Dario Marti 18-109-967
Nûr Johanna Studer 12-122-438

Serie 4

Theoretische Aufgaben

1. COUNTING SORT

Eingabefeld A[1…11]: ganze Zahlen ≤ 8

i	1	2	3	4	5	6	7	8	9	10	11
A	2	7	1	3	2	4	1	8	5	1	4

C: Vorkommnisse der Zahlen 0 - 8

i	1	2	3	4	5	6	7	8
C	3	2	1	2	1	0	1	1

C: Anzahl Zahlen $\leq$ i

	1	2	3	4	5	6	7	8
C	3	5	6	8	9	9	10	11

Ausgabefeld B

i	1	2	3	4	5	6	7	8	9	10	11
								4			
			1					4			
			1					4	5		
			1					4	5		8
		1	1					4	5		8
		1	1				4	4	5		8
		1	1		2		4	4	5		8
		1	1		2	3	4	4	5		8
		1	1		2	3	4	4	5		8
	1	1	1		2	3	4	4	5		8
	1	1	1		2	3	4	4	5	7	8
B	1	1	1	2	2	3	4	4	5	7	8

i	1	2	3	4	5	6	7	8	9	10	11
B	1	1	1	2	2	3	4	4	5	7	8

Dario Marti 18-109-967
Nûr Johanna Studer 12-122-438

2. Algorithmus

Wir haben einen Array A, welcher n ganze Zahlen zwischen 0 und k enthält.
Wir wollen einen Algorithmus welcher die Anzahl Elemente im Intervall [a, b] ausgibt.

Sehr ähnlich wie bei COUNTING SORT mit Hilfsfeld C

```
Vorbearbeitung (A, k)
Sei C[0…k] ein neues Feld
for i = 0 to k
      C[i] = 0                          C mit 0 initialisiert
for j to A.länge
      C[A[j]] = C[A[j]] + 1             C enthält Anzahl Elemente = i
for i = 1 to k
      C[i] = C[i] + C[i-1]              C enthält Anzahl Elemente <= i
```

```
Anfrage (C, a, b)     a < b
Sei x_a die Anzahl Elemente < a
If a > 0
      x_a = C[a – 1]
else
      x_a = 0
return C[b] - x_a
```

3. RADIX SORT

N	D	B
M	N	D
N	S	A
C	I	A
B	N	D
M	A	D
B	F	V
F	S	B
K	G	B
B	V	T
D	N	D
N	I	S
N	S	B

N	S	A
C	I	A
N	D	B
F	S	B
K	G	B
N	S	B
M	N	D
B	N	D
M	A	D
D	N	D
N	I	S
B	V	T
B	F	V

M	A	D
N	D	B
B	F	V
K	G	B
C	I	A
N	I	S
M	N	D
B	N	D
D	N	D
N	S	A
F	S	B
N	S	B
B	V	T

B	F	V
B	N	D
B	V	T
C	I	A
D	N	D
F	S	B
K	G	B
M	A	D
M	N	D
N	D	B
N	I	S
N	S	A
N	S	B

Dario Marti 18-109-967
Nûr Johanna Studer 12-122-438

4. Sortieralgorithmen

Stabiles Sortierverfahren behält die Reihenfolge der Datensätze, deren Sortierschlüssel gleich sind. Wenn nach Teilschlüssel sortiert wird (z.B. Name bei «Name Vorname») gibt es bei gleichen Teilschlüsseln unterschiedliche Möglichkeiten für die Reihenfolge z.B. bei gleichem Namen, Reihenfolge der Vornamen. Ein stabiles Verfahren behält die Originalreihenfolge der Vornamen bei.

a) Stabilität von Sortieralgorithmen

INSERTIONSORT: stabil
Elemente werden nicht vertauscht, wenn sie denselben Wert haben.

MERGESORT: stabil
Es wird als default das linke Feld gewählt bei gleichen Werten.

HEAPSORT: instabil
Elemente mit gleichem Schlüssel können die Position wechseln

QUICKSORT: instabil
Elemente mit gleichem Schlüssel können die Position wechseln (Einordung des Pivot Elementes)

b)
Mit der Einführung eines eindeutigen ID-Schlüssels für die Elemente, nach dem sortiert werden kann bei Gleichheit der Werte.

5. BUCKET SORT

A		B					A sortiert
0.79	[0.0, 0.1)						0.13
0.13	[0.1, 0.2)	0.16	0.13	->	0.13	0.16	0.16
0.16	[0.2, 0.3)	0.20					0.20
0.64	[0.3, 0.4)	0.39					0.39
0.39	[0.4, 0.5)	0.42					0.42
0.20	[0.5, 0.6)	0.53					0.53
0.89	[0.6, 0.7)	0.64					0.64
0.53	[0.7, 0.8)	0.71	0.79	->	0.71	0.79	0.71
0.71	[0.8, 0.9)	0.89					0.79
0.42	[0.9, 1.0)						0.89

6. Feld mit ganzzahligen Werten

Wir wissen, dass bei der Sortierung von Zahlen (ohne führende Nullen), Zahlen mit einer kleineren Anzahl Ziffern kleiner ist, als eine Zahl mit einer grösseren Anzahl Ziffern.

Wir wollen also die Zahlen zuerst nach ihrer Länge sortieren mit Counting-Sort und danach nur Zahlen mit gleicher Länge vergleichen jeweils mit Radix-Sort.

Dario Marti 18-109-967
Nûr Johanna Studer 12-122-438

Praktische Aufgaben:
Die Zeitkomplexität ist O(n*d), wobei d die Länge der Zeichenketten ist.

Wir haben unsere Implementation des Algorithmus für d = 5 und d = 10 mit dem bestehenden Radix Algorithmus verglichen. Falls unsere Implementation des Algorithmus richtig ist, entspricht das Ergebnis den Erwartungen, da der modifizierte Radix Algorithmus für kleine d viel schneller ist.