Script generated by TTT

Title: Täubig: GAD (25.04.2013)

Date: Thu Apr 25 12:16:32 CEST 2013

Duration: 45:27 min

Pages: 15

Übersicht

Laufzeitanalyse

Was wissen wir?

- O-Kalkül
  ($O(f(n))$, $\Omega(f(n))$, $\Theta(f(n))$, ...)
- RAM-Modell
  (load, store, jump, ...)
- Pseudocode
  (if-else, while, new, ...)

Wie analysieren wir damit Programme?

worst case

Berechnung der worst-case-Laufzeit:

- $T(I)$ sei worst-case-Laufzeit für Konstrukt I
- $T(\text{elementare Zuweisung}) = O(1)$
- $T(\text{elementarer Vergleich}) = O(1)$
- $T(\textbf{return } x) = O(1)$
- $T(\textbf{new } \text{Typ}(\ldots)) = O(1) + O(T(\text{Konstruktor}))$
- $T(I_1; I_2) = T(I_1) + T(I_2)$
- $T(\textbf{if } (C)\ I_1\ \textbf{else } I_2) = O(T(C) + \max\{T(I_1), T(I_2)\})$
- $T(\textbf{for}(i = a; i < b; i++)\ I) = O\left(\sum_{i=a}^{b-1}(1 + T(I))\right)$
- $T(e.m(\ldots)) = O(1) + T(ss)$, wobei ss Rumpf von m

Beispiel: Vorzeichenausgabe

`signum(x)`

Input : Zahl $x \in \mathbb{R}$
Output : -1,0 bzw. 1 entsprechend dem Vorzeichen von x

```
if x < 0 then
  return -1
if x > 0 then
  return 1
return 0
```

Wir wissen:

$T(x < 0) = O(1)$
$T(\textbf{return } -1) = O(1)$
$T(\textbf{if } (C)\ I) = O(T(C) + T(I))$

Also: $T(\textbf{if } (x < 0) \textbf{ return } -1) = O(1) + O(1) = O(1)$

Beispiel: Vorzeichenausgabe

`signum(x)`

Input : Zahl $x \in \mathbb{R}$
Output : -1,0 bzw. 1 entsprechend dem Vorzeichen von x

```
if x < 0 then          O(1)
  return -1
if x > 0 then          O(1)
  return 1
return 0               O(1)
```

$O(1 + 1 + 1) = O(1)$

Beispiel: Minimumsuche

`minimum(A, n)`

Input : Zahlenfolge in $A[0], \ldots, A[n-1]$
n: Anzahl der Zahlen
Output : Minimum der Zahlen

```
min = A[0];                                  O(1)
for (i = 1; i < n; i++) do                   O(∑_{i=1}^{n-1} (1 + T(I)))
  if A[i] < min then min = A[i]                O(1)   ↗
return min                                   O(1)
```

$$O\left(1 + \left(\sum_{i=1}^{n-1} 1\right) + 1\right) = O(n)$$

Beispiel: BubbleSort

Sortieren durch Aufsteigen

Vertausche in jeder Runde in der (verbleibenden) Eingabesequenz (hier vom Ende in Richtung Anfang) jeweils zwei benachbarte Elemente, die nicht in der richtigen Reihenfolge stehen

Beispiel

5	10	19	1	14	3
5	10	19	1	3	14
5	10	1	19	3	14
5	1	10	19	3	14
1	5	10	19	3	14

1	5	10	3	19	14
1	5	3	10	19	14
1	**3**	5	10	19	14
1	3	**5**	10	14	19
1	3	5	**10**	**14**	**19**

Beispiel: Sortieren

`BubbleSort(A, n)`

Input : n: Anzahl der Zahlen
$A[0], \ldots, A[n-1]$: Zahlenfolge
Output : Sortierte Zahlenfolge A

```
for (i = 0; i < n − 1; i++) do
    for (j = n − 2; j ≥ i; j--) do
        if A[j] > A[j + 1] then
            x = A[j];
            A[j] = A[j + 1];
            A[j + 1] = x;
```

$O(\sum_{i=0}^{n-2} T(I_1))$
$O(\sum_{j=i}^{n-2} T(I_2))$
$O(1 + T(I_3))$
$O(1)$
$O(1)$
$O(1)$

$$O\left(\sum_{i=0}^{n-2}\sum_{j=i}^{n-2} 1\right)$$

Beispiel: Sortieren

$$\sum_{i=0}^{n-2}\sum_{j=i}^{n-2} 1 = \sum_{i=0}^{n-2}(n-i-1)$$
$$= \sum_{i=1}^{n-1} i$$
$$= \frac{n(n-1)}{2}$$
$$= \frac{n^2}{2} - \frac{n}{2}$$
$$= O\left(n^2\right)$$

Beispiel: Binäre Suche

`BinarySearch(A, n, x)`

Input : n: Anzahl der (sortierten) Zahlen
$A[0], \ldots, A[n-1]$: Zahlenfolge
x: gesuchte Zahl
Output : Index der gesuchten Zahl

```
ℓ = 0;
r = n − 1;
while (ℓ ≤ r) do
    m = ⌊(r + ℓ)/2⌋;
    if A[m] == x then return m;
    if A[m] < x then ℓ = m + 1;
    else r = m − 1;
return −1
```

$O(1)$
$O(1)$
$O(\sum_{i=1}^{k} T(I))$
$O(1)$
$O(1)$
$O(1)$
$O(1)$
$O(1)$

$$O\left(\sum_{i=1}^{k} 1\right) = O(k)$$

Beispiel: Binäre Suche

Aber: Wie groß ist die Anzahl der Schleifendurchläufe k?

Größe des verbliebenen Suchintervalls $(r - \ell + 1)$ nach Iteration i:

$$s_0 = n$$
$$s_{i+1} \leq \lfloor s_i/2 \rfloor$$

Bei $s_i < 1$ endet der Algorithmus.

$$\Rightarrow \quad k \leq \log_2 n$$

Gesamtkomplexität: $O(\log n)$

Beispiel: Bresenham-Algorithmus

Bresenham1

```
x = 0;   y = R;                                                    O(1)
plot(0, R); plot(R, 0); plot(0, −R); plot(−R, 0);                  O(1)
F = 5/4 − R;                                                       O(1)
while x < y do                                                     O(∑_{i=1}^{k} T(I))
    if F < 0 then
        F = F + 2 ∗ x + 1;
    else
        F = F + 2 ∗ x − 2 ∗ y + 2;                                 alles O(1)
        y = y − 1;
    x = x + 1;
    plot(x, y); plot(−x, y); plot(−y, x); plot(−y, −x);
    plot(y, x); plot(y, −x); plot(x, −y); plot(−x, −y);
```

Wie groß ist Anzahl Schleifendurchläufe k? $O\left(\sum_{i=1}^{k} 1\right) = O(k)$

Beispiel: Bresenham-Algorithmus

- Betrachte dazu die Entwicklung der Werte der Funktion

$$\varphi(x, y) = y - x$$

- Anfangswert: $\varphi_0(x, y) = R$
- Monotonie: verringert sich pro Durchlauf um mindestens 1
- Beschränkung: durch die **while**-Bedingung $x < y$ bzw. $0 < y - x$

$\Rightarrow$ maximal R Runden

Beispiel: Fakultätsfunktion

fakultaet(n)

```
Input  : n ∈ ℕ_+
Output : n!
if (n == 1) then                                                   O(1)
    return 1                                                       O(1)
else
    return n ∗ fakultaet(n − 1)                                    O(1 + ...?)
```

- $T(n)$: Laufzeit von fakultaet(n)
- $T(1) = O(1)$
- $T(n) = T(n-1) + O(1)$

$\Rightarrow$ $T(n) = O(n)$

Übersicht