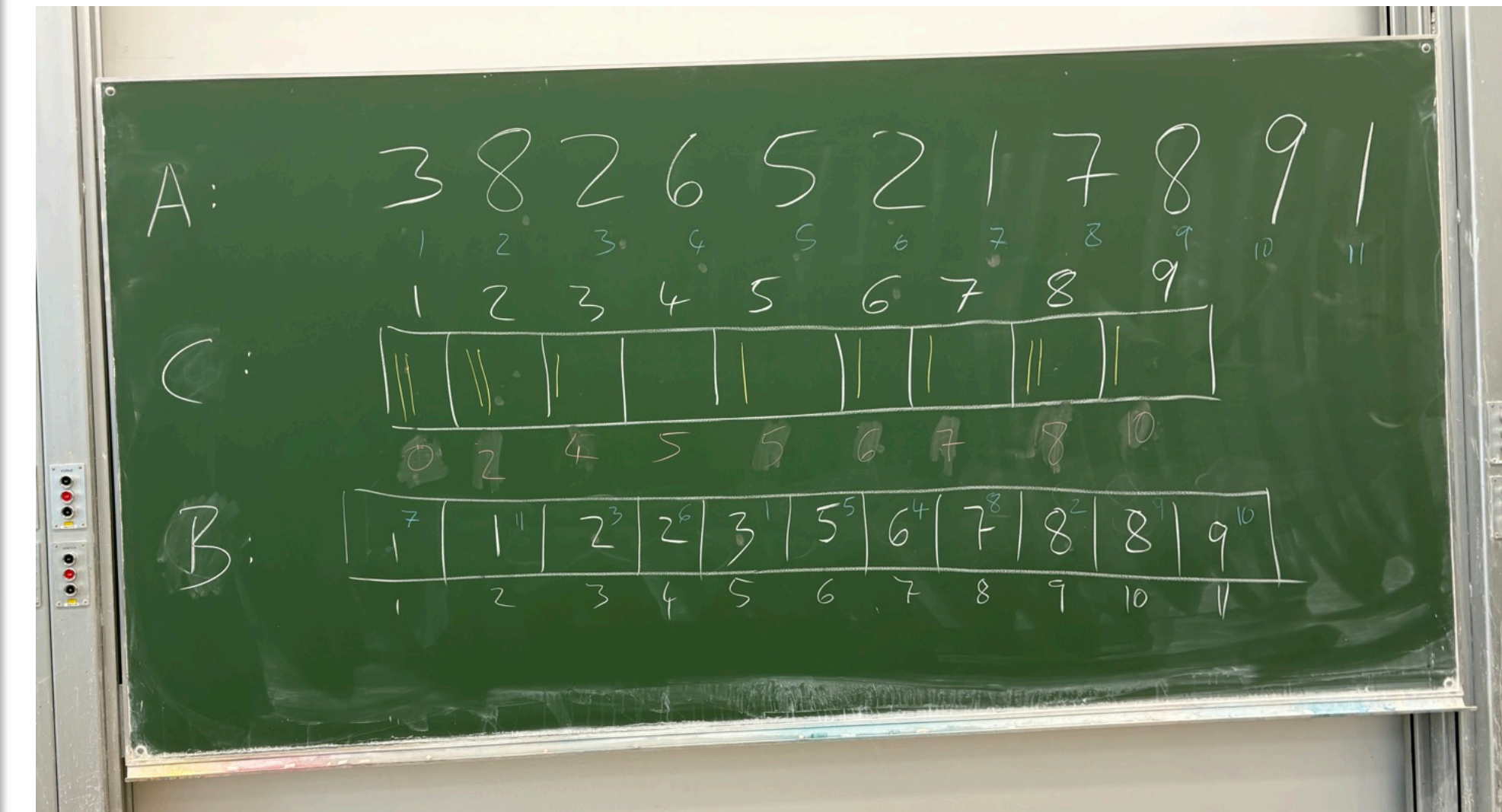


Countingsort

Eingabe: Array $A[1], \dots, A[n]$ mit Schlüsselwerten aus $\{1, \dots, k\}$

Ausgabe: Sortierte Kopie $B[1], \dots, B[n]$ von $A[1], \dots, A[n]$

```
1: function COUNTINGSORT( $A, B, k$ )
2:   for  $i \leftarrow 1$  to  $k$  do
3:      $C[i] \leftarrow 0$                                 ▷ Zählerarray initialisieren
4:   for  $j \leftarrow 1$  to  $n$  do
5:      $C[A[j]] \leftarrow C[A[j]] + 1$                 ▷ Anzahl Elemente mit Wert  $A[j]$  um 1 erhöhen
6:   for  $i \leftarrow 1$  to  $k$  do
7:      $C[i] \leftarrow C[i] + C[i - 1]$                 ▷ Anzahl Elemente mit Wert höchstens  $i$  zählen
8:   for  $j \leftarrow n$  downto  $1$  do
9:      $B[C[A[j]]] \leftarrow A[j]$                     ▷ Element  $A[j]$  an die richtige Stelle in  $B$  schreiben
10:     $C[A[j]] \leftarrow C[A[j]] - 1$ 
```



Satz 5.15. Für n Objekte mit Schlüsseln aus $\{1, \dots, k\}$ benötigt Countingsort (s. Alg. 5.14) die Zeit $O(n + k)$. Insbesondere benötigt man für $k \in O(n)$ die Zeit von $\Theta(n)$.

Radixsort

329	720	720	329
457	355	329	355
657	436	436	436
839	→ 457	→ 839	→ 457
436	657	355	657
720	329	457	720
355	839	657	839

Eingabe: n Zahlen mit je d Ziffern, die k verschiedene Werte annehmen können,
 $A[i], \dots, A[n]$

Ausgabe: Sortiertes Array

```
1: function RADIXSORT( $A, d$ )  
2:   erzeuge (mit 0 initialisiertes) Array  $B$   
3:   for  $i \leftarrow 1$  to  $d$  do  
4:     COUNTINGSORT( $A, B, i$ )
```

Satz 5.19. *Radixsort (s. Alg. 5.18) liefert ein korrektes Ergebnis in der Zeit $\Theta(d \cdot (n+k))$.*

Radixsort



Type 80 Card Sorter, IBM, US, 1925

Sorting data into either numerical or alphabetic order is one of the most basic office automation tasks. The Type 80 sorted by reading the punches in one column of a card, then routing that card to the corresponding output bin. Sorting by multi-digit numbers or names required multiple passes. This sorter was one of IBM's most widely used punched card equipment in the pre-computing era. In 1943, there were over 10,000 units on lease.

102631026

Satz 5.19. *Radix*

Sorting data into either numerical or alphabetic order is one of the most basic office automation tasks. The Type 80 sorted by reading the punches in one column of a card, then routing that card to the corresponding output bin. Sorting by multi-digit numbers or names required multiple passes. This sorter was one of IBM's most widely used punched card equipment in the pre-computing era. In 1943, there were over 10,000 units on lease.

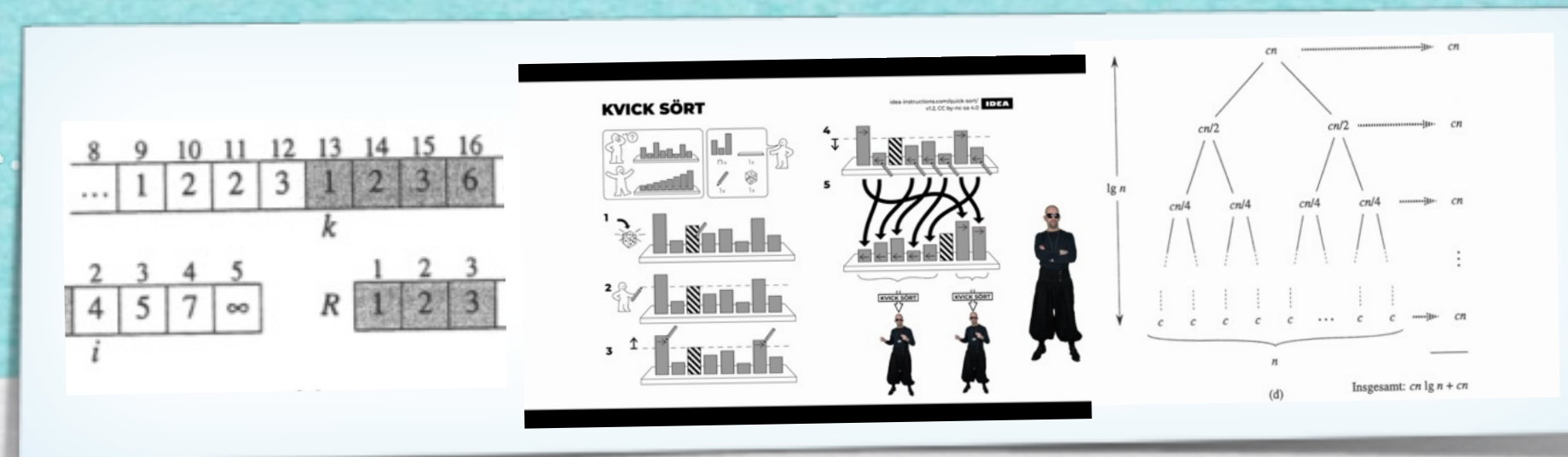


Technische
Universität
Braunschweig

Kapitel 5 - Quiz 6

Quicksort und Mediane



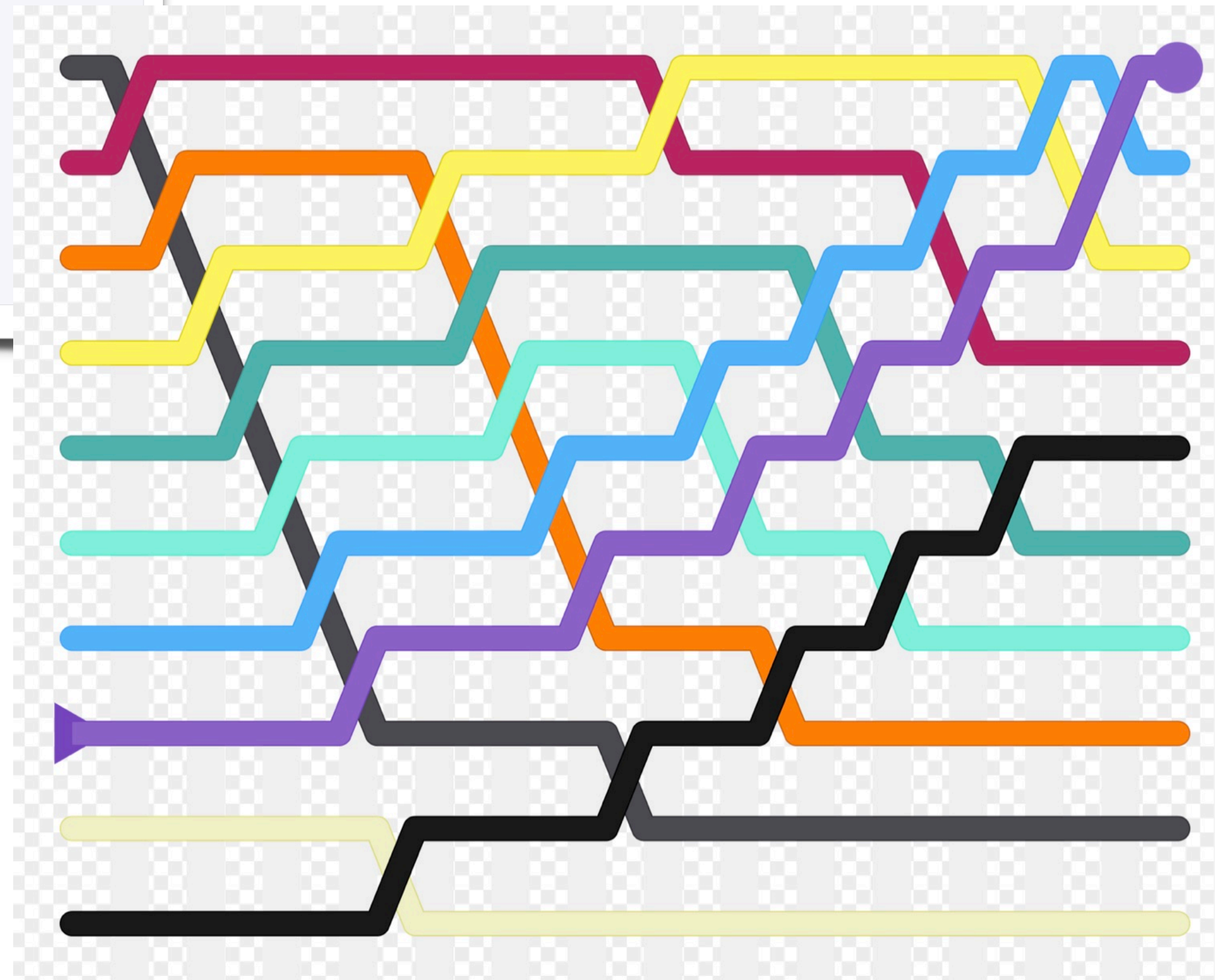


Kapitel 5.8: *Paralleles Sortieren* *Algorithmen und Datenstrukturen* *WS 2025/26*

Prof. Dr. Sándor Fekete

Bubblesort

```
bubbleSort(Array A)
  for (n=A.size; n>1; --n){
    for (i=0; i<n-1; ++i){
      if (A[i] > A[i+1]){
        A.swap(i, i+1)
      } // Ende if
    } // Ende innere for-Schleife
  } // Ende äußere for-Schleife
```

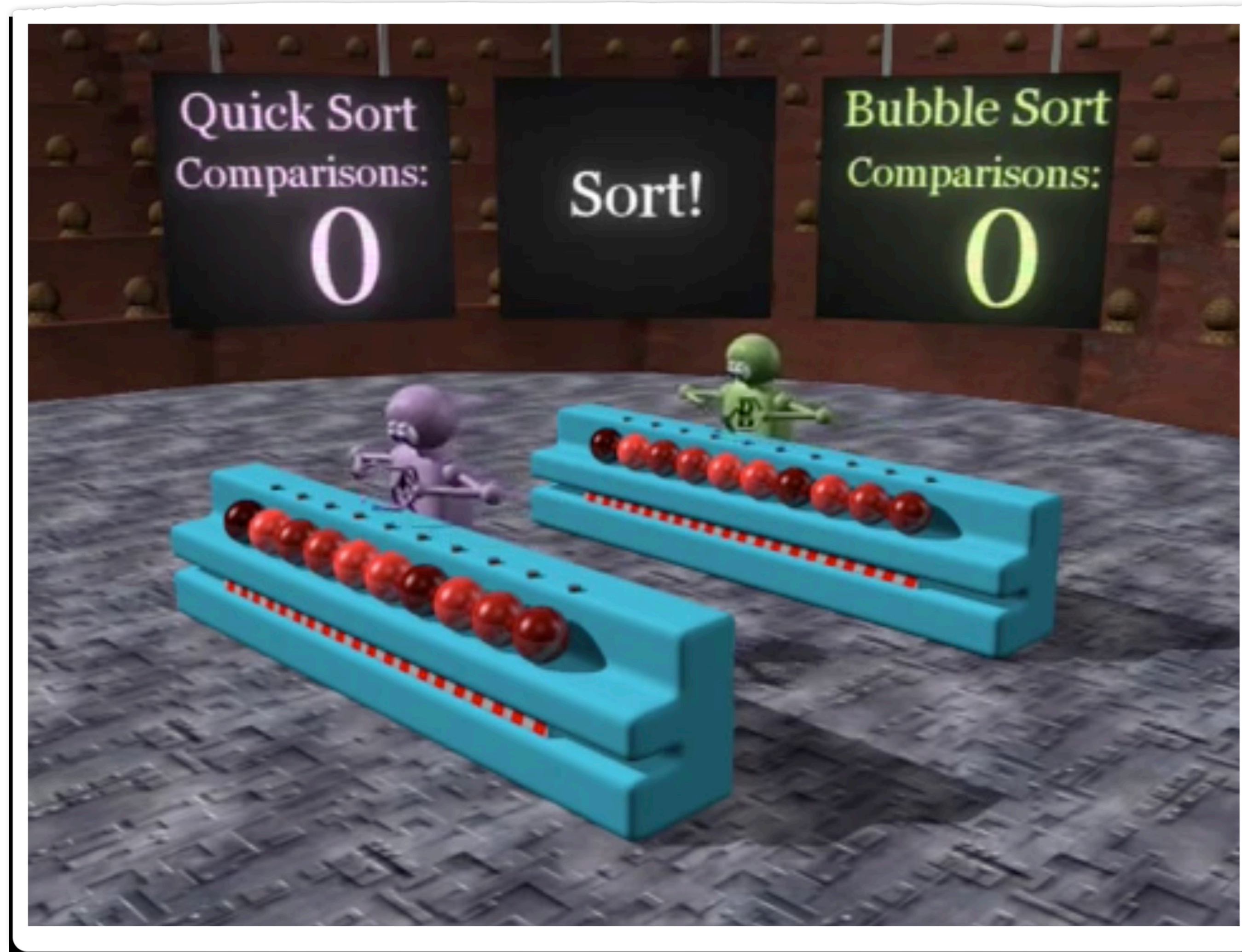


Pmdumuid

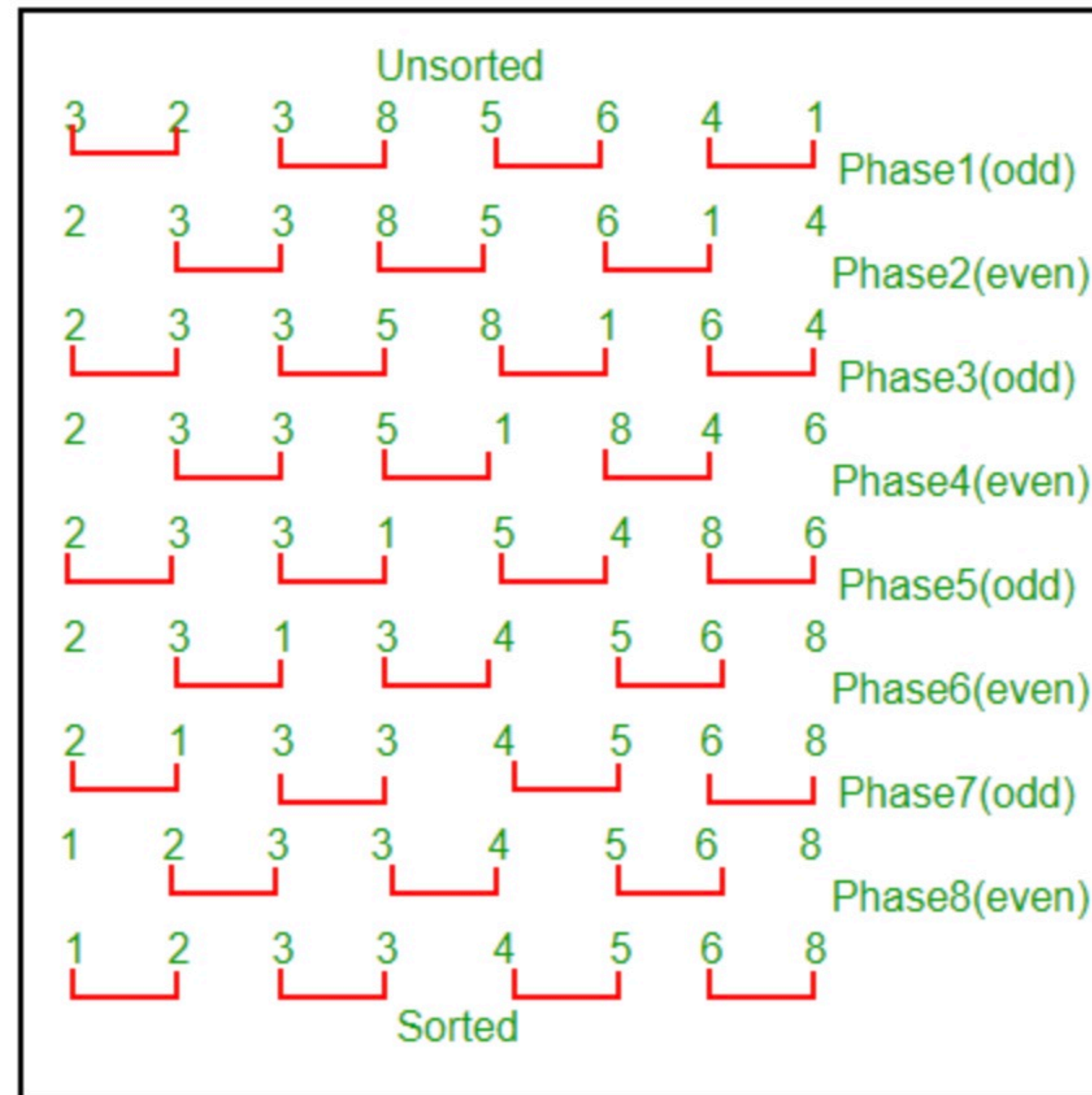
Bubblesort



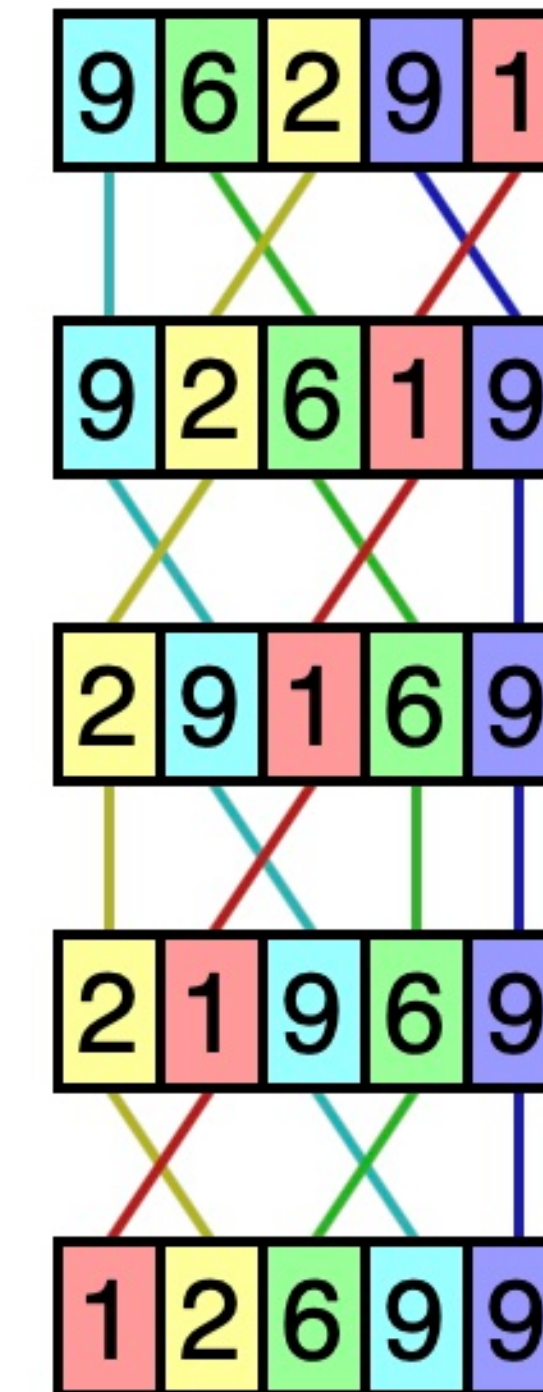
Bubblesort



Paralleles Bubblesort

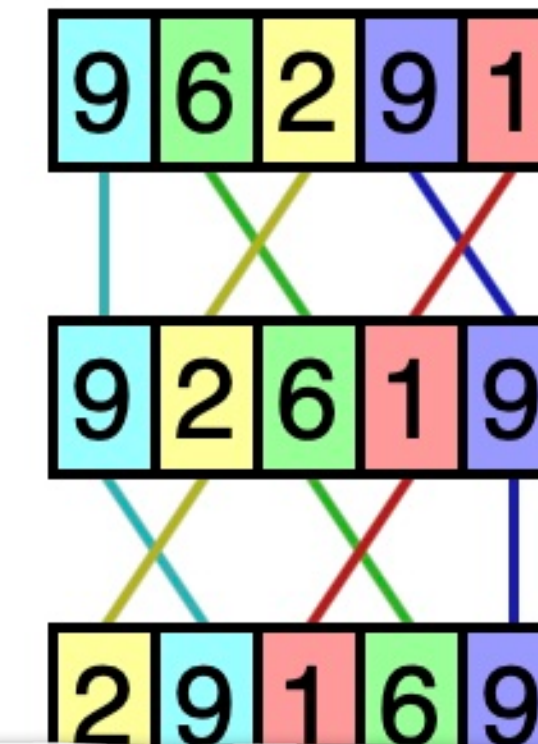
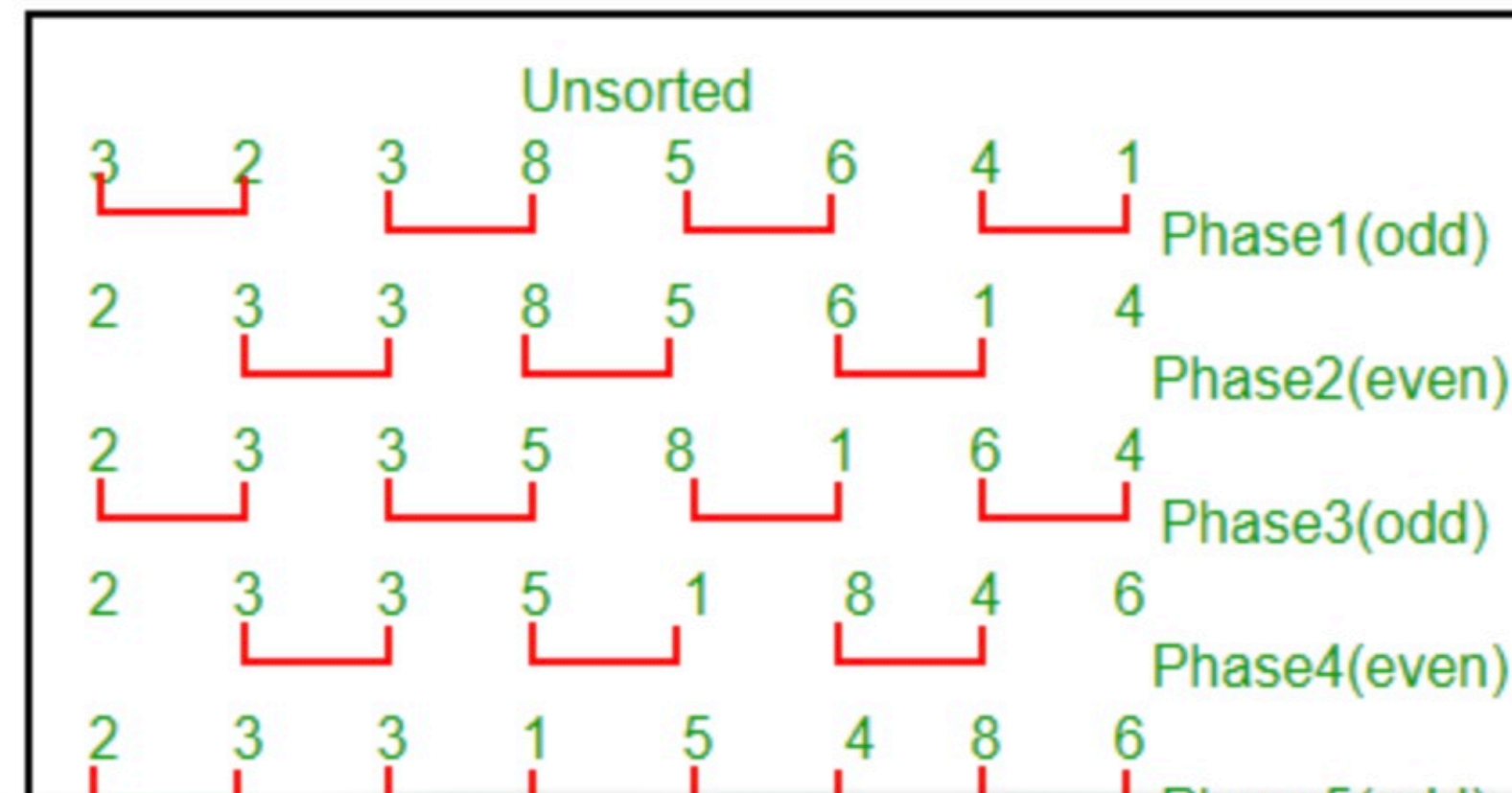


geeksforgeeks



growingwiththeweb

Paralleles Bubblesort

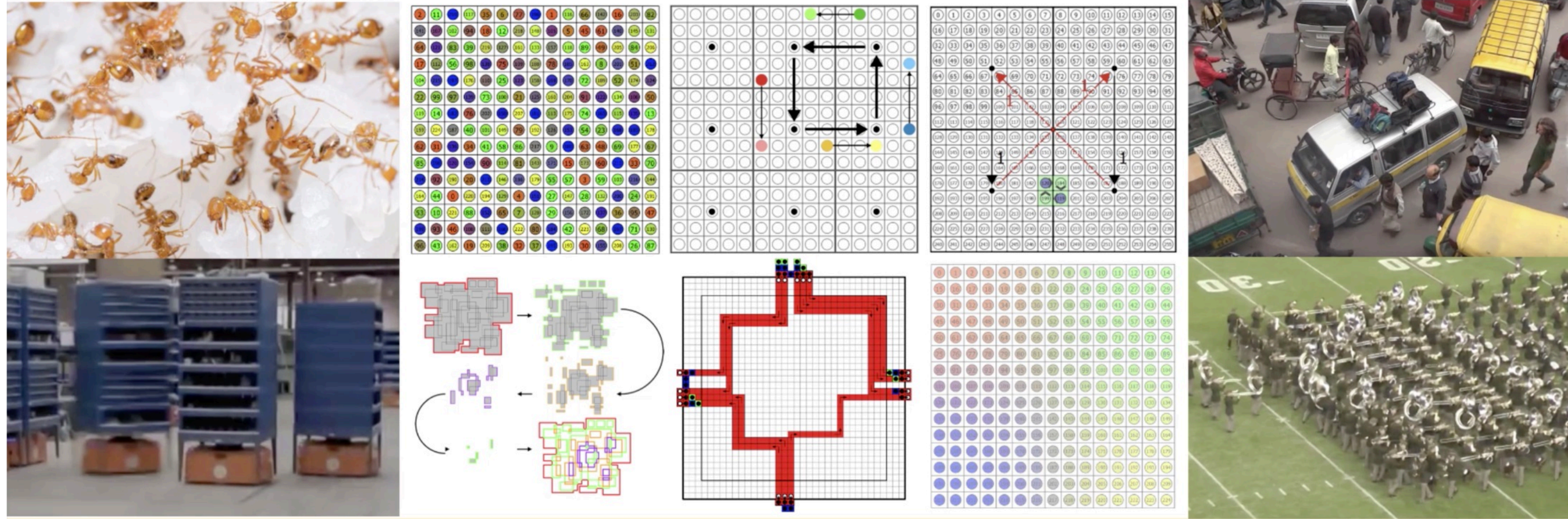


Satz 5.25

Paralleles Bubblesort sortiert die Zahlen in n parallelen Runden.
Die Gesamtzahl der Vergleiche ist $\Theta(n^2)$.

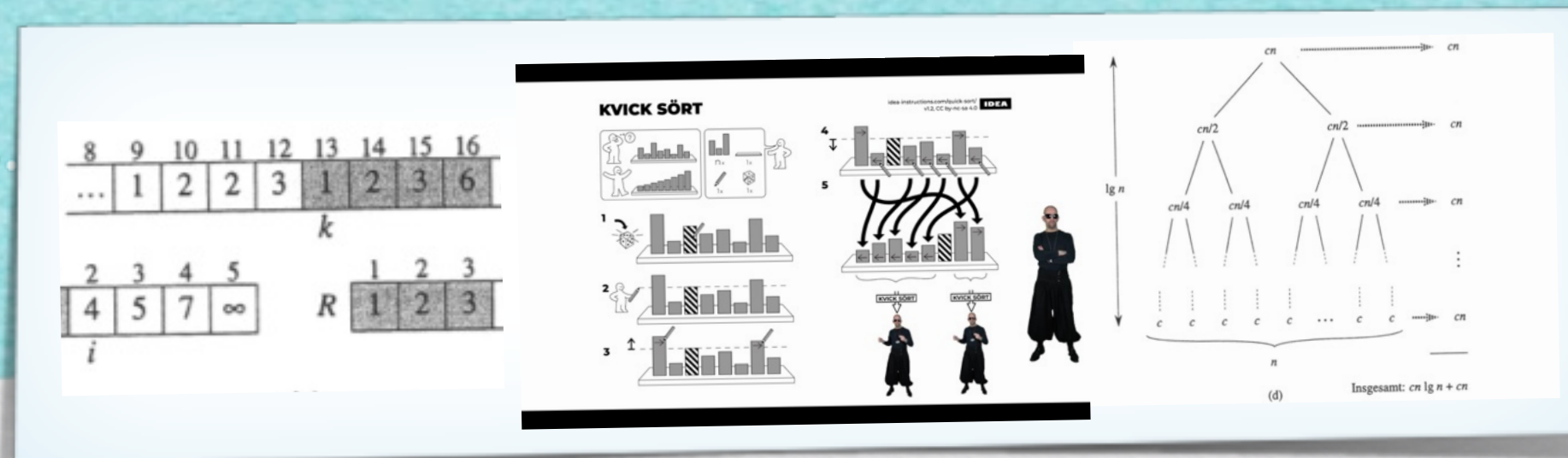
Beweis: Selbst!

Schwarmkoordination



Coordinated Motion Planning: The Video

**Aaron Becker, Sándor P. Fekete, Phillip Keldenich,
Matthias Konitzny, Lillian Lin, Christian Scheffer**



Kapitel 5.9: *Analoge Verfahren* *Algorithmen und Datenstrukturen* *WS 2025/26*

Prof. Dr. Sándor Fekete

Spaghettisort und andere analoge Verfahren

COMPUTER RECREATIONS

Analog gadgets that solve a diversity of problems and raise an array of questions

by A. K. Dewdney

Exactly one year ago a collection of analog gadgets in these pages set off an avalanche of similar devices from inspired readers. I am still extricating myself from a vast heap of wood boards, rubber bands, strings, balls of polystyrene, fish tanks, lead weights, canisters, tubing and stop-cocks. In the process I have, I think, identified the best of these gadgets and have arranged them into a kind of gallery through which the reader is invited to wander.

Analog gadgets are mechanical devices that solve specific problems by virtue of the fact that their construction or behavior is analogous to the elements of the problem. For example, the June 1984 column described SAG, the Spaghetti Analog Gadget. Lengths

of uncooked spaghetti were used as analogs for numbers. To sort the numbers in decreasing order, gather the spaghetti into a bundle held vertically and bring it down rather sharply on a tabletop; select the longest rod. Continued selection of the longest remaining rod produces the desired decreasing sequence of numbers. In addition to SAG, I presented gadgets for finding shortest paths, convex hulls and minimum trees. I even displayed a gadget for factoring numbers that consisted of mirrors and a laser beam.

The latest collection includes several ingenious new gadgets for solving problems in statistics, network theory, algebra and arithmetic. On leaving the gallery I shall reexamine some important issues that arise from the prospect

of analog computing: How accurate are analog computers and how much time do they really take to compute? Are there some analog computers that outperform digital machines?

The first of the new gadgets solves a certain problem in statistics by means of a wood board, rubber bands, nails and a smooth, rigid rod. A set of data points plotted on a sheet of graph paper may present a linear trend to the eye. If a linear relation really governs the points, what straight line best displays the relation? The gadget suggested by Marc Hawley of Mount Vernon, Ind., supplies one possible answer:

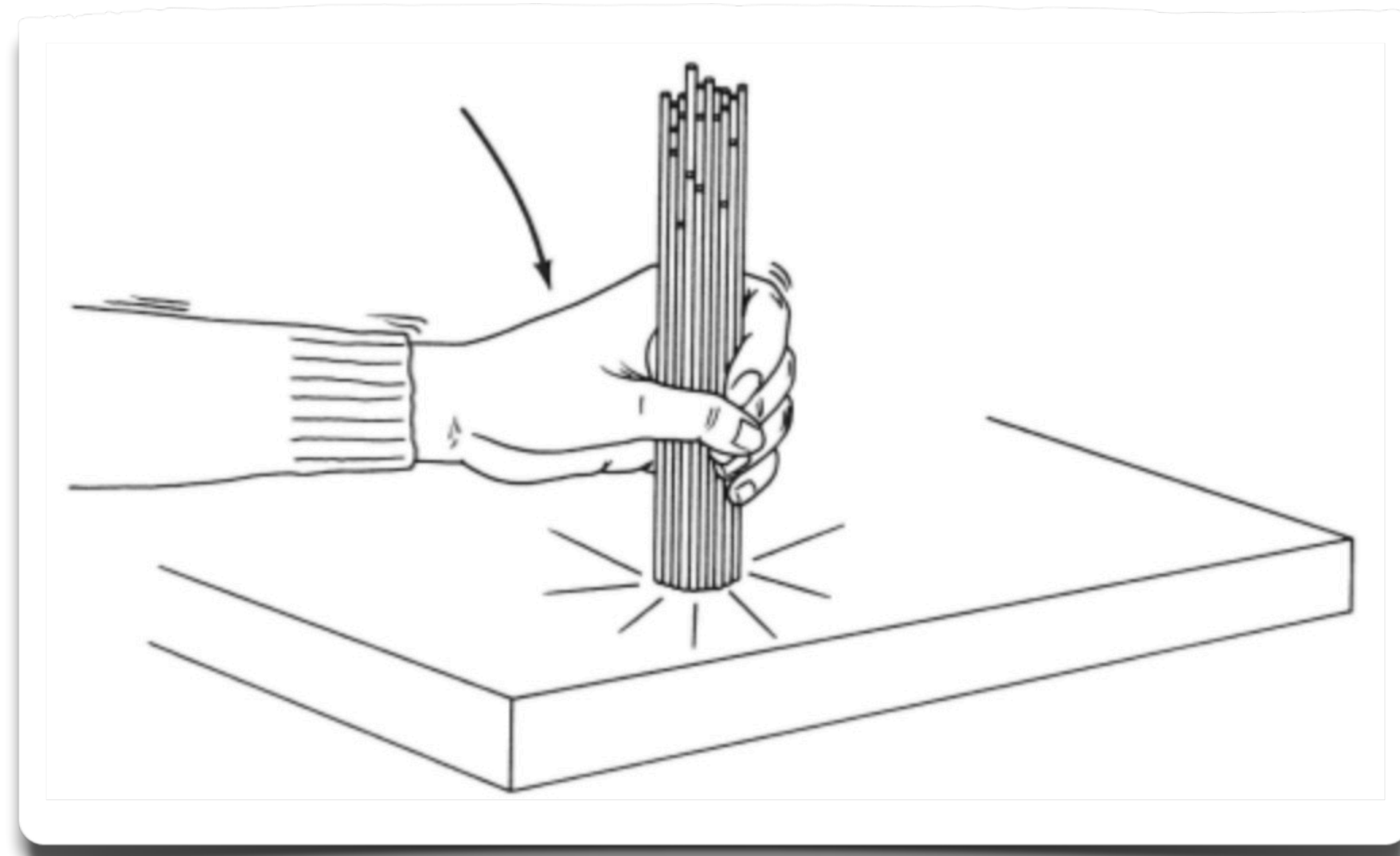
Plot the data points on a wood surface and drive a nail partway into the wood at each point. Next, slip a number of uniform rubber bands onto the rod, one band for each nail. Fit the rod approximately into place and pull each band over one of the nails. When the rod is released, it wiggles and shivers quickly into an equilibrium position [see illustration on this page].

The equilibrium position minimizes the total energy of the system; therefore the sum of the distances from the nails to the rod has also been minimized. In terms of such distances, the rod's final position indicates the straight line that best fits the data. It is not such distances but their squares that appear in the formulas for linear regression used by statisticians. Hawley's gadget computes something at least as complicated.

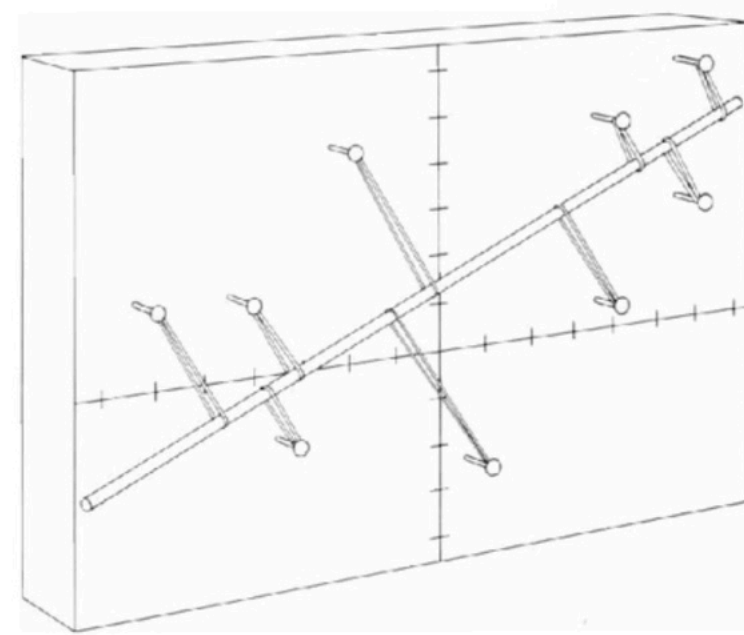
A charming string gadget was suggested by Jos Wennmacker of Nijmegen, Holland. Suppose we wish to know the longest path any message might travel in a communications network shaped like a tree. This path will be what combinatorial mathematicians call the diameter of the tree. In order to find the diameter Wennmacker reconstructs the tree by knotting together an analogy out of pieces of string. Each string is scaled to a specific communications line of the network. Two simple steps complete the computation. Pick up the string tree at any node and allow it to dangle freely. Now pick up the tree anew at the lowest node and dangle it once more. The longest path in the tree runs from the top node to the bottom one [see illustration on page 22].

When I initially encountered Wennmacker's gadget, my immediate reaction was, "It can't be that simple. Surely I must continue to select the bottom node and dangle the tree several more times." But one does not need to do that. Kudos in this column for the most elegant argument.

In last June's column I stated that the problem of finding the longest path in an arbitrary network was what theo-



1. For each number, break off a piece of spaghetti, whose length equals the number. This takes linear time.
2. Take all spaghetti in your fist, and slam their lower sides on the table, so they all point up starting at the table. This takes constant time, thanks to the multiprocessing properties of our universe.
3. Lower your other hand on the bundle of spaghetti. Take out the one which you touch first. This is clearly the longest one. Continue to lower your hand and to remove spaghetti, laying them out next to each other, until all spaghetti have been processed. Again, this takes linear time.
4. Transcribe the spaghetti lengths back to the respective numbers, using linear time. Done!



A gadget for finding the line that best fits a series of data points

18

© 1985 SCIENTIFIC AMERICAN, INC.

Spaghettisort und andere analoge Verfahren

COMPUTER RECREATIONS

Analog gadgets that solve a diversity of problems and raise an array of questions

by A. K. Dewdney

Exactly one year ago a collection of analog gadgets in these pages set off an avalanche of similar devices from inspired readers. I am still extricating myself from a vast heap of wood boards, rubber bands, strings, balls of polystyrene, fish tanks, lead weights, canisters, tubing and stopcocks. In the process I have, I think, identified the best of these gadgets and have arranged them into a kind of gallery through which the reader is invited to wander.

Analog gadgets are mechanical devices that solve specific problems by virtue of the fact that their construction or behavior is analogous to the elements of the problem. For example, the June 1984 column described SAG, the Spaghetti Analog Gadget. Lengths

of uncooked spaghetti were used as analogs for numbers. To sort the numbers in decreasing order, gather the spaghetti into a bundle held vertically and bring it down rather sharply on a tabletop; select the longest rod. Continued selection of the longest remaining rod produces the desired decreasing sequence of numbers. In addition to SAG, I presented gadgets for finding shortest paths, convex hulls and minimum trees. I even displayed a gadget for factoring numbers that consisted of mirrors and a laser beam.

The latest collection includes several ingenious new gadgets for solving problems in statistics, network theory, algebra and arithmetic. On leaving the gallery I shall reexamine some important issues that arise from the prospect

of analog computing: How accurate are analog computers and how much time do they really take to compute? Are there some analog computers that outperform digital machines?

The first of the new gadgets solves a certain problem in statistics by means of a wood board, rubber bands, nails and a smooth, rigid rod. A set of data points plotted on a sheet of graph paper may present a linear trend to the eye. If a linear relation really governs the points, what straight line best displays the relation? The gadget suggested by Marc Hawley of Mount Vernon, Ind., supplies one possible answer:

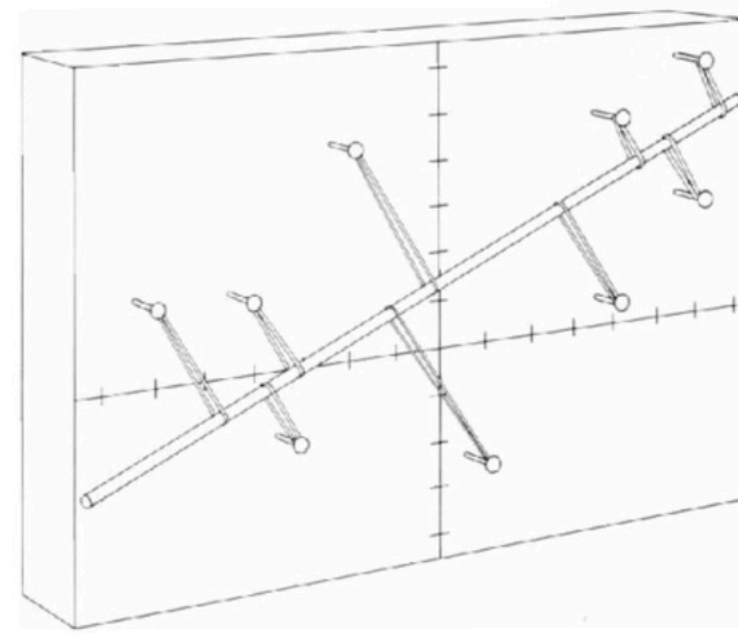
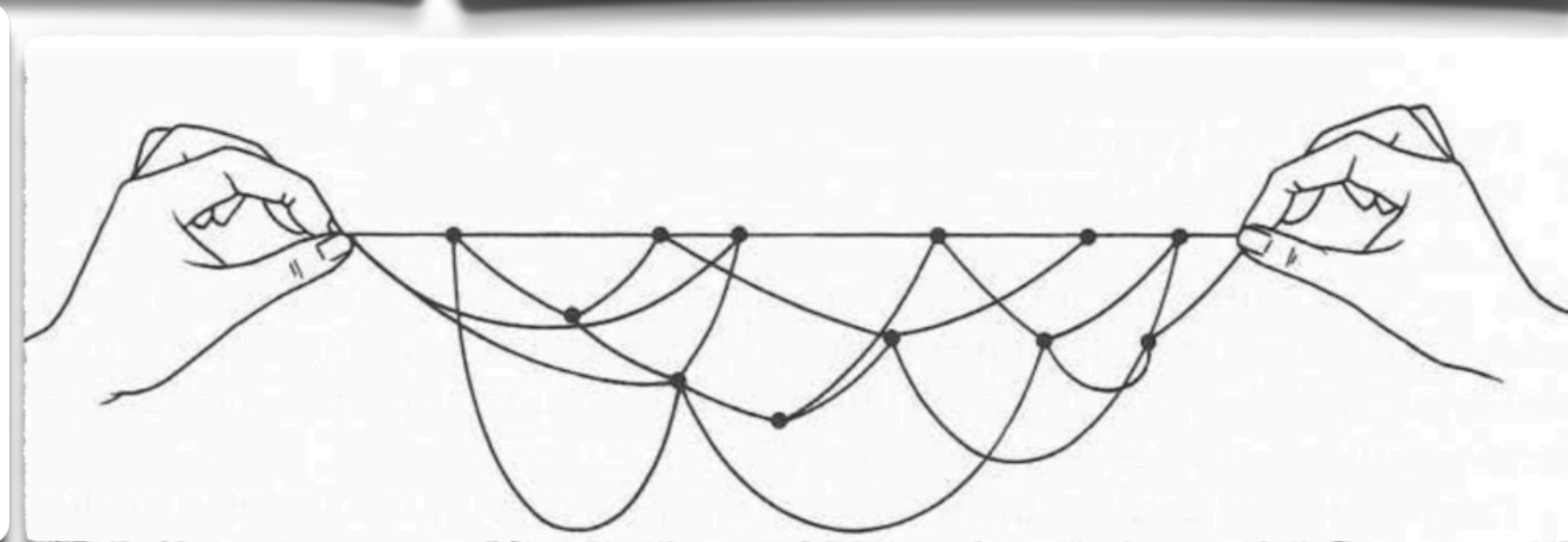
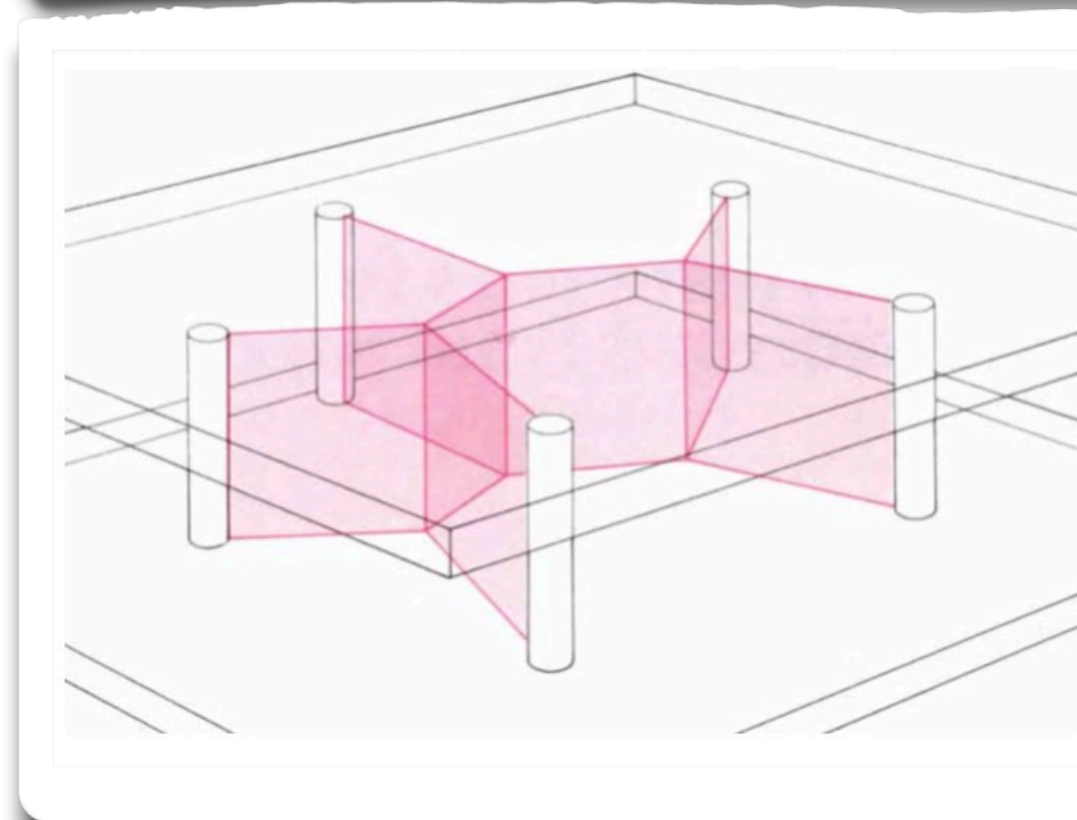
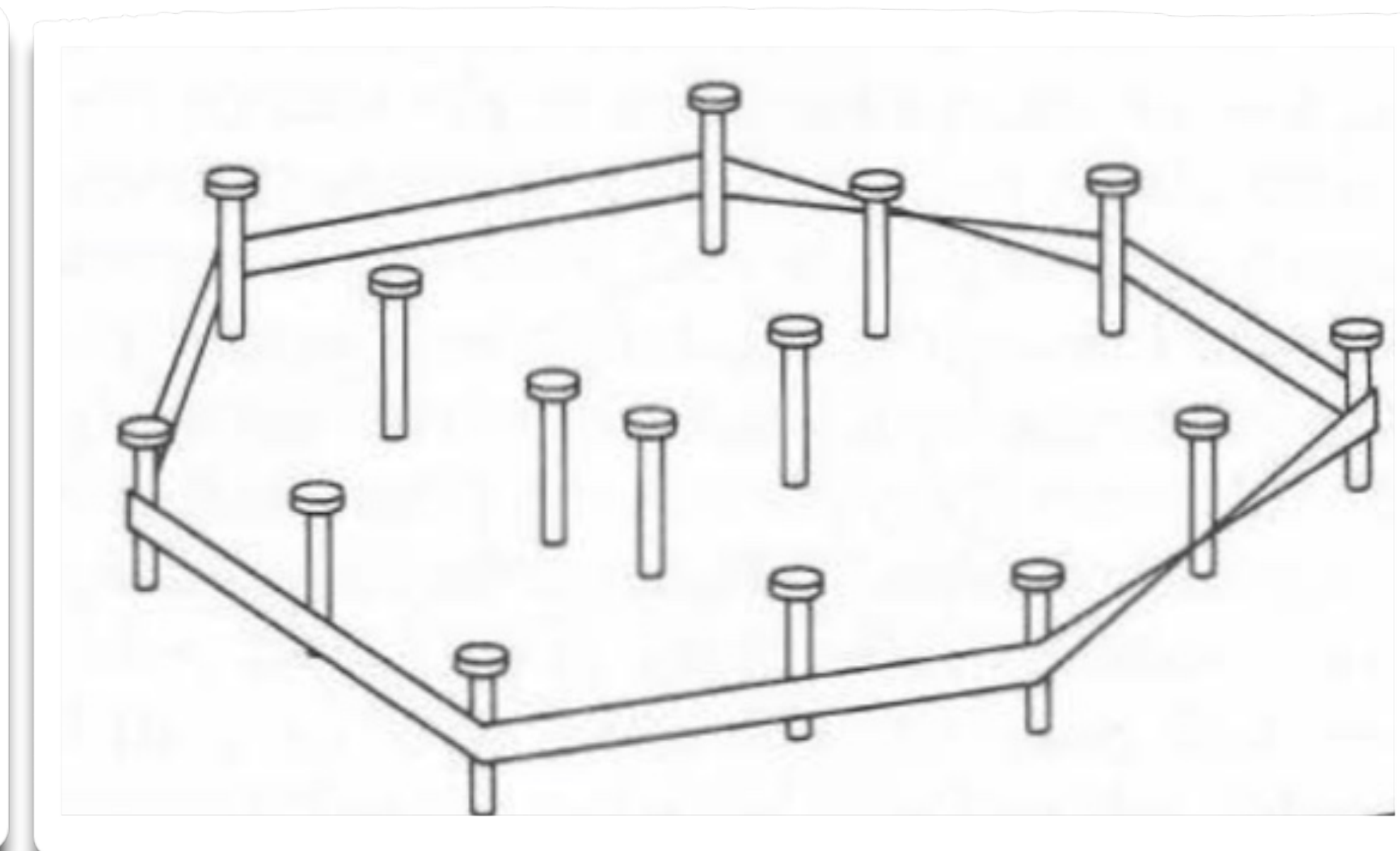
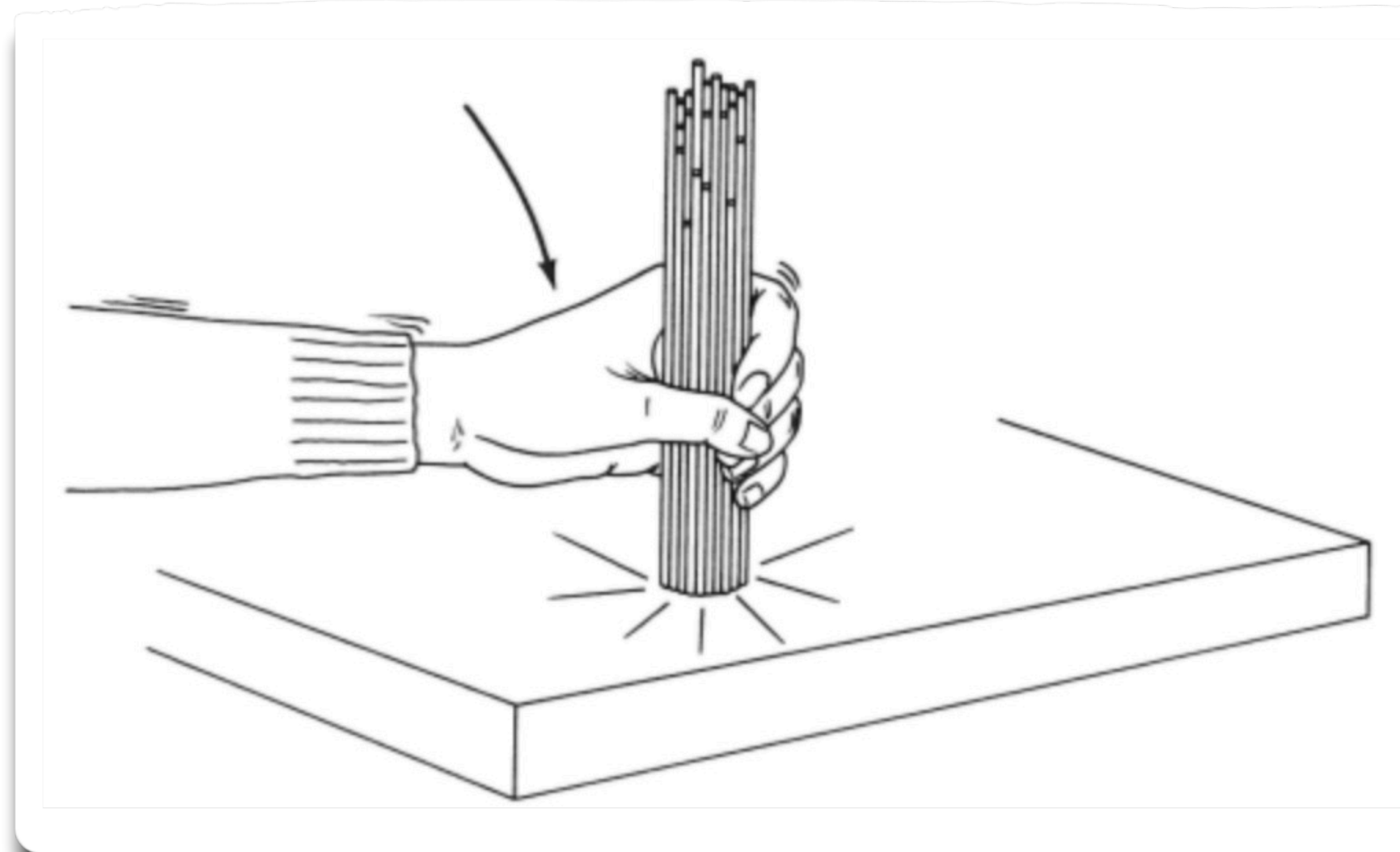
Plot the data points on a wood surface and drive a nail partway into the wood at each point. Next, slip a number of uniform rubber bands onto the rod, one band for each nail. Fit the rod approximately into place and pull each band over one of the nails. When the rod is released, it wiggles and shivers quickly into an equilibrium position [see illustration on this page].

The equilibrium position minimizes the total energy of the system; therefore the sum of the distances from the nails to the rod has also been minimized. In terms of such distances, the rod's final position indicates the straight line that best fits the data. It is not such distances but their squares that appear in the formulas for linear regression used by statisticians. Hawley's gadget computes something at least as complicated.

A charming string gadget was suggested by Jos Wennmacker of Nijmegen, Holland. Suppose we wish to know the longest path any message might travel in a communications network shaped like a tree. This path will be what combinatorial mathematicians call the diameter of the tree. In order to find the diameter Wennmacker reconstructs the tree by knotting together an analogy out of pieces of string. Each string is scaled to a specific communications line of the network. Two simple steps complete the computation. Pick up the string tree at any node and allow it to dangle freely. Now pick up the tree anew at the lowest node and dangle it once more. The longest path in the tree runs from the top node to the bottom one [see illustration on page 22].

When I initially encountered Wennmacker's gadget, my immediate reaction was, "It can't be that simple. Surely I must continue to select the bottom node and dangle the tree several more times." But one does not need to do that. Kudos in this column for the most elegant argument.

In last June's column I stated that the problem of finding the longest path in an arbitrary network was what theo-



A gadget for finding the line that best fits a series of data points

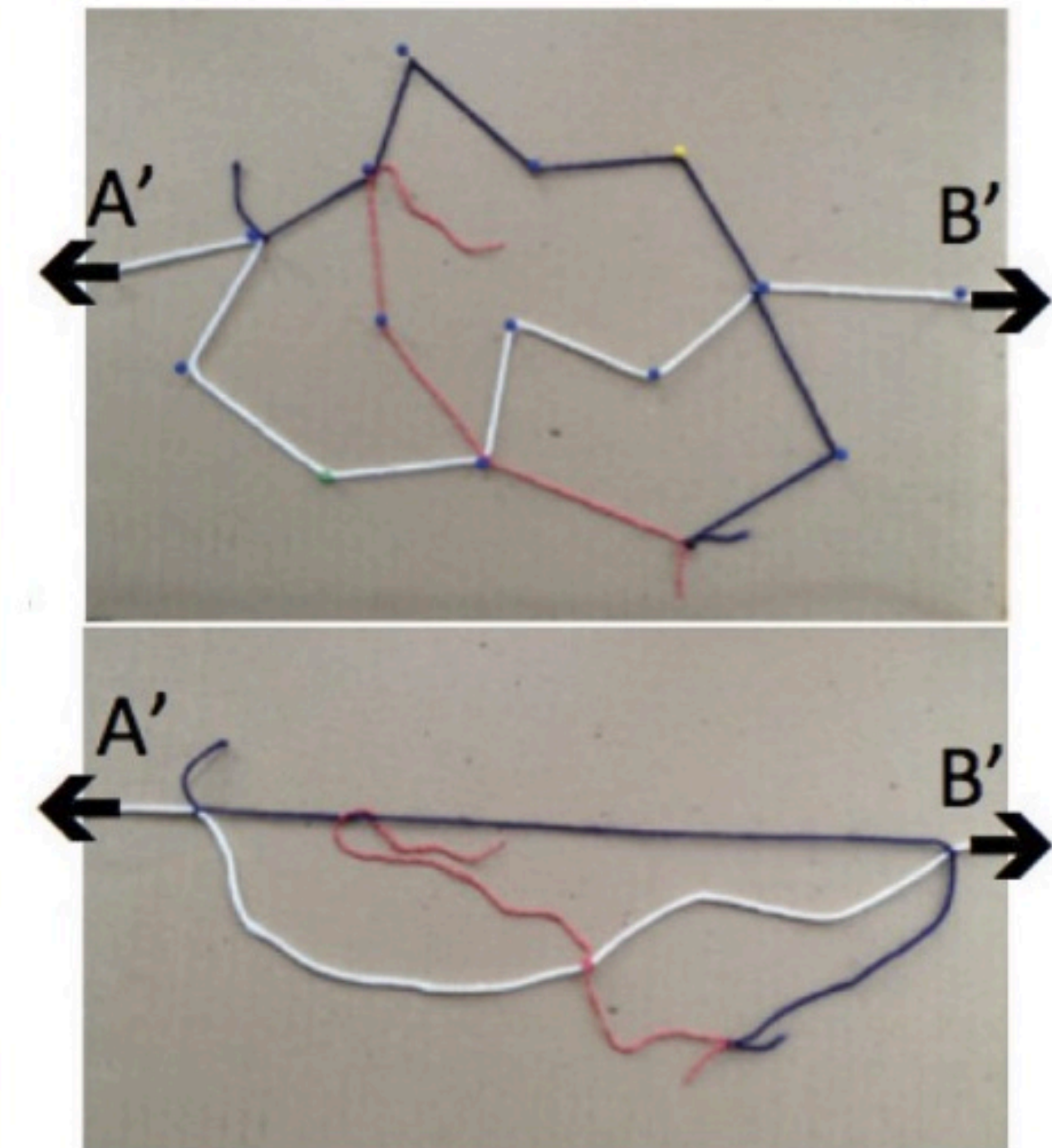
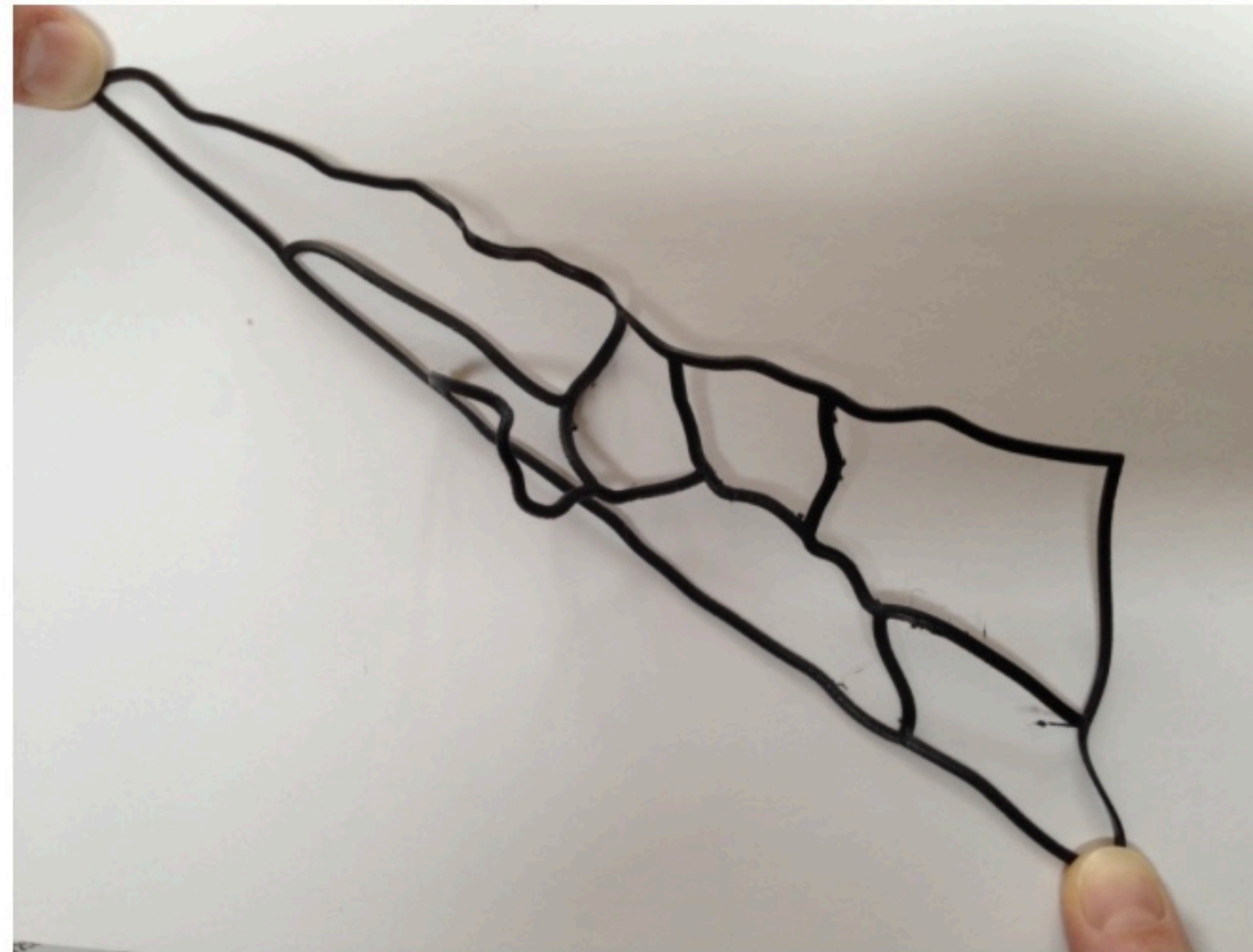
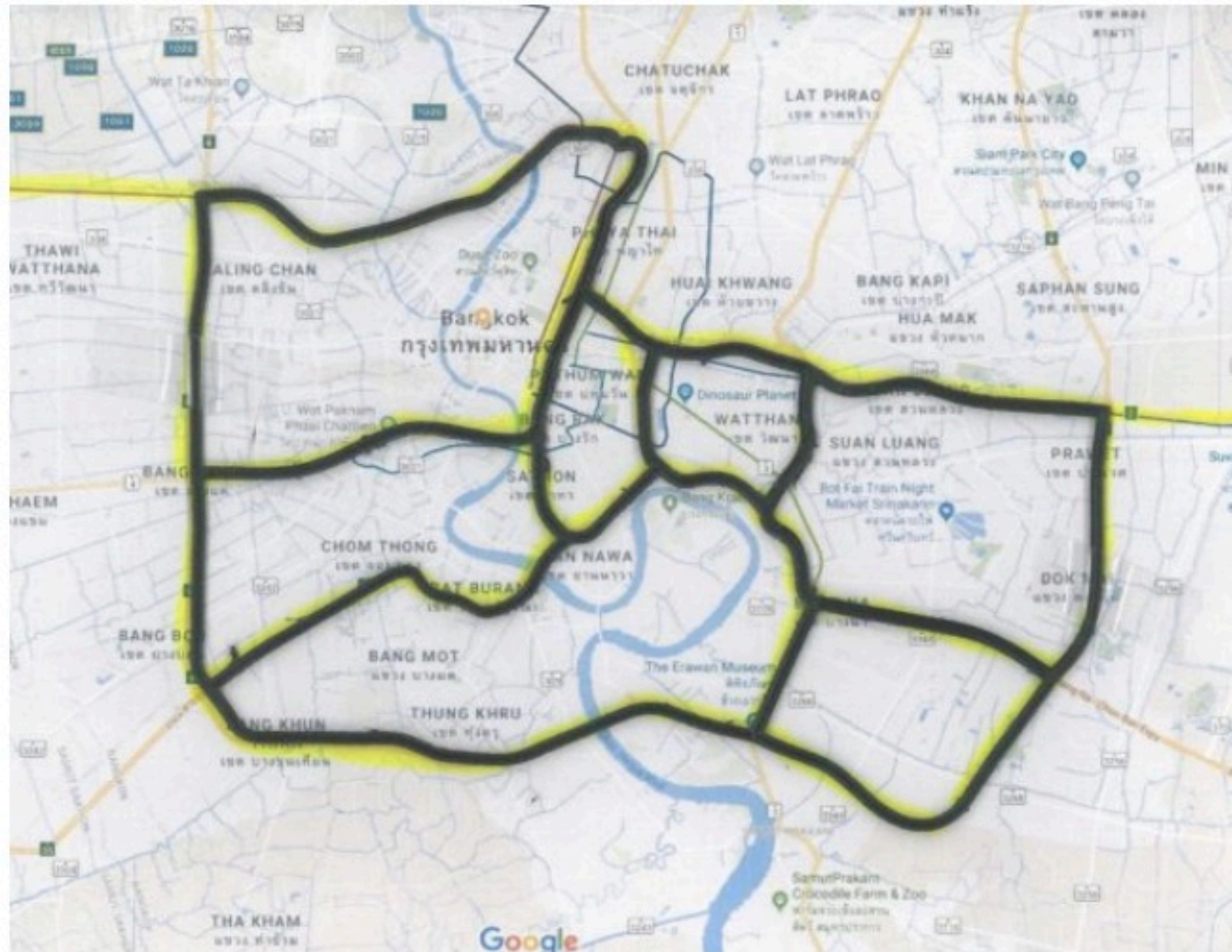
18

© 1985 SCIENTIFIC AMERICAN, INC.

Christian Freksa

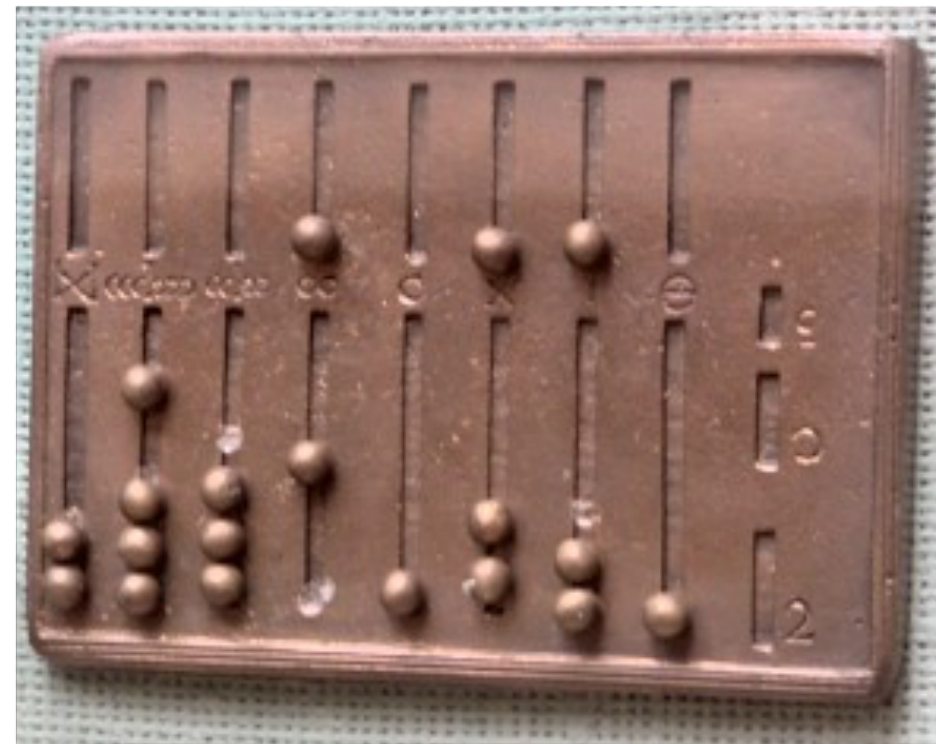
Spaghettisort und andere analoge Verfahren

2018 – Solving the Shortest Route Problem with a 3D Printer

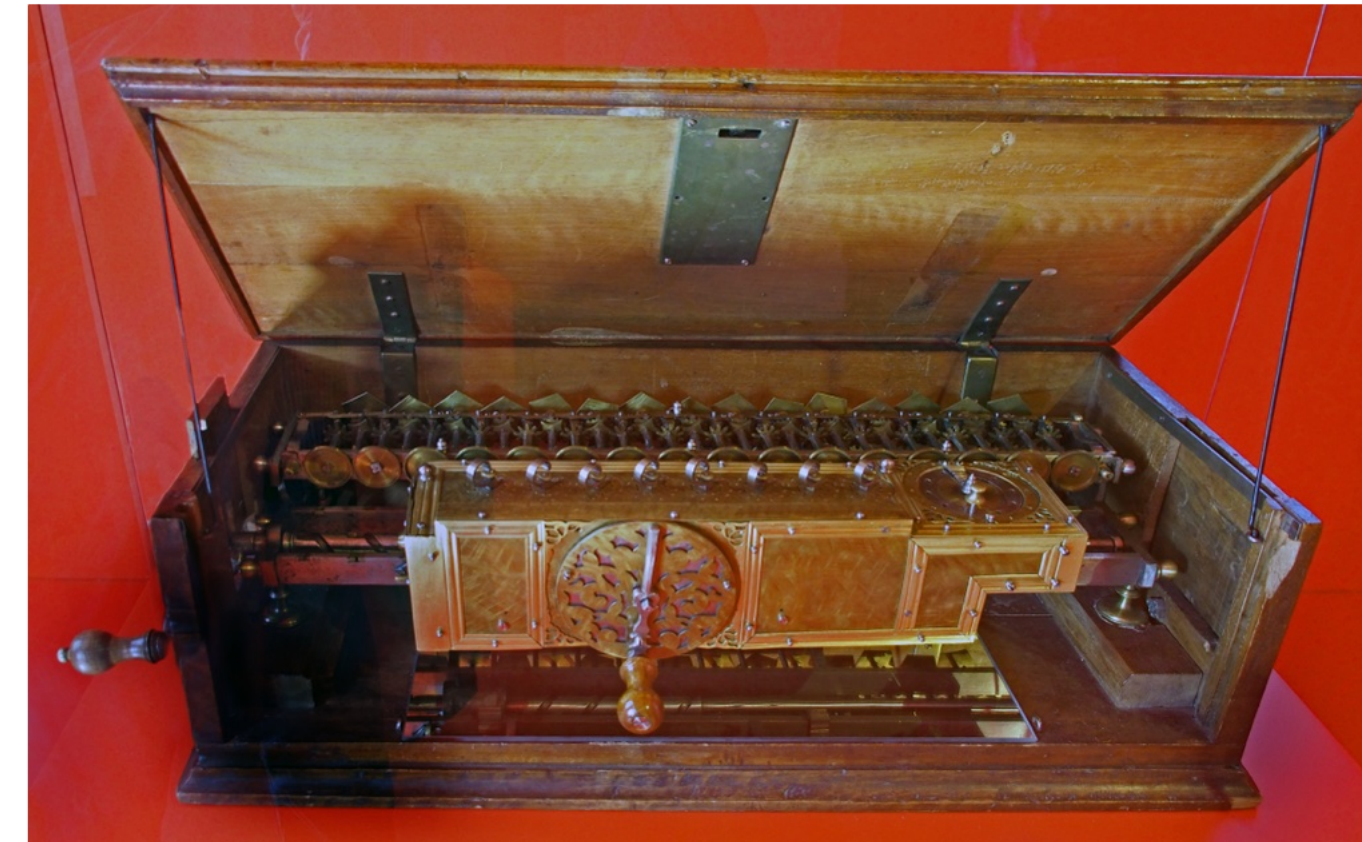


Christian Freksa

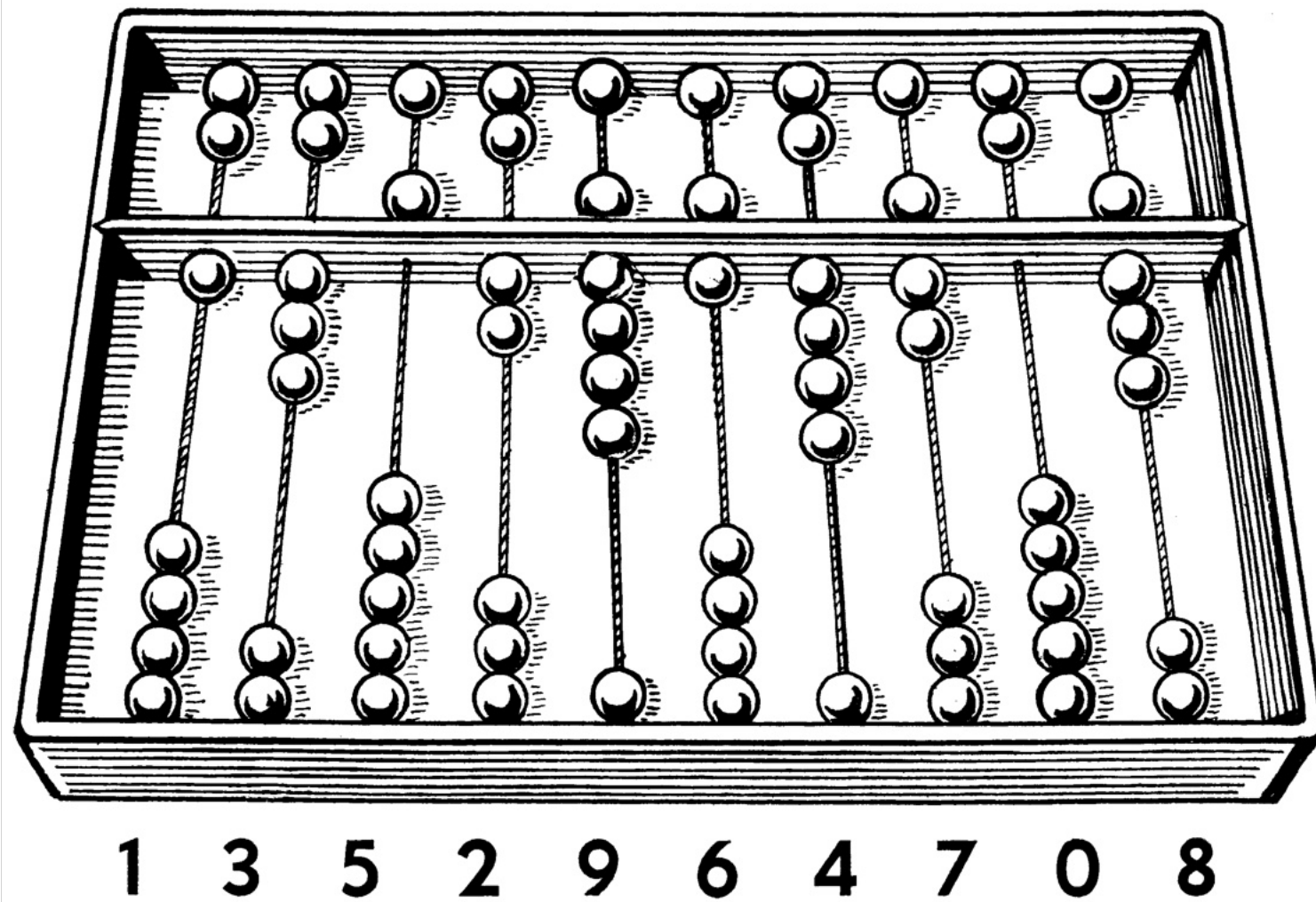
Zählen und Rechnen



Mike Cowlshaw

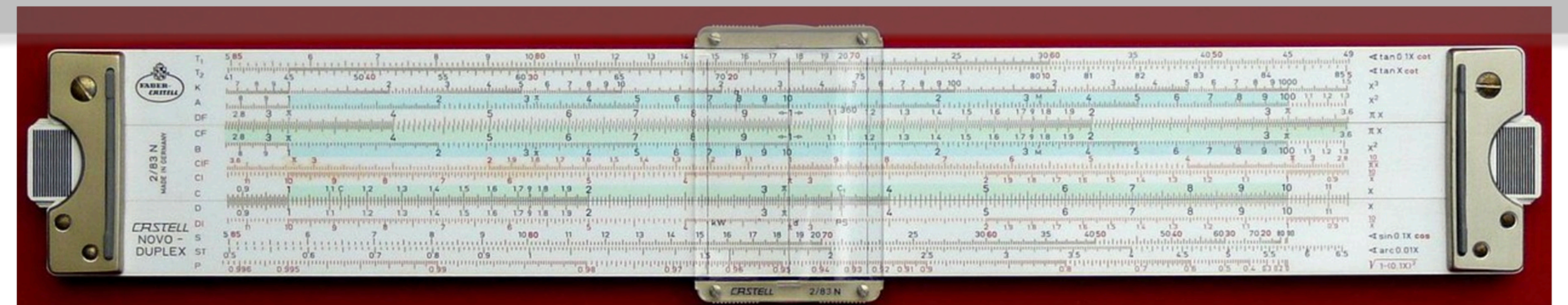


CC Museum Herrenhausen



„Es ist unwürdig, die Zeit von hervorragenden Leuten mit knechtischen Rechenarbeiten zu verschwenden, weil bei Einsatz einer Maschine auch der Einfältigste die Ergebnisse sicher hinschreiben kann.“

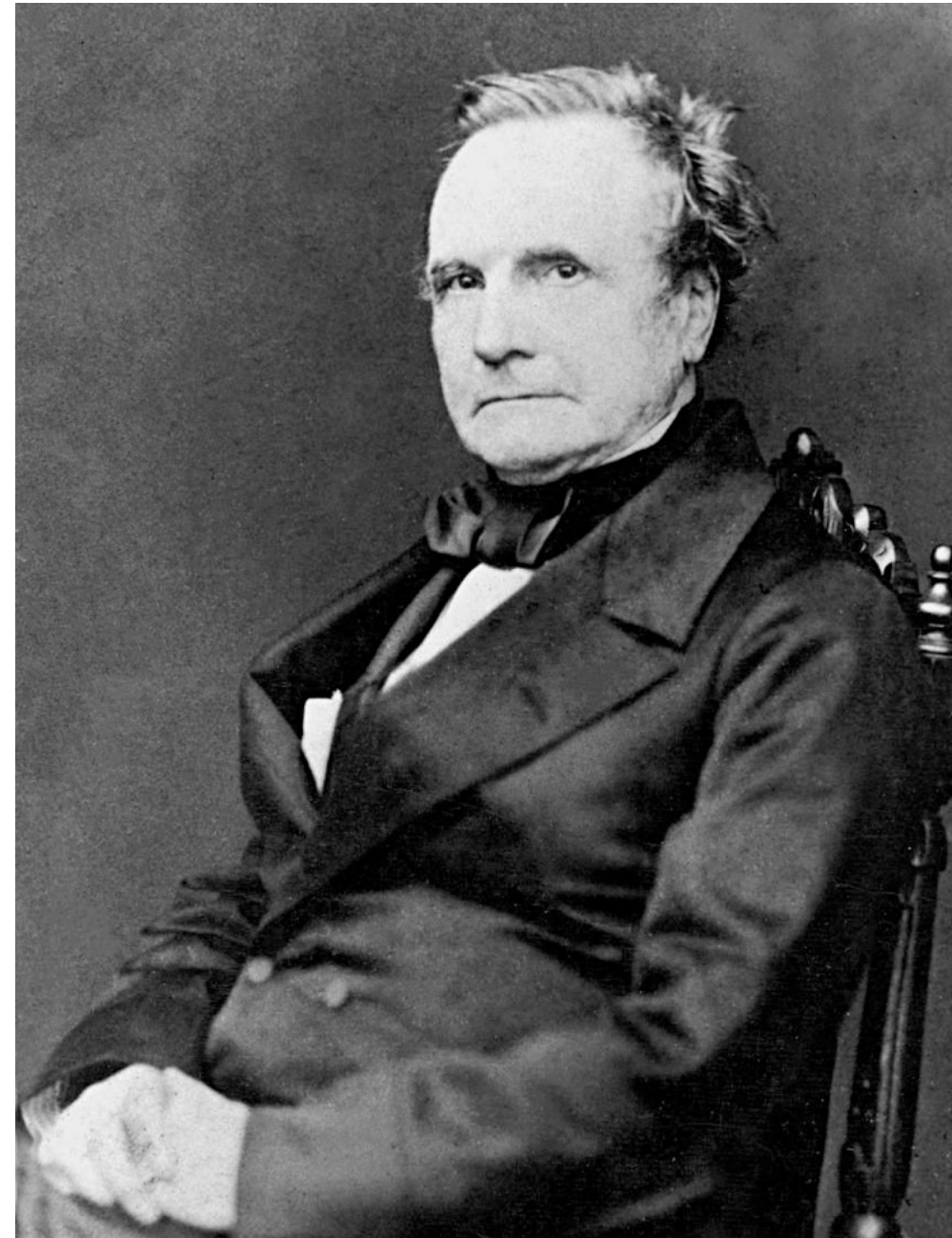
– GOTTFRIED WILHELM LEIBNIZ



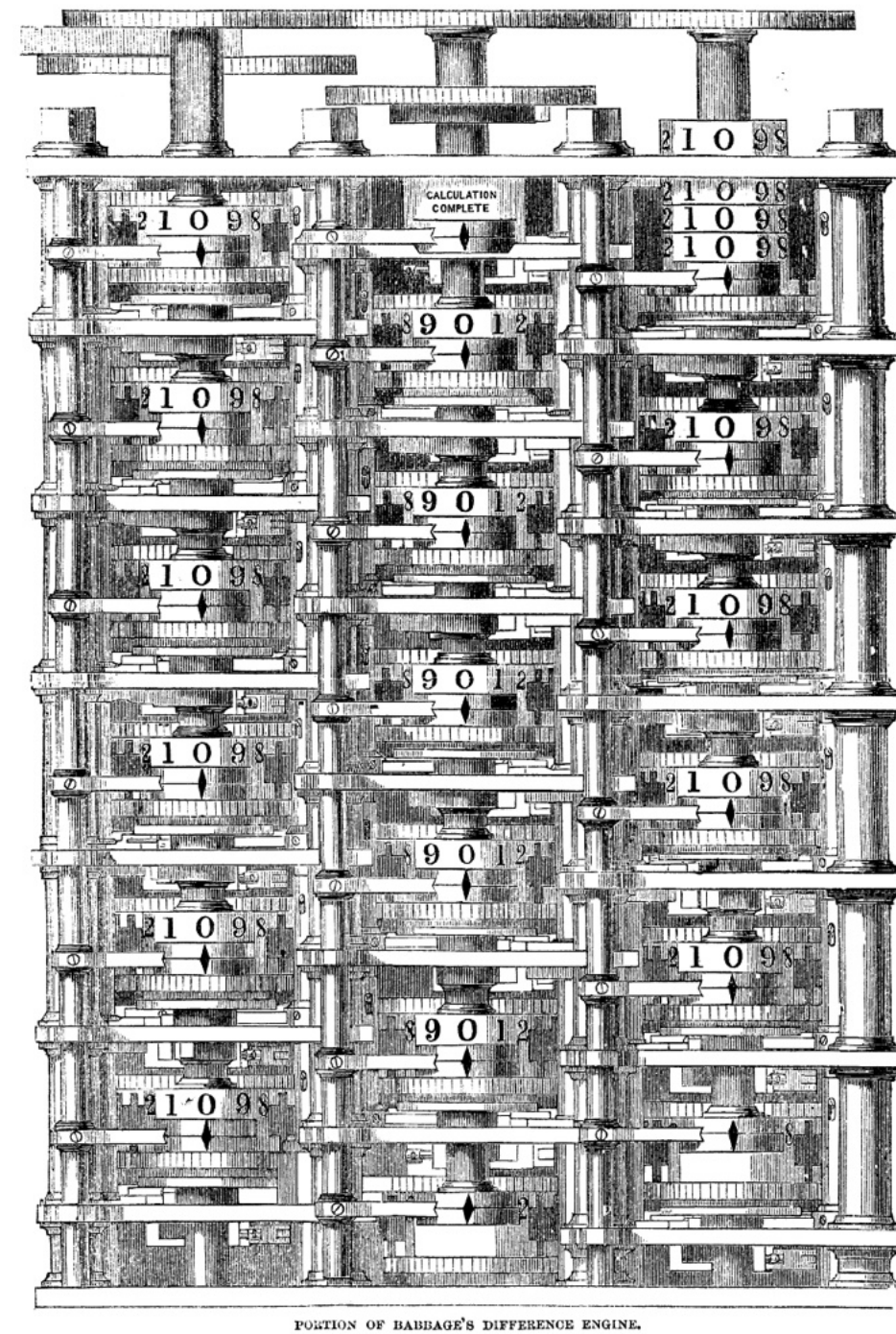
Herausforderungen

1. Realisierbarkeit
2. Skalierbarkeit
3. Genauigkeit
4. Robustheit

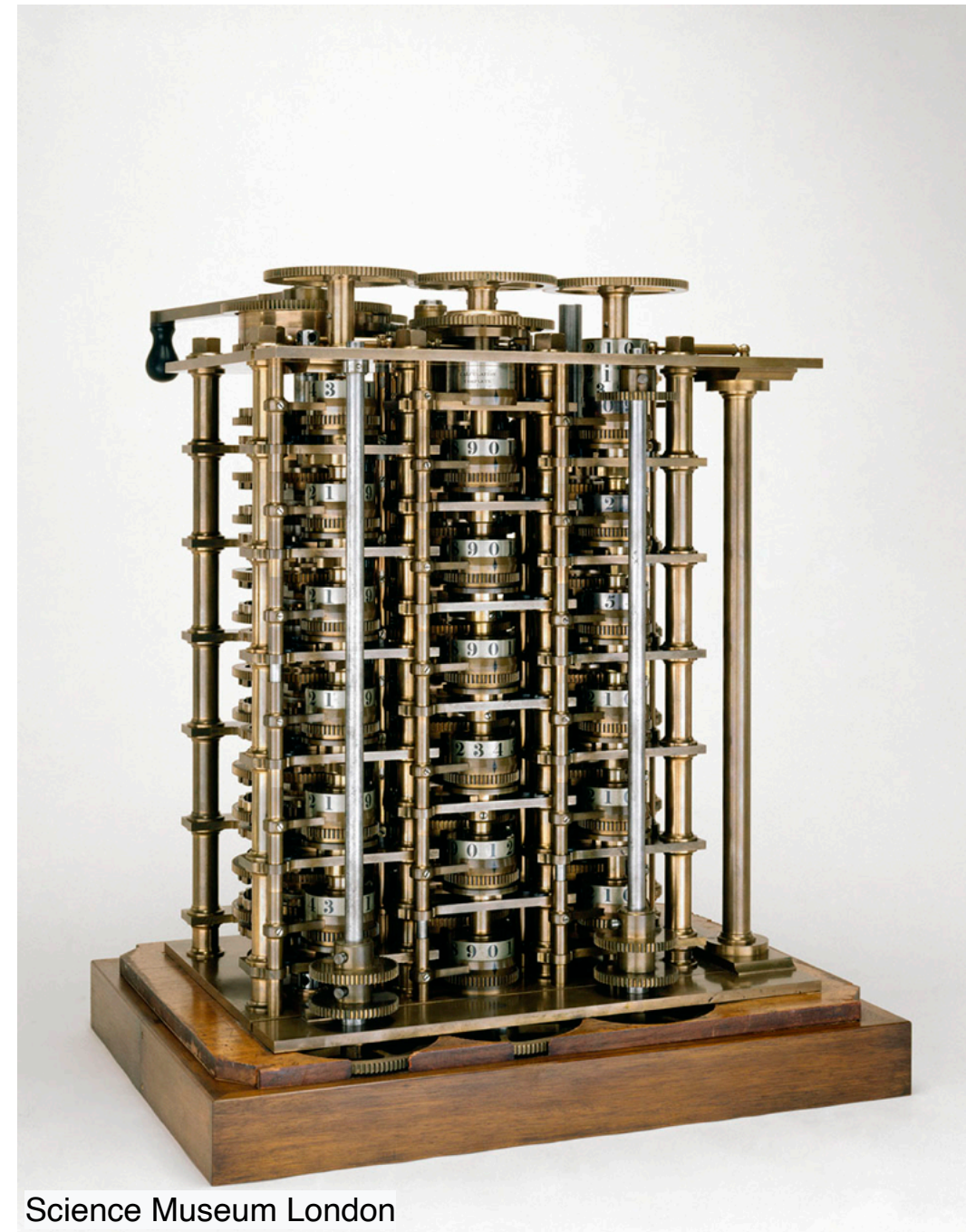
Realisierbarkeit



Charles Babbage (1791-1871)

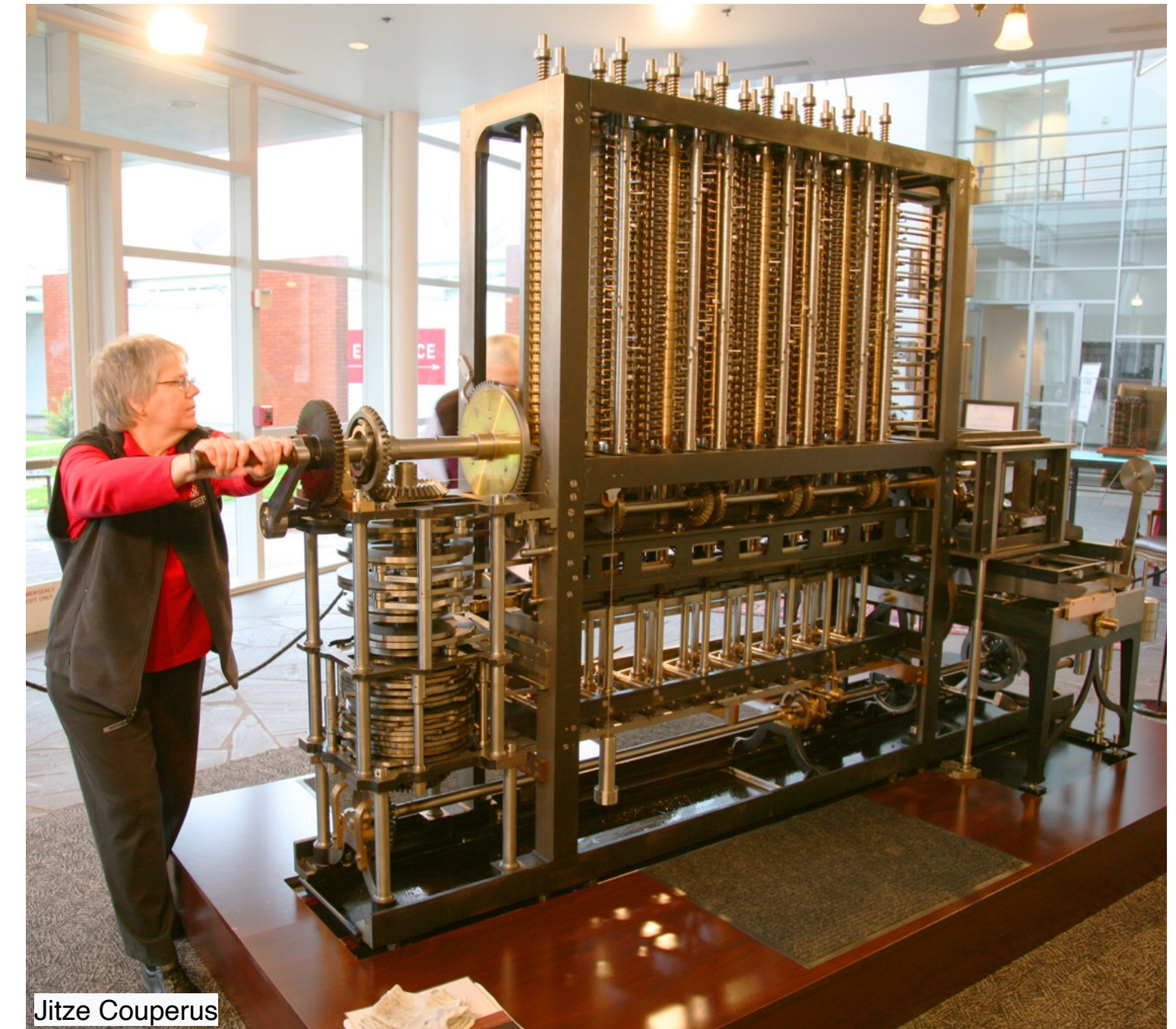


Skizze von 1822



Science Museum London

Teil-Demo von 1832



Jitze Couperus

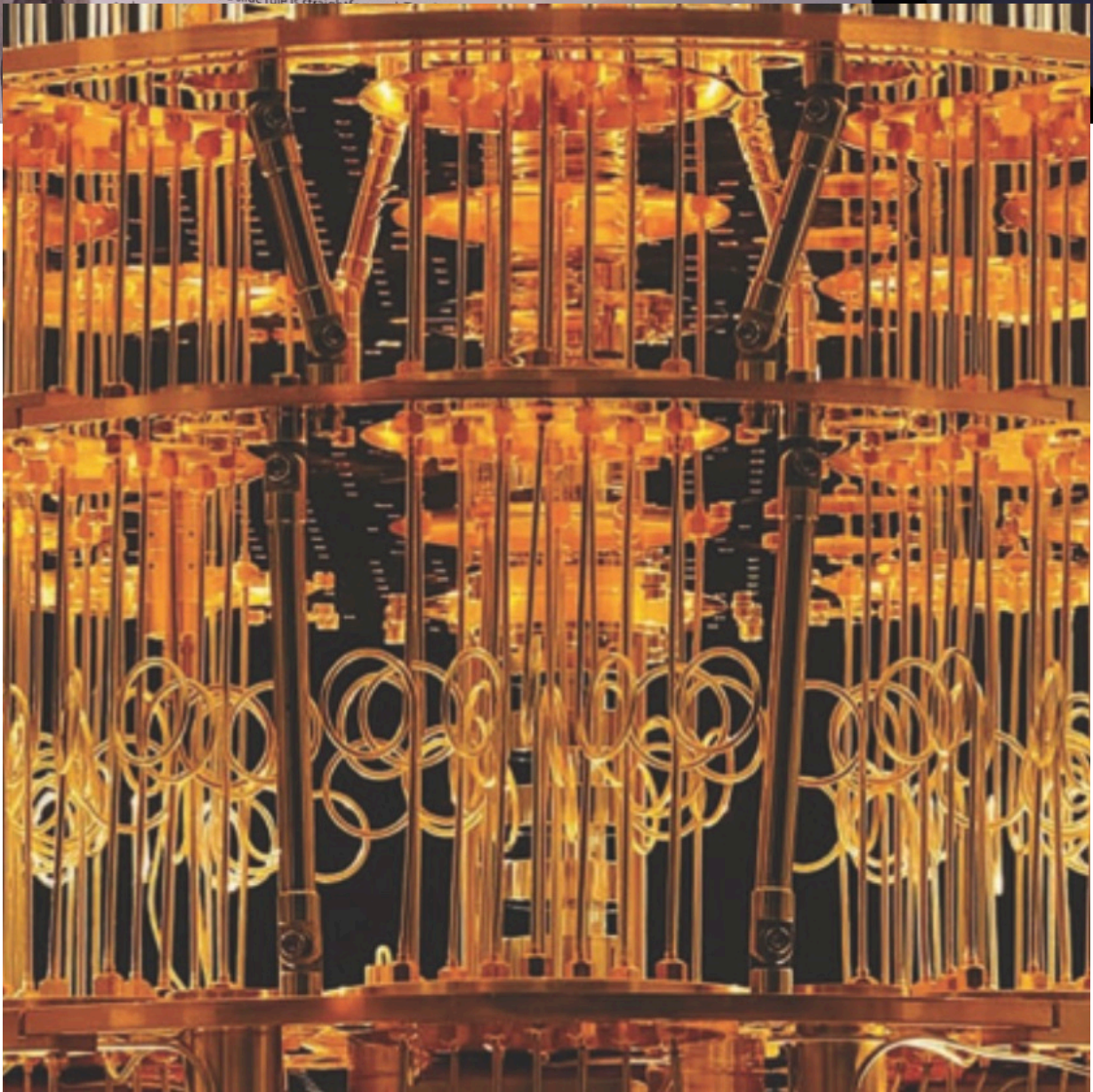
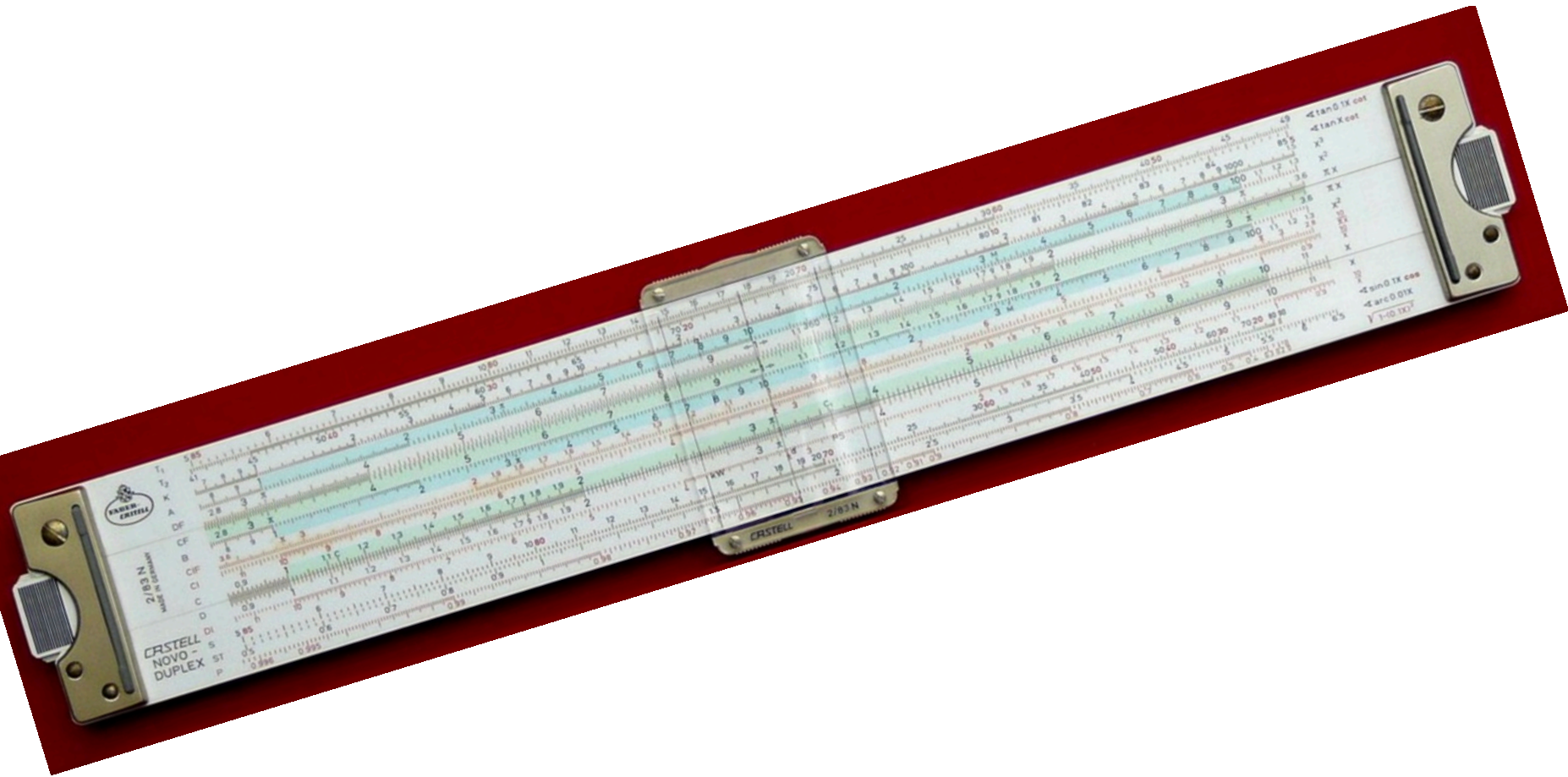
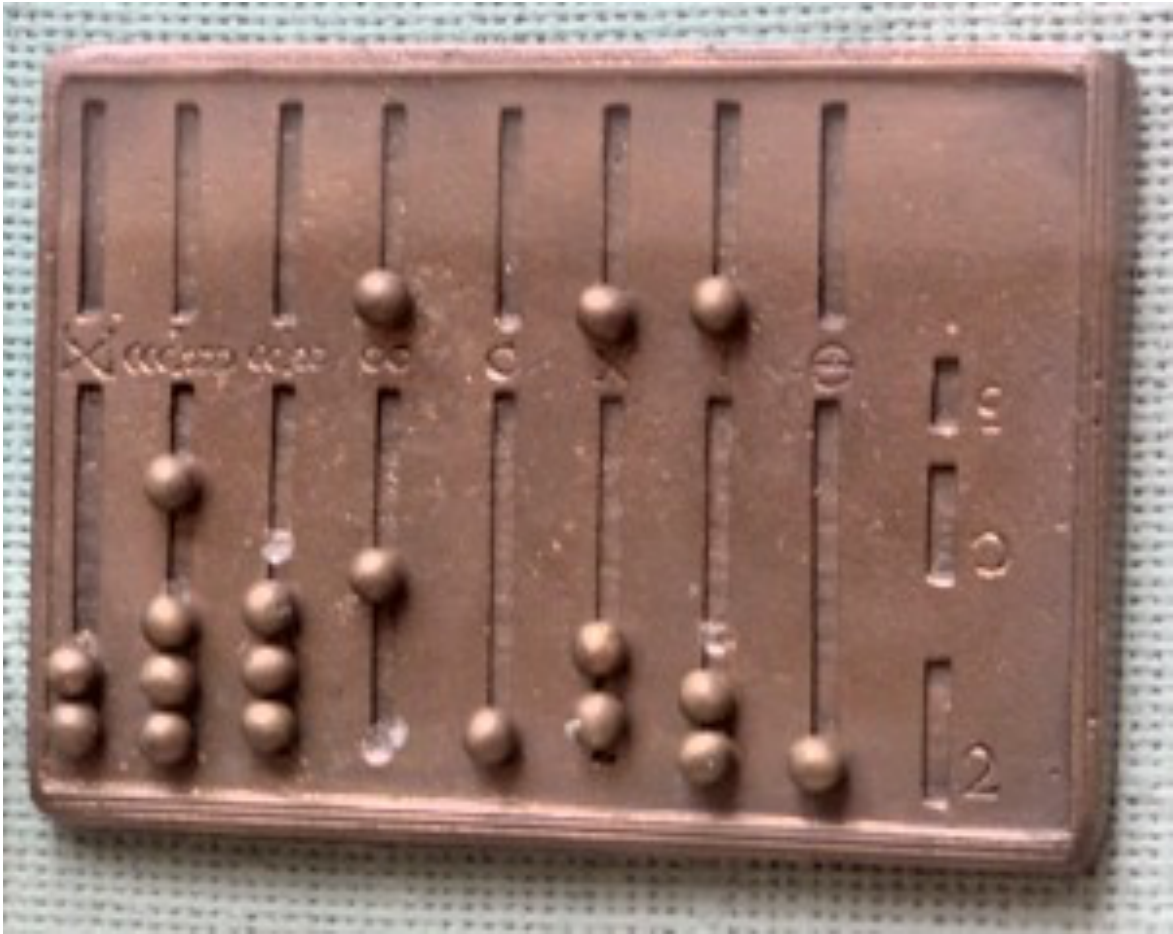
Gebaute Maschine von 1991

Regierungsauftrag: 1823

Gekündigt: 1842

Gesamtausgaben nach heutigem Wert:
2.400.000€

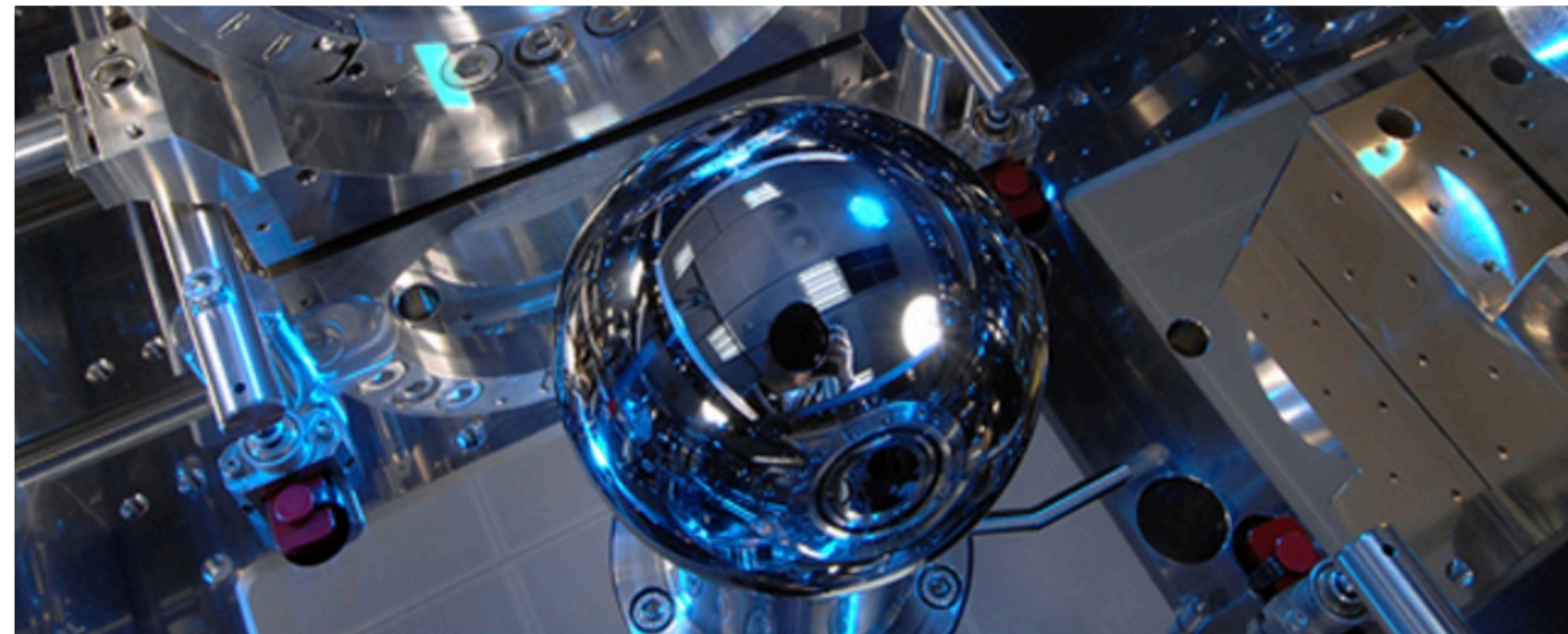
Rechnen mit Physik: Analogrechner



Skalierbarkeit



6,022 140 76 · 10²³ Teilchen



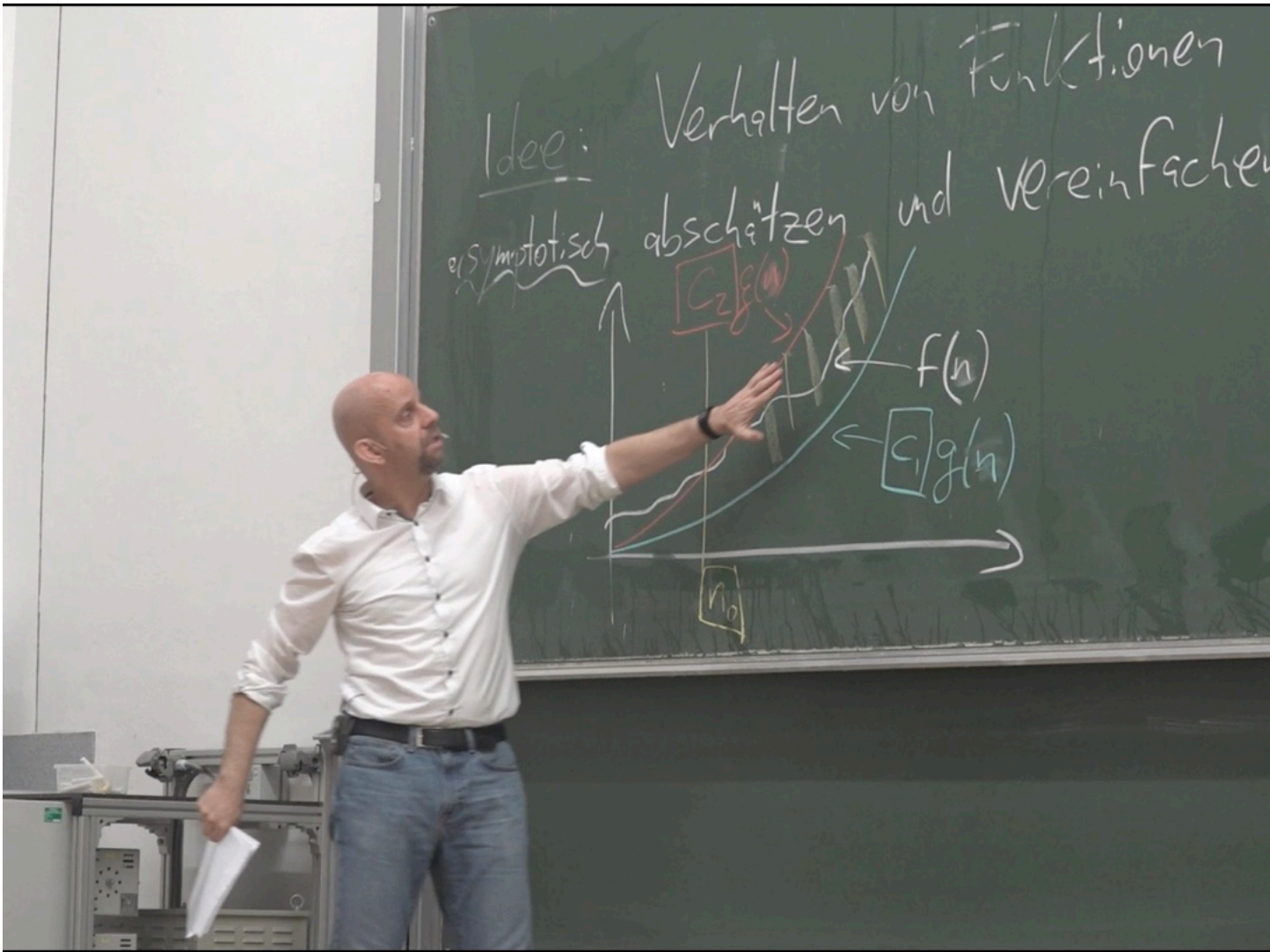
[Forschung zum neuen SI](#) > [PTB-Experimente](#) > Kilogramm und Mol: Atome zählen

Kilogramm und Mol: Atome zählen



1.000.000.000 Teilchen pro Sekunde
1.000 Zähler
über 19.000 Jahre

Komplexität



$$\Theta(n \log n)$$

Ein algorithmisches Problem

Gegeben: n Puzzleteile

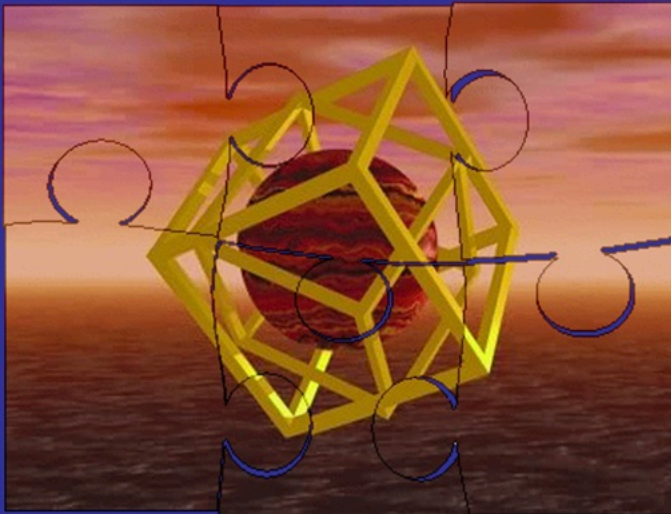
Einfach so: $O(n^2)$

Für n=6:	21
Für n=100:	5.050
Für n=5000:	12.502.500

Raffiniert sortiert:

$O(n \log n)$

Gesucht: Eine systematische Methode zum Puzzeln



Komplexität

Einfach so:

$$O(n^2)$$

Für $n=6$: 21

Für $n=100$: 5.050

Für $n=5000$: 12.502.500

Algorithmisches Problem

Teile

Einfach so: $O(n^2)$

Für $n=6$: 21

Für $n=100$: 5.050

Für $n=5000$: 12.502.500

Raffiniert sortiert:

$$O(n \log n)$$

Automatische Methode zum Puzzeln

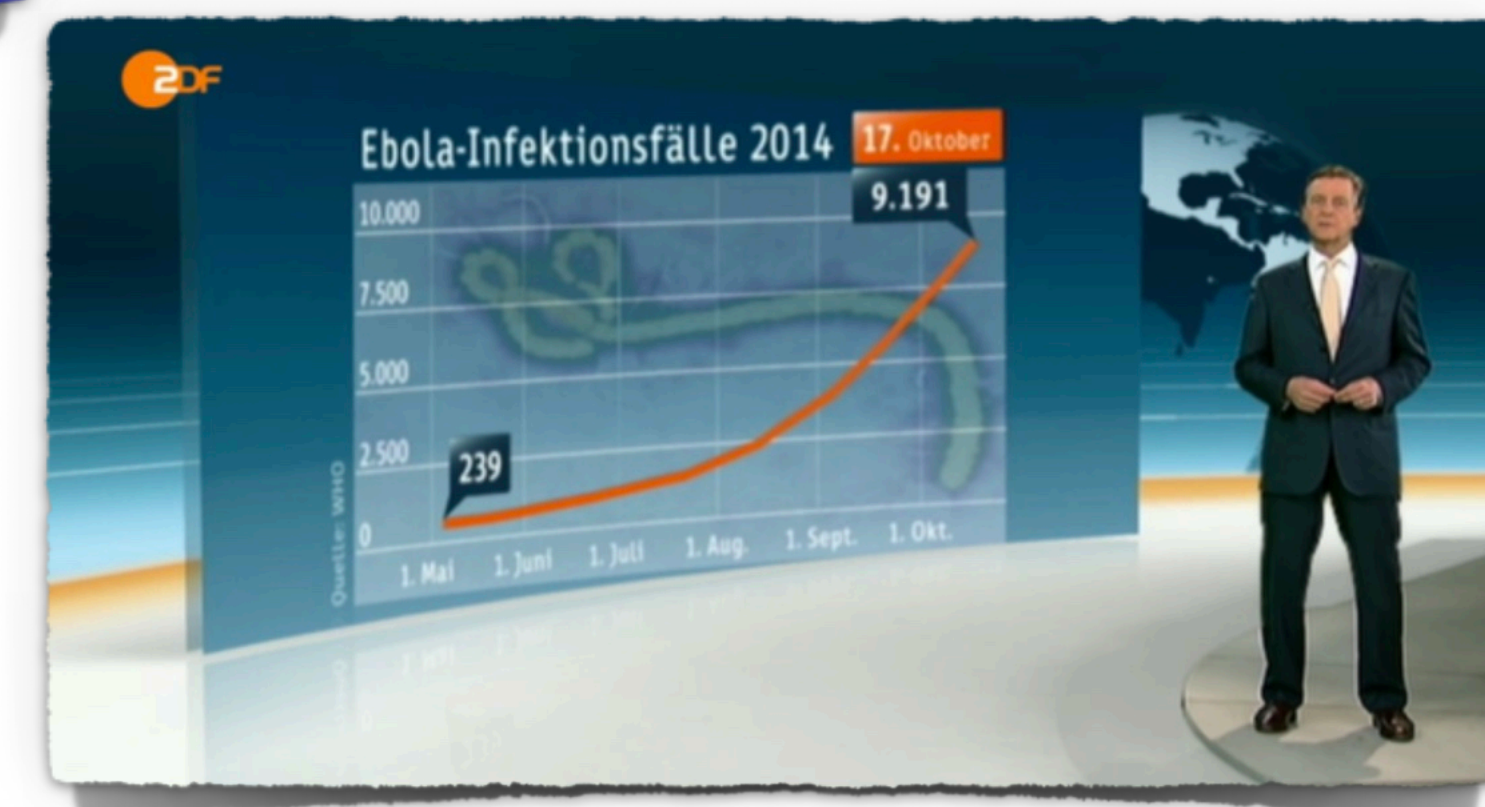
$$\Theta(n \log n)$$

$$\Theta(n^2)$$

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

$$\Theta(2^n)$$

$$2^{2n} = 2^n \cdot 2^n$$

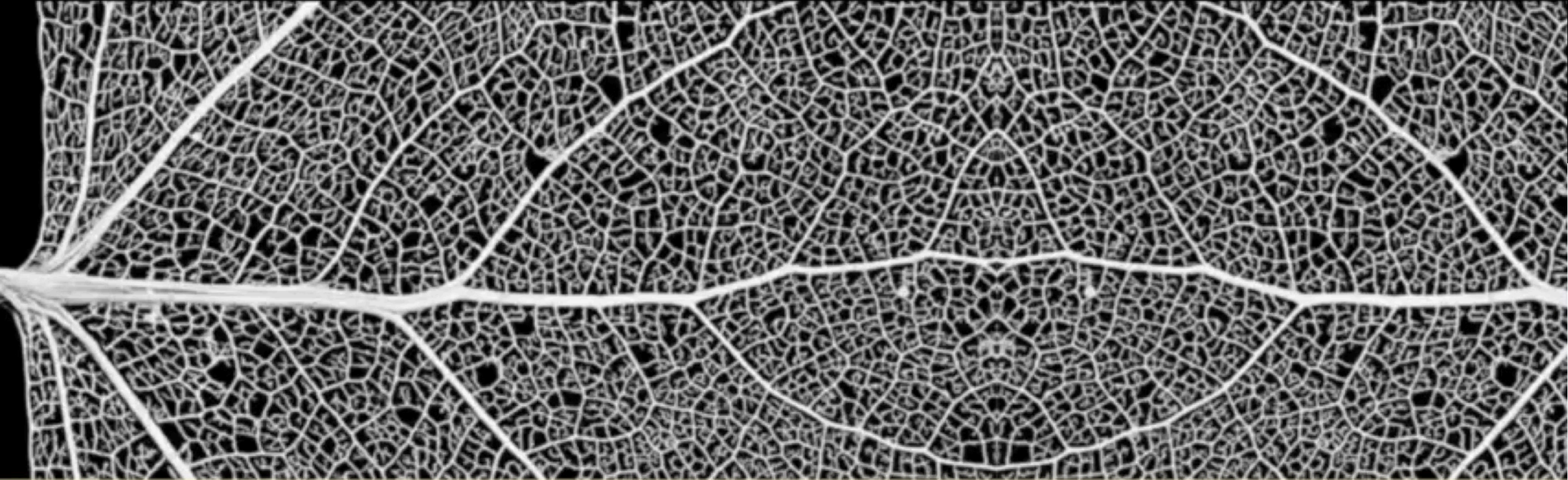


5 Architekturen

5.1 Particle Computation

Mechanische Computer





UNIVERSITY of
HOUSTON



Technische
Universität
Braunschweig

Anwendungen

Targeted Drug Delivery: Algorithmic Methods for Collecting a Swarm of Particles with Uniform, External Forces

Aaron T. Becker^{*1}, Sándor P. Fekete^{*2}, Li Huang^{*1}, Phillip Keldenich^{*2},
Linda Kleist^{*2}, Dominik Krupke^{*2}, Christian Rieck^{*2}, Arne Schmidt^{*2}

Abstract— We investigate algorithmic approaches for targeted drug delivery in a complex, maze-like environment, such as a vascular system. The basic scenario is given by a large swarm of micro-scale particles (“agents”) and a particular target region (“tumor”) within a system of passageways. Agents are too small to contain on-board power or computation and are instead controlled by a global external force that acts uniformly on all particles, such as an applied fluidic flow or electromagnetic field. The challenge is to deliver all agents to the target region with a minimum number of actuation steps. We provide a number of results for this challenge. We show that the underlying problem is NP-hard, which explains why previous work did not provide provably efficient algorithms. We also develop a number of algorithmic approaches that greatly improve the worst-case guarantees for the number of required actuation steps. We evaluate our algorithmic approaches by a number of simulations, both for deterministic algorithms and searches supported by deep learning, which show that the performance is practically promising.

I. INTRODUCTION

A crucial challenge for a wide range of vital medical problems, such as the treatment of cancer, localized infections and inflammation, or internal bleeding is to deliver active substances to a specific location in an organism. The traditional approach of administering a sufficiently large supply of

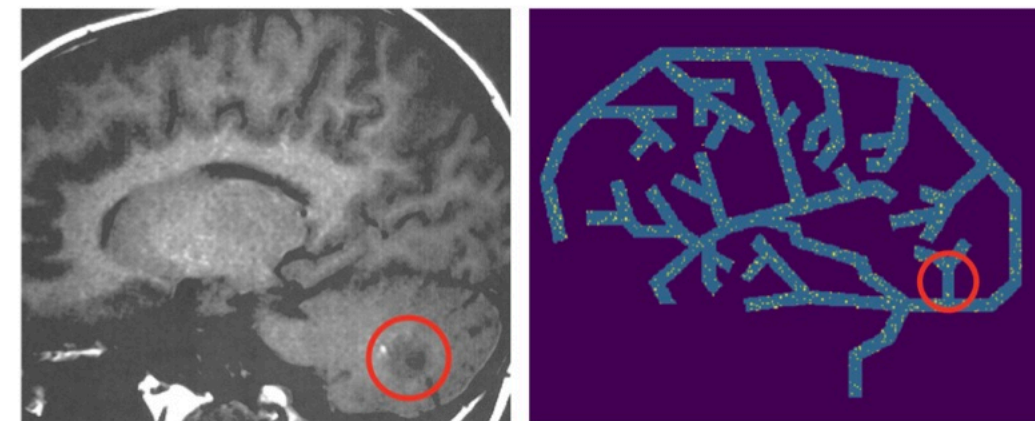


Fig. 1. (Left) An MRI image of a brain tumor (marked by the red circle), located in the cerebellum. (Right) How can the swarm of particles (indicated by yellow dots) be delivered to the target region?

A promising alternative is offered by employing a global external force, e.g., a fluidic flow or an electromagnetic field. When such a force is applied, all particles move in the same direction by the distance, unless they are blocked by obstacles in their way. While this makes it possible to move all particles at once, it introduces the difficulty of using *uniform* forces for many particles in *different* locations with different local topology to navigate them to *one* final destination. In this paper, we investigate how this objective can be achieved with a small number of actuator steps.

Algorithmica (2020) 82:165–187
<https://doi.org/10.1007/s00453-018-0483-9>



Tilt Assembly: Algorithms for Micro-factories That Build Objects with Uniform External Forces

Aaron T. Becker¹  · Sándor P. Fekete²  · Phillip Keldenich²  ·
Dominik Krupke²  · Christian Rieck²  · Christian Scheffer²  ·
Arne Schmidt² 

Received: 12 January 2018 / Accepted: 12 July 2018 / Published online: 3 August 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

We present algorithmic results for the parallel assembly of many micro-scale objects in two and three dimensions from tiny particles, which has been proposed in the context of programmable matter and self-assembly for building high-yield micro-factories. The underlying model has particles moving under the influence of uniform external

Anwendungen

Targeted Drug Delivery: Algorithmic Methods for Collecting a Swarm of Particles with Uniform, External Forces

Aaron T. Becker^{*1}, Sándor P. Fekete^{*2}, Li Huang^{*1}, Phillip Keldenich^{*2},
Linda Kleist^{*2}, Dominik Krupke^{*2}, Christian Rieck^{*2}, Arne Schmidt^{*2}

Abstract— We investigate algorithmic approaches for targeted drug delivery in a complex, maze-like environment, such as a vascular system. The basic scenario is given by a large swarm of micro-scale particles (“agents”) and a particular target region (“tumor”) within a system of passageways. Agents are too small to contain on-board power or computation and are instead controlled by a global external force that acts uniformly on all particles, such as an applied fluidic flow or electromagnetic field. The challenge is to deliver all agents to the target region with a minimum number of actuation steps. We provide a number of results for this challenge. We show that the underlying problem is NP-hard, which explains why previous work did not provide provably efficient algorithms. We also develop a number of algorithmic approaches that greatly improve the worst-case guarantees for the number of required actuation steps. We evaluate our algorithmic approaches by a number of simulations, both for deterministic algorithms and searches supported by deep learning, which show that the performance is practically promising.

I. INTRODUCTION

A crucial challenge for a wide range of vital medical problems, such as the treatment of cancer, localized infections and inflammation, or internal bleeding is to deliver active substances to a specific location in an organism. The traditional approach of administering a sufficiently large supply of

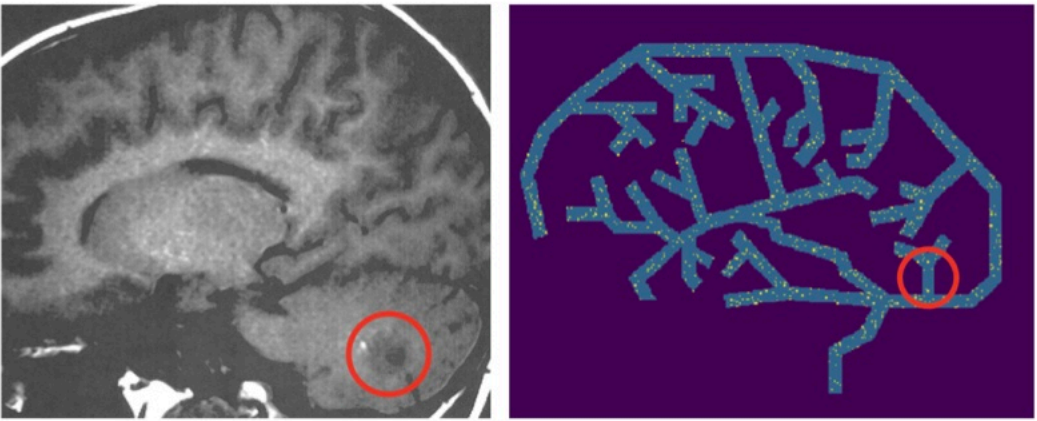


Fig. 1. (Left) An MRI image of a brain tumor (marked by the red circle), located in the cerebellum. (Right) How can the swarm of particles (indicated by yellow dots) be delivered to the target region?

A promising alternative is offered by employing a global external force, e.g., a fluidic flow or an electromagnetic field. When such a force is applied, all particles move in the same direction by the distance, unless they are blocked by obstacles in their way. While this makes it possible to move all particles at once, it introduces the difficulty of using *uniform* forces for many particles in *different* locations to navigate them to *one* final destination. In this paper, we investigate how this objective can be achieved with a small number of actuator steps.

Algorithmica (2020) 82:165–187
<https://doi.org/10.1007/s00453-018-0483-9>



Tilt Assembly: Algorithms for Micro-factories That Build Objects with Uniform External Forces



5.2 Schwarmcomputing

Search

Catalogue

People Index

Location Index

About GEPRIS

Project Details

[→ Index](#)

Projekt

SpaceAnts: Algorithmic Foundations for Constructing and Reconfiguring LargeScale Structures with Simple Robots

Applicants [→ Professor Dr. Sándor Fekete](#); [→ Professor Dr. Christian Scheffer](#)

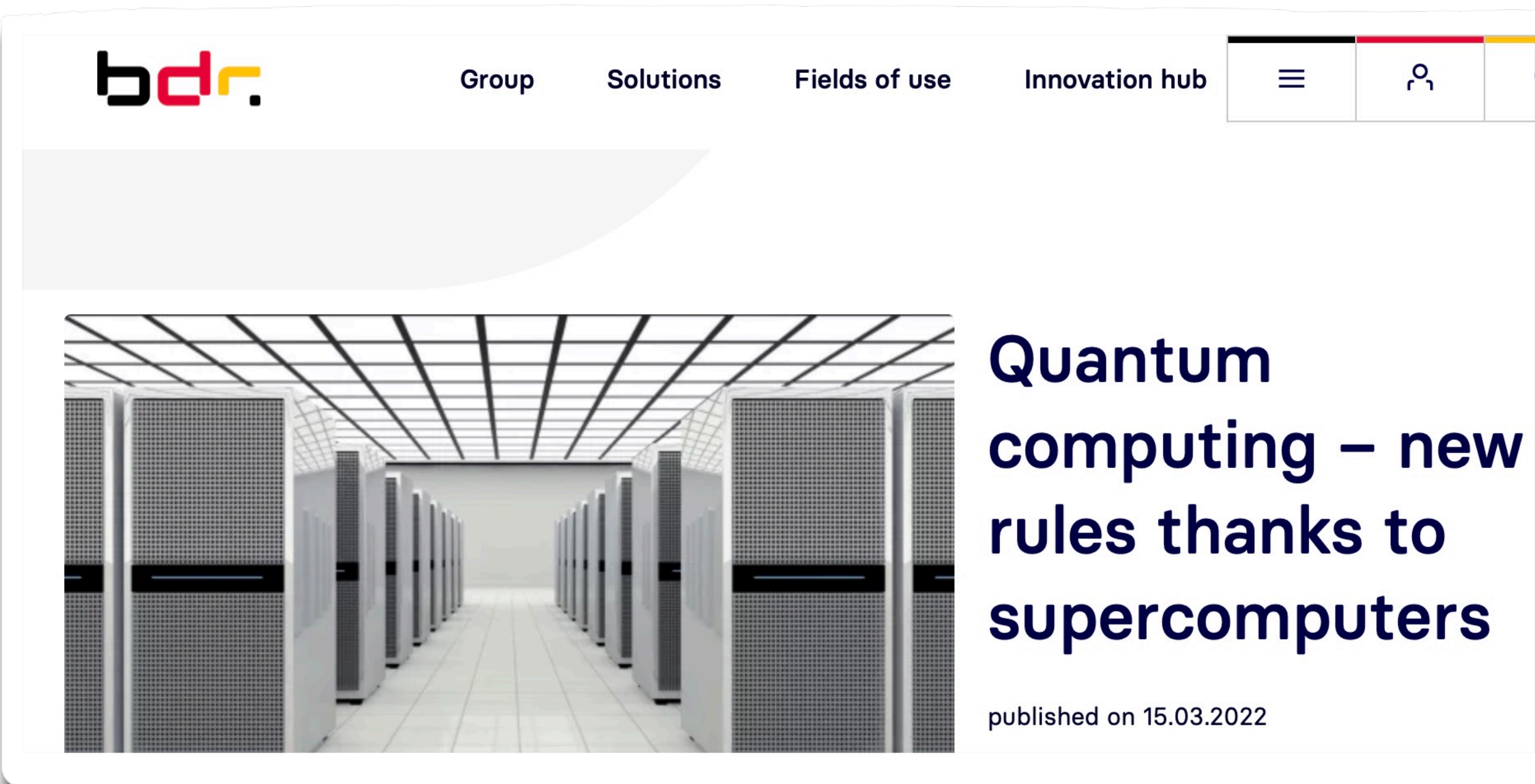
Subject Area Theoretical Computer Science

Term since 2023

Project identifier *Deutsche Forschungsgemeinschaft (DFG) - Project number 530918134*

5.3 Quantencomputer

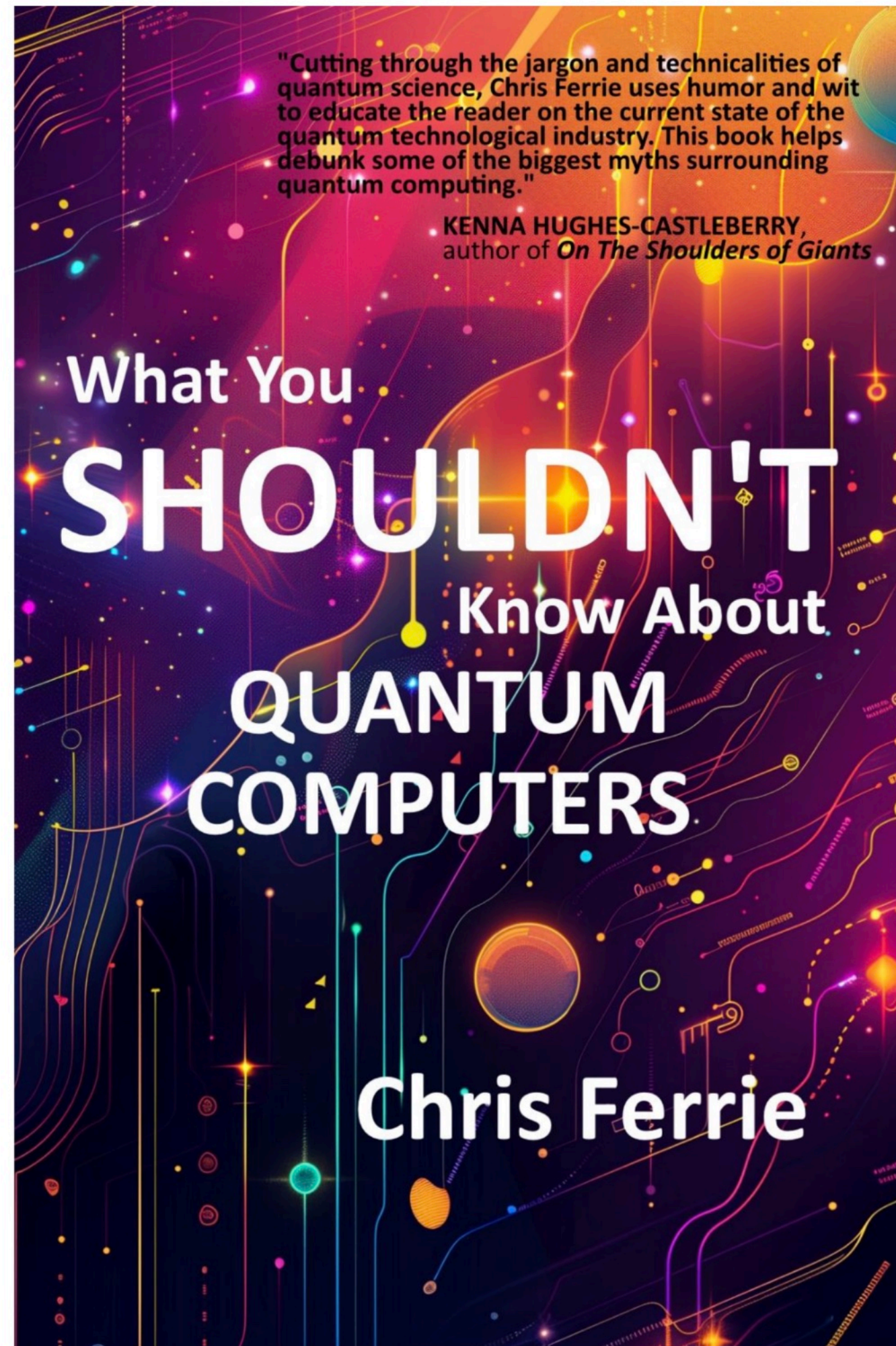
Quantencomputer



The screenshot shows a website header with the logo 'bdr.' and navigation links: 'Group', 'Solutions', 'Fields of use', and 'Innovation hub'. There are also icons for a menu and a user profile. The main content area features a large image of a server room aisle. To the right of the image, the title 'Quantum computing – new rules thanks to supercomputers' is displayed in a large, bold, dark blue font. Below the title, it says 'published on 15.03.2022'.

With these computers, everything centres around a quantum effect, i.e., the so-called superposition in combination with smart parallel information processing on the qubits of the quantum computer. Roughly speaking, superposition means that parallel computing is possible. This is precisely what distinguishes quantum computers from conventional computers that process information sequentially, one at a time, like pearls being strung on a necklace. A quantum computer, on the other hand, can calculate all values parallel, which would mean an exponential increase in performance.

Quantencomputer



Myth 2: Qubits Are
Zero And One At The
Same Time

Myth 3: Quantum
Computers Try All
Solutions At Once

Myth 5: Quantum
Computers Will
Replace Digital
Computers

Myth 6: Quantum
Computers Will Break
the Internet

Projekte



Projekt
QuBRA

Quantum Readiness for Optimization Providers Enabling the Enablers



Sándor Fekete
TU Braunschweig

[Read more](#)



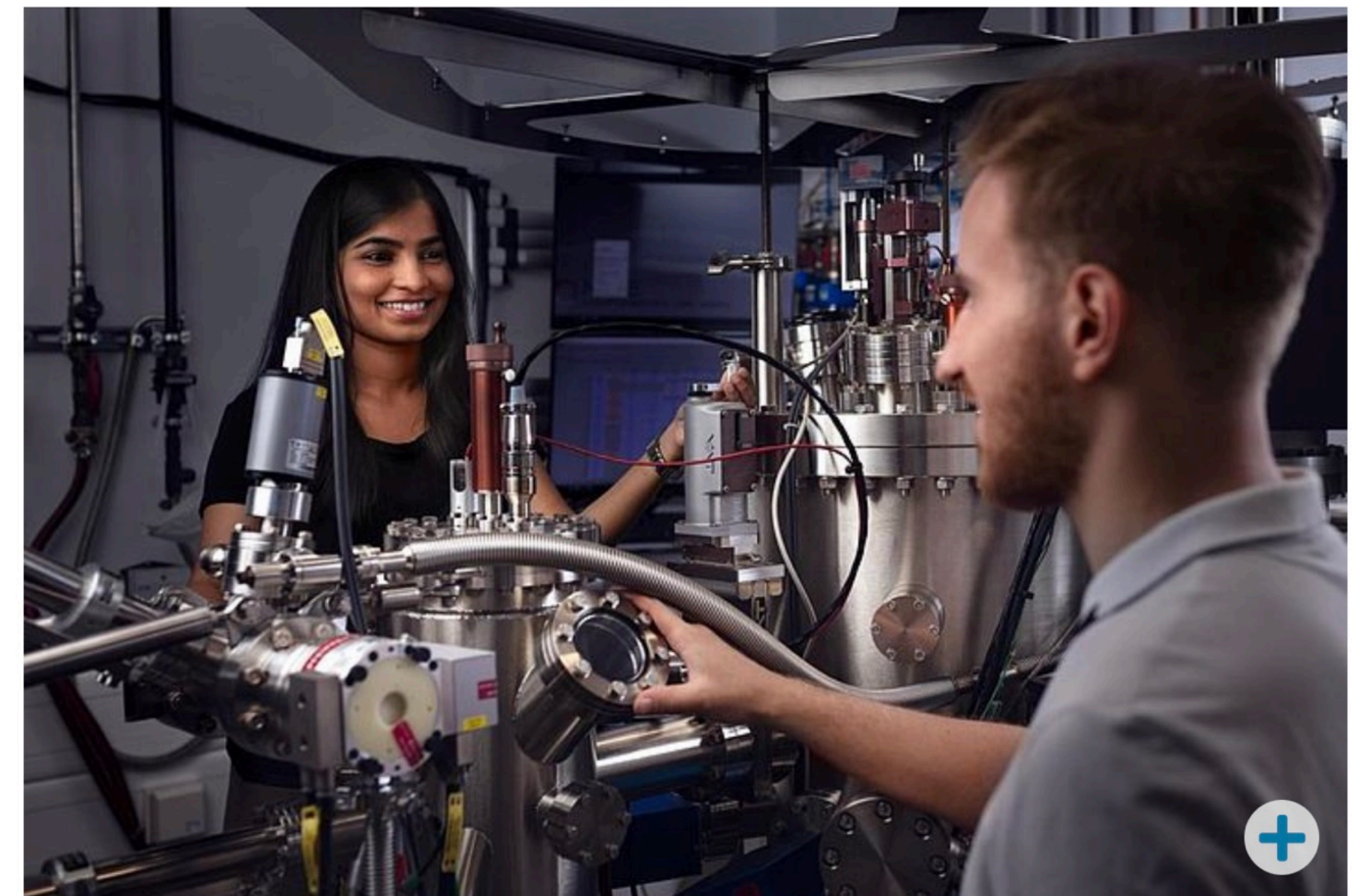
David Gross
Universität zu Köln

[Read more](#)

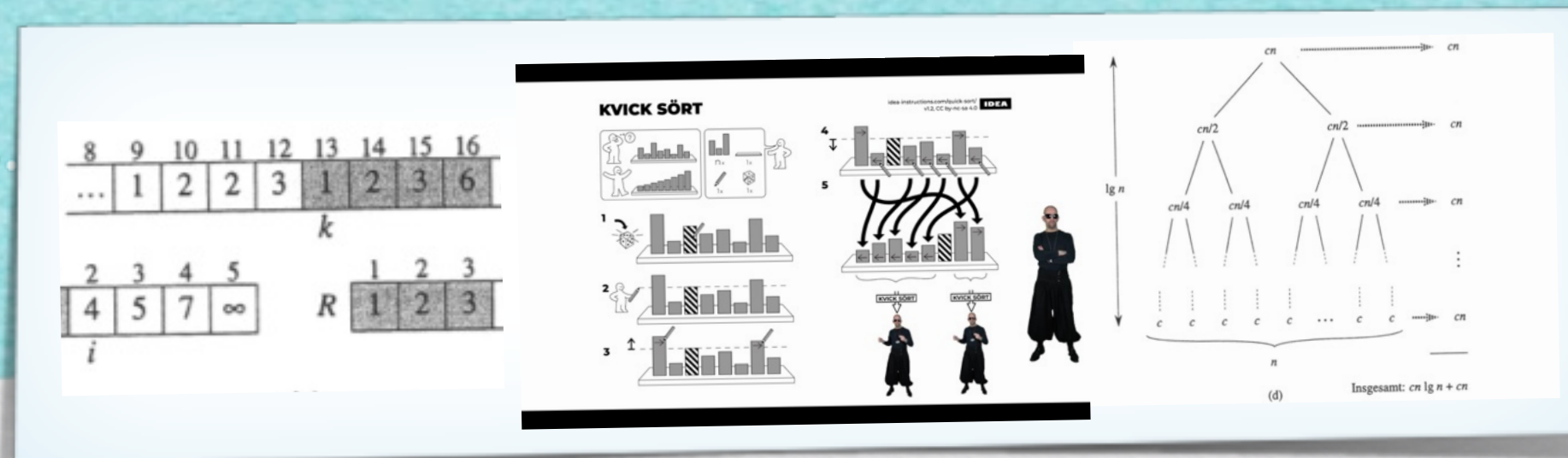


Tobias Osborne
Leibniz Universität Hannover

[Read more](#)



Quantencomputing verspricht Durchbrüche in der Entwicklung innovativer Lösungen für industrielle Herausforderungen.



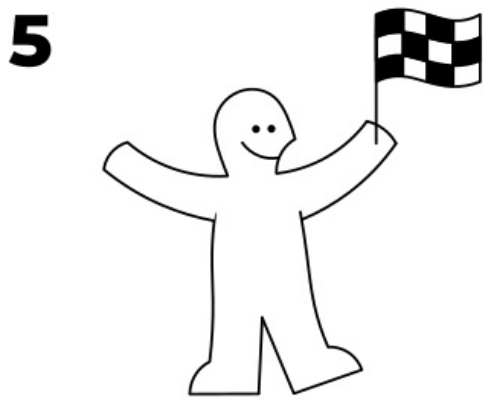
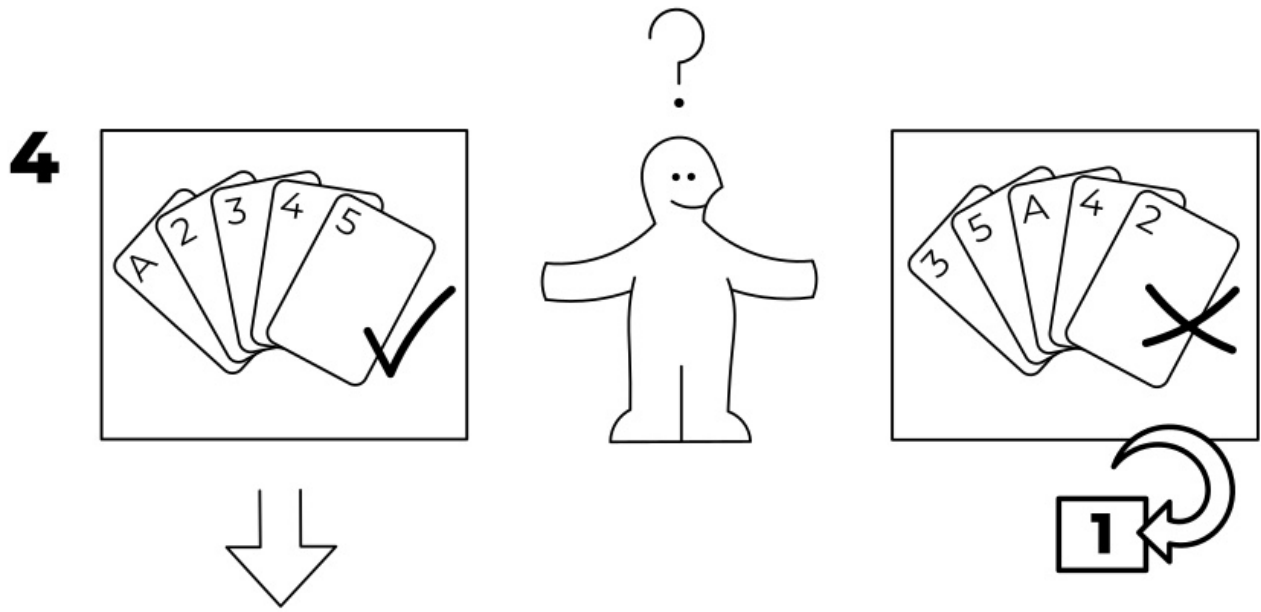
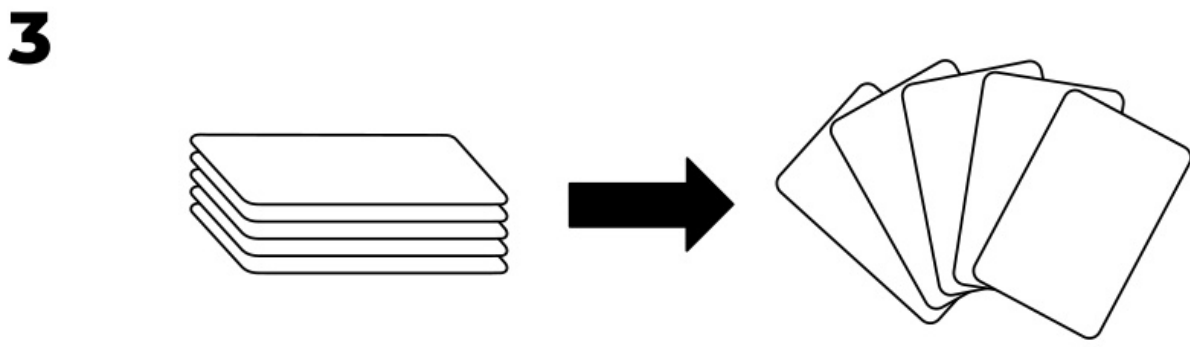
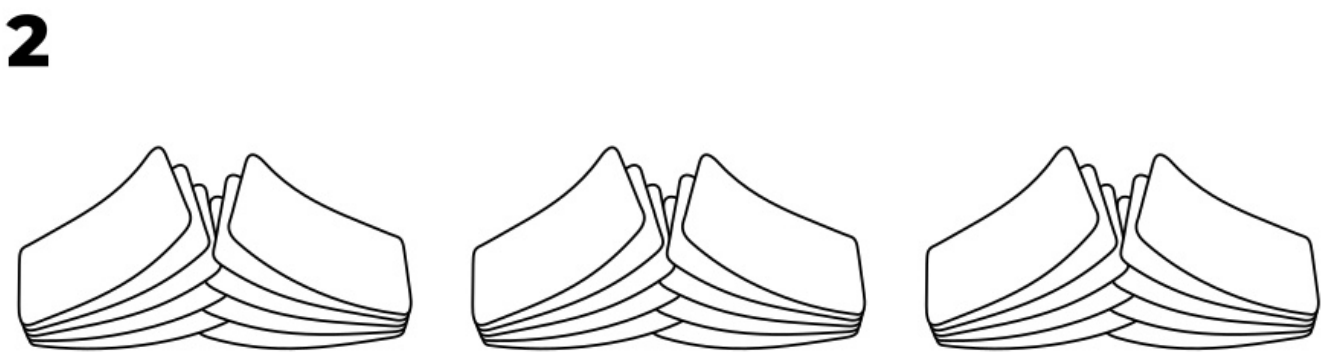
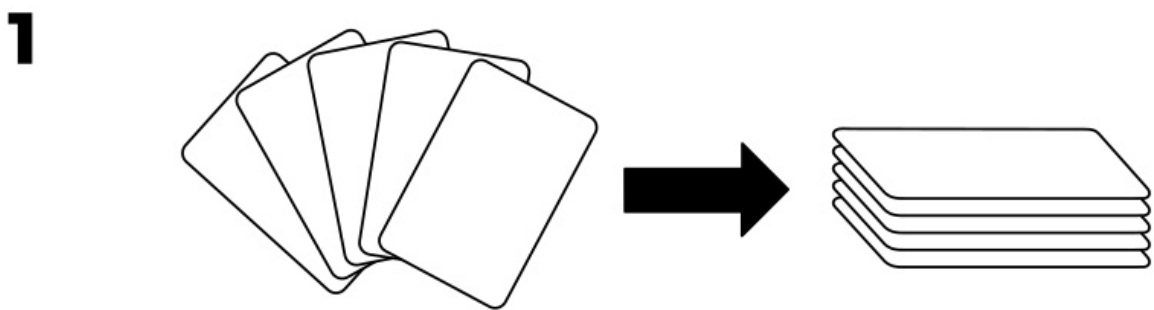
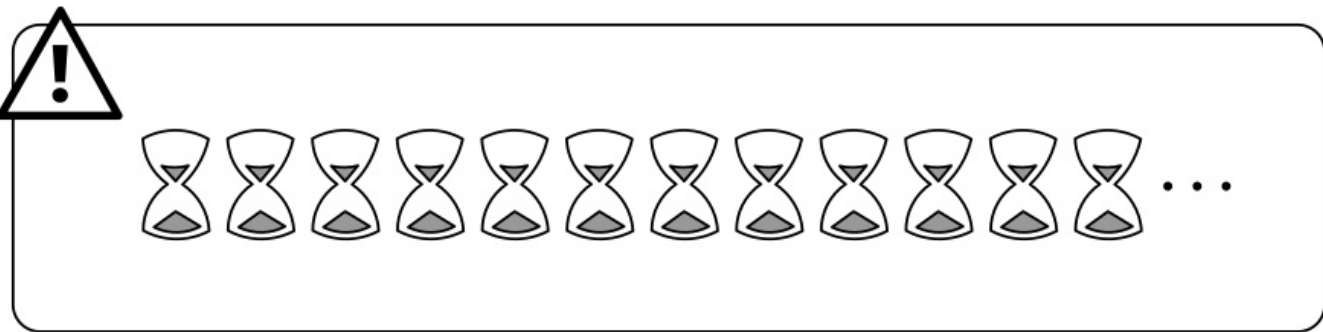
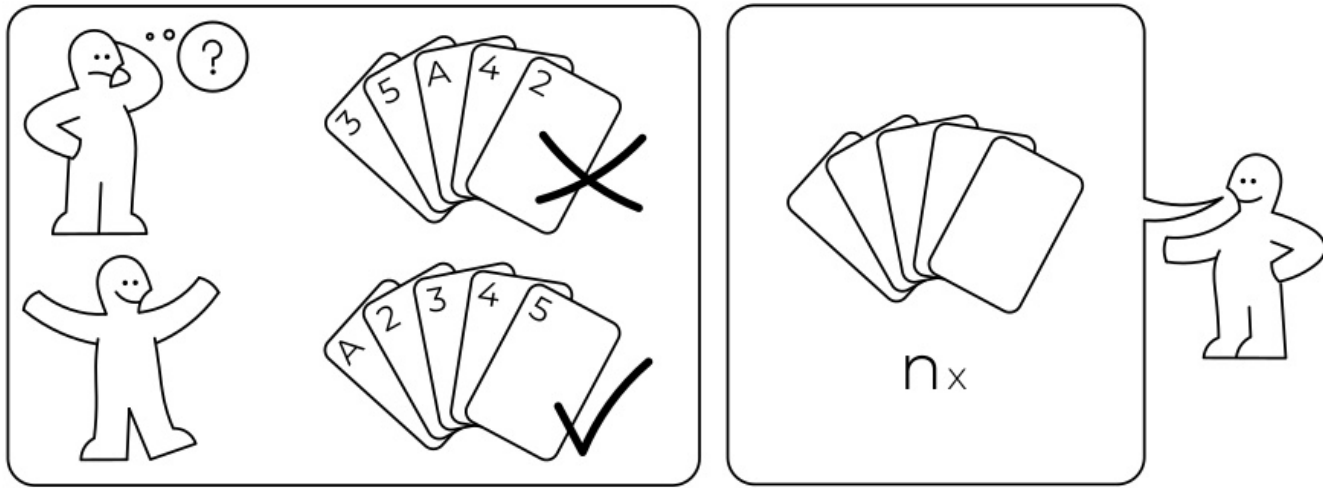
Kapitel 5.10: *Unernste Verfahren* *Algorithmen und Datenstrukturen* *WS 2025/26*

Prof. Dr. Sándor Fekete

Bogosort

BOGO SÖRT

idea-instructions.com/bogo-sort/
v1.2, CC by-nc-sa 4.0 **IDEA**



Bogobogosort

ALGORITHMUS 5.28 (Bogobogosort)

INPUT: n Zahlen $A[1], \dots, A[n]$

OUTPUT: Sortierter Array

1. $i := 2$

2. WHILE ($i \leq n$) DO

2.1 wähle Zufallspermutation der Elemente $A[1], \dots, A[i]$

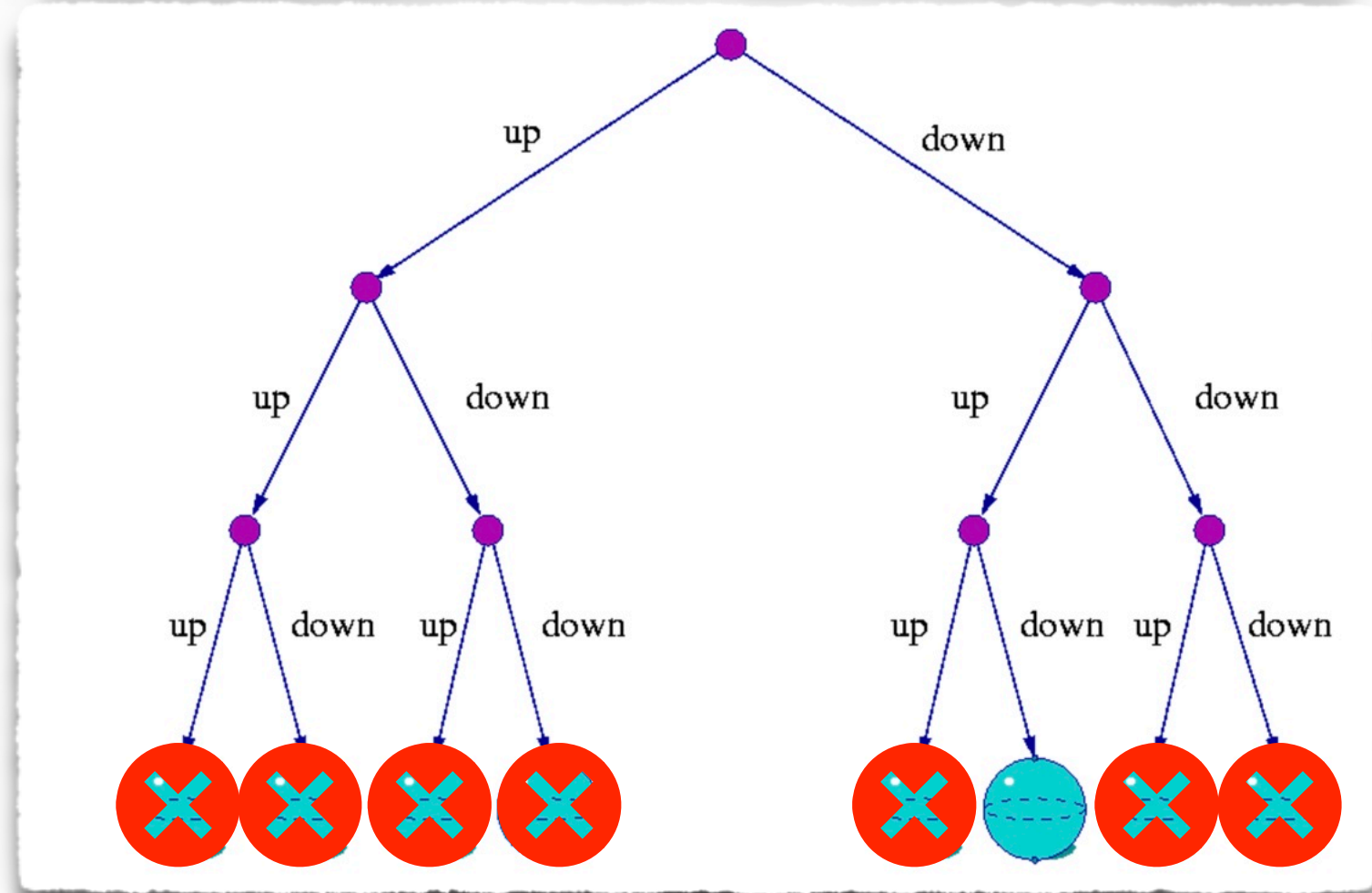
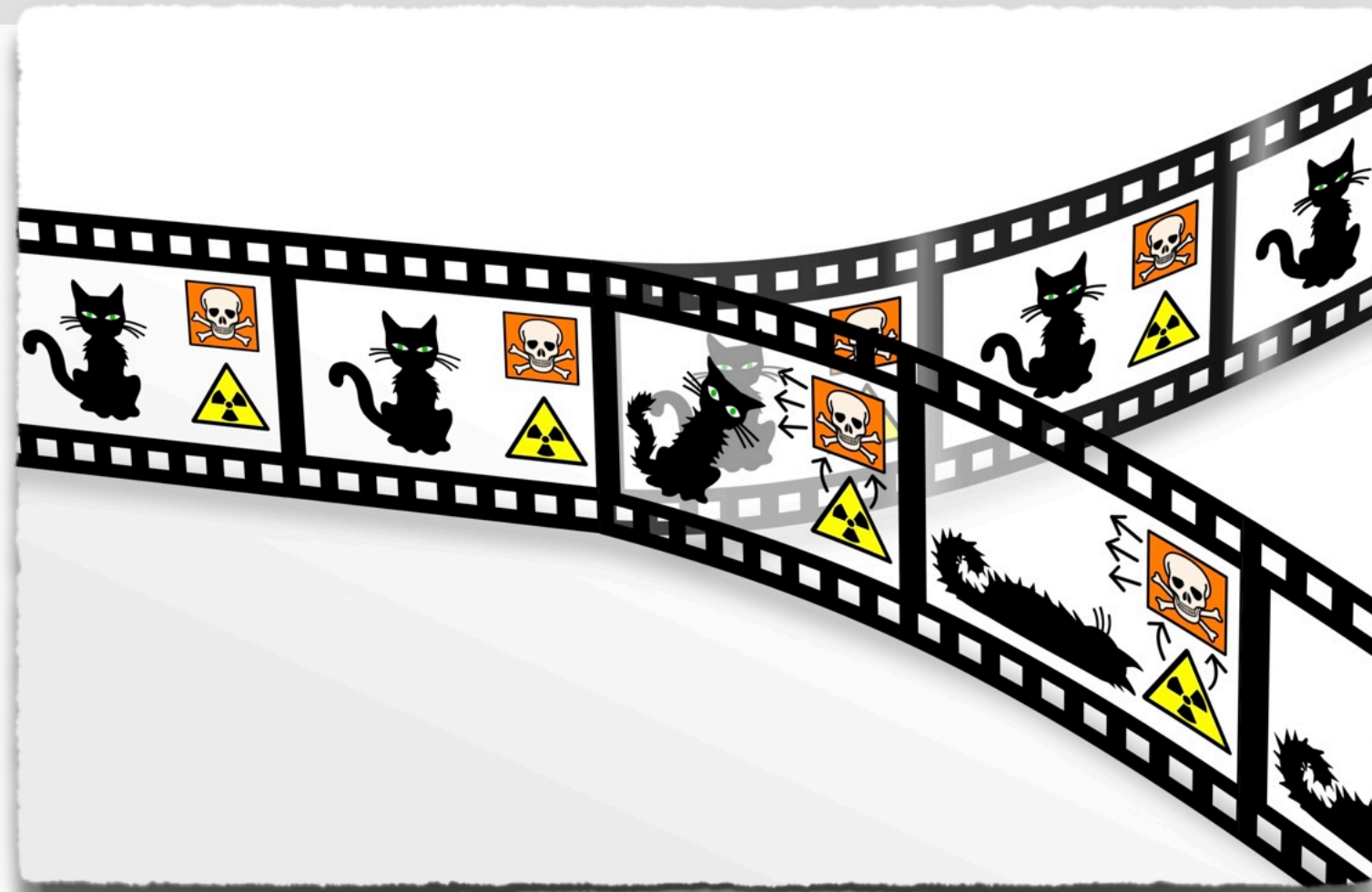
2.2 IF (Permutation ist sortiert)

2.2.1 $i := i + 1$

2.3 ELSE

2.3.1 $i := 2$

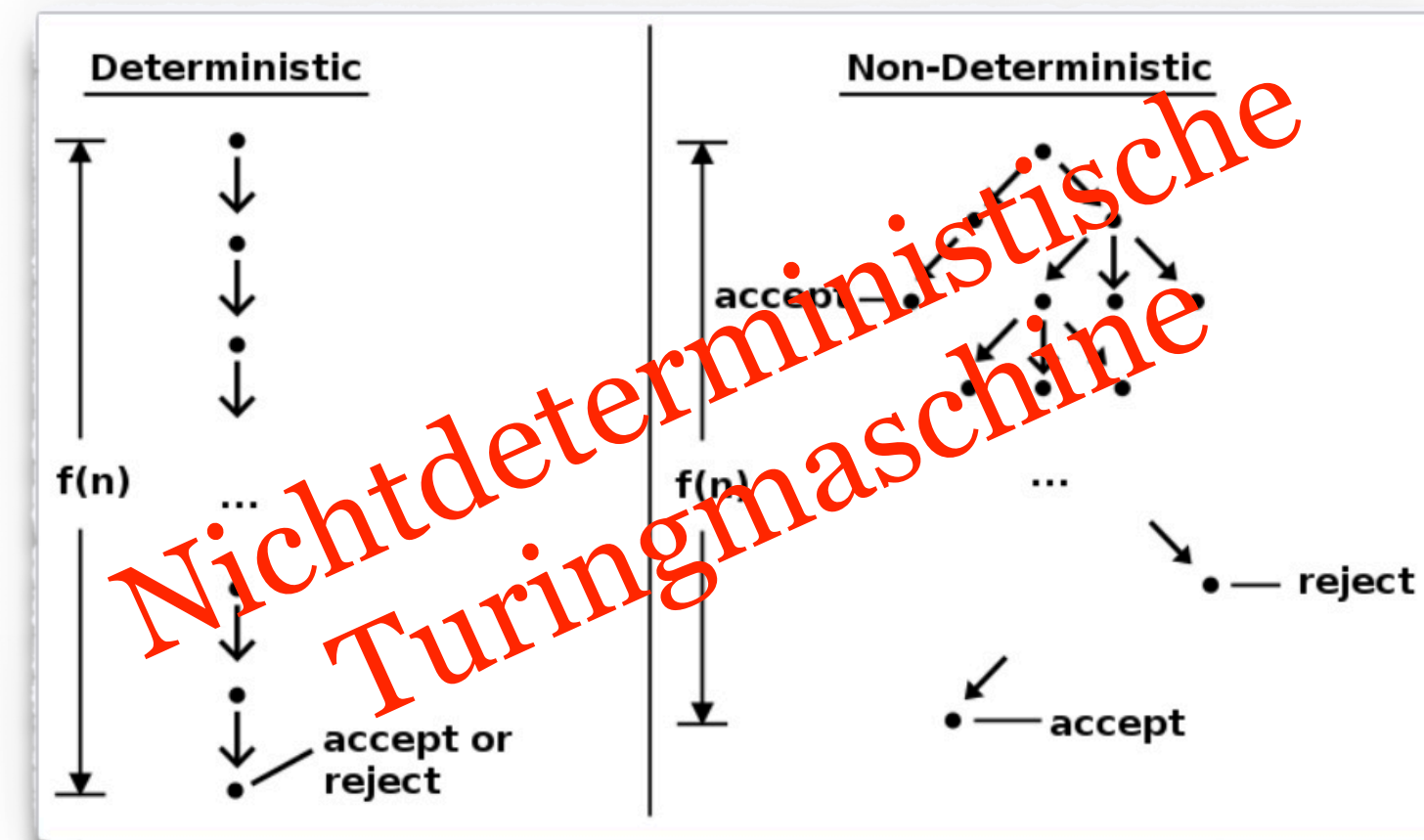
Quantum Bogosort

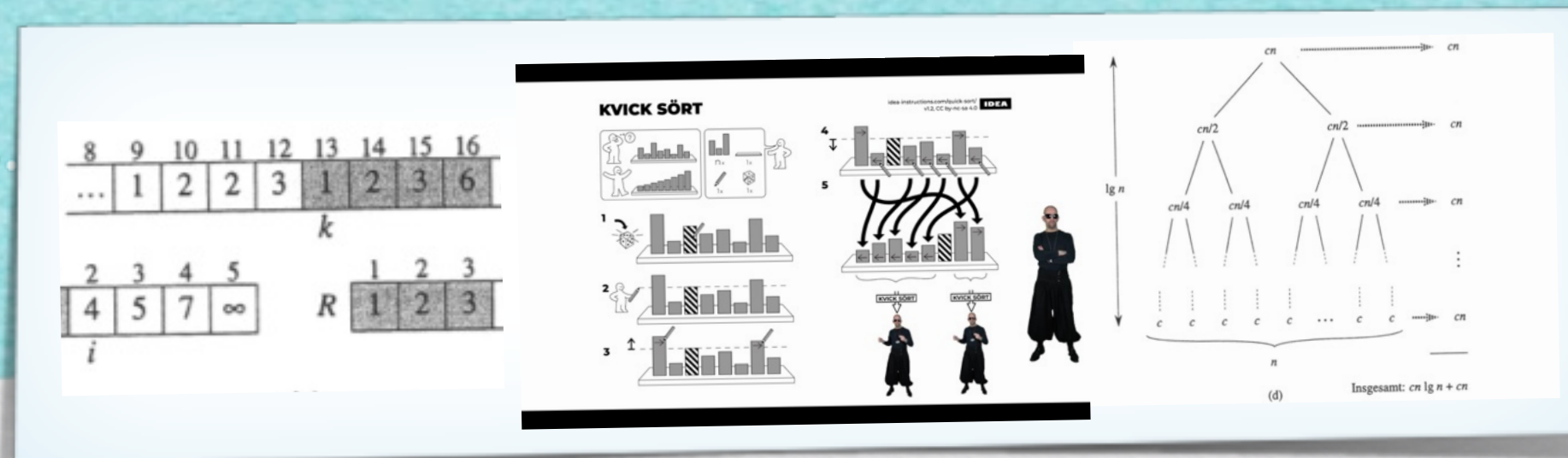


ALGORITHMUS 5.30 (Quantum Bogosort)

INPUT : n Zahlen $A[1], \dots, A[n]$
 OUTPUT : Sortierter Array in geeignetem Universum

1. Permutiere zufällig
2. IF (Permutation unsortiert)
 - 2.1 Zerstöre Universum

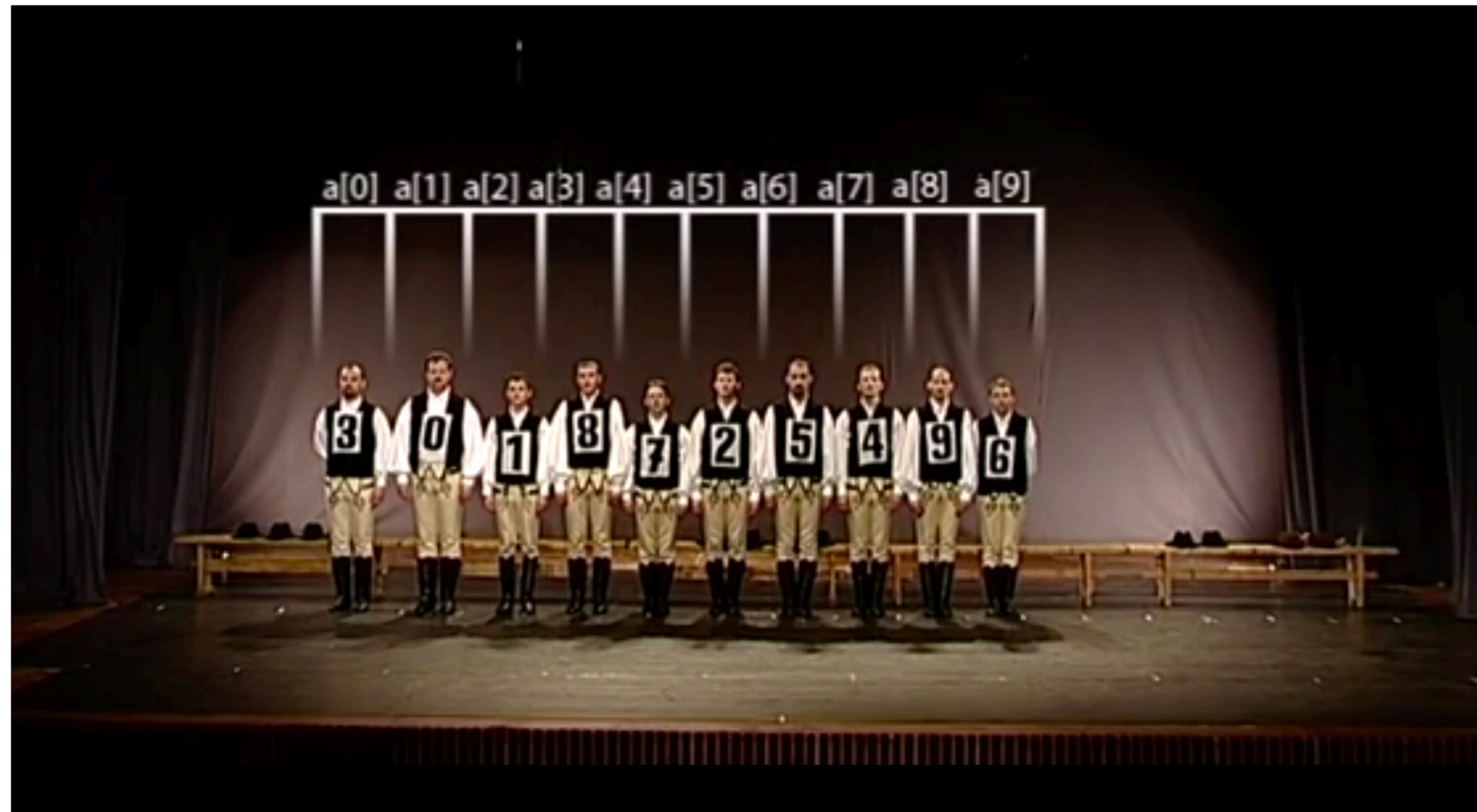


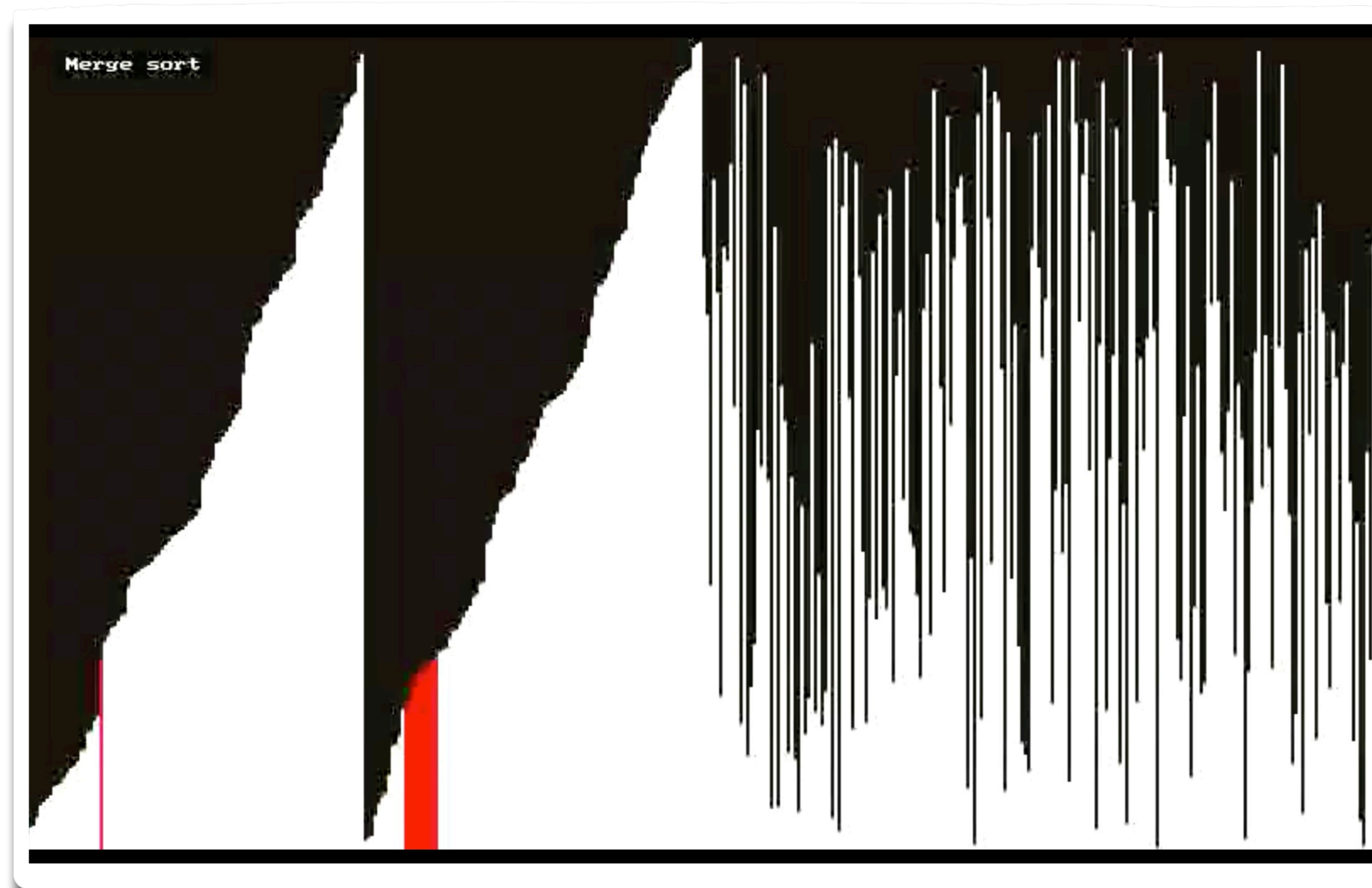


Kapitel 5.11: *Tanz und Musik!* *Algorithmen und Datenstrukturen* *WS 2025/26*

Prof. Dr. Sándor Fekete

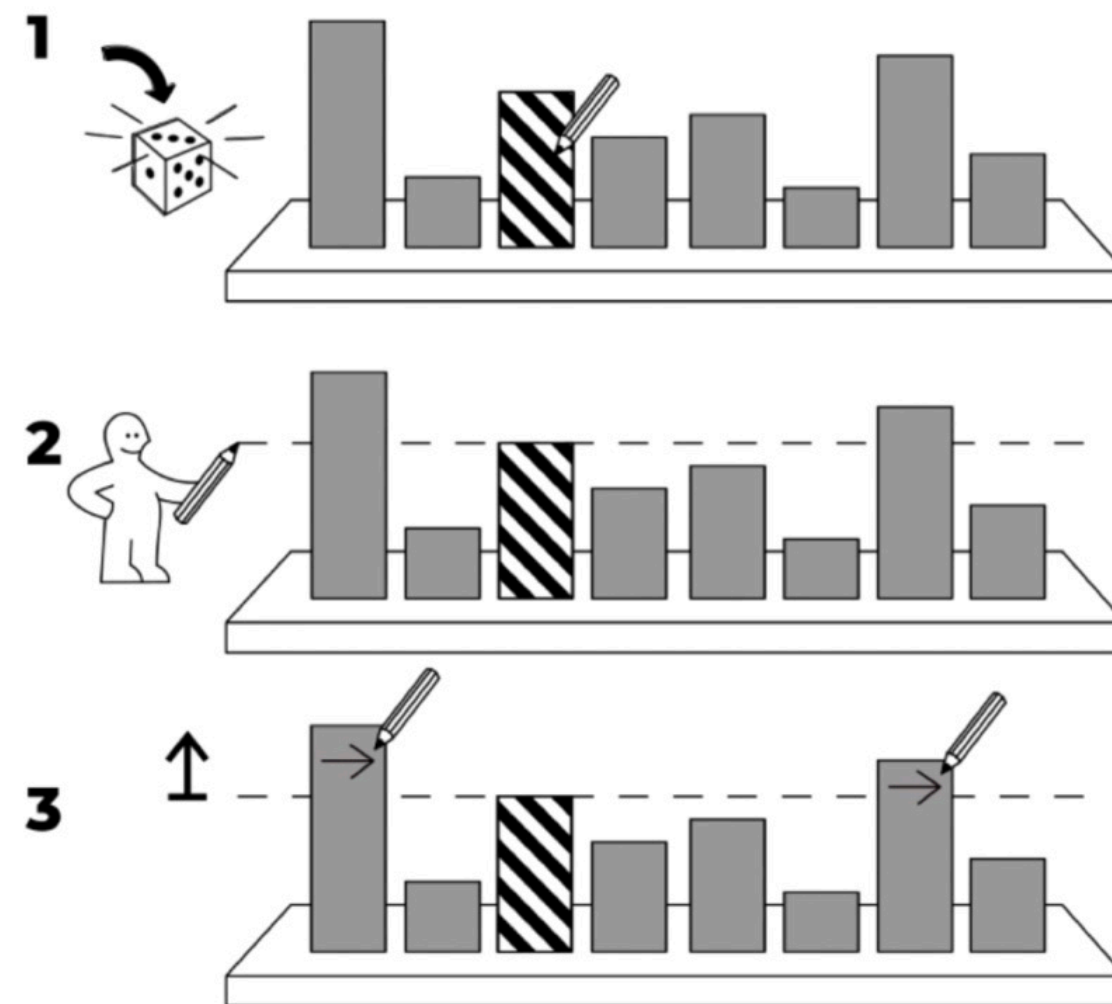
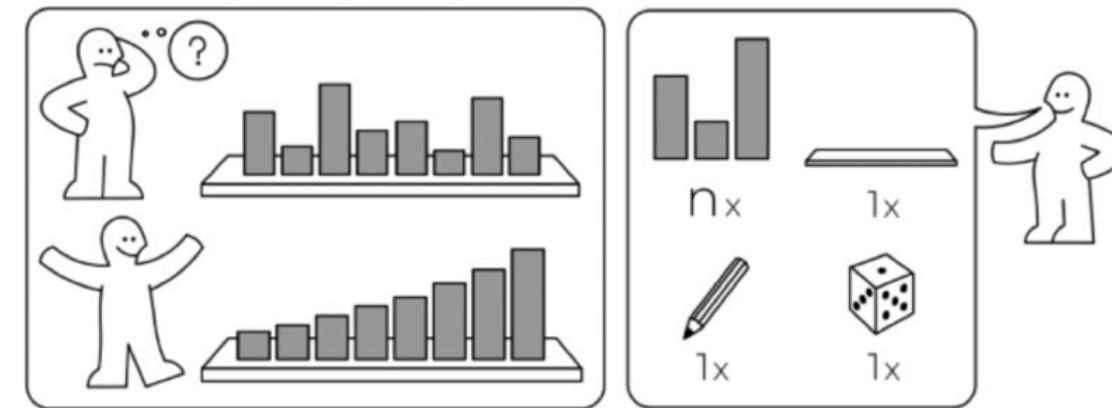
Volkstanz!



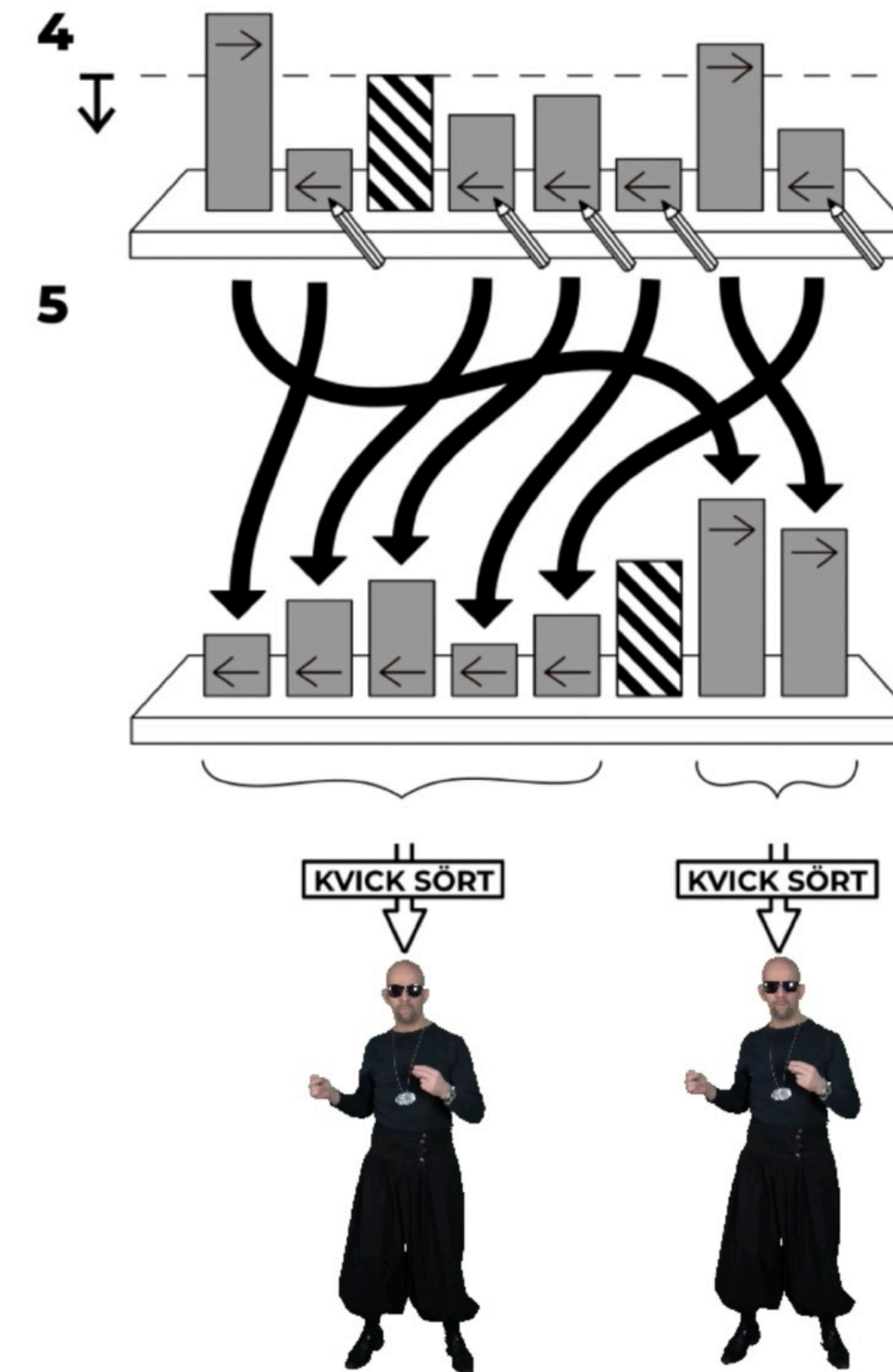


Zusammenfassung

KVICK SÖRT



idea-instructions.com/quick-sort/
v1.2, CC by-nc-sa 4.0 **IDEA**



Ausblick!

AuD 2

Algorithmen und Datenstrukturen 2

Sommersemester 2021

Startseite

Vorlesungen

Organisation

Kapitel ▾

Kontakt

Archiv

Startseite

Startseite / Algorithmen und Datenstrukturen 2

Algorithmen und Datenstrukturen 2

Die Vorlesung **Algorithmen und Datenstrukturen 2** ist eine Wahlpflichtveranstaltung für Studierende der Informatik, Wirtschaftsinformatik, Informations- und Systemtechnik; außerdem ist sie offen für interessierte Studierende anderer Studiengänge.

Algorithmen sind das methodische Herz der theoretischen und praktischen Informatik; Datenstrukturen ermöglichen die effiziente Umsetzung von Algorithmen und den effizienten Zugriff auf Input- und Outputdaten. In dieser weiterführenden Vorlesung werden die folgenden grundlegenden Begriffe erarbeitet:

- Elementare Aspekte zu Heuristiken
- Exakte Verfahren: Dynamic Programming, Branch-and-Bound
- Approximationsalgorithmen
- Komplexitätsaspekte

Neu

He

Voraussichtl.
Veranstaltu

Bitte meld
Wir nutzen
Informat
soweit m
Brauns



Vielen Dank!

s.fekete@tu-bs.de