

Übung 11 – Algorithmen II

Daniel Seemaier, Tobias Heuer – daniel.seemaier@kit.edu, tobias.heuer@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS20.php

Institut für Theoretische Informatik - Algorithmik II

```
    result = current_weight;
    return true;
}

for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ) );
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ) );
        Weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ) );
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ) );
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::INCORRECTLY_RELAXED_EDGES ) );
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

Organisatorisches

Klausurtermin

Mo 15.03.2021 16:00 Uhr

Hörsaaleinteilung wird rechtzeitig bekannt gegeben

Klausuranmeldung

Vorraussichtlich bis zehn Tage vor Klausurtermin
(≈ 05.03.2020)

Evaluation der Übung
nachher

Themenübersicht

- *in-place Multikey Quicksort*
(Sortierung von Zeichenketten *in-place*)
- KMP Algorithmus

in-place Multikey Quicksort

Wiederholung

Bentley, Sedgewick (1997)

(Three-way Radix Quicksort)

- sortiert Elemente mit **mehreren Schlüsseln** wie msd-*Radixsort*
→ z.B. Stellen einer Zahl, Zeichen eines Strings
- für einen Schlüssel wird *Quicksort* mit **drei Fällen** ausgeführt
→ **kleiner als, gleich, größer als** das Pivotelement
- PartitionierungsSchritt kann ***in-place*** erfolgen
→ ähnlich wie bei normalem *Quicksort*

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

 $\text{mkqSort}(\langle e \in S : e[i] < p[i] \rangle, i),$
 $\text{mkqSort}(\langle e \in S : e[i] = p[i] \rangle, i + 1),$
 $\text{mkqSort}(\langle e \in S : e[i] > p[i] \rangle, i)$

S	B	E	H	A	M	T	M	H	S	H	A	H	U	N
A	I	H	A	R	I	A	O	A	E	U	U	A	H	A
A	E	R	U	M	E	S	R	N	E	N	A	L	R	C
L	N	E	S		S	S	D	D		D		L		H
	E					E						E		T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

$$\begin{aligned} &\text{mkqSort}(\langle e \in S : e[i] < p[i] \rangle, i), \\ &\text{mkqSort}(\langle e \in S : e[i] = p[i] \rangle, i + 1), \\ &\text{mkqSort}(\langle e \in S : e[i] > p[i] \rangle, i) \end{aligned}$$

S	B	E	H	A	M	T	M	H	S	H	A	H	U	N
A	I	H	A	R	I	A	O	A	E	U	U	A	H	A
A	E	R	U	M	E	S	R	N	E	N	A	L	R	C
L	N	E	S		S	S	D	D		D		L		H
	E					E					E			T

$i = 1$

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)

p

S	B	E	H	A	M	T	M	H	S	H	A	H	U	N
A	I	H	A	R	I	A	O	A	E	U	U	A	H	A
A	E	R	U	M	E	S	R	N	E	N	A	L	R	C
L	N	E	S		S	S	D	D	D		L		H	T
	E				E						E			

$i = 1$

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

B	E	A	A
I	H	R	U
E	R	M	A
N	E		
E			

$i = 1$

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

B	E	A	A
I	H	R	U
E	R	M	A
N	E		
E			

$i = 1$

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

A	A
R	U
M	A

$i = 1$

B
I
H
E
R
N
E

E
H
A
D
D

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

A	A
R	U
M	A

B
I
H
E
R
N
E

E
H
R
N
E

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

$i = 1$

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

A	A
R	U
M	A

$i = 1$

B
I
H
E
N
E

E
H
R
E
E

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

A	A
R	U
M	A

$i = 2$

B
I
H
E
N
E

E
H
R
E

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

p

A	A
R	U
M	A

B	E
I	H
E	R
N	
E	

$i = 2$

H	H	H	H
A	A	U	A
U	N	N	L
S	D	D	L
			E

S	M	T	M	S	U	N
A	I	A	O	E	H	A
A	E	S	R	E	R	C
L	S	S	D			H
			E			T

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

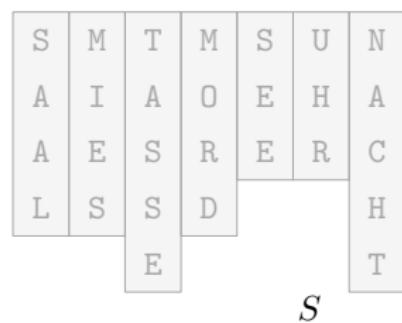
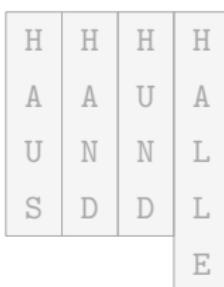
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

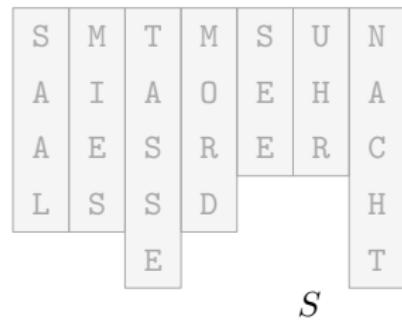
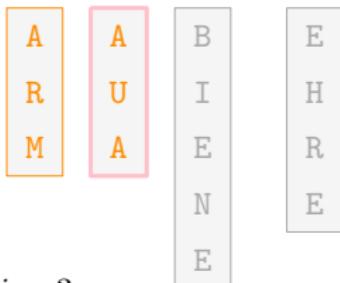
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

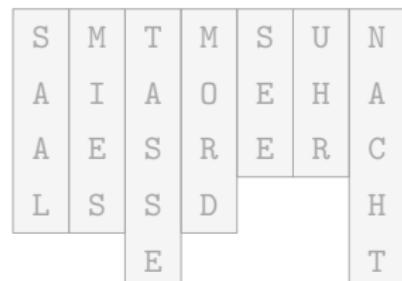
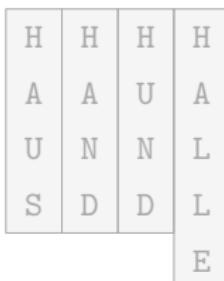
mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



$i = 2$



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

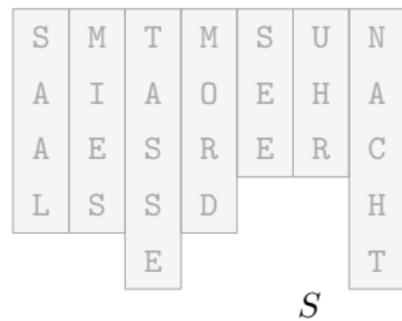
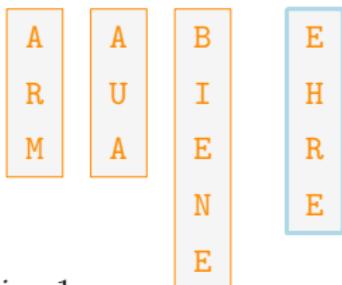
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

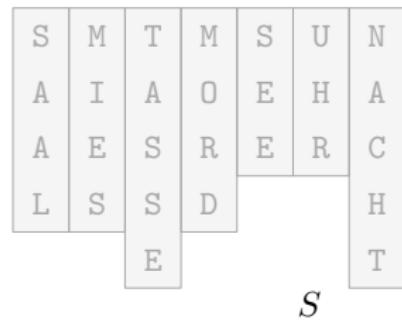
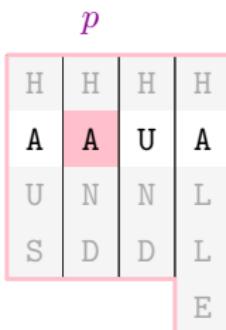
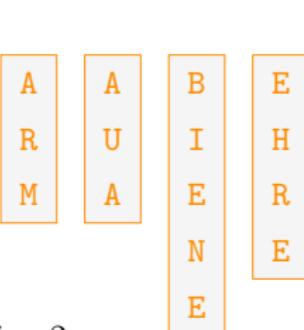
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

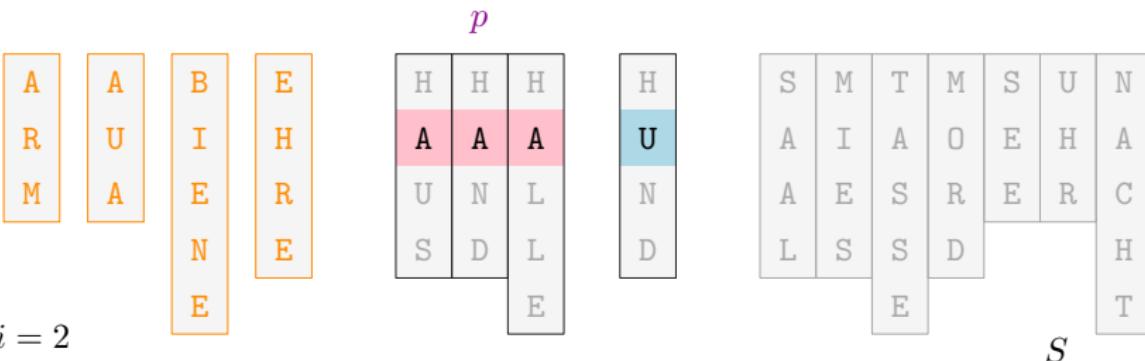
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

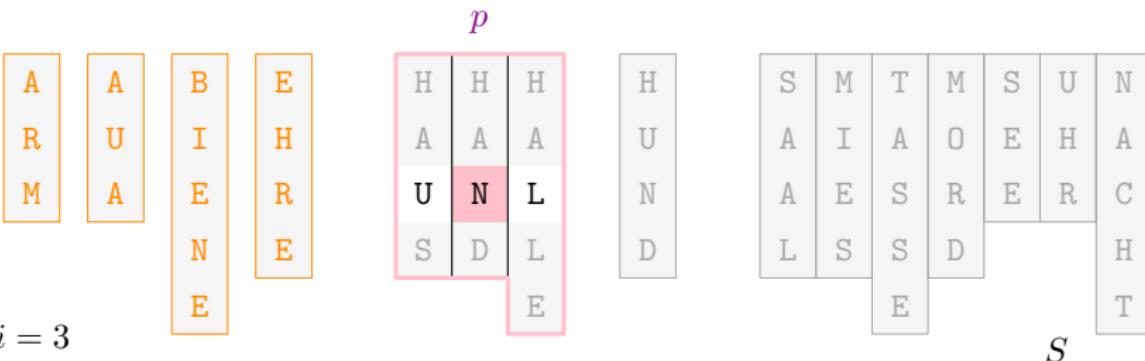
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

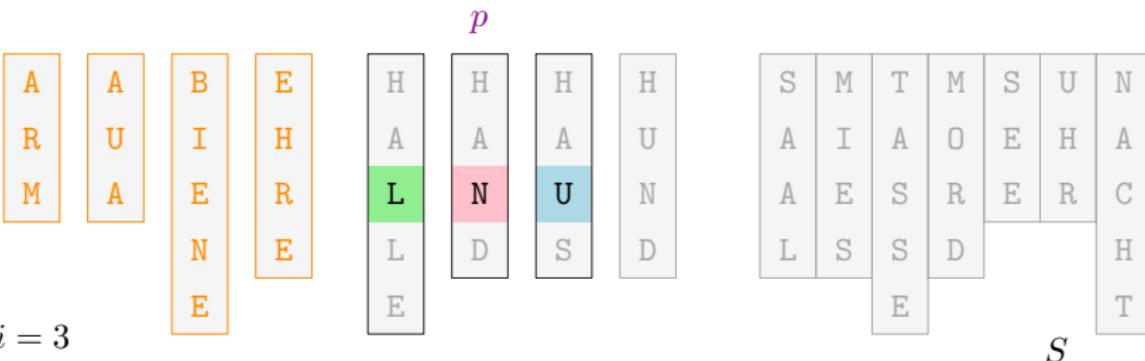
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

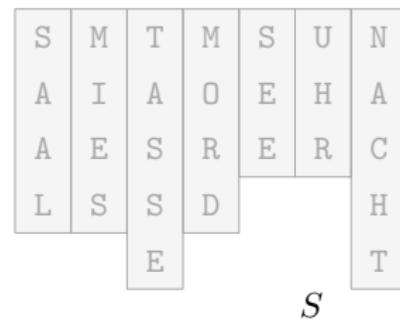
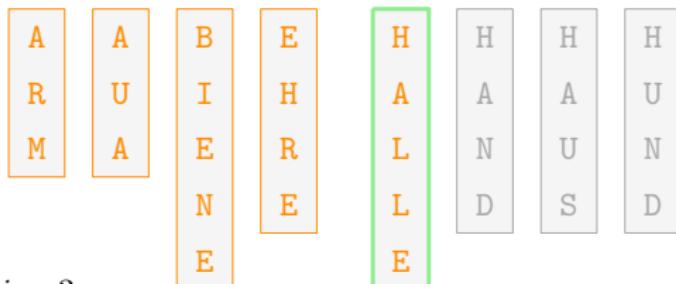
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

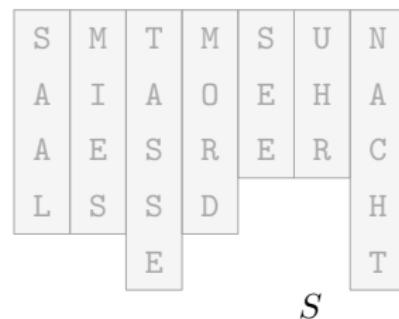
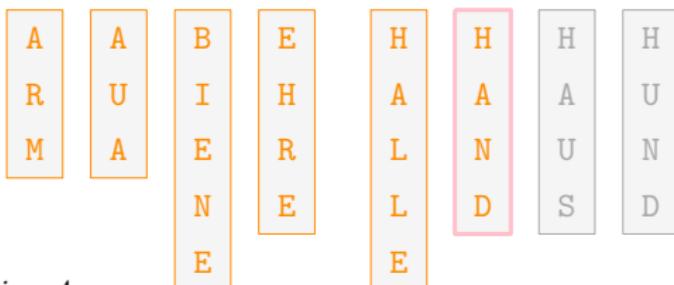
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

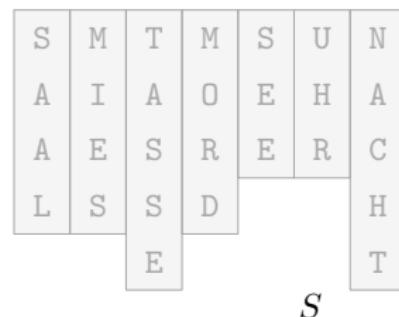
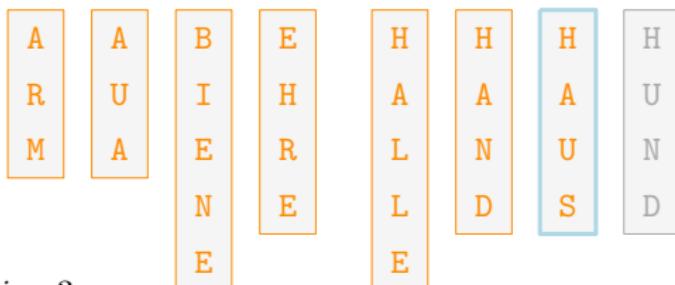
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

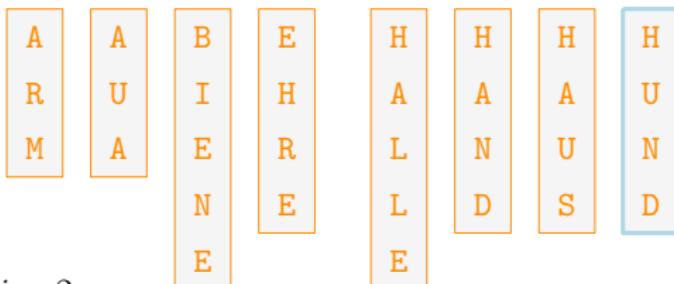
return concatenation of

(Rekursion)

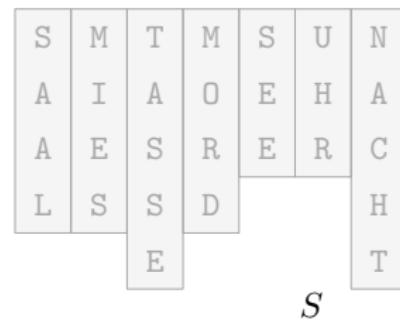
mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



$i = 2$



S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

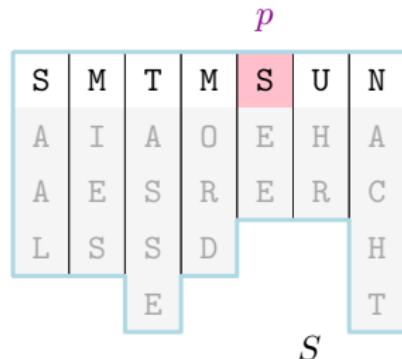
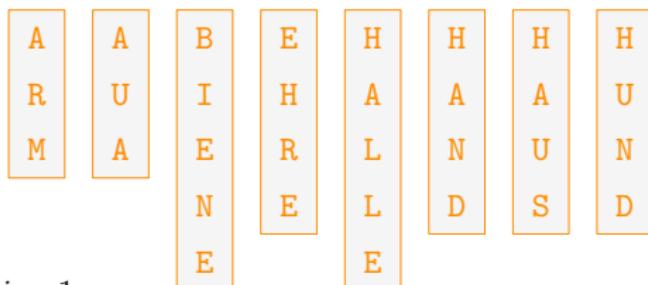
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

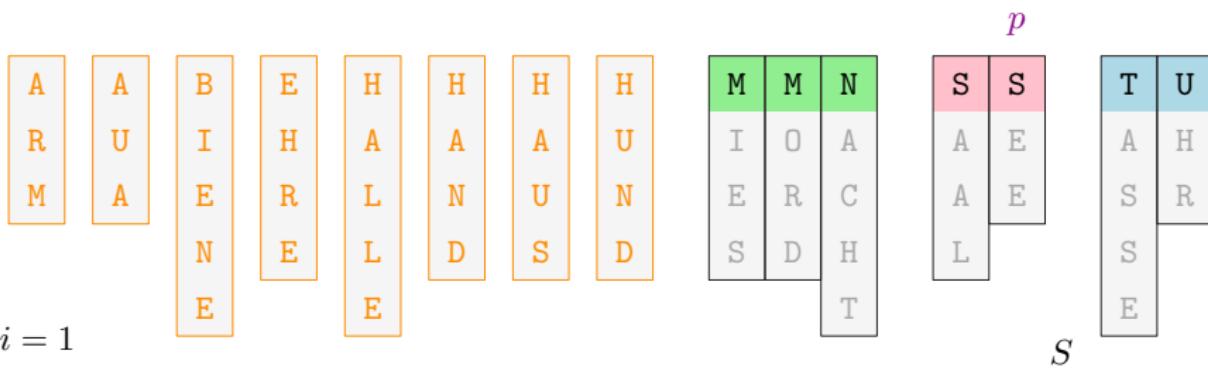
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



$i = 1$

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

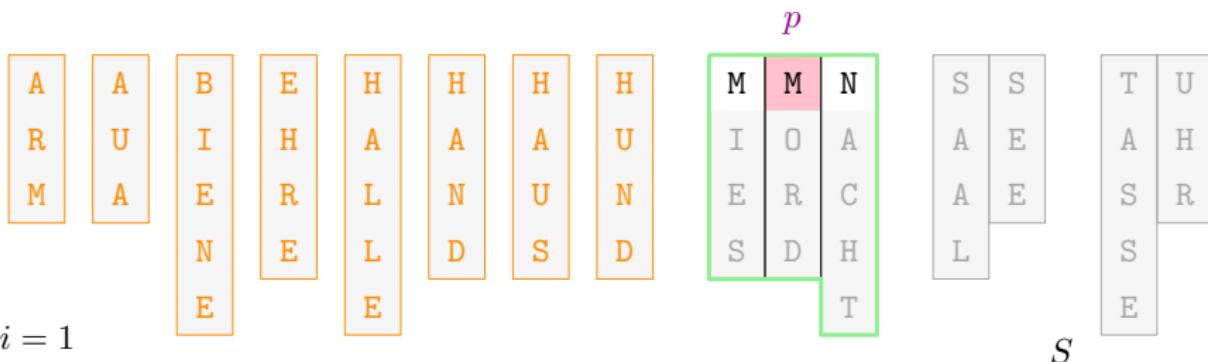
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

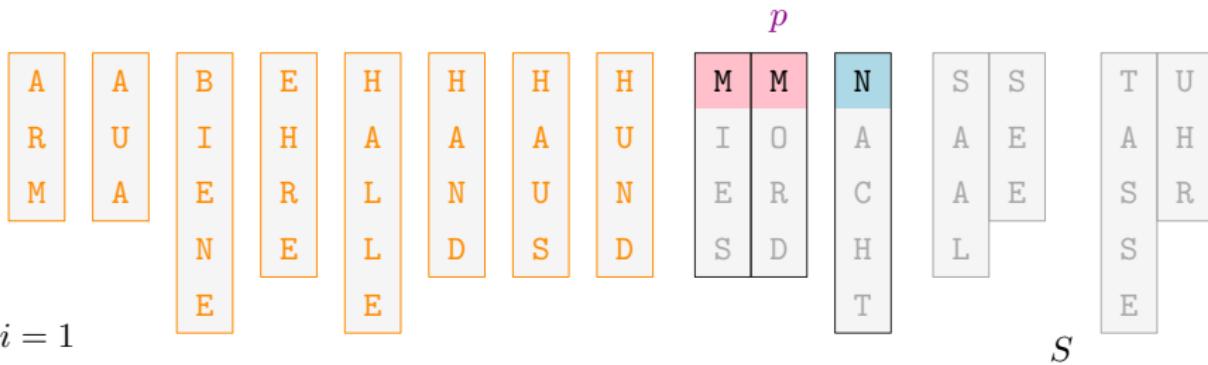
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

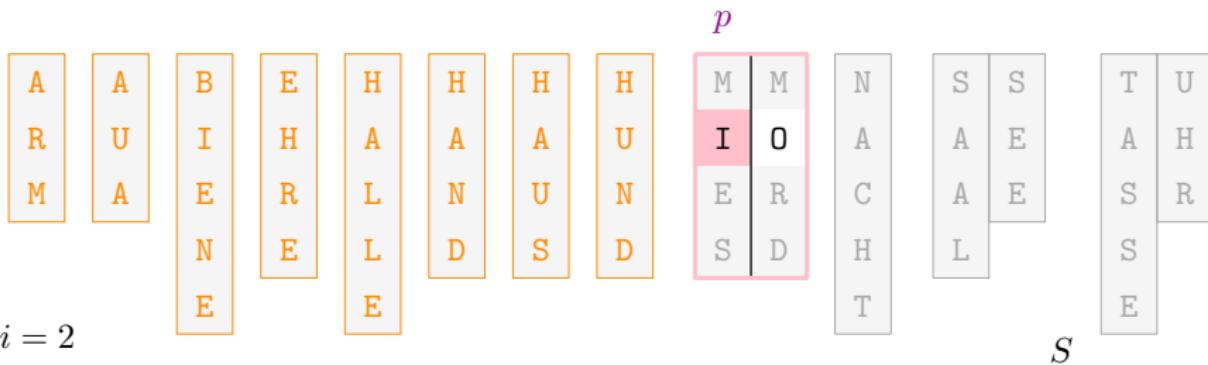
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

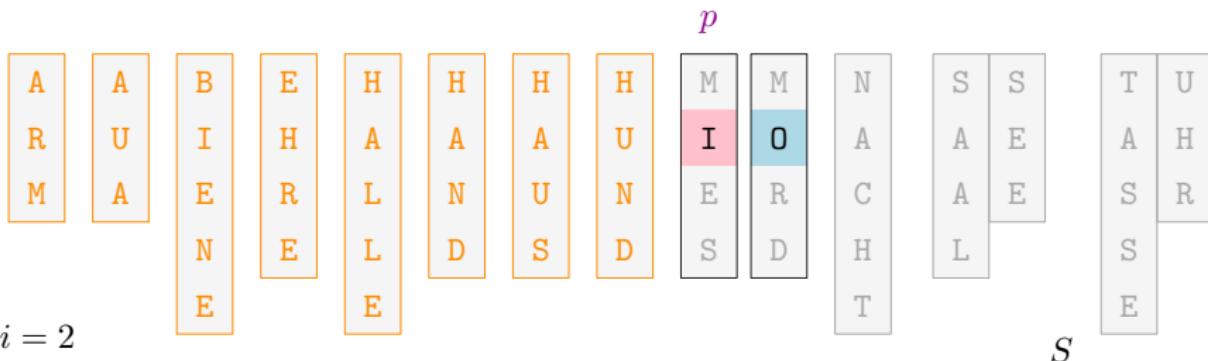
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

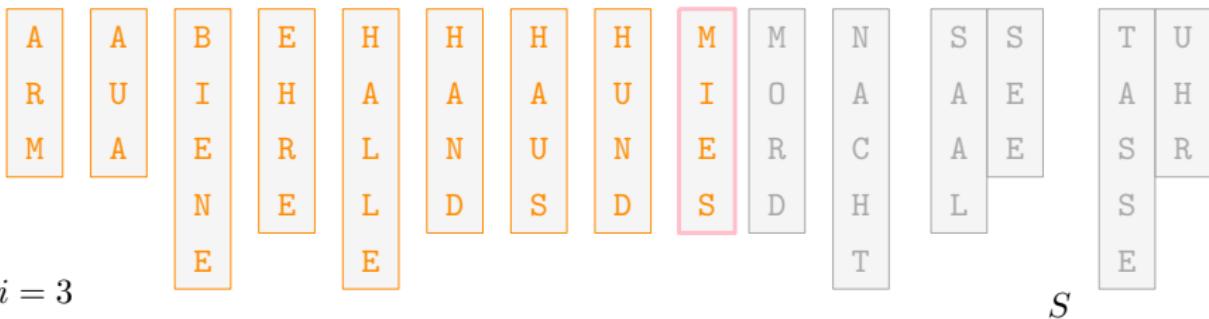
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



$i = 3$

S

in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

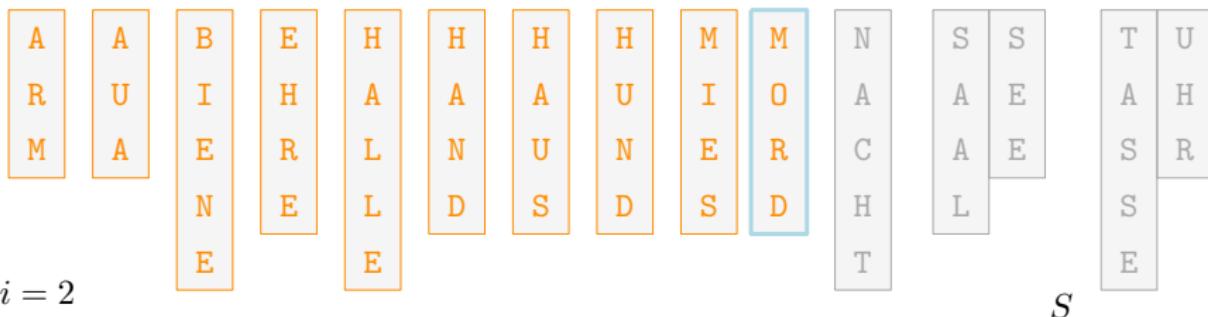
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

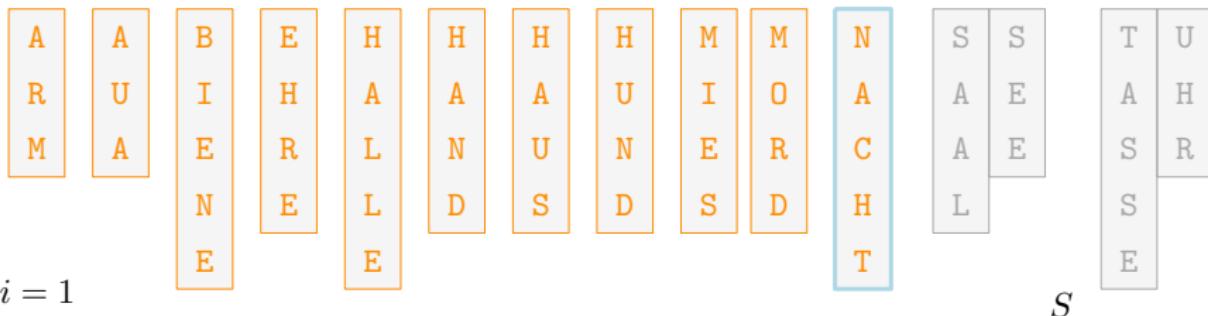
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

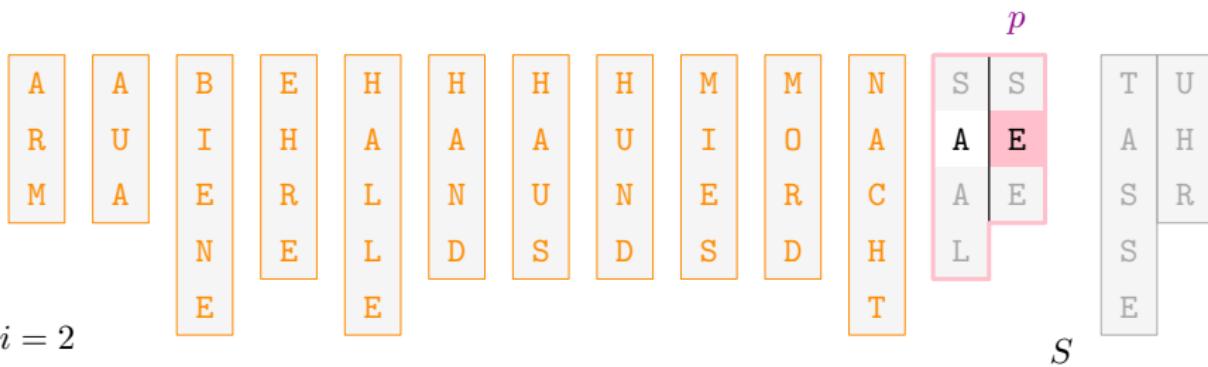
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

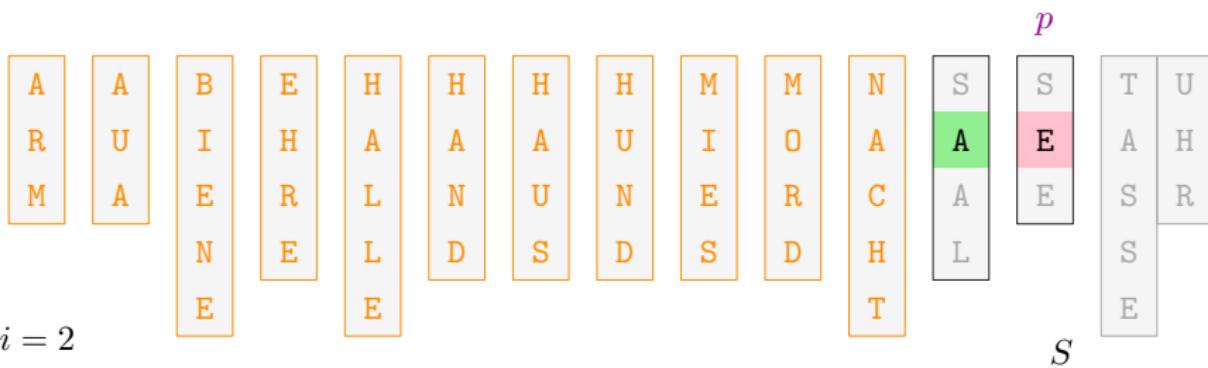
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

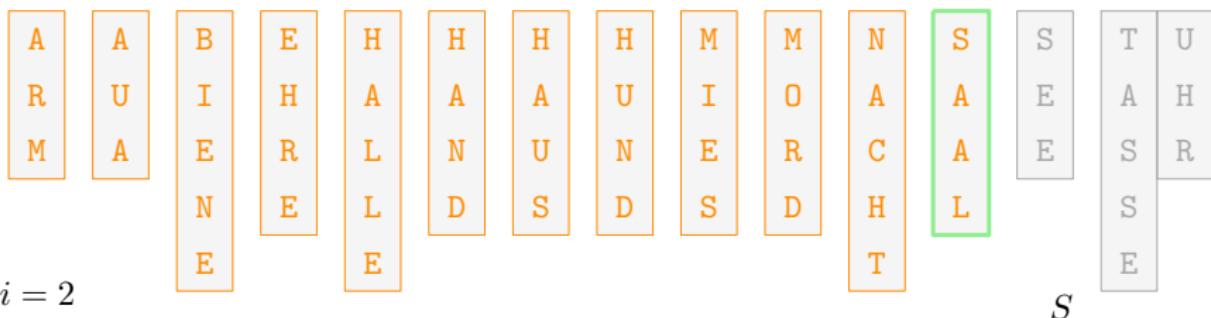
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

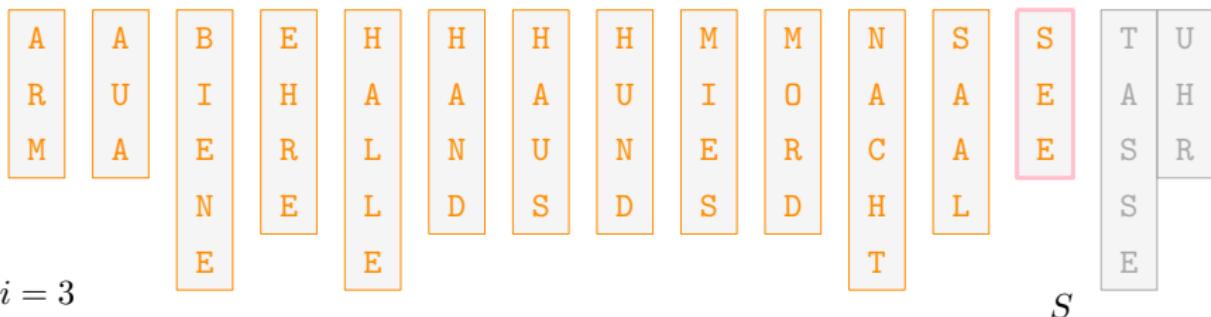
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle, i$),

mkqSort($\langle e \in S : e[i] = p[i] \rangle, i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle, i$)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

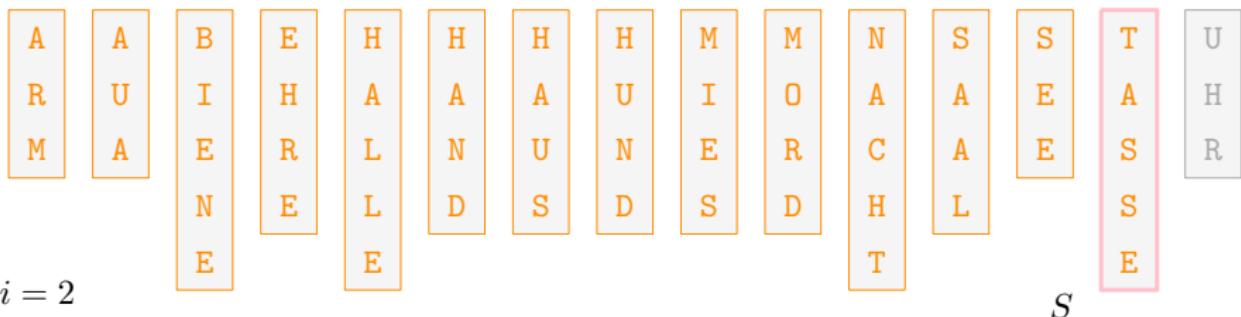
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

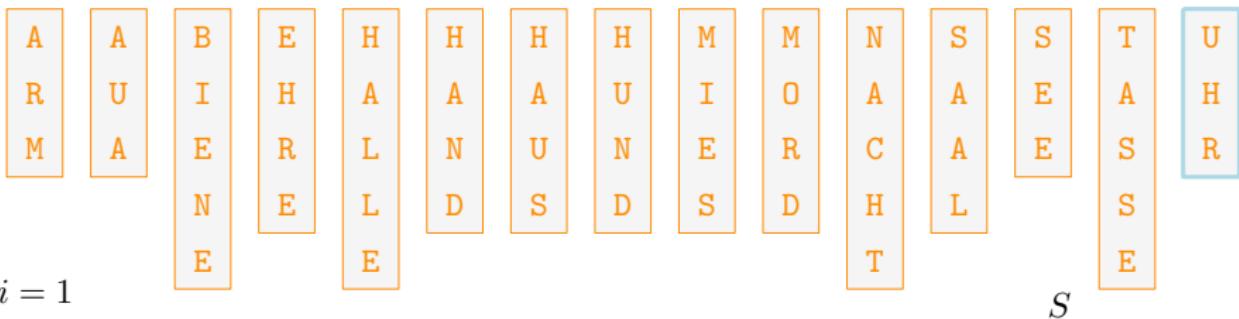
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)



in-place Multikey Quicksort

Ablauf

Function mkqSort(S : Array of String, i : Integer) : Array of String

if $|S| \leq 1$ **then return** S

(Basisfall)

choose $p \in S$ uniformly at random

(Pivotelement)

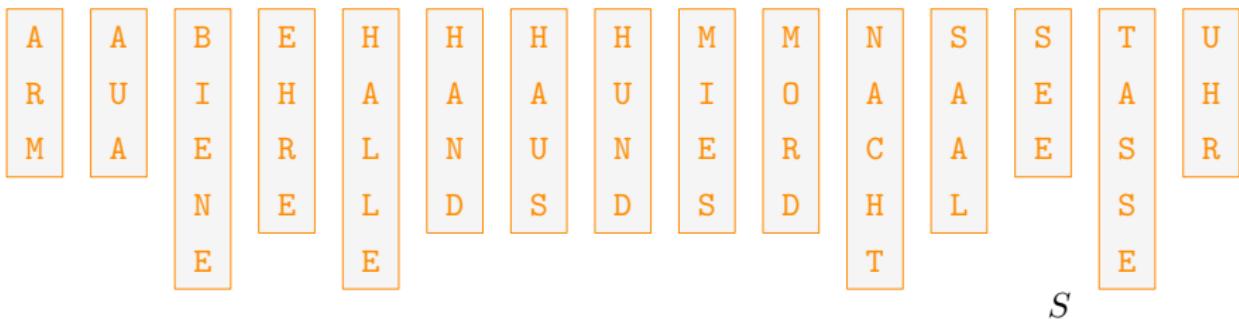
return concatenation of

(Rekursion)

mkqSort($\langle e \in S : e[i] < p[i] \rangle$, i),

mkqSort($\langle e \in S : e[i] = p[i] \rangle$, $i + 1$),

mkqSort($\langle e \in S : e[i] > p[i] \rangle$, i)

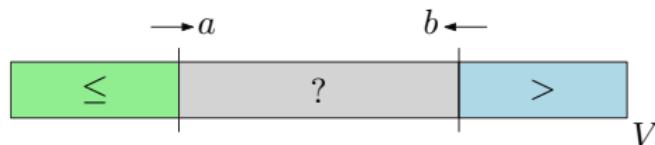


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

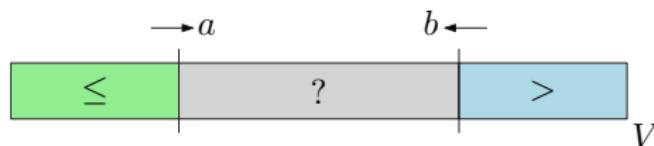


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
 - zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
-
- Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$



in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p,$
 $b \rightarrow b - 1$, solange $V[b] > p,$
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

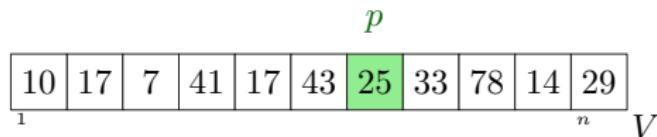
10	17	7	41	17	43	25	33	78	14	29
$_1$									$_n$	V

in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

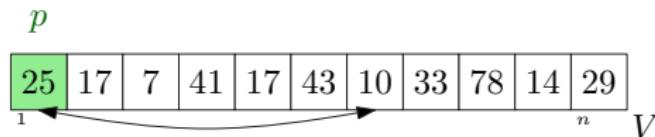


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$



in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p,$
 $b \rightarrow b - 1$, solange $V[b] > p,$
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

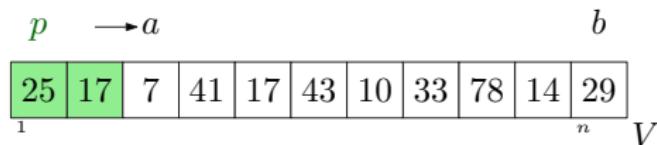
p	a									b
25	17	7	41	17	43	10	33	78	14	29

in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p,$
 $b \rightarrow b - 1$, solange $V[b] > p,$
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

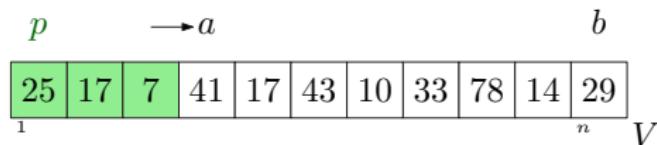


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p,$
 $b \rightarrow b - 1$, solange $V[b] > p,$
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

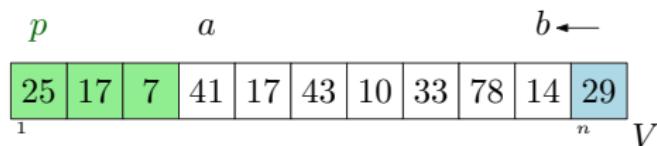


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

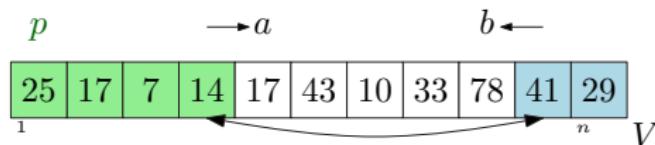


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

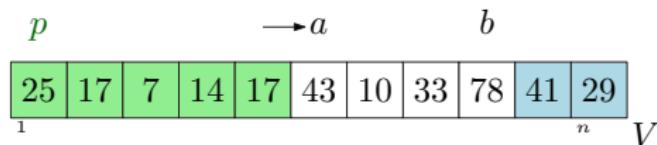


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

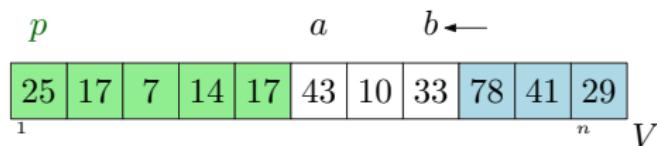


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p, V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p,$
 $b \rightarrow b - 1$, solange $V[b] > p,$
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

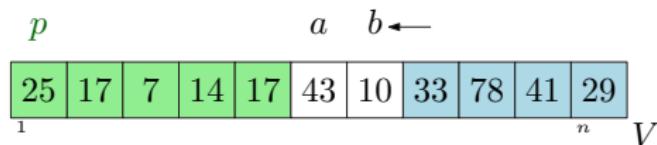


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

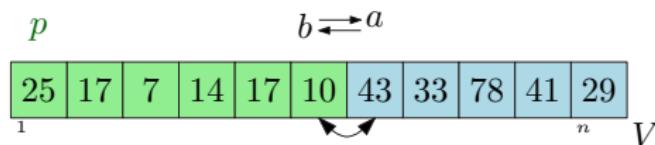


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

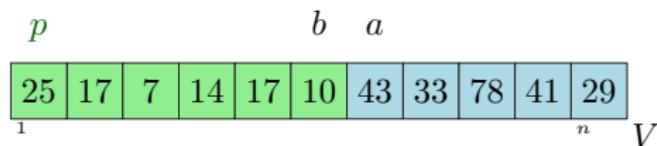


in-place Multikey Quicksort

Wiederholung: Partitionierung

in-place bei Quicksort (für Integer)

- teilt Elemente in kleiner gleich und größer als Pivotelement p
- zwei Zeiger a, b wandern von außen "in die Mitte"
→ Invariante: $V[i < a] \leq p$, $V[i > b] > p$
 - Wähle Pivot p und tausche mit erstem Element,
setze $a = 2, b = n$
 - $a \rightarrow a + 1$, solange $V[a] \leq p$,
 $b \rightarrow b - 1$, solange $V[b] > p$,
 - Tausch, wenn $V[a] > p$ und $V[b] \leq p$
 - Ende, wenn $a > b$

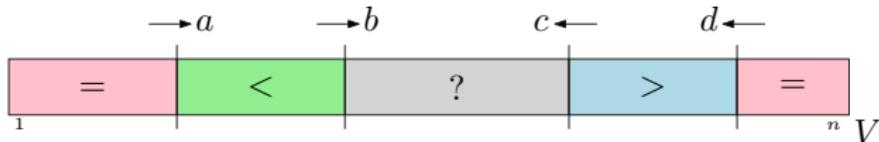


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort

- teilt Elemente in kleiner, gleich und größer als Pivotelement p
- zwei Zeiger b, c wandern von außen “in die Mitte”
- gleiche Elemente werden mit Zeiger a, d “außen” gesammelt
→ Invariante: $V[i \in [a, b] \text{ } a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d) \text{ } c \neq d] > p$

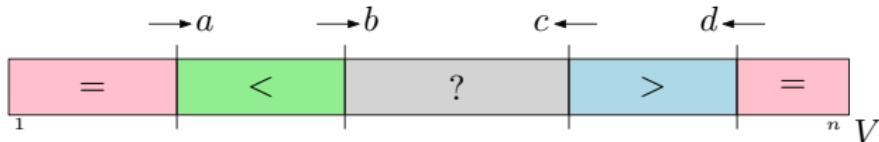


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort

- teilt Elemente in kleiner, gleich und größer als Pivotelement p
- zwei Zeiger b, c wandern von außen “in die Mitte”
- gleiche Elemente werden mit Zeiger a, d “außen” gesammelt
→ Invariante: $V[i \in [a, b) \ a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d] \ c \neq d] > p$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort

- teilt Elemente in kleiner, gleich und größer als Pivotelement p
- zwei Zeiger b, c wandern von außen “in die Mitte”
- gleiche Elemente werden mit Zeiger a, d “außen” gesammelt
→ Invariante: $V[i \in [a, b) \ a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d) \ c \neq d] > p$

1	S	B	E	H	A	M	T	M	H	S	H	A	H	U	N	n	S
	A	I	H	A	R	I	A	O	A	E	U	U	A	H	A		
	A	E	R	U	M	E	S	R	N	E	N	A	L	R	C		
	L	N	E	S		S	S	D	D		D		L		H		
	E					E						E			T		

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort

- teilt Elemente in kleiner, gleich und größer als Pivotelement p
- zwei Zeiger b, c wandern von außen “in die Mitte”
- gleiche Elemente werden mit Zeiger a, d “außen” gesammelt
→ Invariante: $V[i \in [a, b) \ a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d) \ c \neq d] > p$

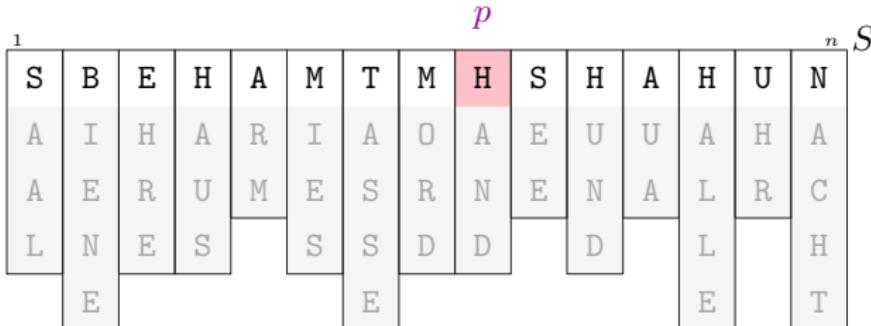
$V \hat{=}$	1	S	B	E	H	A	M	T	M	H	S	H	A	H	U	N	n	S
		A	I	H	A	R	I	A	O	A	E	U	U	A	H	A		
		A	E	R	U	M	E	S	R	N	E	N	A	L	R	C		
		L	N	E	S		S	S	D	D		D		L		H		
		E					E							E		T		

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

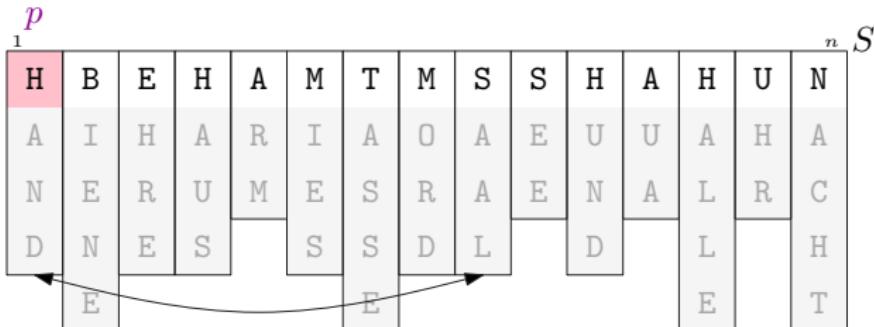


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

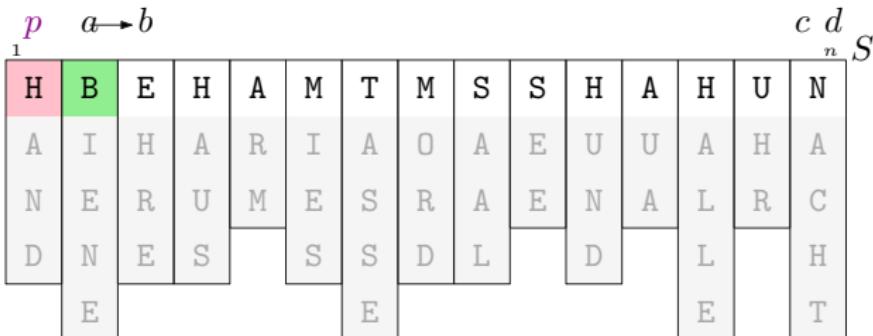
p	a	b	S																
1			c	d	n	S													
H	B	E	H	A	M	T	M	S	S	H	A	H	U	N					
A	I	H	A	R	I	A	O	A	E	U	U	A	H	A					
N	E	R	U	M	E	S	R	A	E	N	A	L	R	C					
D	N	E	S		S	S	D	L		D		L	L	H					
E					E						E			T					

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

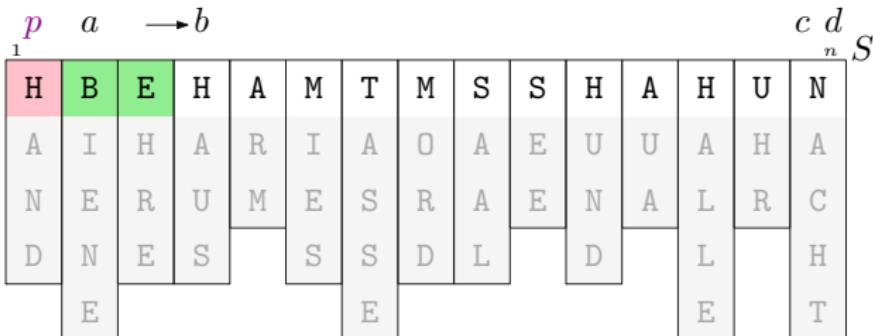


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

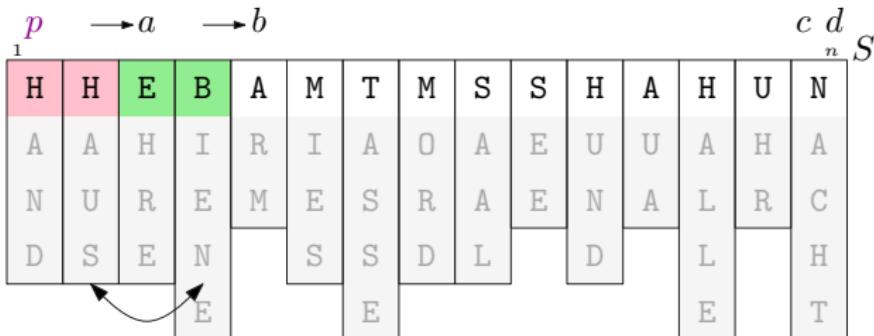
p	a	$\longrightarrow b$													c	d	S
1																	n
H	B	E	H	A	M	T	M	S	S	H	A	H	U	N			
A	I	H	A	R	I	A	O	A	E	U	U	A	H	A			
N	E	R	U	M	E	S	R	A	E	N	A	L	R	C			
D	N	E	S		S	S	D	L		D		L	L	H			
E					E						E			T			

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

<i>p</i>	<i>a</i>	$\rightarrow b$		<i>c</i>		<i>d</i>	<i>S</i>
1						n	
H	H	E	B	A	M	T	M
A	A	H	I	R	I	A	O
N	U	R	E	M	E	S	R
D	S	E	N		S	D	L
		E			E		

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

p	a	b	$c \leftarrow d$		n
H	H	E	M	T	S
A	A	H	I	O	E
N	U	R	M	A	H
D	S	E	N	S	A
		E	S	D	L
			E	D	T
				L	
				E	
					S

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

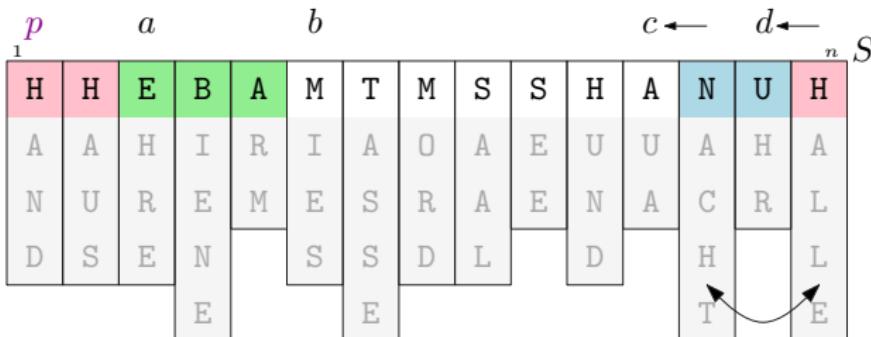
p	a	b			$c \leftarrow$	d	n	S
H	H	E	B	A	M	T	M	S
A	A	H	I	R	I	A	O	A
N	U	R	E	M	E	S	R	A
D	S	E	N		S	S	D	L
		E			E		D	
							L	
							E	
							T	

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

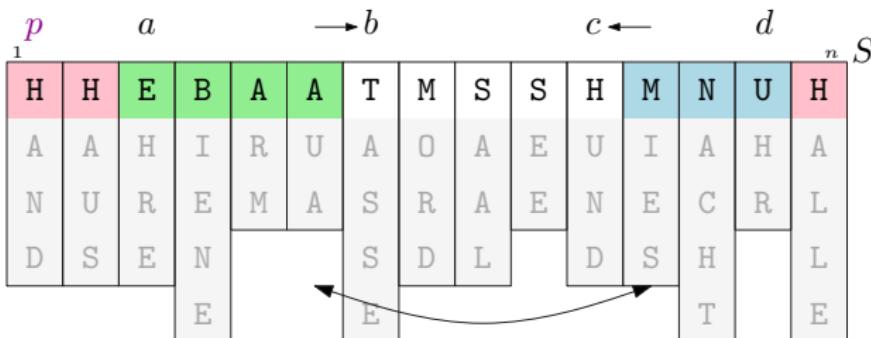


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

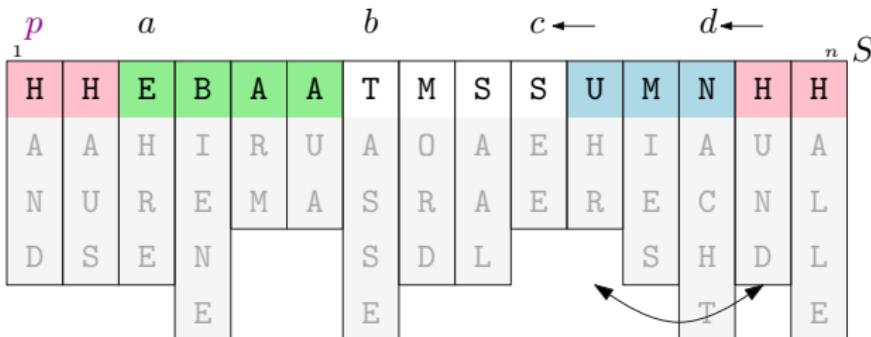


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

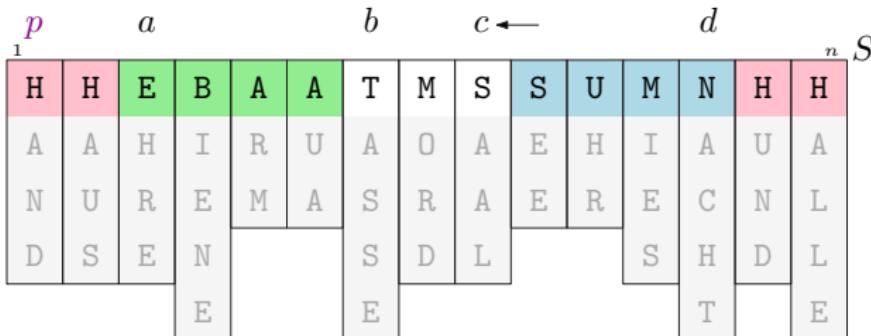


in-place Multikey Quicksort

Partitionierung

***in-place* bei Multikey Quicksort (Algorithmus)**

- Wähle Pivot p und tausche mit erstem Element,
setze $a = b = 2, c = d = n$
 - $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$,
 $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
 - Tausch, wenn $V[b] > p$ und $V[c] < p$
 - Ende, wenn $b > c$

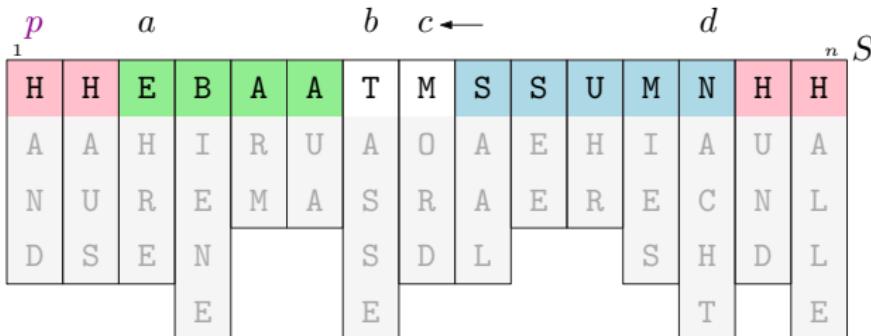


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

p	a	b	$c \leftarrow$	d	n	S
H	H	E	B	A	A	
A	A	H	I	R	U	
N	U	R	E	M	A	
D	S	E	N			
		E		S	D	L
			E		T	E

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Algorithmus)

- Wähle Pivot p und tausche mit erstem Element, setze $a = b = 2, c = d = n$
- $b \rightarrow b + 1$, solange $V[b] \leq p$, wenn $V[b] = p$: Tausch mit $V[a]$, $a \rightarrow a + 1$, $c \rightarrow c - 1$, solange $V[c] \geq p$, wenn $V[c] = p$: Tausch mit $V[d]$, $d \rightarrow d - 1$
- Tausch, wenn $V[b] > p$ und $V[c] < p$
- Ende, wenn $b > c$

p	a	$c \leftarrow b$	d	n
H	H	E	B	A
A	A	H	I	R
N	U	R	E	M
D	S	E	N	
		E		
			S	
			D	
			L	
			E	
				S
				H
				H

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Umgruppierung)

- $r = \min(a - 1, b - a)$
Tausch von r Zeichen zwischen $[1, r)$ und $[b - r, b)$
- $r = \min(d - c, n - d)$
Tausch von r Zeichen zwischen $[c + 1, c + r)$ und $[n - r + 1, n + 1)$

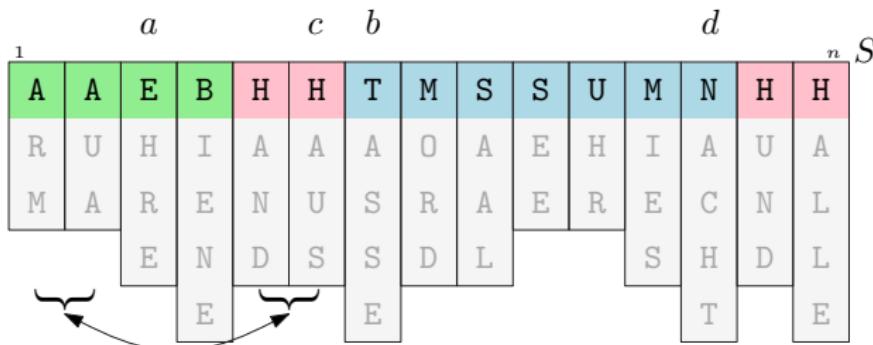
1	a		c		b		d		n		S			
H	H	E	B	A	A	T	M	S	S	U	M	N	H	H
A	A	H	I	R	U	A	O	A	E	H	I	A	U	A
N	U	R	E	M	A	S	R	A	E	R	E	C	N	L
D	S	E	N			S	D	L			S	H	D	L
		E				E					T		E	

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Umgruppierung)

- $r = \min(a - 1, b - a)$
Tausch von r Zeichen zwischen $[1, r)$ und $[b - r, b)$
- $r = \min(d - c, n - d)$
Tausch von r Zeichen zwischen $[c + 1, c + r)$ und $[n - r + 1, n + 1)$



in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Umgruppierung)

- $r = \min(a - 1, b - a)$
Tausch von r Zeichen zwischen $[1, r)$ und $[b - r, b)$
- $r = \min(d - c, n - d)$
Tausch von r Zeichen zwischen $[c + 1, c + r)$ und $[n - r + 1, n + 1)$

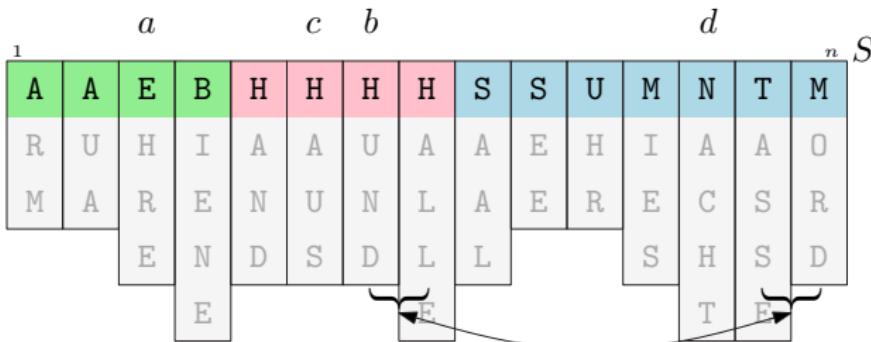
a	c	b	d
A	H	T	M
R	A	O	N
M	S	E	H
A	R	H	A
R	E	E	U
E	N	S	L
E	D	S	L
E	S	D	L
E	T	H	E

in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Umgruppierung)

- $r = \min(a - 1, b - a)$
Tausch von r Zeichen zwischen $[1, r)$ und $[b - r, b)$
- $r = \min(d - c, n - d)$
Tausch von r Zeichen zwischen $[c + 1, c + r)$ und $[n - r + 1, n + 1)$

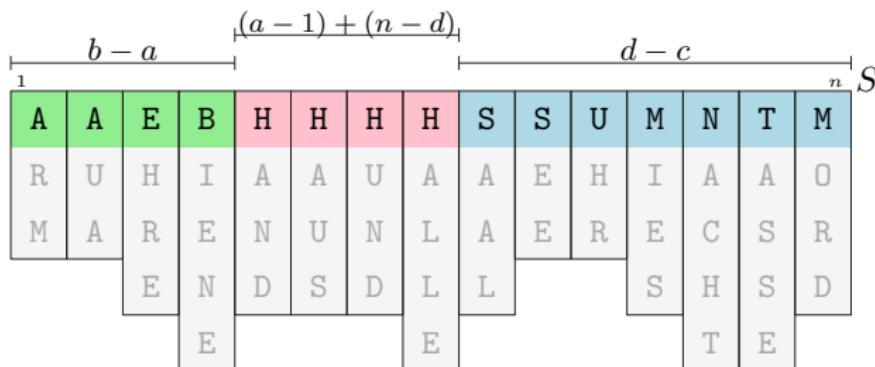


in-place Multikey Quicksort

Partitionierung

in-place bei Multikey Quicksort (Umgruppierung)

- $r = \min(a - 1, b - a)$
Tausch von r Zeichen zwischen $[1, r)$ und $[b - r, b)$
- $r = \min(d - c, n - d)$
Tausch von r Zeichen zwischen $[c + 1, c + r)$ und $[n - r + 1, n + 1)$



in-place Multikey Quicksort

Zusammenfassung

- *Three-way Radix Quicksort*

(Partitionierung in kleiner, gleich, größer über alle Stellen analog zu msd-Radixsort)

- effizient $\mathcal{O}(|S| \log |S| + d)$

($d \triangleq$ Summe der Länge der unterscheidenden Präfixe)

- *in-place* Partitionierung möglich

(durch geschicktes Speichern und Verschieben der gleichen Elemente)

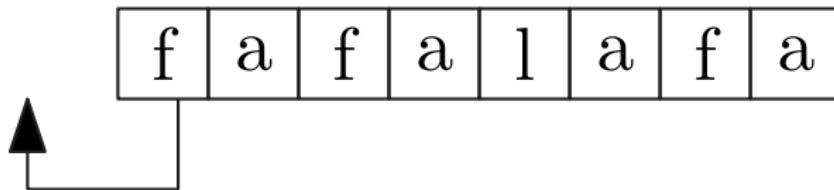
- sehr einfache Implementierung

(nur 40 Zeilen Quellcode, siehe Anhang)

KMP Algorithmus

Border Array

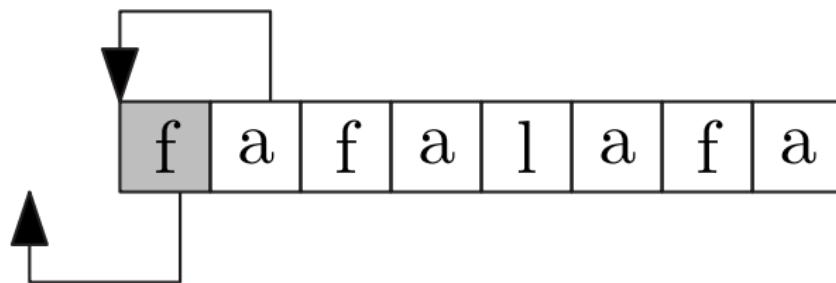
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

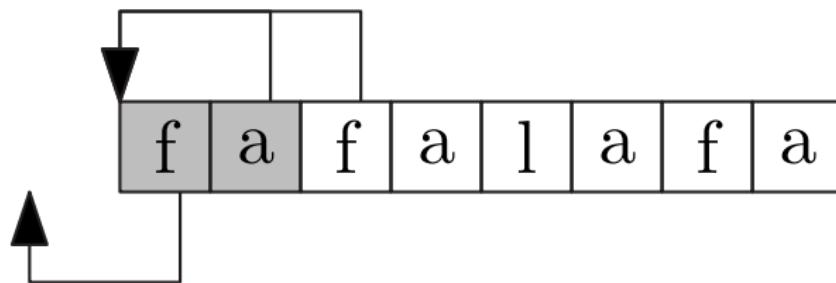
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

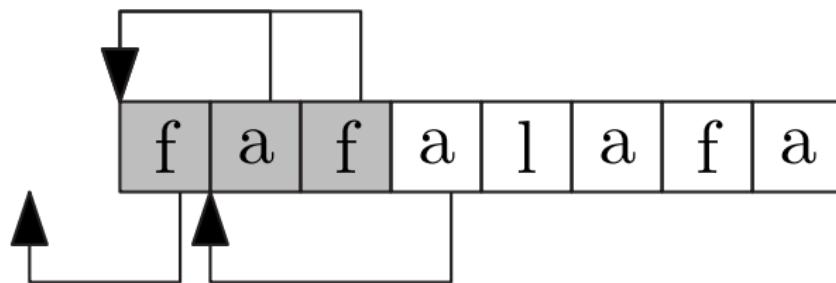
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

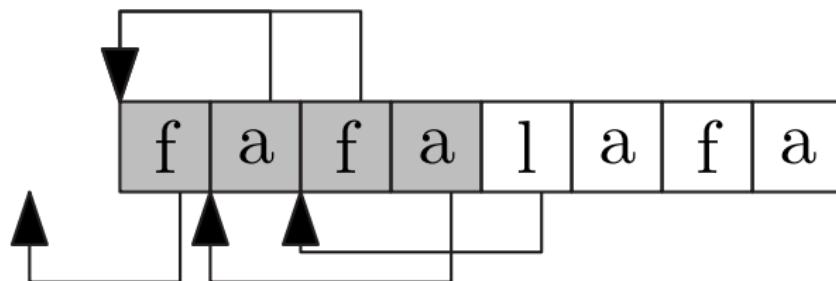
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

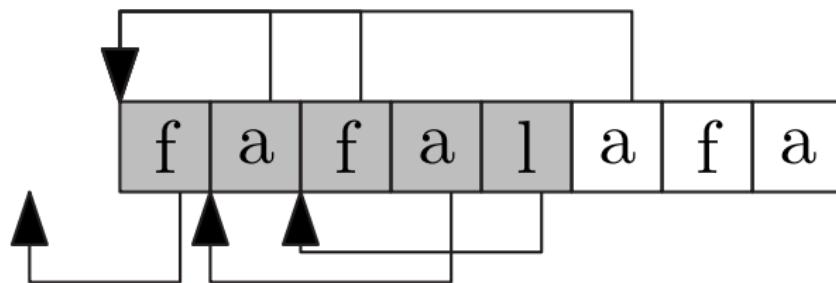
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

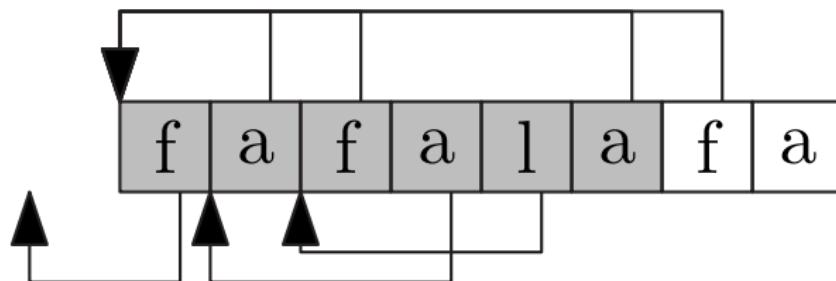
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

Border Array

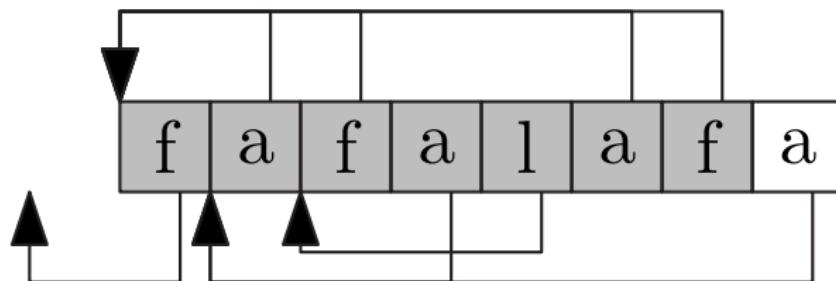
- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



KMP Algorithmus

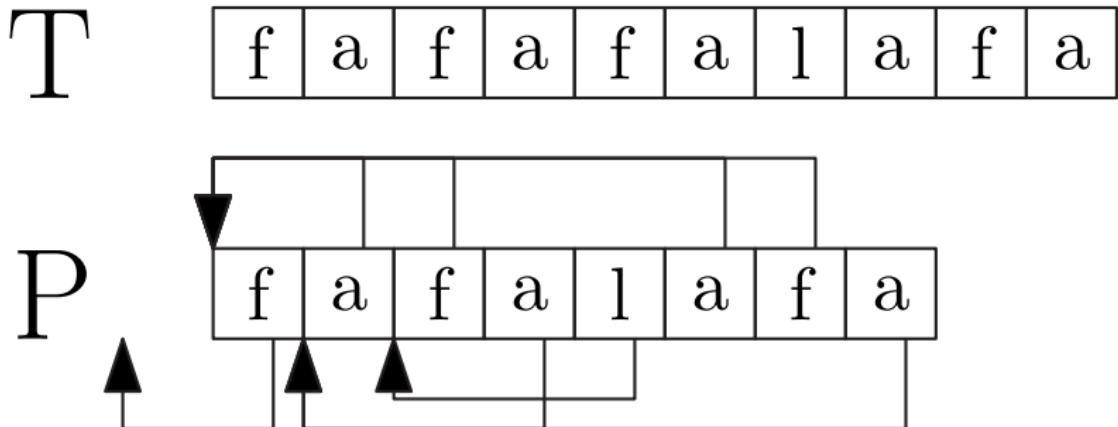
Border Array

- Pattern P
- $\text{border}[j] = \text{Länge des Präfixes von } P_{1\dots j-1}, \text{ das auch echtes Suffix von } P_{1\dots j-1} \text{ ist}$



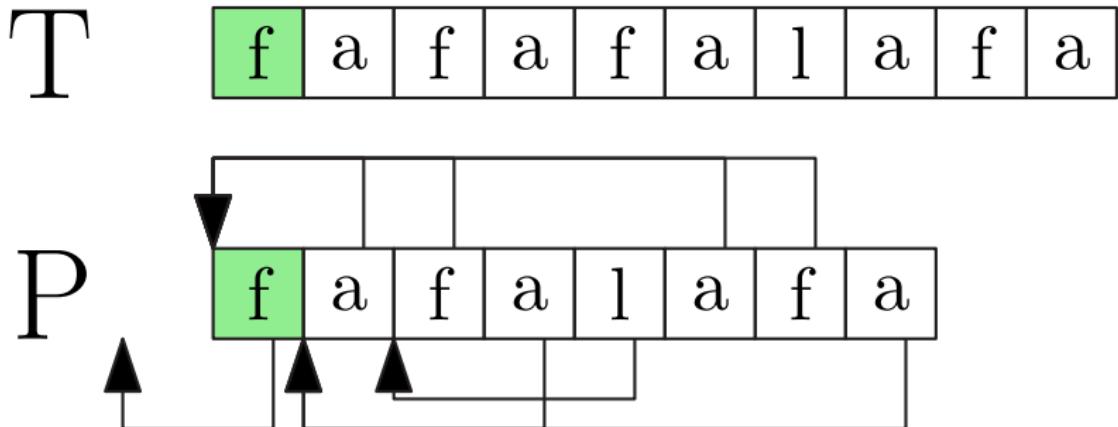
KMP Algorithmus

Pattern Suche



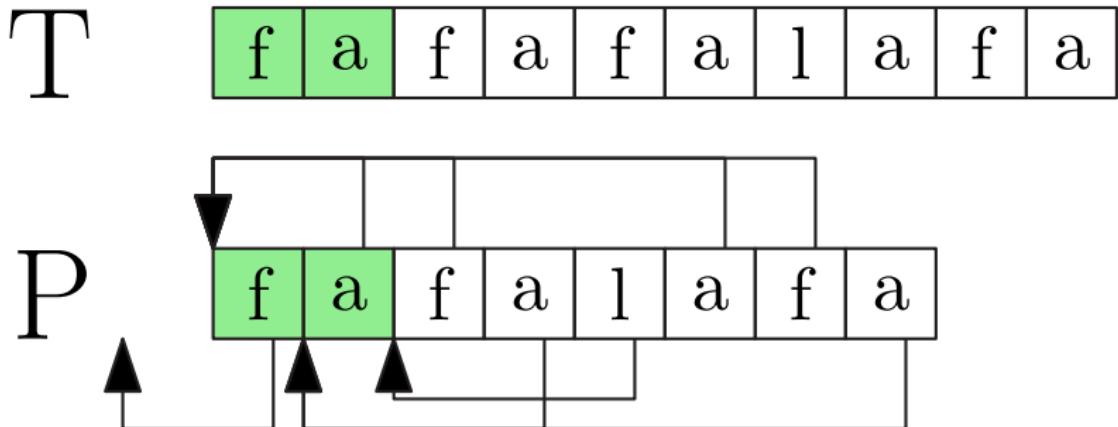
KMP Algorithmus

Pattern Suche



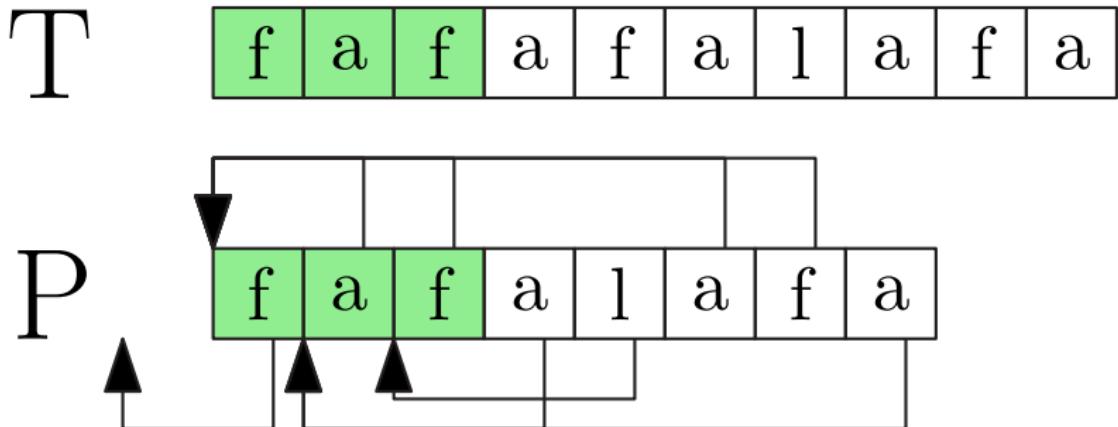
KMP Algorithmus

Pattern Suche



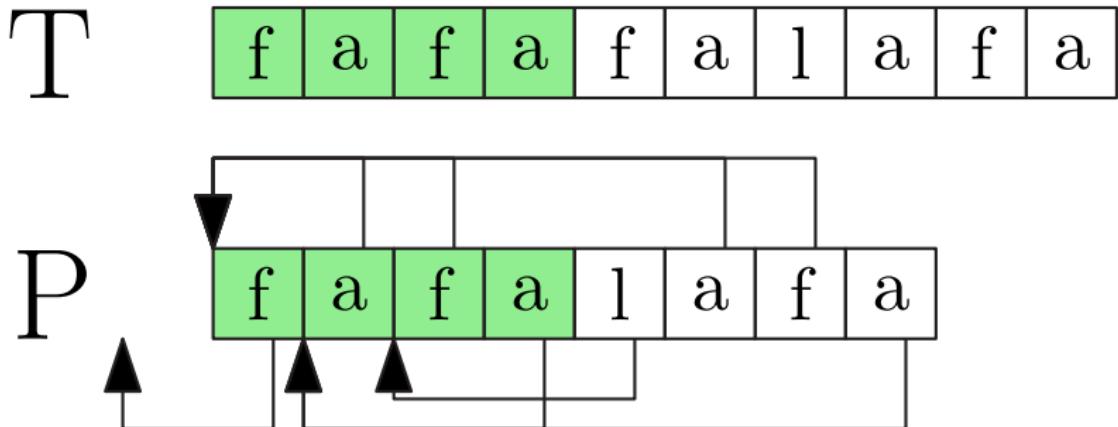
KMP Algorithmus

Pattern Suche



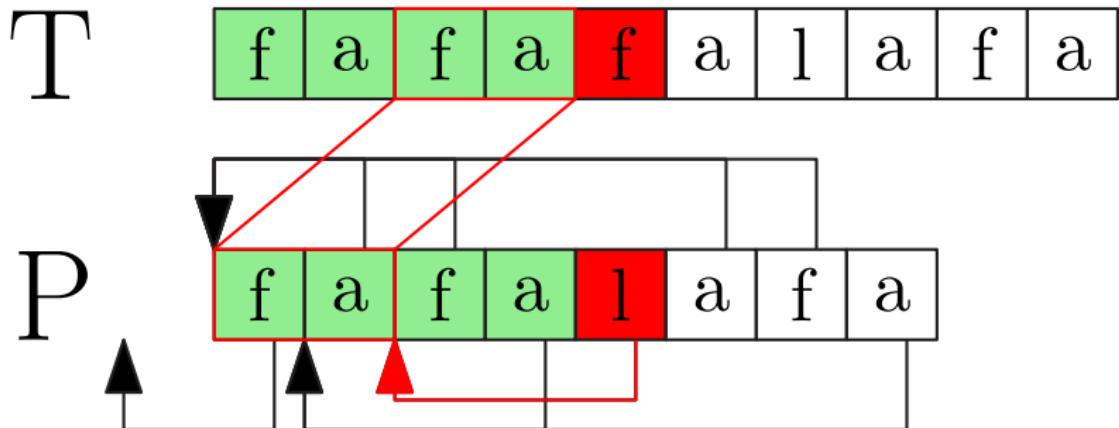
KMP Algorithmus

Pattern Suche



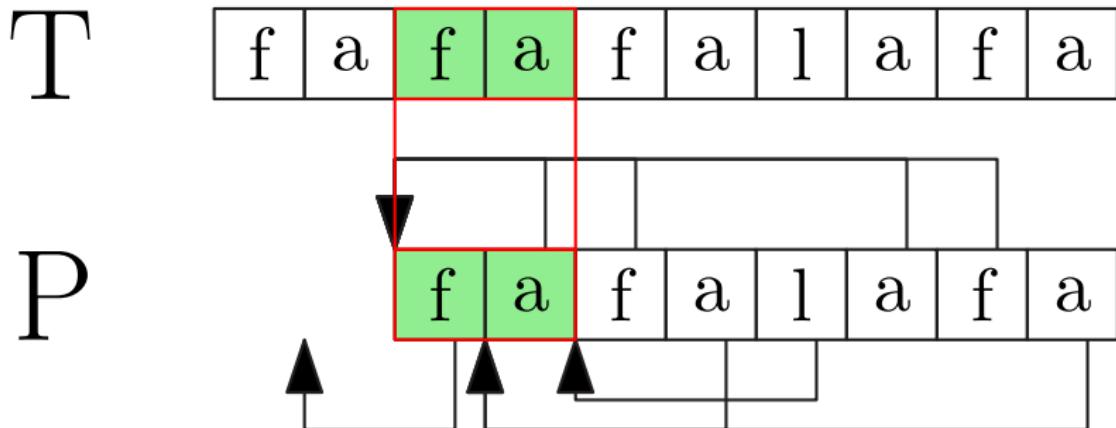
KMP Algorithmus

Pattern Suche



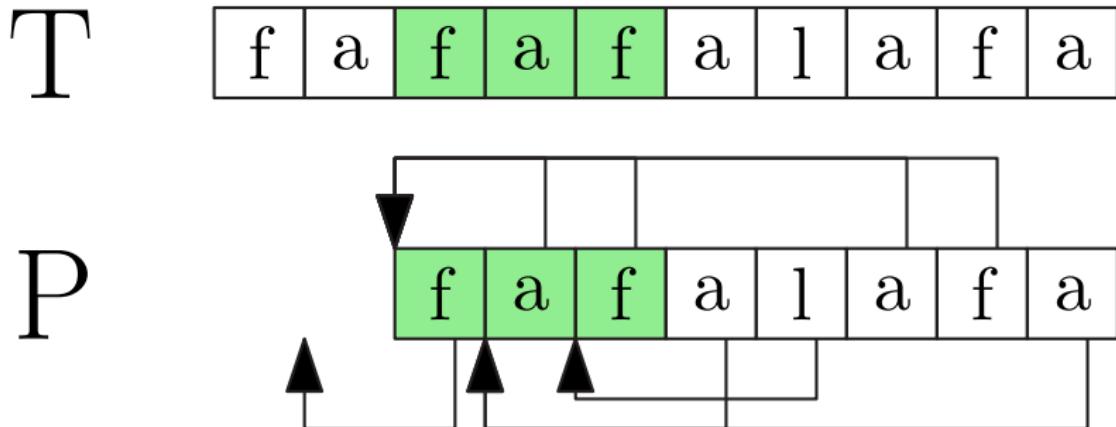
KMP Algorithmus

Pattern Suche



KMP Algorithmus

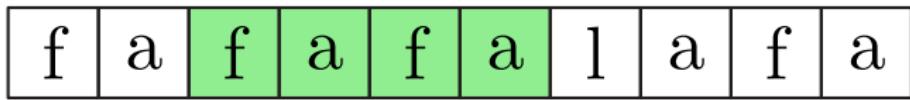
Pattern Suche



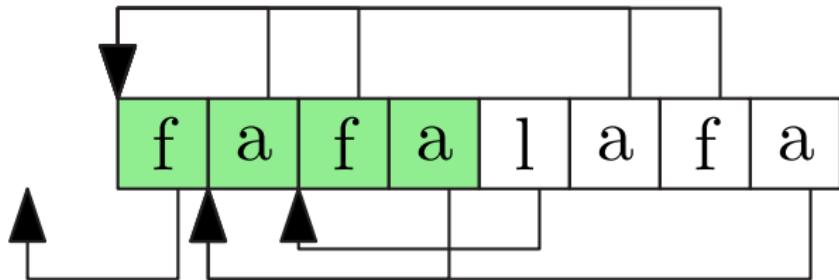
KMP Algorithmus

Pattern Suche

T



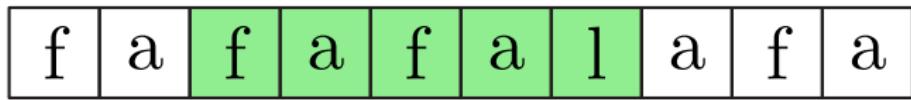
P



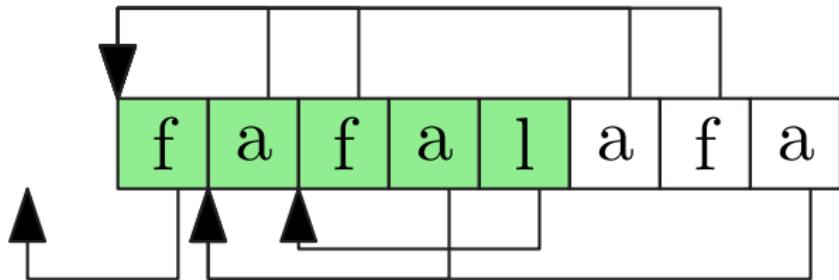
KMP Algorithmus

Pattern Suche

T



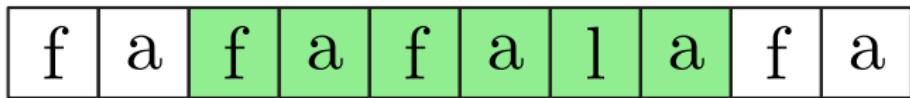
P



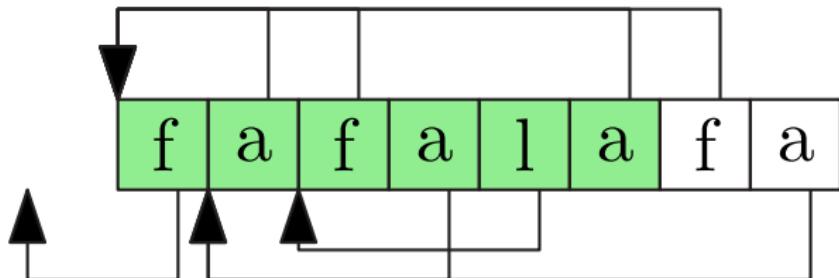
KMP Algorithmus

Pattern Suche

T



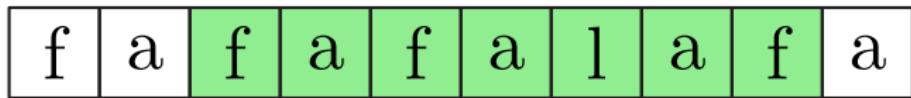
P



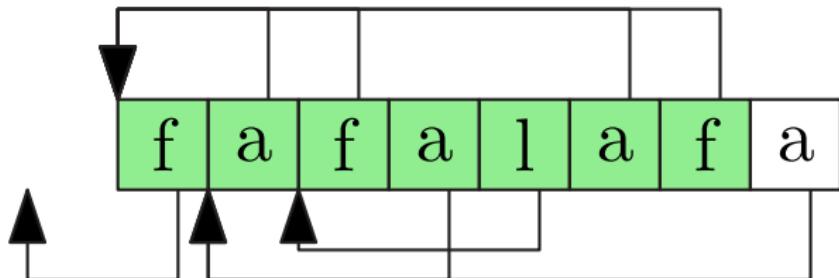
KMP Algorithmus

Pattern Suche

T



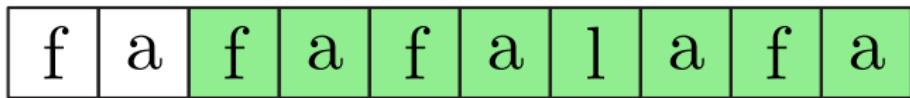
P



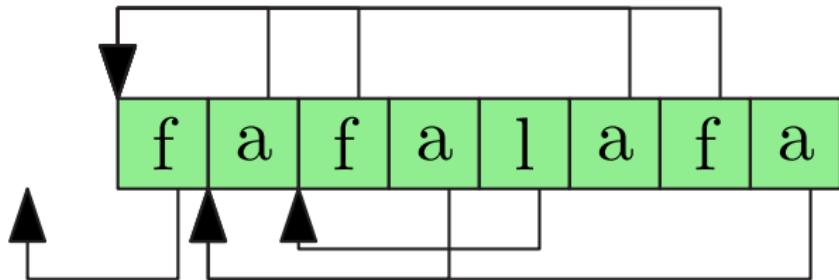
KMP Algorithmus

Pattern Suche

T



P



Evaluation der Übung

<https://onlineumfrage.kit.edu/evasys/online.php?p=QSDQE>
(bis 18 Uhr)



Feierabend!

in-place Multikey Quicksort

Definitionen

- nach C-Standard ist ein string ist Zeiger auf char
(die Daten werden als char Array mit *Sentinel* '\0' gespeichert)

```
void swap(int a, int b, char *x[]){
    char *t=x[a];
    x[a]=x[b];
    x[b]=t;
}
```

```
#define i2c(i) x[i][depth] //Buchstabe an Stelle 'depth' aus i-tem String
```

```
void vecswap(int i, int j, int n, char *x[]) {
    while(n--> 0) {
        swap(i,j);
        i++; j++;
    }
}
```

in-place Multikey Quicksort Algorithmus

```
void mkqsort( char *x[], int n, int depth){  
    if(n <= 1) return;  
    a = rand() % n;  
    swap(0,a,x);  
    v=i2c(0);  
    a = b = 1;  
    c = d = n-1;  
    while(true) {  
        while(b <= c && (r = i2c(b) -v) <= 0) {  
            if(r == 0) { swap(a,b,x); a++ }  
            b++;  
        }  
        while (b <= c && (r= i2c(c) -v) >= 0) {  
            if(r == 0) { swap(c,d,x); d--; }  
            c--;  
        }  
        if(b>c) break;  
        swap(b,c,x); b++; c--;  
    }  
    r = min(a, b-a); vecswap(0, b-r, r, x);  
    r = min(d-c, n-d-1); vecswap(b, n-r, r, x);  
    r = b-a; mkqsort(x, r, depth);  
    if(i2c(r) != 0) { mkqsort(x+r, a+n-d-1, depth+1); } // "mittlere" Rekursion, falls notwendig  
    r = d-c; mkqsort(x+n-r, r, depth);  
}
```

// Weniger als zwei Strings zu sortieren?
// Wahl des Pivot-Elements
// Pivot an den Anfang des Arrays verschieben
// Wert der aktuellen Stelle im Pivot
// Anfang des zu sortierenden (Sub-)Arrays
// Ende des zu sortierenden (Sub-)Arrays

// Von links Elemente scannen, bis Char > Pivot
// Char == Pivot? Dann nach links kopieren
// Grenze der gleichen Elemente eins nach rechts

// Von rechts Elemente scannen, bis Char < Pivot
// Char == Pivot? Dann nach rechts kopieren
// Grenze der gleichen Elemente eins nach links

// Zeiger aneinander vorbei? Dann Abbruch
// Inversion gefunden, tauschen, Zeiger je 1 weiter

// "gleiche" Elemente von links zur Mitte kopieren
// "gleiche" Elemente von rechts zur Mitte kopieren
// "linke" Rekursion
// "mittlere" Rekursion, falls notwendig
// "rechte" Rekursion

[Bentley&Sedgewick1997]