

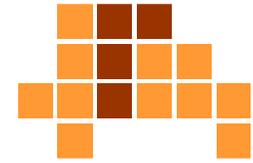
Algorithmen und Datenstrukturen

Laufzeitabschätzung

Matthias Teschner
Graphische Datenverarbeitung
Institut für Informatik
Universität Freiburg

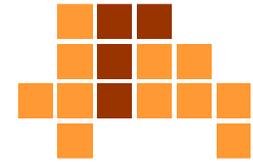
SS 12

Lernziele der Vorlesung



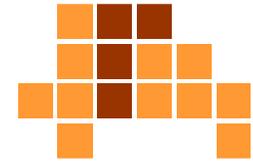
- **Algorithmen**
 - Sortieren, Suchen, Optimieren
- **Datenstrukturen**
 - Repräsentation von Daten
 - Listen, Stapel, Schlangen, Bäume
- **Techniken zum Entwurf von Algorithmen**
 - Algorithmenmuster
 - Greedy, Backtracking, Divide-and-Conquer
- **Analyse von Algorithmen**
 - Korrektheit, Effizienz

Analyse von Algorithmen



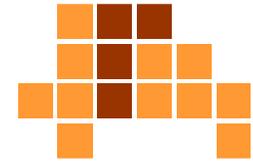
- **Korrektheit**
 - Ein korrekter Algorithmus stoppt (terminiert) für jede Eingabeinstanz mit der durch die Eingabe-Ausgabe-Relation definierten Ausgabe.
 - Ein inkorrekt Algorithmus stoppt nicht oder stoppt mit einer nicht durch die Eingabe-Ausgabe-Relation vorgegebenen Ausgabe.
- **Effizienz**
 - Bedarf an Speicherplatz und Rechenzeit
 - Wachstum (Wachstumsgrad, Wachstumsrate) der Rechenzeit bei steigender Anzahl der Eingabe-Elemente (Laufzeitkomplexität)

Effizienz



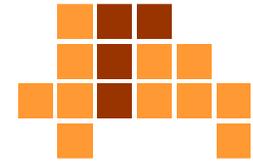
- Beispiel
 - Sortieralgorithmus A benötigt n^2 Schritte für n Elemente
 - Sortieralgorithmus B benötigt $n \log_2 n$ Schritte für n Elemente
- Wie bestimmt man die Komplexität der Zahl der Schritte?
- Wie schätzt man die Laufzeit (-komplexität) ab?

Überblick



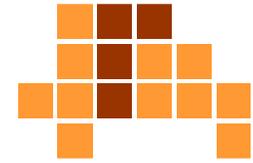
- Modell
- Laufzeiten für Sprachkonstrukte
- Beispiel 1
- Arten von Laufzeiten
- Beispiel 2
- Beispiel 3

Maschinenmodell



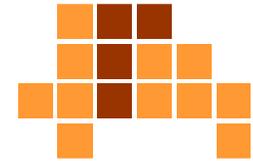
- Maschine
 - mit wahlfreiem Speicherzugriff (RAM)
 - mit einem Prozessor
 - Speicherhierarchie wird vernachlässigt (z. B. Cache)
 - kann Algorithmus als Programm ausführen
- Schritte eines Algorithmus sind zeitkonstante Anweisungen
 - werden in konstanter Zeit ausgeführt
 - Laufzeit unabhängig von der Eingabe
 - Laufzeitunterschiede einzelner Schritte werden vernachlässigt.
 - Addieren, Runden, Kopieren, bedingte Verzweigung, Aufruf eines Unterprogramms
 - Sortieren ist kein Schritt
(sinnvolle Definition von Schritten!)

Überblick



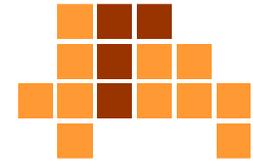
- Modell
- Laufzeiten für Sprachkonstrukte
- Beispiel 1
- Arten von Laufzeiten
- Beispiel 2
- Beispiel 3

Laufzeitabschätzung



- für Konstrukte einer einfachen imperativen Programmiersprache
 - elementare Operationen
 - sequenzielle Ausführung
 - bedingte Ausführung
 - Schleife
 - Rekursion (siehe Rekursionsgleichungen zur Laufzeitbestimmung bei Teile-und-Herrsche-Algorithmen)

Laufzeitkomplexität für imperative Sprachkonstrukte



- Problemgröße n in allen Beispielen
- elementare Operationen

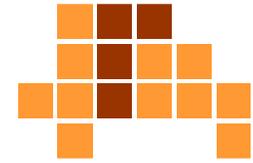
<code>i1 := 0;</code>	$O(1)$
-----------------------	--------

- Sequenz elementarer Operationen

<code>i1 := 0;</code>	$O(1)$	$327 \cdot O(1) = O(1)$
<code>i2 := 0;</code>	$O(1)$	
...	...	
<code>i327 := 0;</code>	$O(1)$	

wenn 327 konstant
bzw. unabhängig
von n ist

Laufzeitkomplexität für imperative Sprachkonstrukte

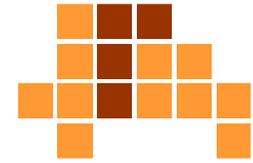


■ Schleife

<pre>for i := 1 to n do begin a[i] := 0; end;</pre>	O(n)	$O(1) \cdot O(n) = O(n)$
	O(1)	

<pre>for i := 1 to n do begin a1[i] := 0; ... a37[i] := 0; end;</pre>	O(n)	O(n)	$O(1) \cdot O(n) = O(n)$
	O(1)	$37 \cdot O(1) = O(1)$	
	...		
	O(1)		

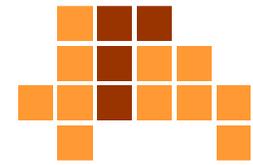
Laufzeitkomplexität für imperative Sprachkonstrukte



■ Schleife

<pre>for i := 1 to n do for j := 1 to n-1 do begin a1[i][j] := j; ... a137[i][j] := i; end;</pre>	$O(n)$	$O(n) \cdot O(n-1) =$	$O(1) \cdot O(n^2)$ $= O(n^2)$
	$O(n-1)$	$O(n) \cdot O(n) = O(n^2)$	
	$O(1)$		
	...	$137 \cdot O(1) = O(1)$	
	$O(1)$		

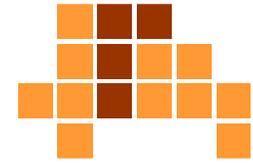
Laufzeitkomplexität für imperative Sprachkonstrukte



- bedingte Anweisung

<pre>if x<100 then y:=x; else for i:=1 to n if a[i]>y then y:=a[i];</pre>	O(1)	O(1)	O(max{1,n}) = O(n)
	O(1)		
		O(n) · O(1) = O(n)	
	O(n)		
	O(1)		
	O(1)		

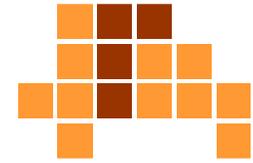
Überblick



- Modell
- Laufzeiten für Sprachkonstrukte
- **Beispiel 1**
- Arten von Laufzeiten
- Beispiel 2
- Beispiel 3

Beispiel

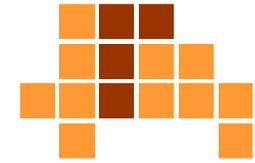
Arithmetisches Mittel



- **Eingabe:** Feld x mit n Zahlen
- **Ausgabe:** Feld a mit n Zahlen, wobei $a[i]$ dem arithmetischen Mittel von $x[0]$ bis $x[i]$ entspricht

```
for i:=0 to n-1 do
  begin
    s:=0;
    for j:=0 to i do
      begin
        s:=s+x[j];
      end;
    a[i]:=s/(i+1);
  end;
return a;
```

Laufzeitkomplexität



for i:=0 to n-1 do	O(n)		O(n) · O(i) = O(n ²)
begin		O(n)	
s:=0;	O(1)		
for j:=0 to i do	O(i+1)	O(i)	
s:=s+x[j];	O(3)		
a[i]:=s/(i+1);	O(4)	O(1)	
end;			

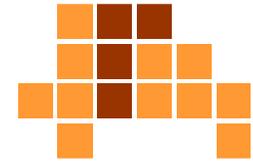
i ist von n abhängig, kann nicht als konstant angesehen werden.

- Wie oft werden Anweisungen in der inneren Schleife in Abhängigkeit von der Problemgröße n ausgeführt?

$$1 + 2 + \dots + n = \frac{n(n+1)}{2} = O(n^2)$$

Gaußsche Summenformel

Motivation für effizientere Variante



$$a[i] = \frac{1}{i+1} (x[0] + x[1] + \dots + x[i])$$

$$a[i] \cdot (i + 1) = x[0] + x[1] + \dots + x[i]$$

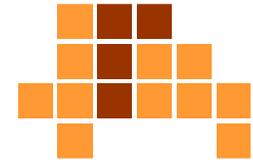
$$a[i + 1] = \frac{1}{i+2} (x[0] + x[1] + \dots + x[i] + x[i + 1])$$

$$a[i + 1] = \frac{1}{i+2} (a[i] \cdot (i + 1) + x[i + 1])$$

- $a[i+1]$ kann aus $a[i]$ und $x[i+1]$ in konstanter Zeit berechnet werden. Das Feld muss nicht durchlaufen werden.

Arithmetisches Mittel

Verbesserte Variante



- **Eingabe:** Feld x mit n Zahlen
- **Ausgabe:** Feld a mit n Zahlen, wobei $a[i]$ dem arithmetischen Mittel von $x[0]$ bis $x[i]$ entspricht

```
a[0]:=x[0];  
for i:=1 to n-1 do  
    a[i]:=(a[i-1]*i+x[i])/(i+1);  
return a;
```

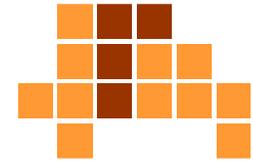
$O(n)$

- **alternativ**

```
a[0]:=x[0];  
for i:=0 to n-2 do  
    a[i+1]:=(a[i]*(i+1)+x[i+1])/(i+2);  
return a;
```

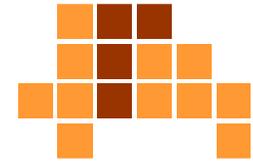
$O(n)$

Überblick



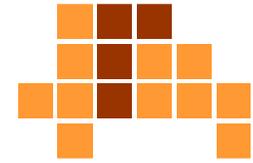
- Modell
- Laufzeiten für Sprachkonstrukte
- Beispiel 1
- Arten von Laufzeiten
- Beispiel 2
- Beispiel 3

Verschiedene Laufzeiten



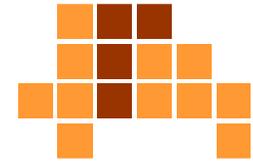
- Laufzeit hängt nicht immer ausschließlich von der Größe des Problems ab, sondern auch von der Beschaffenheit der Eingabemenge
- Daraus ergeben sich
 - **beste Laufzeit**
beste Laufzeitkomplexität für eine Eingabeinstanz der Größe n
 - **schlechteste Laufzeit**
schlechteste Laufzeitkomplexität für eine Instanz der Größe n
 - **mittlere oder erwartete Laufzeit**
gemittelte Laufzeitkomplexität für alle Eingabeinstanzen der Größe n

Überblick



- Modell
- Laufzeiten für Sprachkonstrukte
- Beispiel 1
- Arten von Laufzeiten
- **Beispiel 2**
- Beispiel 3

Beispiel

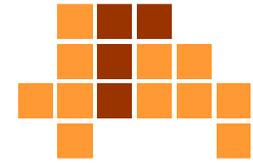


- Eingabe: Feld a mit n Elementen $a[i] \in \mathbb{N} \quad 1 \leq a[i] \leq n$
- Ausgabe: Feld a mit n Elementen mit $a[0] \neq 0$

<code>if (a[0]==1)</code>	$O(1)$	$O(1)$?
<code>then</code>		$O(1)$	
<code> a[0]:=1;</code>	$O(1)$		
<code>else</code>			
<code> for i:=0 to n-1 do</code>	$O(n)$	$O(n)$	
<code> a[i]:=2;</code>	$O(1)$		

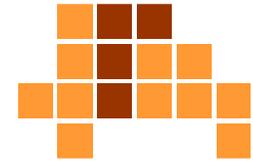
- **beste Laufzeit:** $O(1) + O(1) = O(1)$
- **schlechteste Laufzeit:** $O(1) + O(n) = O(n)$

Mittlere Laufzeit



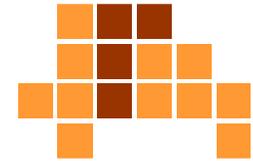
- Mittlere Laufzeit wird durch die Mittelung der notwendigen Schritte für **alle Eingabeinstanzen der Größe n** ermittelt.
- Jedes Element $a[i]$ kann n Werte annehmen.
- n Elemente: $n \cdot n \cdot \dots \cdot n = n^n$ verschiedene Eingabeinstanzen der Größe n
- $a[i]=1$ in n^{n-1} Instanzen
- $a[i] \neq 1$ in $n^n - n^{n-1} = n^{n-1} \cdot (n-1)$ Instanzen
- Gesamtschritte:
 n^{n-1} Instanzen \cdot 1 Schritt + $n^{n-1} \cdot (n-1)$ Instanzen \cdot n Schritte
 $= n^{n-1} + n^n \cdot (n-1)$
- **Mittlere Laufzeit:** $\frac{n^{n-1} + n^n \cdot (n-1)}{n^n} = \frac{1}{n} + n - 1 = O(n)$
(Schritte / Instanzen)

Überblick



- Modell
- Laufzeiten für Sprachkonstrukte
- Beispiel 1
- Arten von Laufzeiten
- Beispiel 2
- **Beispiel 3**

Beispiel

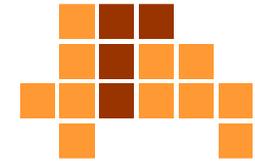


- Eingabe: n-stellige Dualzahl a
- Ausgabe: n-stellige Dualzahl a+1
- Zahl der Schritte des Algorithmus ergibt sich aus der Zahl der "0"- "1"- bzw. "1"- "0"-Wechsel

Problemgröße n	Eingabe	Operation	Ausgabe	Zahl der Schritte
10	1011100100	+1	1011100101	1
4	1011	+1	1100	3
8	11111111	+1	00000000	8 (=n)

- **beste Laufzeit:** 1 Schritt = $O(1)$
- **schlechteste Laufzeit:** n Schritte = $O(n)$

Mittlere Laufzeit



- ermittle die durchschnittliche Schrittzahl für alle Eingabeinstanzen der Größe n

n=1		
Eingabe	Ausgabe	Schritte
0	1	1
1	0	1

$$\text{Durchschnitt } \frac{1+1}{2} = 1$$

n=2		
Eingabe	Ausgabe	Schritte
00	01	1
01	10	2
10	11	1
11	00	2

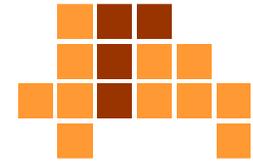
$$\text{Durchschnitt } \frac{1+2+1+2}{4} = \frac{3}{2}$$

n=3		
Eingabe	Ausgabe	Schritte
000	001	1
001	010	2
010	011	1
011	100	3
100	101	1
101	110	2
110	111	1
111	000	3

$$\text{Durchschnitt } \frac{7}{4}$$

$$\text{Mittlere Laufzeit } 2 - \frac{1}{2^{n-1}} = O(1)$$

Mittlere Laufzeit Fallunterscheidung



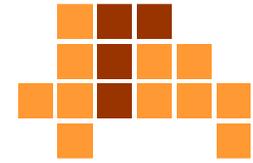
- Fallunterscheidung für Instanzen der Größe n

n-stellige Eingabe	Instanzen	Zahl der Schritte
xxx...x0	2^{n-1}	1
xxx...x01	2^{n-2}	2
xxx...x011	2^{n-3}	3
...
x0111...1	$2 = 2^1$	n-1
0111...1	$1 = 2^0$	n
111...1	1	n

- **Mittlere Laufzeit (Schritte / Instanzen):**

$$\frac{1 \cdot 2^{n-1} + 2 \cdot 2^{n-2} + \dots + (n-1) \cdot 2^1 + n \cdot 2^0 + n \cdot 1}{2^{n-1} + 2^{n-2} + \dots + 2^1 + 2^0 + 1} = \frac{\sum_{i=1}^n (i \cdot 2^{n-i}) + n}{\sum_{i=0}^{n-1} (2^i) + 1}$$

Zwischenrechnung



$$\frac{\sum_{i=1}^n (i \cdot 2^{n-i}) + n}{\sum_{i=0}^{n-1} (2^i) + 1} \quad ?$$

Nenner:

$$\sum_{i=0}^{n-1} (2^i) + 1 = 2^n - 1 + 1 = 2^n$$

geometrische
Reihe

intuitiv
klar

Zähler:

$$\sum_{i=1}^n (i \cdot 2^{n-i}) + n = 2 \cdot \sum_{i=1}^n (i \cdot 2^{n-i}) - \sum_{i=1}^n (i \cdot 2^{n-i}) + n$$

$a = 2a - a$

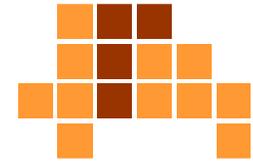
$$= 1 \cdot 2^n + 2 \cdot 2^{n-1} + 3 \cdot 2^{n-2} + \dots + (n-1) \cdot 2^2 + n \cdot 2^1$$
$$- (1 \cdot 2^{n-1} + 2 \cdot 2^{n-2} + 3 \cdot 2^{n-3} + \dots + (n-1) \cdot 2^1 + n \cdot 2^0) + n$$

$$= 2^n + 2^{n-1} + 2^{n-2} + \dots + 2^2 + 2^1 - n + n$$

$$= 2^n + 2^{n-1} + 2^{n-2} + \dots + 2^2 + 2^1 + 2^0 - 2^0 = 2^{n+1} - 1 - 2^0$$

$$+ 1 - 1 = 0$$

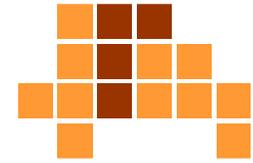
Ergebnis



- Mittlere Laufzeit (Schritte / Instanzen)

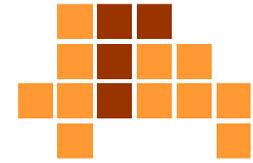
$$\frac{2^{n+1}-2}{2^n} = 2 - \frac{1}{2^{n-1}} \rightarrow 2 = O(1)$$

Zusammenfassung



- Laufzeitabschätzung in Abhängigkeit der Problemgröße
- Berücksichtigung des Maschinenmodells
- Laufzeitabschätzung für imperative Konstrukte
 - elementare Operationen
 - sequenzielle Ausführung
 - bedingte Ausführung
 - Schleife
- Laufzeiten
 - beste
 - schlechteste
 - mittlere / erwartete

Nächstes Thema



- Entwurfstechniken
 - Teile und Herrsche / Rekursionsgleichungen