AlgoDat: 1.Hausaufgabe (19.04.23) - Cora Zeitler

Dienstag, 4. April 2023 09:13

Aufgabe 1:
Sie haben Sortieralgorithmen A_1, A_2 und A_3. A_1 braucht zum Sortieren einer Folge von n Zahlen $30n\lfloor\log_3 n\rfloor$ Vergleiche. Algorithmus A_2 braucht dafür $2n^2$ Vergleiche und A_3 $3n^2$ Vergleiche.

(a) Stellen Sie die Anzahl der Vergleiche die A_1 bzw. A_2 benötigen für $n = 3, 9, 27, 81, 3^5$ in einer Tabelle zusammen.

(b) Ermitteln Sie das kleinste $n > 1$, für welches A_1 weniger Vergleiche ausführt als A_2.

(c) Ermitteln Sie das kleinste $n > 1$, für welches A_1 weniger Vergleiche ausführt als A_3.

(10 Punkte)

// $\log_a(b) = x \iff a^x = b$

1a)

n	$A_1(30n\lfloor\log_3 n\rfloor)$	$A_2(2n^2)$	$A_3(3n^2)$
3	90	18	27
9	540	162	243
27	2430	1.458	2.187
81	9.720	13.122	19.683
3^5	36.450	118.098	177.147

2/2

1b) → nach Tabelle aus 1a) sieht man das A_1 nach $n=27$ kleiner wird im Vergleich zu A_2
→ kleinste n muss zw 27 und 81 liegen nicht unbedingt, du musst auch andere Bereiche prüfen

$n = 30$: A_1) $30 \cdot 30 \cdot \lfloor\log_3(30)\rfloor = 30 \cdot 30 \cdot 3 = 2700$
A_2) $2 \cdot 30^2 = 1800$

$n = 35$: A_1) $30 \cdot 35 \cdot \lfloor\log_3(35)\rfloor = 30 \cdot 35 \cdot 3 = 3150$
A_2) $2 \cdot 35^2 = 2450$

$n = 40$: A_1) $30 \cdot 40 \cdot \lfloor\log_3(40)\rfloor = 30 \cdot 40 \cdot 3 = 3600$
A_2) $2 \cdot 40^2 = 3200$

$n = 45$: A_1) $30 \cdot 45 \cdot \lfloor\log_3(45)\rfloor = 30 \cdot 45 \cdot 3 = 4050$
A_2) $2 \cdot 45^2 = 4050$ ↳ gleich, d.h bei $n > 45$ hat A_1 weniger Vergleiche

$n = 46$: A_1) $30 \cdot 46 \cdot \lfloor\log_3(46)\rfloor = 30 \cdot 46 \cdot 3 = 4140$
A_2) $2 \cdot 46^2 = 4232$ ↳ bei $n = 46$ hat A_1 weniger Vergleiche als A_2 ✓

4/4

1c) → nach Tabelle sollte das kleinste n zw. 27 und 81 liegen. f $n = 21$

$n = 28$: A_1) $30 \cdot 28 \cdot \lfloor\log_3(28)\rfloor = 30 \cdot 28 \cdot 3 = 2520$
A) $3 \cdot 28^2 = 2352$

$n = 30$: A_1) 2700
A_3) $3 \cdot 30^2 = 2700$ } gleicher Wert, dh. ab $n > 30$ hat A_1 weniger Vergleiche als A_3

$n = 31$: A_1) $30 \cdot 31 \cdot \lfloor\log_3(31)\rfloor = 30 \cdot 31 \cdot 3 = 2790$
A_3) $3 \cdot 31^2 = 2883$ ↳ bei $n = 31$ hat A_1 weniger Vergleiche als A_3

0/4

Aufgabe 2:
Wir betrachten folgenden Algorithmus, welcher Sortieren durch Einfügen implementiert.

```
INSERTIONSORT(A[1, ..., n])
for j = 2 to n do
    k := A[j]
    i := j − 1
    while (i > 0 and A[i] > k) do
        A[i + 1] := A[i]
        i := i − 1
    A[i + 1] := k
```

Bestimmen Sie die genaue Anzahl von Vergleichen, die INSERTIONSORT für folgende Eingaben ausführt. $(i > 0$ and $A[i] > k)$ zählt dabei als 1 Vergleich. Zudem sei n eine

Bsp: | 5 | 9 | 1 | 4 | 6 | 7 | 3 | 2 | 8 | → 5 und 9 sind schon sortiert

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

6. $i := i - 1$
7. $A[i+1] := k$

Bestimmen Sie die genaue Anzahl von Vergleichen, die INSERTIONSORT für folgende Eingaben ausführt. ($i > 0$ and $A[i] > k$) zählt dabei als 1 Vergleich. Zudem sei n eine durch vier teilbare natürliche Zahl.

(a) $\langle 1, 2, 3, \ldots, n-1, n \rangle$

(b) $\langle 2, 1, 4, 3, 6, 5, \ldots, n, n-1 \rangle$

(c) $\langle 3, 1, 4, 2, 7, 5, 8, 6, 11, 9, 12, 10, \ldots, n-1, n-3, n, n-2 \rangle$

(d) $\langle \frac{n}{2}+1, \frac{n}{2}+2, \frac{n}{2}+3, \ldots, n, 1, 2, 3, \ldots, \frac{n}{2} \rangle$

(10 Punkte)

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

→ von links nach recht durchlaufen
→ ausgewählte Zahl immer links an geeigneten Platz einfügen

→ Laufzeit: $O(n^2)$
→ Best case: $O(n)$ → Liste schon fast sortiert

2a) Bsp: n = 8: ⟨1,2,3,4,5,6,7,8⟩

1. 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

Bsp: n = 4: ⟨1,2,3,4⟩

1 → 2 → 3 → 4

→ Zahlenreihe ist bereits sortiert
→ es gibt immer n−1 Vergleiche
→ da schon sortiert gibt es nur eine Reihe, die einmal durchlaufen wird
→ d.h. die while-Schleife bricht jedes mal ab, weil die 2 Zahlen, welche verglichen werden, schon sortiert sind (→ also gibt es immer nur einen Vergleich)

↳ $1 \cdot (n-1) = \underline{n-1}$ Vergleiche ✓

2/2

2b) n = 4: ⟨2,1,4,3⟩

2 → 1 → 4 → 3
1 → 2 → 4 → 3
1 → 2 → 3 → 4

→ Liste nicht sortiert von anfang an

n = 8: ⟨2,1,4,3,6,5,8,7⟩

2 → 1 → 4 → 3 → 6 → 5 → 8 → 7
1 → 2 → 4 → 3 → 6 → 5 → 8 → 7
1 → 2 → 3 → 4 → 6 → 5 → 8 → 7
1 → 2 → 3 → 4 → 5 → 6 → 8 → 7
1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

↳ Pro Vertauschung wird 2 mal Verglichen (es geht 2 mal durch die while-Schleife)
→ bei allen nicht-Vertauschungen, bei dem die 2 Zahlen, die verglichen werden, schon sortiert ist, gibt es nur einen Vergleich.
(es geht nur einmal durch die while-Schleife, weil sie beim ersten Vergleich abbricht)

Bsp: n = 4: ⟨2,1,4,3⟩ ⇒ 5 Vergleiche

1. Vergl.: 2 mit 1 → while-Schleife Vertauscht, weil i > 0 und A[j] > k
2. Vergl.: 1 mit 2 → while-Schleife bricht ab naja, es wird verglichen, ob i > 0
3. Vergl.: 2 mit 4 → −॥− , weil schon sortiert
4. Vergl.: 4 mit 3 → while-Schleife vertauscht
5. Vergl.: 3 mit 4 → while-Schleife bricht ab

2/2

→ es gibt $\frac{n}{2}$ Vertauschungen, also n Vergleiche, für die unsortierten Zahlen
→ und nicht-Vertauschungen dazwischen sind $(\frac{n}{2}-1)$ Vergleiche, für die sortierten Zahlen (nach Vertauschung stehende Zahlen)

↳ $\underline{\underline{n + (\frac{n}{2} - 1)}}$ Vergleiche ✓

2c) Bsp: n = 4: ⟨3,1,4,2⟩

3 → 1 → 4 → 2
1 → 3 → 4 → 2
1 → 2 → 3 → 4

n = 8: ⟨3,1,4,2,7,5,8,6⟩

3 → 1 → 4 → 2 → 7 → 5 → 8 → 6
1 → 3 → 4 → 2 → 7 → 5 → 8 → 6
1 → 2 → 3 → 4 → 7 → 5 → 8 → 6
1 → 2 → 3 → 4 → 5 → 7 → 8 → 6
1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

↳ Bsp: n = 4: ⟨3,1,4,2⟩

1. Vergl: 3 mit 1 → while-Schleife vertauscht
2. Vergl: 1 mit 3 → while-Schleife bricht ab
3. Vergl.: 3 mit 4 → while-S. bricht ab
4. Vergl.: 4 mit 2 → w-S: vertauscht, k wird zu 3

3. Vergl.: 3 mit 4 → while-S. bricht ab

4. Vergl.: 4 mit 2 → w-S: vertauscht, k wird zu 3

5. Vergl.: 3 mit 2 → w-S: vertauscht

+ 6. Vergl.: 1 mit 2 → w-S: bricht ab

→ es gibt $\frac{n}{2}$ Vertauschungen, also n Vergleiche, für die unsortierten Zahlen

→ und nicht-Vertauschungen dazwischen sind $(\frac{n}{2}-1)$ Vergleiche, für die sortierten Zahlen (nach Vertauschung stehende Zahlen

→ jede 2te Vertauschung muss eine Zahl 3 mal Verglichen werden, weil die Distanz zw. den unsortierten Zahlen größer ist und die kleinere Zahl deswegen weiter nach links sortiert werden muss

→ da es nicht jede Vertauschung $(\frac{n}{2})$ passiert, sondern jede 2te Vertauschung $(\frac{n}{2}:2=\frac{n}{4})$, gibt es $(n+\frac{n}{4})$ Vergleiche für den Abschnitt

3/3

$\Rightarrow$ $(n+\frac{n}{4})+(\frac{n}{2}-1)$ ✓ Vergleiche

$= \frac{7}{4}n - 1$

2d) Bsp: $n=4$: $\langle 3,4,1,2\rangle$

$3\to4\to1\to2$

$1\to3\to4\to2$

$1\to2\to3\to4$ // 7 Vergl.

$n=8$: $\langle 5,6,7,8,1,2,3,4\rangle$

$5\to6\to7\to8\to1\to2\to3\to4$

$1\to5\to6\to7\to8\to2\to3\to4$

$1\to2\to5\to6\to7\to8\to3\to4$

$1\to2\to3\to5\to6\to7\to8\to4$

$1\to2\to3\to4\to5\to6\to7\to8$ // 23 Vergleiche

$\Rightarrow$ Bsp: $n=4$: $\langle 3,4,1,2\rangle$

1. Vergl.: 3 mit 4 → w.S.: Abbruch

$\frac{n}{2}$ 2. Vergl.: 4 mit 1 → w.S.: vertauschen ‖ $\langle 3,1,4,2\rangle$

3. Vergl.: 3 mit 1 → w.S.: vertauschen ‖ $\langle 1,3,4,2\rangle$

$\frac{n}{2}\cdot\frac{n}{2}$ 4. Vergl.: 1 mit 3 → w.S.: Abbruch

$\frac{n}{2}$ 5. Vergl.: 4 mit 2 → w.S.: Vertauschen ‖ $\langle 1,3,2,4\rangle$

6. Vergl.: 3 mit 2 → w.S: Vertauschen ‖ $\langle 1,2,3,4\rangle$

7. Vergl.: 1 mit 2 → w.S.: Abbruch

$n-1$

3/3

$\Rightarrow$ $\frac{n}{2}\cdot\frac{n}{2}+n-1$

$=\frac{n^2}{4}+n-1$ Vergleiche ✓

INSERTIONSORT($A[1,...,n]$)

```
1. for j = 2 to n do
2.   k := A[j]
3.   i := j − 1
4.   while (i > 0 and A[i] > k) do
5.     A[i + 1] := A[i]
6.     i := i − 1
7.   A[i + 1] := k
```

$n=4$: geg: $\langle 3,4,1,2\rangle$

$j=2$
$k=4$
$i=1$

1. Vergl.: w(✓, $3 > 4$ ✗) // Abbruch $k=4$

$j=3$
$k=1$
$i=2$

2. Vergl.: w(✓, $4>1$ ✓) ‖ Vertauschen: $\langle 3,1,4,2\rangle$, neues $i=1$

3. Vergl.: w(✓, $3>1$) ‖ Vertauschen: $\langle 1,3,4,2\rangle$, neues $i=0$

4. Vergl.: w (✗, ✗) ‖ Abbruch $k=1$

$j=4$
$k=2$
$i=3$

5. Vergl.: w(✓, $4>2$ ✓) ‖ Vertauschen: $\langle 1,3,2,4\rangle$, neues $i=2$

6. Vergl.: w(✓, $3>2$ ✓) ‖ Vertauschen: $\langle 1,2,3,4\rangle$, neues $i=1$

5.Vergl: w(✓, 4 $\overset{\checkmark}{>}$ 2) ‖ Vertauschen: ⟨1,3,2,4⟩, neues i=2
6.Vergl: w(✓, 3 $\overset{\checkmark}{>}$ 2) ‖ Vertauschen: ⟨1,2,3,4⟩, neues i=1
7.Vergl: w(✓, 1 $\overset{\times}{>}$ 2) ‖ Abbruch k=2

Aufgabe 3:
Folgendes Spiel ist eine Variante der *Türme von Hanoi*. Es gibt insgesamt 4 Stäbe. Auf einem Startstab S befinden sich eine gerade Anzahl von Scheiben. Sie sind der Größe nach geordnet (größte unten) und von der kleinsten zur größten aufsteigend nummeriert. Weiter gibt es zwei Zielstäbe Z_g und Z_u. Auf sie sollen jeweils alle Scheiben mit gerader bzw. alle Scheiben ungerader Nummer gestapelt werden, wieder sollen sie der Größe nach sortiert sein und die jeweils größte Scheibe soll unten liegen. Der vierte Stab ist ein Hilfsstab H. Es gelten die üblichen Regeln der Originalaufgabe: Pro Zug nur eine Scheibe bewegen und niemals eine größere Scheibe auf eine kleinere legen.

(a) Geben Sie für Türme der Höhen 2, 4, 6 jeweils eine Lösung, das heißt eine Folge von Zügen an.

(b) Beschreiben Sie einen rekursiven Algorithmus für einen Turm der Höhe $2k$.
(4 Punkte)

(c) Geben Sie eine Rekursionsgleichung für die Anzahl der Züge, die Ihr Algorithmus ausführt, an. (4 Punkte)

(d) Raten Sie eine Lösung dieser Gleichung und bestätigen Sie sie durch vollständige Induktion.

Hinweis: Nutzen Sie für b) und c) das Ergebnis der Originalaufgabe.

3b) Scheiben nummeriert von k (größte) bis 1 (kleinste

T(k)

① Wenn k > 2
  Wenn Turm auf S steht
    Verschiebe (k-2) bis 1 mit dem regulären Hanoi-Algorithmus auf H
  Wenn Turm auf H steht
    Verschiebe (k-2) bis 1 mit dem regulären Hanoi-Algorithmus auf S

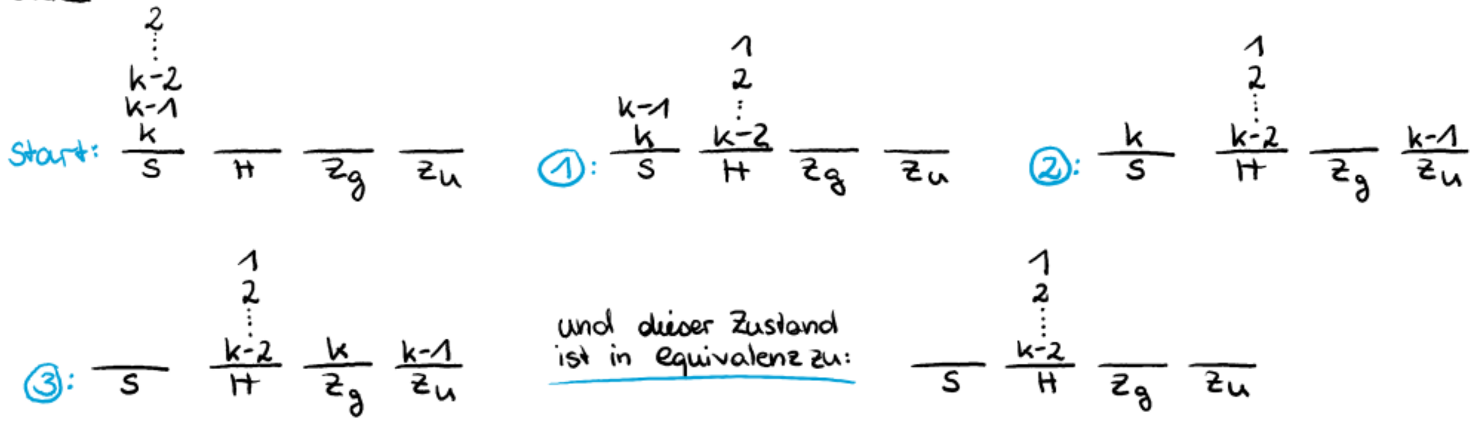
② Verschiebe k-1 (ungerade) auf Z_u

③ Verschiebe k (gerade) auf Z_g

④ Wenn (k > 2)
  T(k-2)

✓ 4/4

Skizze:



3c)

	Schritt 1	Schritt 2	Schritt 3	Schritt 4
T(k) =	H(k-2) +	1 +	1 +	T(k-2)
T(k) =	$2^{k-2}-1$ +	1 +	1 +	T(k-2)

Rekursionsschema:

$T(2) = 2$

$T(k) = 2^{k-2} + 1 + T(k-2)$ ✓

4/4