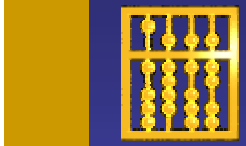
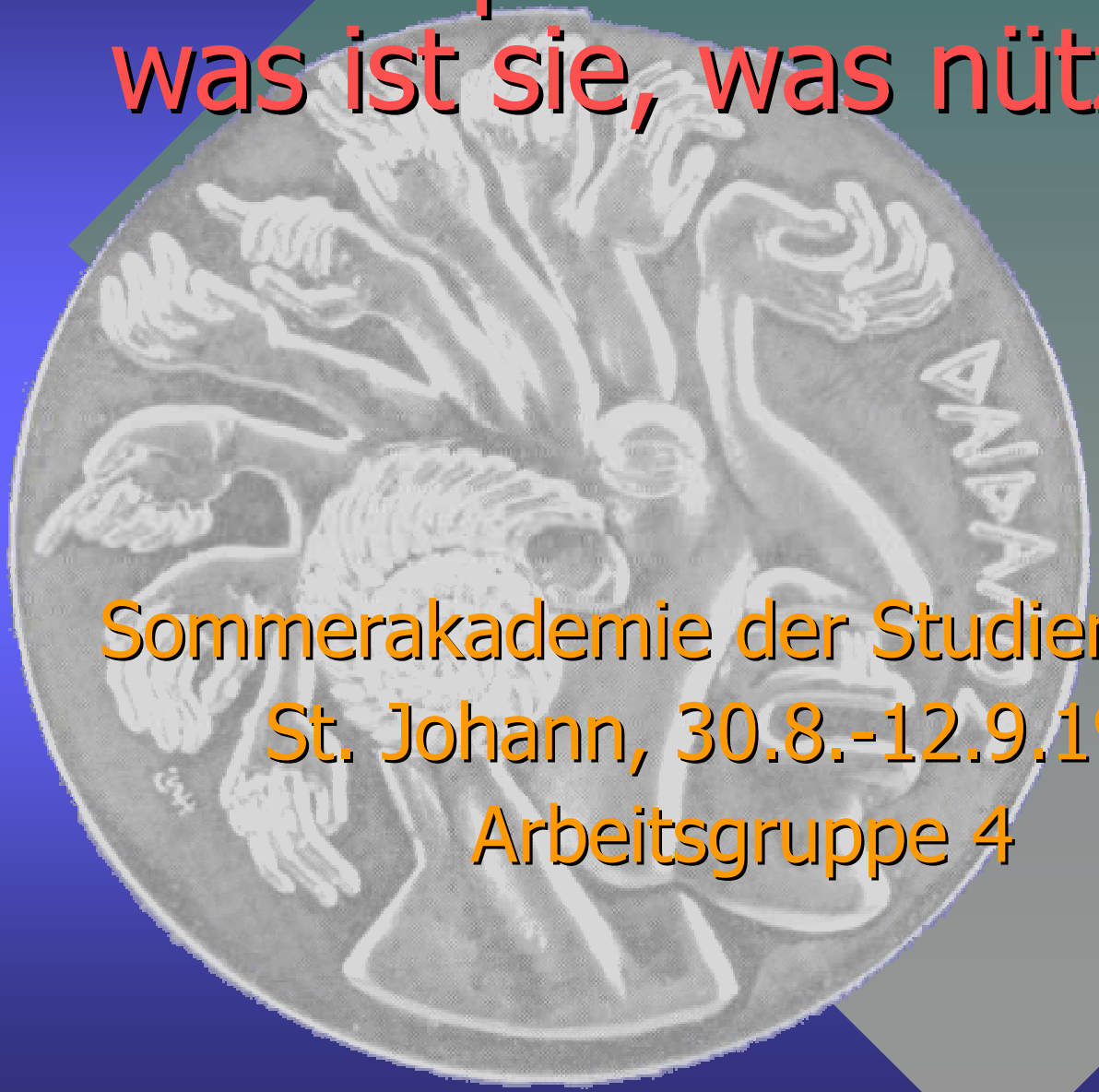
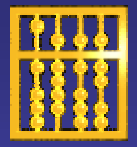


Komplexitätstheorie, was ist sie, was nützt sie?



Sommerakademie der Studienstiftung
St. Johann, 30.8.-12.9.1998
Arbeitsgruppe 4



Begriffe

Problem

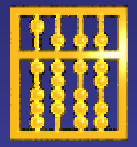
hat x die Eigenschaft Φ ?

Algorithmus

Lösung

ja/nein

Komplexität = Maß für den Ressourcenverbrauch des Algorithmus (asymptotisch)



Beispiel

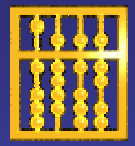
Sieb des Erathosthenes

Ist die Zahl

n

prim?

Komplexität = Maß für den Zeitverbrauch des Algorithmus



Beispiel

Ist die Zahl

4,294,967,297

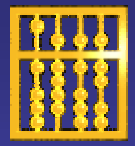
prim?

Sieb des Erathosthenes

Nein!

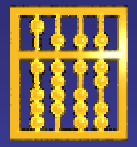
Komplexität: Zeitbedarf $\approx \sqrt{n}$,

Eingabelänge aber nur $\approx \log n$



Primzahlzerlegung *vs.* „Ist n prim?"

n=5760884999930980850257857886029248934033
5354332081765064711052492324191483894490
3183209495641255436025171261371741533464
0466664880477671008051435715108796508411
2063958709148703830550132156030420790837

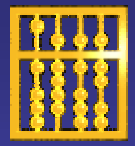


Exponentielles Wachstum!

Hier ist $\sqrt{n}$ etwa 10^{100}, das ergibt bei einer Milliarde Computern und je einer Milliarde Operationen pro Sekunde einen Zeitbedarf von etwa

$$3.17 \cdot 10^{74}$$

Jahren!

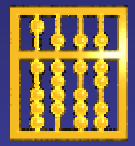


FLT: Fermat's kleiner Satz

Eine natürliche Zahl n ist prim genau dann,

wenn für alle a, $0 < a < n$, gilt:

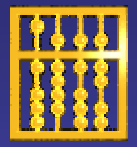
$$a^{n-1} \equiv 1 \bmod n$$



n=5760884999930980850257857886029248934033535433208176506471105249232419148389449031832094956412554360251712613717415334640466664880477671008051435715108796508411206395870914870383055013215603042079083 7;

$$a^{n-1} \bmod n$$

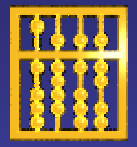
2 42100784141761969074870960679818570731641083699225154157399 4...59030
3 41540960072765620032378538462249790455718803648983376130608 1...86475
4 48988428624815119279000072499558578832191982718225724904445 3...28129
5 61330770172731381069661275375518292804918889244089413115099 2...19429
6 26755657746588612841605124621829464540985871389058585404027 2...41732
7 12312643616061582164785549906640228304641462667669399582017 9...84218
8 25132635732204893616635882273963990437848670471711218350710 6...19983
9 30677895005209678252119054338704521813796052016871699607783 5...60251
10 15316681629704231067565743976291150307482391308428037988811...00434



p=6263904117823298199092877027816256491029047673053750292420032980172129122095688727006929964350357293;

$a^{p-1} \bmod p$

2 1
3 1
4 1
5 1
6 1
7 1
8 1
9 1
10 1

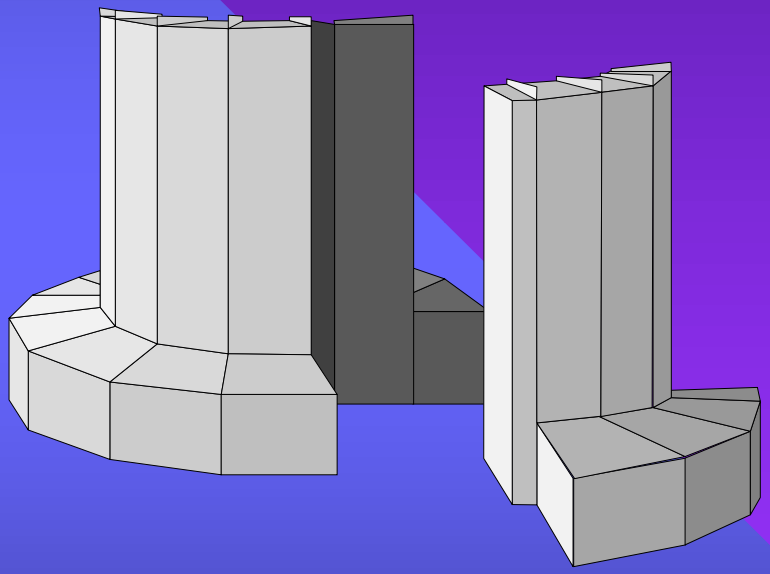


Was ist ein Computer, was ist ein Algorithmus?

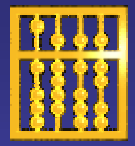
Das formale Modell eines Computers ist die Turingmaschine.

Eine Turingmaschine

Die „bessere” Turingmaschine



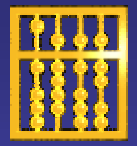
- Random Access Machine
- Lisp
- Quantencomputer
- DNA-Computer
- Church’sche These



Komplexitätsklassen

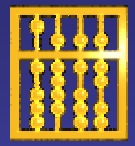
- $\mathcal{P}$ ist die Klasse all der Probleme, die von einer TM in polynomieller Zeit entschieden werden können
- Diese Definition ist sehr robust, aber ...

  seit kurzem scheint es Ausnahmen zu geben!



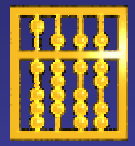
Wichtige Probleme in $\mathcal{P}$

- Sortieren
- Lineare Gleichungssysteme
- Lineare Optimierung
- Unifikation
- ...



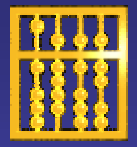
Wichtige Probleme nicht in $\mathcal{P}$

- Erfüllbarkeitsproblem der Aussagenlogik
- Ganzzahlige Lösungen linearer Ungleichungssysteme
- Spielprobleme (wie Go, Schach u.ä.)
- Nichtlineare algebraische Gleichungssysteme
- ...



Konsequenz nichtpolynomieller Laufzeiten

	n=1000	n=1001	n=2000
$1000 \cdot n$	1h	1h3.6s	2h
$100 \cdot n \cdot \log n$	59m47.7s	59m51.8s	2h11m35s
n^2	1h	1h7.2s	4h
1.1^n	$28 \cdot 10^{30}$y	$31 \cdot 10^{30}$y	$69.6 \cdot 10^{71}$y



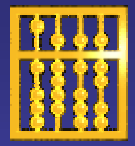
Aufgaben der Komplexitätstheorie

- Obere Schranken/Algorithmen
- Komplexitätsklassen
- Mächtigkeit von Berechnungsmodellen
- Untere Schranken
- Trade-offs

Nichtdeterministische TM

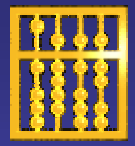
Beispiel: zusammengesetzte Zahlen

4 | 2 | 9 | 4 | 9 | 6 | 7 | 2 | 9 | 7 | | | . . .
Lesekopf
Programm mit endlich vielen Zuständen
Lese-/Schreibkopf
A | R | B | E | I | T | S | B | A | N | D | | . . .
6 | 4 | 1 | | |



Komplexitätsklassen II

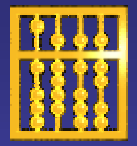
- $\mathcal{NP}$ ist die Klasse all der Probleme, die von einer nichtdet. TM in polynomieller Zeit akzeptiert werden können.
- Damit ist $\mathcal{NP}$ die Klasse all der Probleme, deren Lösungen kurze Zertifikate/Beweise haben.



Wichtige Probleme in $\mathcal{NP}$

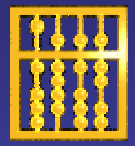
- Primfaktorzerlegung
- Erfüllbarkeitsproblem der Aussagenlogik
- Ganzzahlige Lösungen linearer Ungleichungssysteme
- Scheduling/kombinatorische Optimierung
- ...

Probabilistische Maschinen



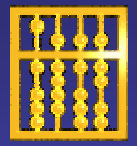
Komplexitätsklassen III

- Probabilistische TMs können die richtigen Eingaben mit einem guten Prozentteil der „Eingebungen” akzeptieren!
- Es gibt hier für eine Lösung also viele Zertifikate.
- Beispiel: Zusammengesetztsein von Zahlen



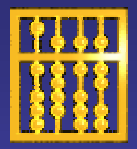
Untere Schranken

- Sie sind die „Edelsteine” der Komplexitäts-theorie
- Sie sind bei höheren Komplexitäten oft leichter, bei niedrigen (z.B. polynomiell) notorisch schwer zu bekommen
- Sie gestatten die Auszeichnung optimaler Algorithmen



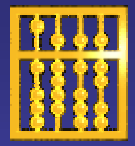
Beispiele für Untere Schranken

- Sortieren benötigt $n \log n$ Vergleiche
- ebenso die Berechnung der konvexen Hülle
- schnelle Chips für Multiplikation benötigen quadratische Größe
- algebraische Geometrie benötigt exponentiellen Platz



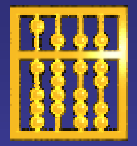
Trade-offs

- Sie besagen, daß sich eine Ressource gegen eine andere austauschen läßt
- Beispiel: Speicherplatz × Rechenzeit ist mindestens/höchstens gleich einer gewissen Schranke
- Beispiel: Multiplizier-Chips können kleiner werden, wenn sie länger brauchen dürfen



Einige Highlights

- Lineare Programmierung in $\mathcal{P}$
- Randomisierte Algorithmen
- Charakterisierung von Approximierbarkeit
- nichtalgorithmische Charakterisierung von Komplexitätsklassen
- Relativierung von Beweistechniken



Große offene Fragen

- $\mathcal{P}$ *vs* $\mathcal{NP}$ (oder: wieviel hilft Eingebung?)
- um wieviel ist Speicherplatz als Ressource mehr wert als Rechenzeit?
- Wie kann Parallelismus als Ressource effizient genutzt werden?
- Sind Quantencomputer realisierbar?
- Wie „rechnet" das menschliche Gehirn?