ETH zürich

D&A - Übungsstunde 3

Diese Folien basieren auf denjenigen der Vorlesung, wurden aber durch den Assistenten Adel Gavranović adaptiert und erweitert

Übersicht

Heutiges Programm

Follow-up
Feedback zu [code]expert
Landau Recap
Laufzeitanalyse rekursiver Funktionen
Lösen von Rekurrenzgleichungen
Sortieralgorithmen
    Quiz
    Stabile und In-Situ Algorithmen
Outro

n.ethz.ch/~agavranovic

- Link zum Material für die Übungsstunden
- Webseite des Assistenten
- Mail an Assistenten

1. Follow-up

Follow-up aus vorheriger Übungsstunde

- Die Laufzeit für die Codes aus der Ersten Übungsstunde habe ich noch immer nicht...

Follow-up aus vorheriger Übungsstunde

Cookies

2. Feedback zu [code]expert

Allgemeines zu [code]expert

- Habe euer Feedback bez. der Aufgabenstellung zu "Prefix sums in two dimensions" weitergeleitet
- Ist jemand in dieser Übungsstunde, aber nicht in meiner [code]expert -Gruppe?
- Files raufladen, aber wie?

bezüglich meinem [code]expert -Feedback

- um 3/3 Punkte zu erreichen, muss es sehr nah an perfekt sein
- Notation und Formattierung werden bei mir nie (bewusst) in die Bewertung einfliessen, aber bitte gebt euch trotzden Mühe
- Falls ihr die Bewertung unfair findet, dann bitte sagt mir das (bedenkt aber, dass man ab 1/3 bereits die XP bekommt)
- Falls ich irgendwo erwähnt habe, dass ihr nach dir Übungsstunde kurz zu mir kommen sollt, dann macht das: Oft sind das Sachen, die man wirklich können sollte

Aufgabe "big-O"

- Wurde zwar letztes Mal kurz angesprochen, aber:

$$\log_b(n) \in \Theta(\log(n)), \quad \forall b \in \mathbb{R}$$

Aufgabe "Asymptotic Growth"

Wie löst man so "ordne diese Funktionen"-Aufgaben?

$$c, \log\log n, \log^c n, \sqrt{n}, n, n\log n, n^c, c^n, n!, n^n$$

- Hier mein Ansatz:
  1. Rangliste auf Zusammenfassung bereitmachen
  2. erstmal alle mit Exponenten die abhängig sind von n ganz nach rechts
  3. Konstanten (egal wie gross) ganz nach links!
  4. alle offensichtlichen $\log$-Sachen eher nach links
  5. Binomialzeugs auflösen/umschreiben zu einem Polynom
  6. nicht vergessen, dass $\sqrt{n} = n^{\frac{1}{2}}$
  7. alle offensichtlichen polynomial-in-n-Sachen eher nach rechts
  8. Da wo es nicht offensichtlich ist:
     - Hirn anschalten und vergleiche anstellen

Vielleicht noch hilfreich

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!} \in \Theta(n^k)$$

Aufgabe "The Set $\Theta(f)$"

Was genau beinhaltet $\Theta(f)$?

Aufgabe "Some Proofs"

Beweise

- Um eine mathematische Aussage zu widerlegen, reicht ein einziges Gegenbeispiel
- Um eine mathematische Aussage zu beweisen, braucht ihr viel Hirn…
- …oder einen sauberen Verweis auf Vorlesungsfolien oder -skripts ;)

Speziell bei dieser Aufgabe

- Studiert die Musterlösung gut
- Stellt sicher, dass ihr all die wahren Statements auf der Zusammenfassung habt

Aufgabe "Prefix Sum in 2D"

- Wenn ihr auf 100% gekommen seid und ich keine Fragen gesehen habe, gabs kein/wenig Feedback
- Falls ihr trotz 100% Fragen habt, stellt sie am besten via Mail, damit ich mich richtig einlesen kann
- Studiert trotzdem immer die Musterlösung
- Jemand (ich) wird euren Code lesen und verstehen müssen:
  - Bitte formattiert ihn gut und schreibt Kommentare!

Fragen zu [code]expert eurerseits?

3. Landau Recap

Quiz zur Landau-Notation

Ist $f \in \mathcal{O}(n^2)$, wenn $f(n) = \ldots$?

- n ✓
- $n^2 + 1$ ✓
- $\log^4(n^2)$ ✓
- $n \log(n^2)$ ✓
- n^π ✗ $(\pi \approx 3.14 > 2)$
- $n \cdot 2^{16}$ ✓
- $n^2 \cdot 2^{16}$ ✓
- 2^n ✗

Ist $g \in \Omega(n)$, wenn $f(n) = \ldots$?

- 1 ✗
- n ✓
- $\pi \cdot n$ ✓
- $\pi^{42} \cdot n$ ✓
- $\log(n)$ ✗
- $\sqrt{n}$ ✗

Fragen/Unklarheiten?

4. Laufzeitanalyse rekursiver Funktionen

Analyse

Wie oft wird `f()` aufgerufen?

```
for(unsigned i = 1; i <= n/3; i += 3)
  for(unsigned j = 1; j <= i; ++j)
    f();
```

Das Code-Fragment ruft `f()` $\Theta(n^2)$ mal auf: die äußere Schleife wird $n/9$ mal durchlaufen, und die innere Schleife ruft `f()` i mal auf.

Wie oft wird `f()` aufgerufen?

```
for(unsigned i = 0; i < n; ++i) {
  for(unsigned j = 100; j*j >= 1; --j)
    f();
  for(unsigned k = 1; k <= n; k *= 2)
    f();
}
```

Wir können die erste innere Schleife ignorieren, weil sie `f()` nur konstant oft aufruft.
Die zweite innere Schleife ruft `f()` $\lfloor \log_2(n) \rfloor + 1$ mal auf, in Summe haben wir $\Theta(n \log(n))$ Aufrufe.

Wie oft wird `f()` aufgerufen?

```
void g(unsigned n) {
  if (n>0){
    g(n-1);
    f();
  }
}
```

$$M(n) = M(n-1) + 1 = M(n-2) + 2 = ... = M(0) + n = n \in \Theta(n)$$

Wie oft wird `f()` aufgerufen?

```
// pre: n is a power of 2
//      n = 2^k
void g(int n){
  if (n>0){
    g(n/2);
    f()
  }
}
```

$$M(n) = 1 + M(n/2) = 1 + 1 + M(n/4) = k + M(n/2^k) \in \Theta(\log n)$$

Wie oft wird `f()` aufgerufen?

```
// pre: n is a power of 2
void g(int n){
  if (n>0){
    f();
    g(n/2);
    f();
    g(n/2);
  }
}
```

$$M(n) = 2M\left(\frac{n}{2}\right) + 2 = 4M\left(\frac{n}{4}\right) + 4 + 2 = 8M\left(\frac{n}{8}\right) + 8 + 4$$
$$= n + n/2 + ... + 2 \in \Theta(n)$$

Wie oft wird `f()` aufgerufen?

```
// pre: n is a power of 2
//      n = 2^k
void g(int n){
  if (n>0){
    g(n/2);
    g(n/2);
  }
  for (int i = 0; i < n; ++i){
    f();
  }
}
```

$$M(n) = 2M(n/2) + n = 4M(n/4) + n + 2n/2 = ... = (k+1)n \in \Theta(n \log n)$$

Wie oft wird `f()` aufgerufen?

```
void g(unsigned n) {
  for (unsigned i = 0; i<n ; ++i) {
    g(i)
  }
  f();
}
```

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

n	0	1	2	3	4
$T(n)$	1	2	4	8	16

Hypothese: $T(n) = 2^n$.

Induktion

Hypothese: $T(n) = 2^n$.
Induktionsschritt:

$$\begin{aligned} T(n) &= 1 + \sum_{i=0}^{n-1} 2^i \\ &= 1 + 2^n - 1 = 2^n \end{aligned}$$

Wie oft wird `f()` aufgerufen?

```
void g(unsigned n) {
  for (unsigned i = 0; i<n ; ++i) {
    g(i)
  }
  f();
}
```

Man sieht es auch direkt:

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

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

$$\Rightarrow T(n) = T(n-1) + T(n-1) = 2T(n-1)$$

Fragen/Unklarheiten?

5. Lösen von Rekurrenzgleichungen

Rekursionsgleichung

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + \frac{n}{2} + 1, & n > 1 \\ 3 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (nicht rekursive), einfache Formel für $T(n)$ an und beweisen Sie diese mittels vollständiger Induktion. Gehen Sie davon aus, dass n eine Potenz von 2 ist.

Rekursionsgleichung

$$
\begin{aligned}
T(2^k) &= 2T(2^{k-1}) + 2^k/2 + 1 \\
&= 2(2(T(2^{k-2}) + 2^{k-1}/2 + 1) + 2^k/2 + 1 = ... \\
&= 2^k T(2^{k-k}) + \underbrace{2^k/2 + ... + 2^k/2}_{k} + 1 + 2 + ... + 2^{k-1} \\
&= 3n + \frac{n}{2}\log_2 n + n - 1
\end{aligned}
$$

$\Rightarrow$ Annahme $T(n) = 4n + \frac{n}{2}\log_2 n - 1$

Induktion

1. Hypothesis $T(n) = f(n) := 4n + \frac{n}{2}\log_2 n - 1$
2. Base Case $T(1) = 3 = f(1) = 4 - 1$.
3. Step $T(n) = f(n) \longrightarrow T(2 \cdot n) = f(2n)$ ($n = 2^k$ for some $k \in \mathbb{N}$):

$$
\begin{aligned}
T(2n) &= 2T(n) + n + 1 \\
&\overset{i.h.}{=} 2(4n + \frac{n}{2}\log_2 n - 1) + n + 1 \\
&= 8n + n\log_2 n - 2 + n + 1 \\
&= 8n + n\log_2 n + n\log_2 2 - 1 \\
&= 8n + n\log_2 2n - 1 \\
&= f(2n).
\end{aligned}
$$

Master Methode

$$T(n) = \begin{cases} aT(\frac{n}{b}) + f(n) & n > 1 \\ f(1) & n = 1 \end{cases} \qquad (a, b \in \mathbb{N}^+)$$

1. $f(n) = \mathcal{O}(n^{\log_b a - \epsilon})$ für eine Konstante $\epsilon > 0 \Longrightarrow T(n) \in \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a}) \Longrightarrow T(n) \in \Theta(n^{\log_b a} \log n)$
3. $f(n) = \Omega(n^{\log_b a + \epsilon})$ für eine Konstante $\epsilon > 0$, und wenn $af(\frac{n}{b}) \leq cf(n)$ für eine Konstante $c < 1$ und alle genügend grossen $n \Longrightarrow T(n) \in \Theta(f(n))$

Beispiele

Maximum Subarray / Mergesort

$$T(n) = 2T(n/2) + \Theta(n)$$

$a = 2, b = 2,\ f(n) = cn = cn^1 = cn^{\log_2 2} \overset{[2]}{\Longrightarrow} T(n) = \Theta(n \log n)$

Beispiele

Naive Matrix Multiplication Divide & Conquer[1]

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$a = 8, b = 2$, $f(n) = cn^2 \in \mathcal{O}(n^{\log_2 8 - 1}) \overset{[1]}{\Longrightarrow} T(n) \in \Theta(n^3)$

[1] Wird später im Kurs betrachtet

Beispiele

Strassens Matrix Multiplication Divide & Conquer[2]

$$T(n) = 7T(n/2) + \Theta(n^2)$$

$a = 7, b = 2$, $f(n) = cn^2 \in \mathcal{O}(n^{\log_2 7 - \epsilon}) \overset{[1]}{\Longrightarrow} T(n) \in \Theta(n^{\log_2 7}) \approx \Theta(n^{2.8})$

[2]Wird später im Kurs betrachtet

Beispiele

$$T(n) = 2T(n/4) + \Theta(n)$$

$$a = 2, b = 4,\ f(n) = cn \in \Omega(n^{\log_4 2+0.5}),\ 2f(n/4) = c\tfrac{n}{2} \leq \tfrac{c}{2}n^1 \overset{[3]}{\Longrightarrow} T(n) \in \Theta(n)$$

Beispiele

$$T(n) = 2T(n/4) + \Theta(n^2)$$

$a = 2, b = 4$, $f(n) = cn^2 \in \Omega(n^{\log_4 2+1.5})$, $2f(n/4) = \frac{n^2}{8} \leq \frac{1}{8}n^2 \overset{[3]}{\Longrightarrow}$
$T(n) \in \Theta(n^2)$

Fragen/Unklarheiten?

6. Sortieralgorithmen

Quiz

Nachfolgend sehen Sie drei Folgen von Momentaufnahmen (Schritten) der Algorithmen (a) Sortieren durch Einfügen, (b) Sortieren durch Auswahl und (c) Bubblesort. Geben Sie unter den Folgen jeweils den Namen des zugehörigen Algorithmus an.

5	4	1	3	2
1	4	5	3	2
1	2	5	3	4
1	2	3	5	4
1	2	3	4	5

Auswahl

5	4	1	3	2
4	1	3	2	5
1	3	2	4	5
1	2	3	4	5

Bubblesort

5	4	1	3	2
4	5	1	3	2
1	4	5	3	2
1	3	4	5	2
1	2	3	4	5

Einfügen

Fragen/Unklarheiten?

Quiz

Führen Sie auf dem folgenden Array zwei weitere Iterationen des Algorithmus Quicksort aus. Als Pivot wird jeweils das erste Element des (Sub-)Arrays genommen.

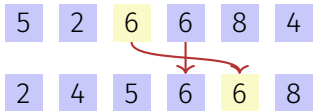
8	7	10	15	3	6	9	5	2	13
2	7	5	6	3	**8**	9	15	10	13
2	7	5	6	3	**8**	**9**	15	10	13
2	3	5	6	**7**	**8**	**9**	13	10	**15**

Fragen/Unklarheiten?

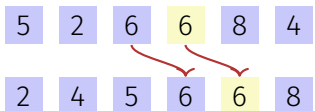
6. Sortieralgorithmen

Stabile und in-situ-Sortieralgorithmen

- Stabile Sortieralgorithmen ändern die relative Position von zwei gleichen Elementen nicht.



nicht stabil



stabil

- In-situ-Algorithmen brauchen nur konstant viel zusätzlichen Speicher.

  Welche der Sortieralgorithmen sind stabil? Welche in-situ? (Wie) kann man sie stabil / in-situ machen?

Fragen/Unklarheiten?

Allgemeine Fragen?

Bis zum nächsten Mal

Schönes Wochenende!