IF LK Schuljahr 2020/21

Nachweise der quadratischen Laufzeit bei Bubble-Sort

Bisher haben wir uns ja eigentlich nur experimentell Gewissheit über die Laufzeit von Bubble-Sort verschafft.

Dabei ergaben sich die Ergebnisse der letzten Stunde. Mit der nun vorhandenen mathematischen Definition der O-Notation können wir diese quadratische Laufzeit auch nachweisen. Dazu schauen wir uns den Quelltext der Sortierroutine an.

```
public String gibSortiermethode()
 2
 3
            return "BubbleSort":
 5
       public void sortieren()
            zZugriffe = 0;
 9
            // gibt den Bereich, in dem Verglichen werden soll an
10
            // Bereich wird jede Runde um 1 verkleinert
11
            for(int aussen=zZahlen.length-1;aussen>0;aussen--)
12
              for(int innen=0;innen<aussen;innen++) // zu vergleichender Index wird bestimmt
14
15
                zZugriffe = zZugriffe + 2;
16
                if(zZahlen[innen]>zZahlen[innen+1]) // Array wird an den zuvor bestimmten Stellen verglichen
17
18
                   tauschen (innen,innen+1); // wenn Zahl link größer als Zahl rechts, dann werden sie getauscht
19
20
              }
2.1
            }
        }
```

Interessant scheinen die Zeilen 16-20 zu sein, da hier die eigentliche Sortierarbeit aufgebracht werden muss. Diese Zeile wird bei n Zahlen

- beim ersten Sortierdurchlauf (n-1)-mal,
- beim zweiten Sortierdurchlauf (n-2)-mal,
- beim dritten Sortierdurchlauf (n-3)-mal
- ...
- beim letzten Sortierdurchlauf 1 mal

aufgerufen.

Insgesamt gibt es also (n-1) Sortierdurchläufe bei n zu sortierenden Zahlen (was auch die fori-Schleife in Zeile 5 zeigt), so dass (n-1)+(n-2)+(n-3)+....+1 die Zeile 10 aufgerufen wird. Diese Summe der ersten (n-1) Zahlen ergibt aber $0,5*(n-1)*(n-2) = 0,5*n^2-1,5*n+1$.

Nach unseren Vorgaben müssen wir nun vom "worst case" ausgehen, also davon ausgehen, dass in den Zeilen 16-20 immer(!) getauscht werden muss. Der Vergleich kostet zwei Speicherzugriffe, Vertauschen sechs Zugriffe, so dass ein Aufruf der Zeilen 16-20 acht Zugriffe kostet.

Somit ergibt sich eine Laufzeit von $4n^2-12n+8$ und dies liegt in $O(n^2)$.

IF LK Schuljahr 2020/21

Aufgabe:

a) Führe einen Nachweis der Laufzeit für den Algorithmus "Minsort" (Quellecodes siehe unten) durch.

b) Führe einen Nachweis der Laufzeit für den Algorithmus "rekursiver BubbleSort" (Quellecodes siehe unten) durch.

<u>Hinweis:</u> Bilde den Ablauf des rekursiven Bubble-Sort-Algorithmus zunächst mit Hilfe des Schachtelmodells ab (siehe nächste Seite)

c) Kannst Du auch Minsort rekursiv implementieren? Was bedeutet dies für die Laufzeit?

```
1
       public String gibSortiermethode()
 2
 3
            return "MinSort";
 4
 5
 6
       public void sortieren()
 7
           zZugriffe = 0;
 9
           int hMinIndex = 0;
10
           // Durchgehen des kompletten Arrays, um für die Stelle j die passende Zahl zu finden
11
           for(int j=0;j<zZahlen.length;j++)</pre>
12
13
                hMinIndex = j;
                // Suchen der kleinsten Zahl hinter den bereits sortierten Zahlen
14
15
                for(int i=j;i<zZahlen.length;i++)</pre>
16
17
                    zZugriffe = zZugriffe + 2;
18
                    // wenn die Zahl kleiner als das bisherige Minimum ist
19
                    if(zZahlen[i]<zZahlen[hMinIndex])</pre>
20
2.1
                         // Speichern des Index des neuen Minimums
22
                        hMinIndex=i;
2.3
                    }
24
25
                tausche(j, hMinIndex); // das Minimum wird an die Stelle j gesetzt
26
           }
2.7
       }
       public String gibSortiermethode()
 3
            return "BubbleSort-Rekursiv";
 4
 5
       private void sortieren(int pBereich)
 6
 8
         if (pBereich > 1) then
 9
         {
10
           for (int j = 1; j < pBereich; j++)
11
             zZugriffe = zZugriffe + 2;
12
13
             if(zZahlen[j]>zZahlen[j+1]) // Array wird an den zuvor bestimmten Stellen verglichen
14
15
                tauschen(j,j+1); // wenn Zahl link größer als Zahl rechts, dann werden sie getauscht
16
           }
17
18
            sortieren (pBereich-1);
19
       }
20
21
       public void sortieren()
22
23
            zZugriffe = 0;
24
            this.sortieren(zZahlen.length-1);
25
       }
```

IF LK Schuljahr 2020/21

zZahlen:

0	τ	2	ε	7	5
131	3	92	9	172	48

Bereich = 3			iere)									
Description	J	Bere	eich	=											
O	Description	-				zZahle	en			<u> </u>	Zugriffe				
perion	prtieren() proper continue (0		1	2	3	4	5		Zugiiie				
pBereich = j	Description	0		-						_					
pBereich = j	Description														
pBereich = j	Description														
pBereich = j	Description														
pBereich = j	Description														
<pre> j</pre>	<pre>j</pre>	ort	iere	า ()							
<pre> j</pre>	<pre>j</pre>	nBei	reich	n =											
Sortieren(J	PDCI	LCICI												
Sortieren() pBereich = j	Sortieren() pBereich = j zZahlen zZugriffe 0	j	- 0	<u> </u>	1			4		5	zZugrifi	fe			
pBereich = j	pBereich = j	0													
pBereich = j	pBereich = j														
pBereich = j	pBereich = j														
pBereich = j	pBereich = j														
pBereich = j	pBereich = j														
pBereich = j	pBereich = j	a o1	Li 0	n /				,							
j zZahlen zZugriffe 0 1 2 3 4 5 zZugriffe sortieren() pBereich = zzugriffe	j 0 1 2 3 4 5 zZugriffe 0 1 2 3 4 5 zZugriffe sortieren() pBereich = j 0 1 2 3 4 5 zZugriffe sortieren() pBereich = j 0 1 2 3 4 5 zZugriffe)							
J	J	рВе	ereio	ch =											
J	J		.			zZa	hlen				Τ				
<pre>sortieren(</pre>	<pre>sortieren(</pre>			0	1				4	5	zZugri	ffe			
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0	(0				-								
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0	-						+							
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0														
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0														
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0														
<pre>pBereich = j</pre>	pBereich = j vzahlen vzzugriffe 0 1 2 3 4 5 zzugriffe 0	sor	rtier	cen()						
<pre> j</pre>	j zZahlen 0 1 0 3 4 5 zZugriffe sortieren() pBereich = j zZahlen 0 1 zZahlen zzugriffe														
0	J 0 1 2 3 4 5	PE	serei	LCII	_										
sortieren() pBereich =	sortieren() pBereich =		j	0	1				4	E	zZugr	iffe			
sortieren() pBereich =	<pre>sortieren(</pre>	-		U	1	2	+	3	4	5	+				
pBereich =	pBereich = j zzahlen zZugriffe zZugriffe														
pBereich =	pBereich = j zzahlen zZugriffe zZugriffe														
pBereich =	pBereich = j zzahlen zZugriffe zZugriffe	\parallel			-	-	-			-					
pBereich =	pBereich = j zzahlen zZugriffe zZugriffe	-								L					
pBereich =	pBereich = j zzahlen zZugriffe zZugriffe														
	j zZahlen zZugriffe	sc	ortie	eren	()						
	j zZahlen zZugriffe	r	Bere	eich	=										٦l
- Cahlan	U O 1 2 3 4 5 ZZugriile						= 0 c 1- 1				ı		7		
			j	0	1				4		zZu	griffe	1		
			0]		
													4		