

Übung 3 – Algorithmen II

Hans-Peter Lehmann, Daniel Seemaier – {hans-peter.lehmann, daniel.seemaier}@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS21.php

Institut für Theoretische Informatik - Algorithmik II

```
    result < current_weight;
    return false;
}

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 );
            }
        }
    }
}
```

Themenübersicht

- *in-place Multikey Quicksort*
Sortierung von Zeichenketten *in-place*
- Suche mit Hilfe von Suffix-Arrays
 - Beschleunigung mittels LCP-Array

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

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)

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

Multikey Quicksort

Ablauf

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

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

Basisfall

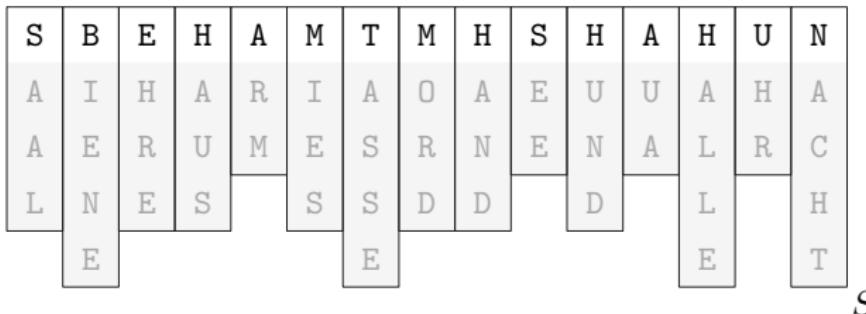
choose $p \in S$ uniformly at random

Pivotelement

return concatenation of

Rekursion

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



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 | R | | H |
| | E | | | | | E | | | | | E | | | T |

$i = 1$

S

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

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($\{e \in S : e[i] < p[i]\}$, i),

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

mkqSort($\{e \in S : e[i] > p[i]\}$, 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

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 |
| 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

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($\{e \in S : e[i] < p[i]\}$, i),

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

mkqSort($\{e \in S : e[i] > p[i]\}$, 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

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

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 |

| |
|---|
| E |
| H |
| R |
| E |

| | | | |
|---|---|---|---|
| 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

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 | |

| |
|---|
| H |
| |
| A |
| A |
| U |
| S |
| D |
| D |
| L |
| 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 |

$i = 2$

S

Multikey Quicksort

Ablauf

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

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

Basisfall

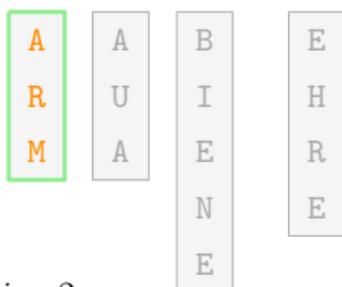
choose $p \in S$ uniformly at random

Pivotelement

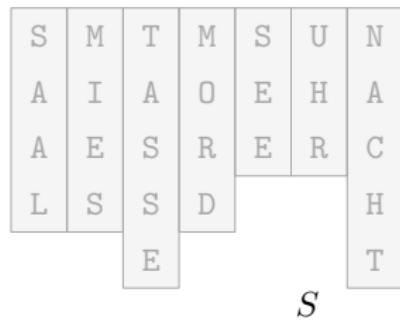
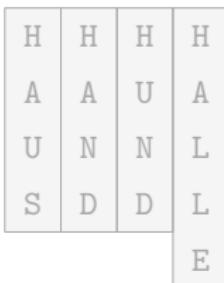
return concatenation of

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

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

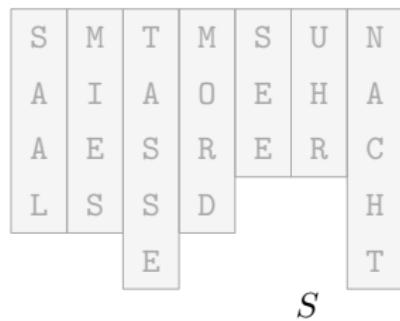
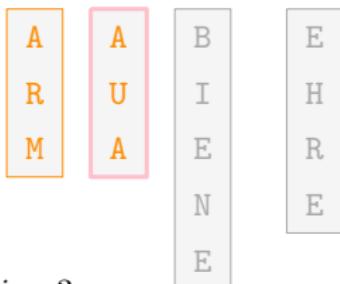
return concatenation of

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)



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

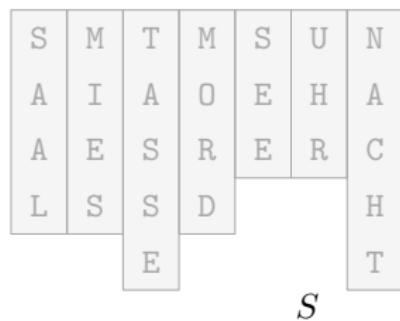
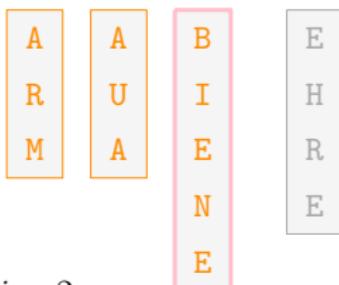
return concatenation of

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)



Multikey Quicksort

Ablauf

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

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

Basisfall

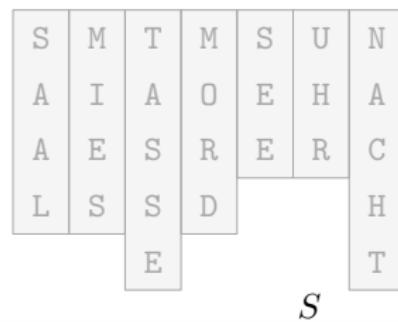
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

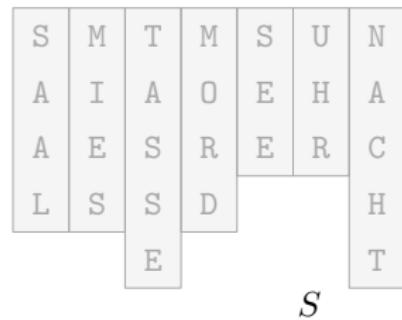
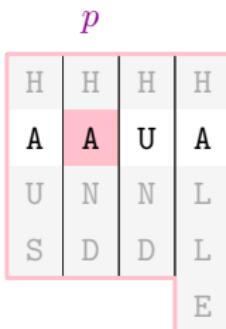
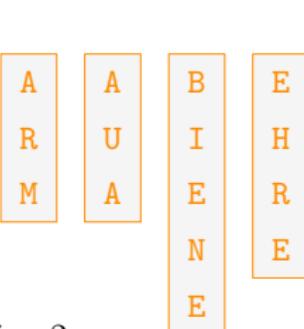
choose $p \in S$ uniformly at random

Pivotelement

return concatenation of

Rekursion

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



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

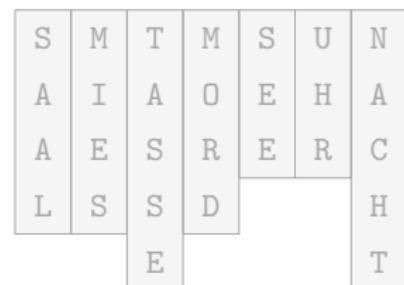
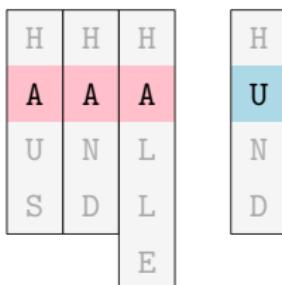
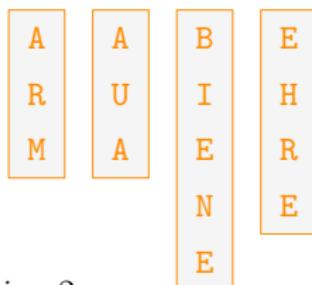
Pivotelement

return concatenation of

Rekursion

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

p



Multikey Quicksort

Ablauf

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

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

Basisfall

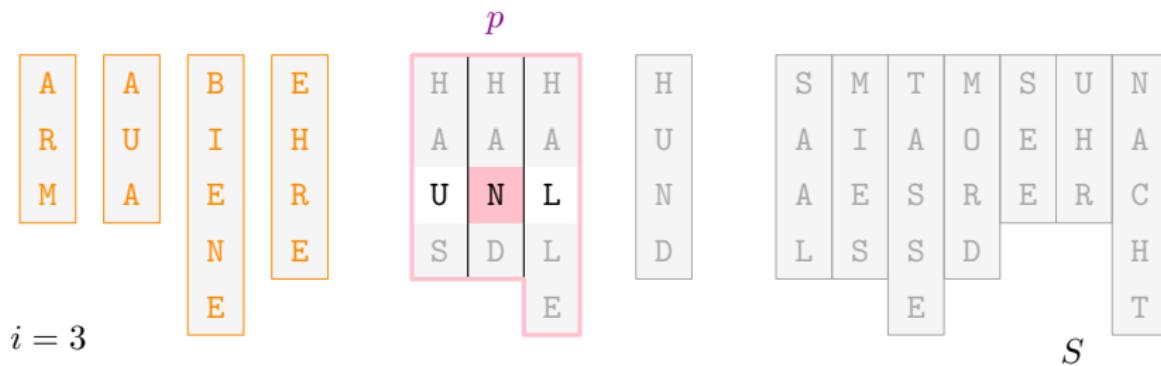
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

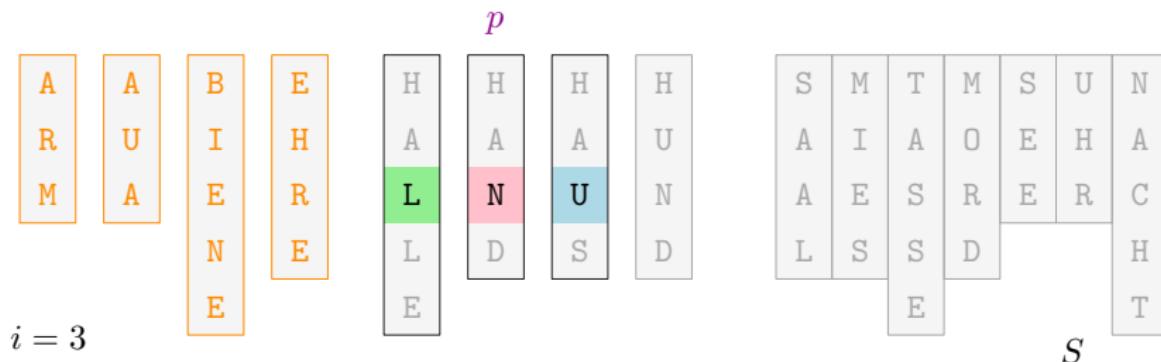
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

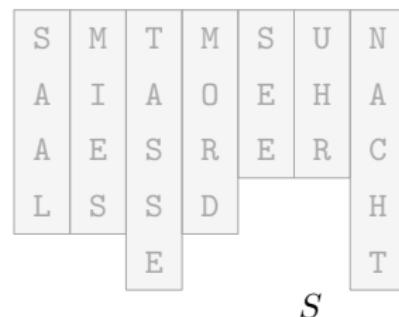
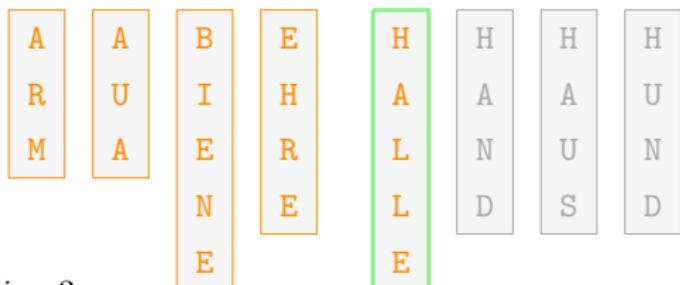
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

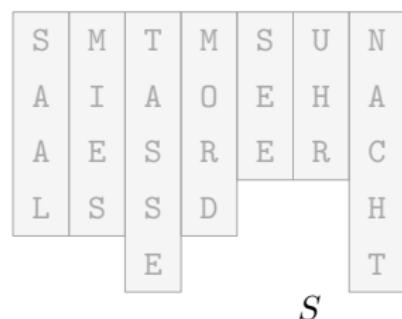
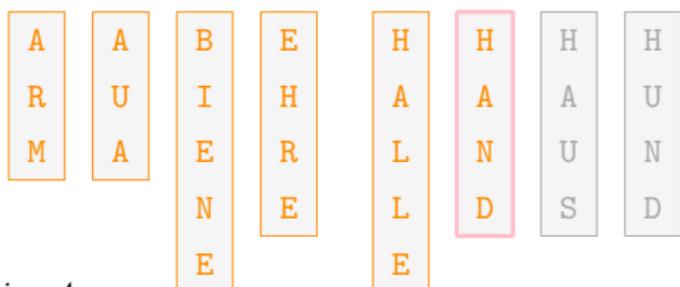
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

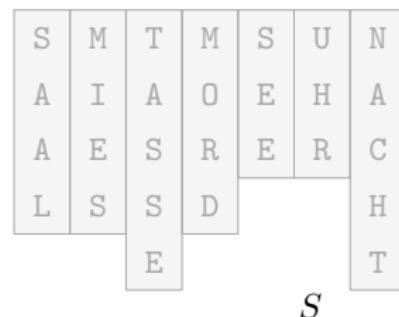
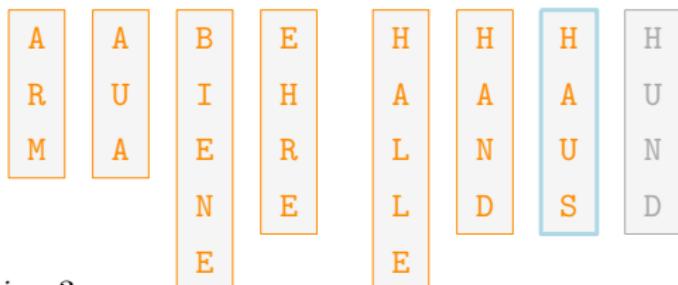
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

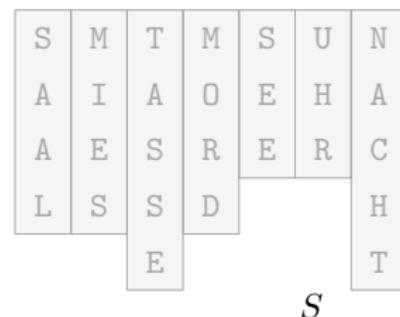
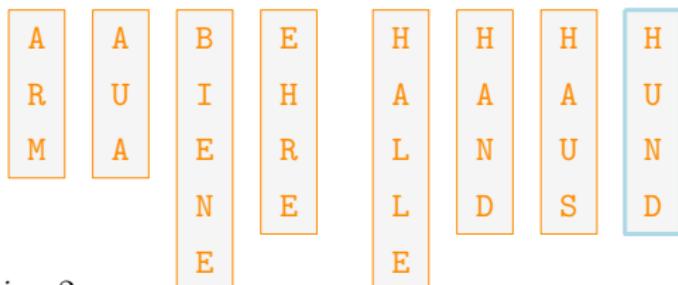
choose $p \in S$ uniformly at random

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



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

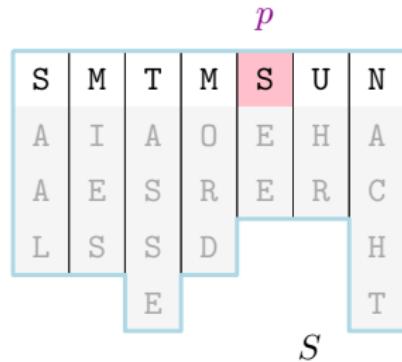
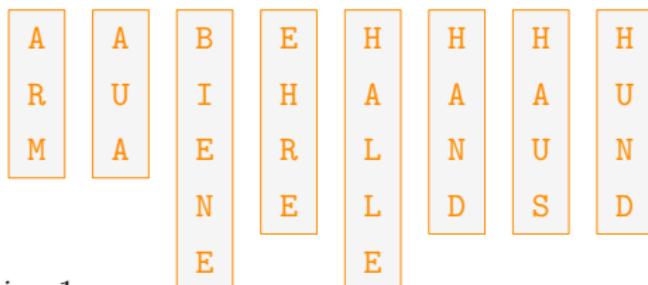
return concatenation of

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)



Multikey Quicksort

Ablauf

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

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

Basisfall

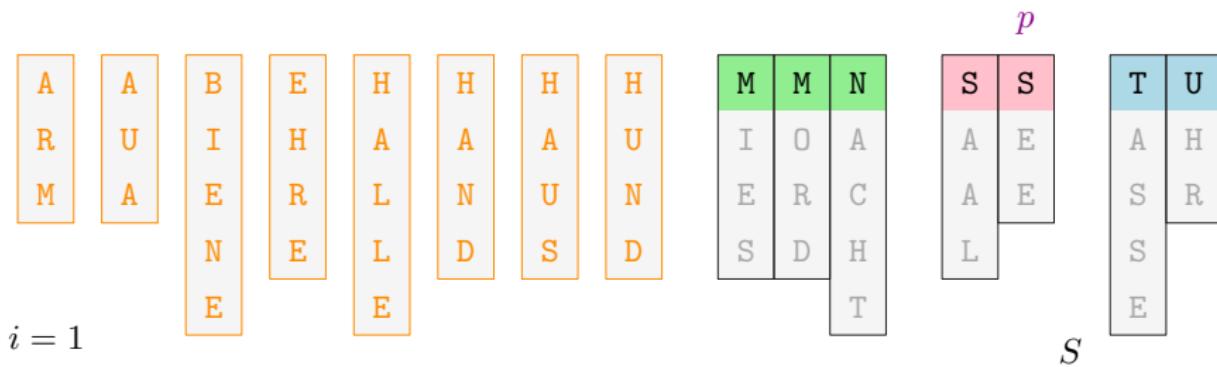
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

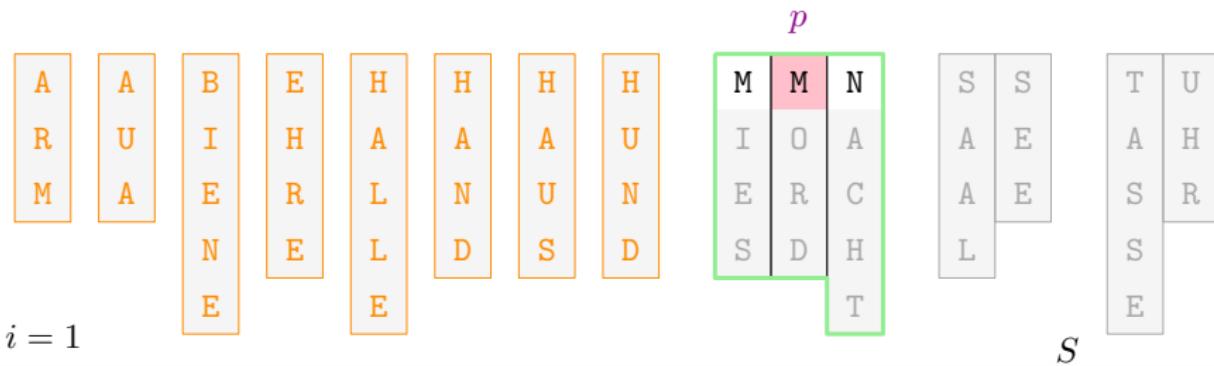
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

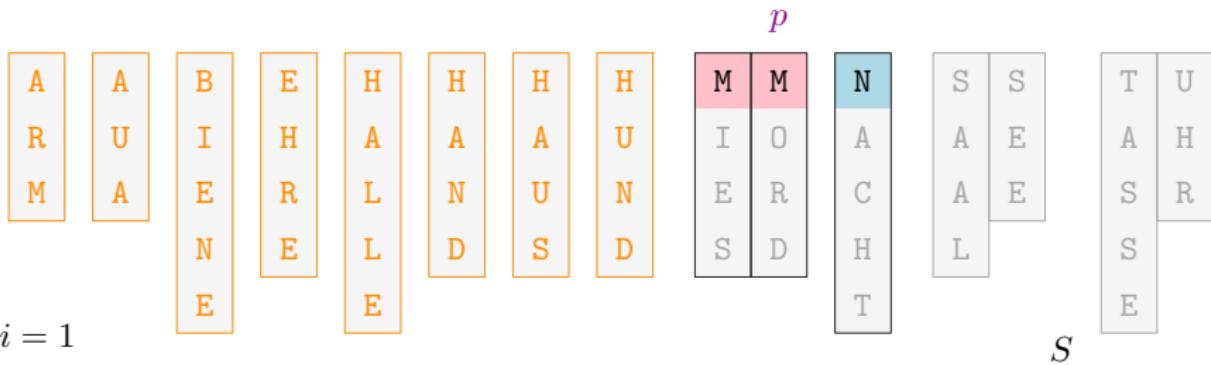
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

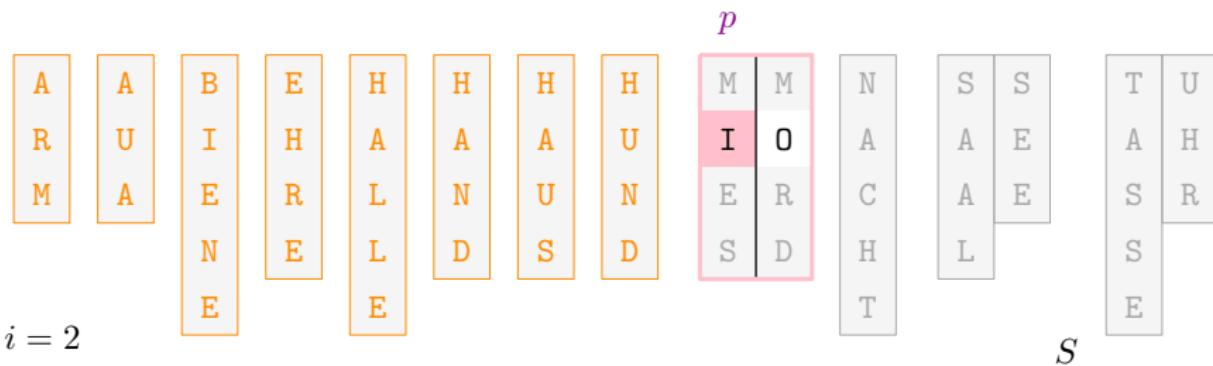
choose $p \in S$ uniformly at random

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



$i = 2$

Multikey Quicksort

Ablauf

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

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

Basisfall

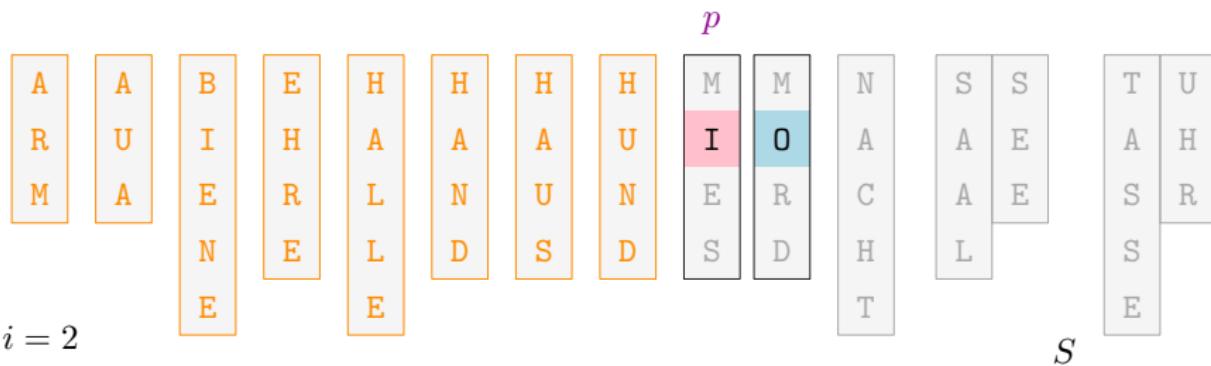
choose $p \in S$ uniformly at random

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



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



Multikey Quicksort

Ablauf

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

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

Basisfall

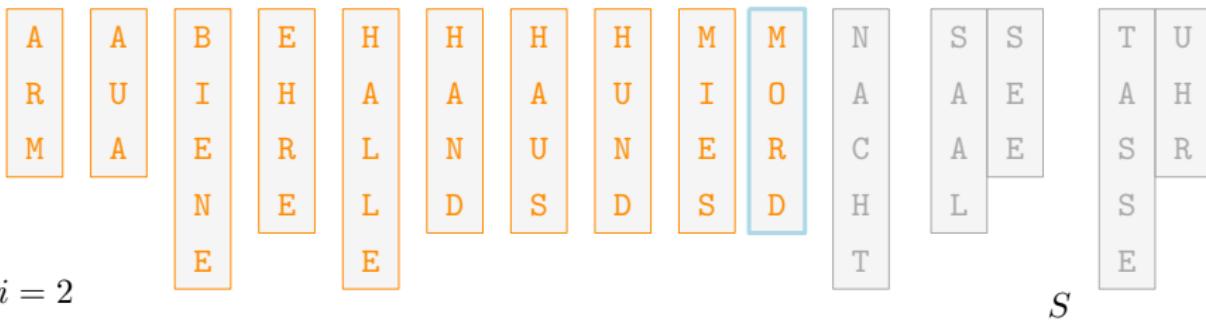
choose $p \in S$ uniformly at random

Pivotelement

return concatenation of

Rekursion

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



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

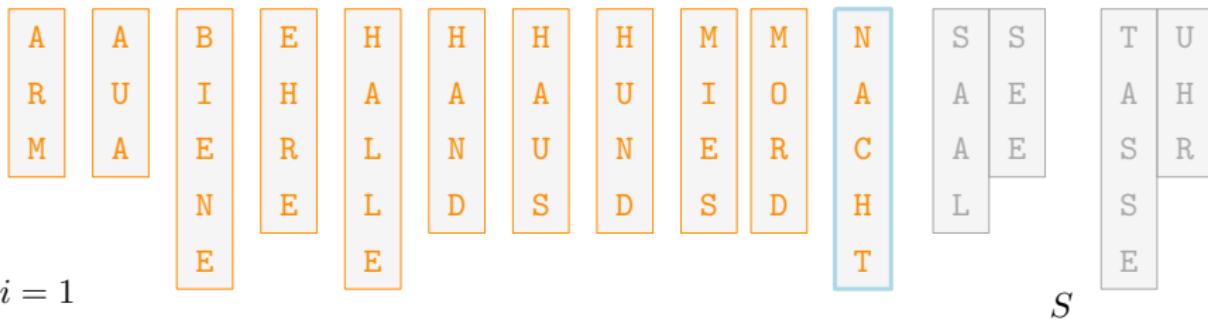
return concatenation of

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)



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

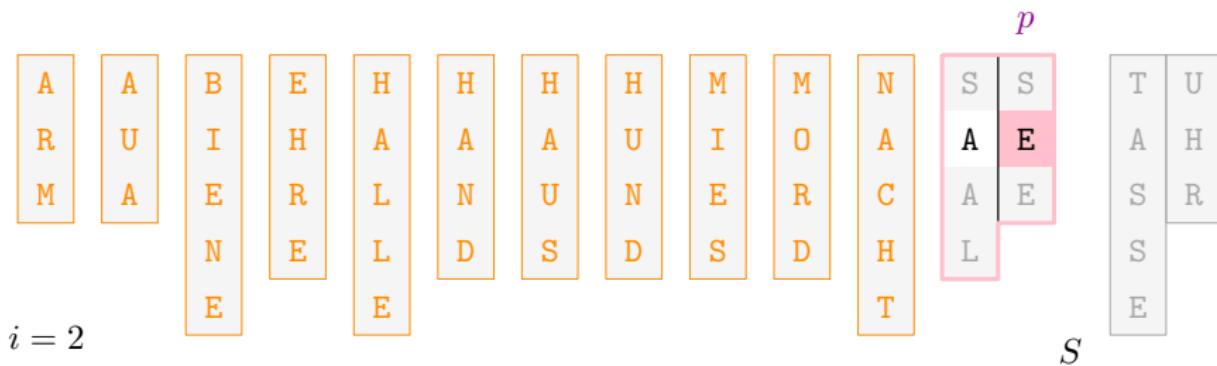
return concatenation of

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)



Multikey Quicksort

Ablauf

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

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

Basisfall

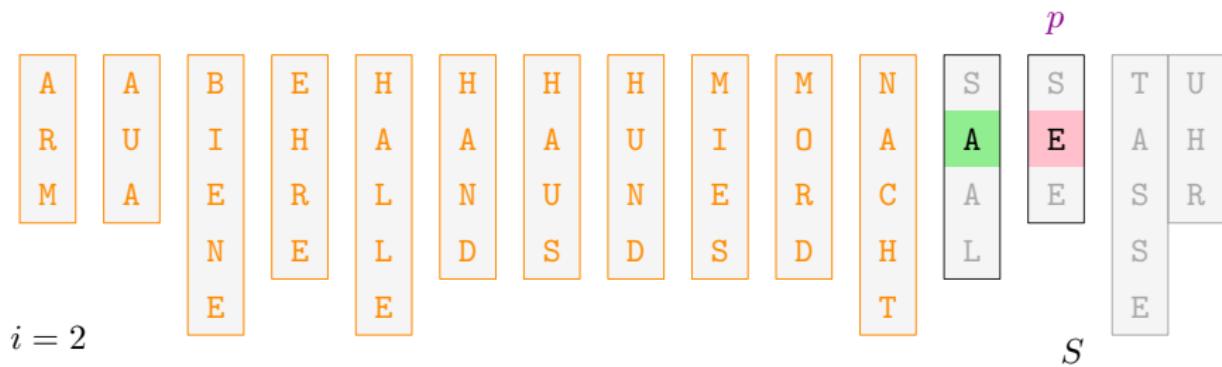
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

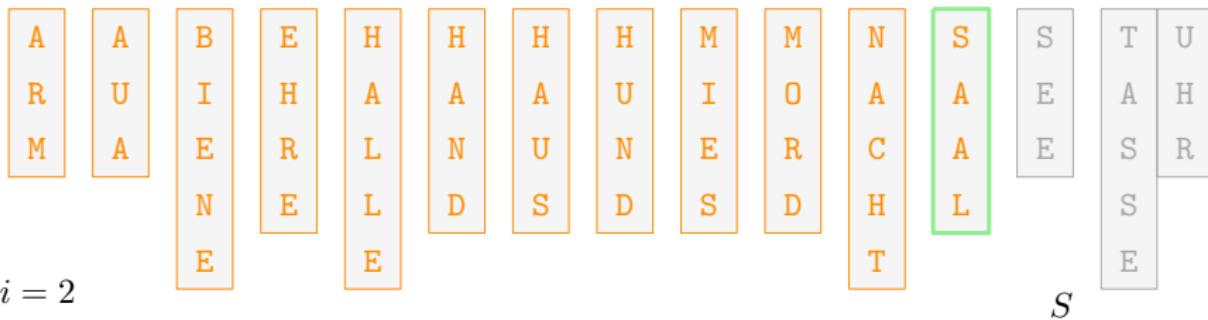
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

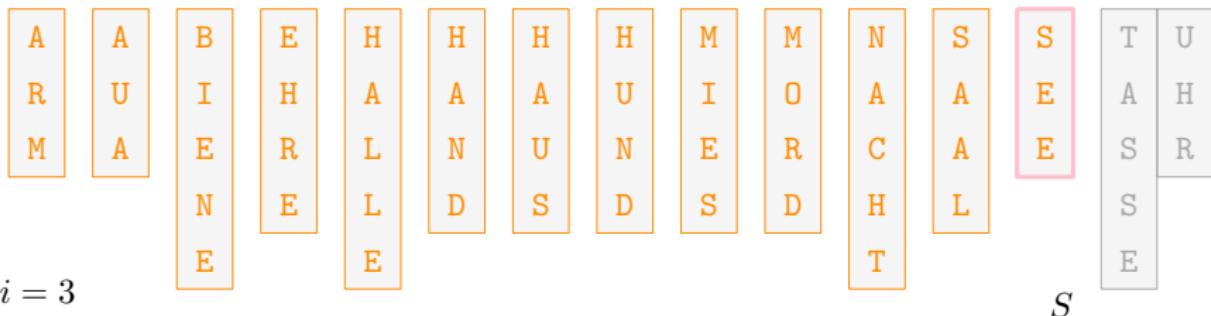
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

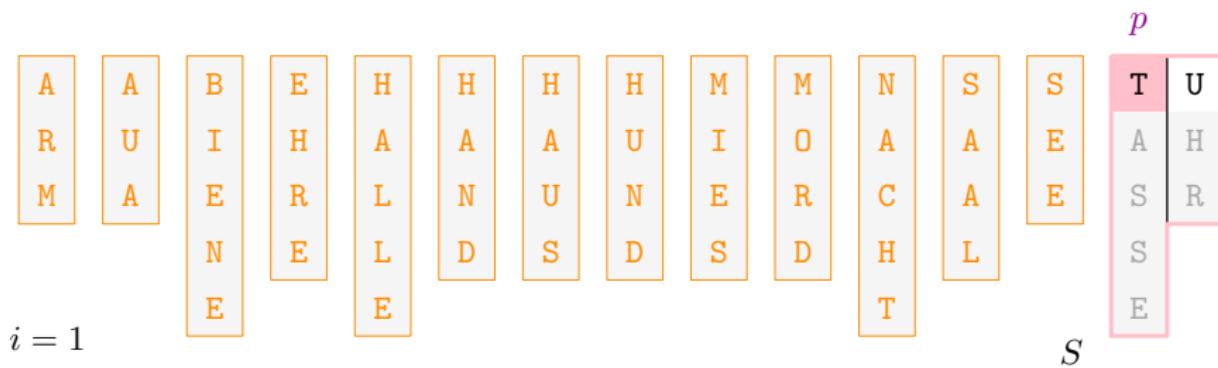
choose $p \in S$ uniformly at random

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



Multikey Quicksort

Ablauf

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

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

Basisfall

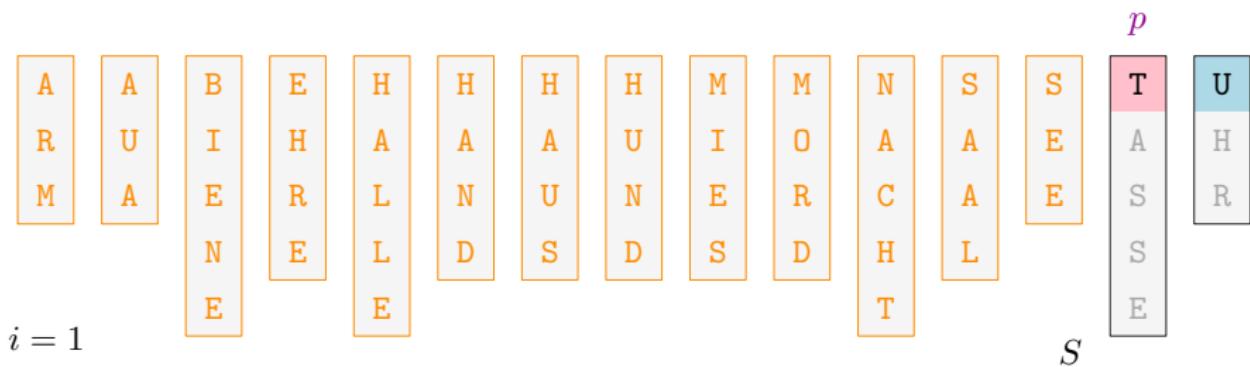
choose $p \in S$ uniformly at random

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



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

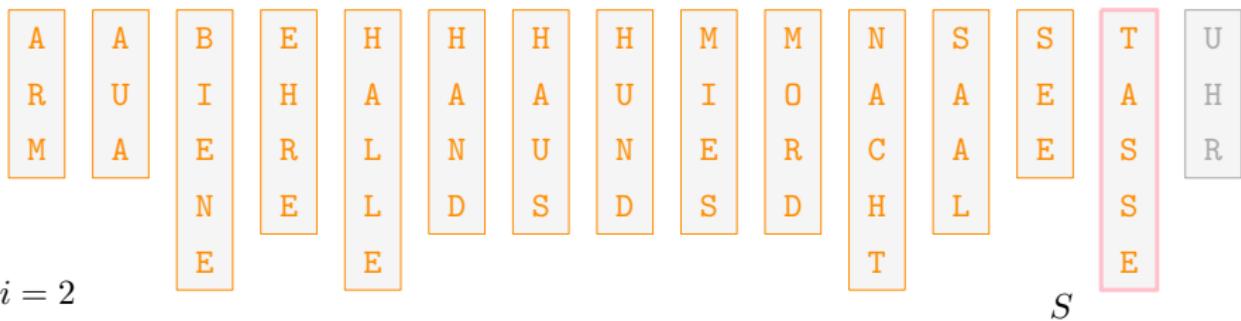
return concatenation of

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)



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



Multikey Quicksort

Ablauf

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

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

Basisfall

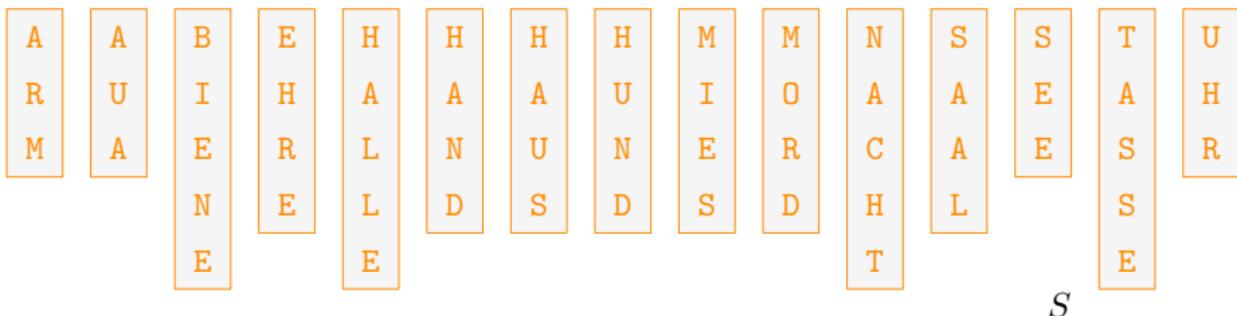
choose $p \in S$ uniformly at random

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

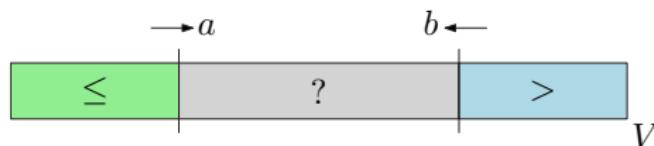


in-place Multikey Quicksort

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 $a > b$

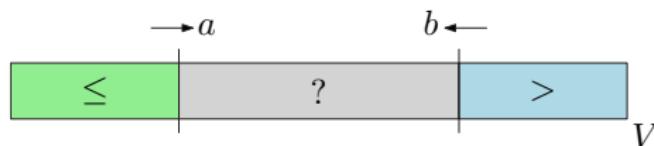


in-place Multikey Quicksort

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 $a > b$

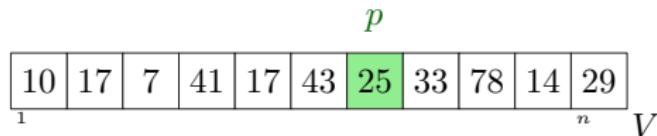
| | | | | | | | | | | |
|------|----|---|----|----|----|----|----|----|------|-----|
| 10 | 17 | 7 | 41 | 17 | 43 | 25 | 33 | 78 | 14 | 29 |
| $_1$ | | | | | | | | | $_n$ | V |

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 $a > b$

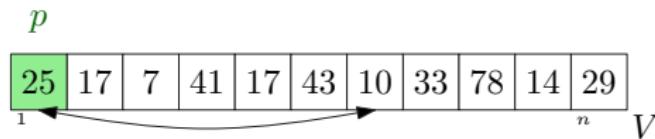


in-place Multikey Quicksort

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 $a > b$

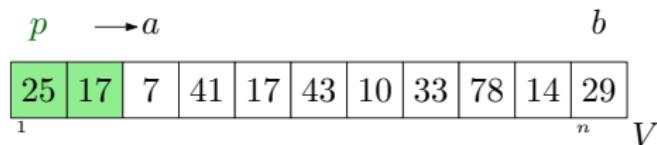
| p | a | b | | | | | | | | |
|-----|-----|-----|----|----|----|----|----|----|-----|-----|
| 25 | 17 | 7 | 41 | 17 | 43 | 10 | 33 | 78 | 14 | 29 |
| 1 | | | | | | | | | n | V |

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 $a > b$

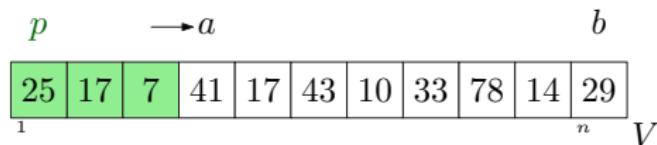


in-place Multikey Quicksort

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 $a > b$

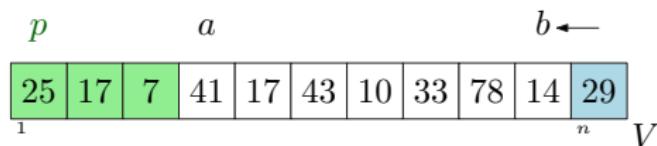


in-place Multikey Quicksort

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 $a > b$

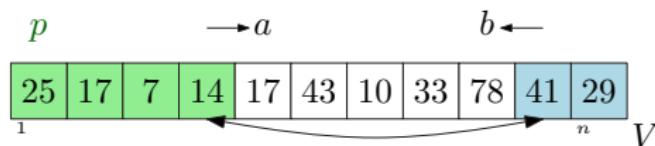


in-place Multikey Quicksort

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 $a > b$

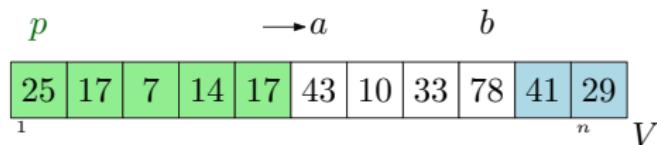


in-place Multikey Quicksort

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 $a > b$

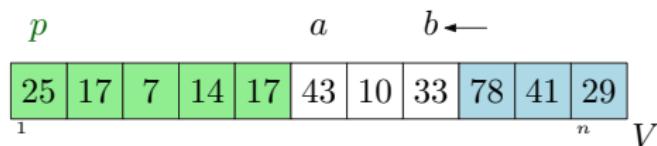


in-place Multikey Quicksort

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 $a > b$

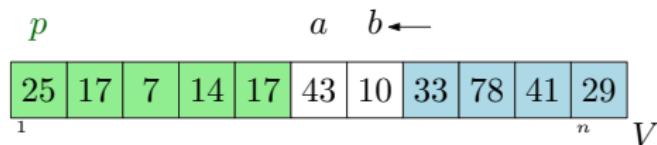


in-place Multikey Quicksort

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 $a > b$

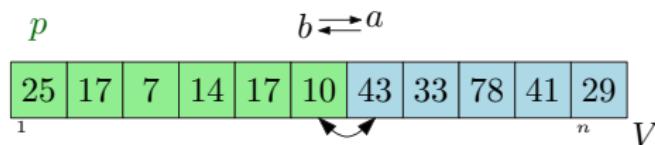


in-place Multikey Quicksort

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 $a > b$

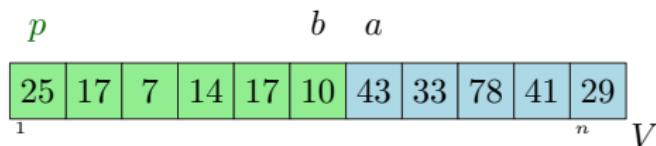


in-place Multikey Quicksort

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 $a > b$

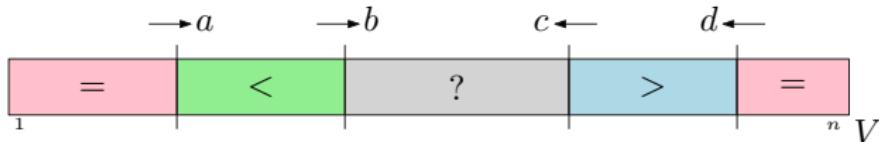


in-place Multikey Quicksort

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: $V[i \in [a, b] \text{ } a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d) \text{ } c \neq d] > p$

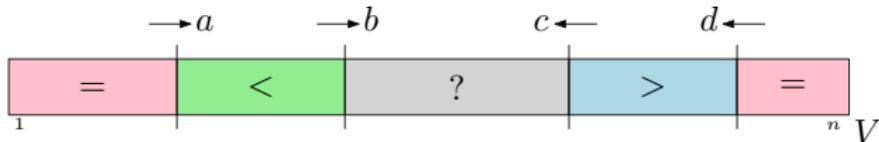


in-place Multikey Quicksort

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 <th>n</th> | 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 | |

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: $V[i \in [a, b) \ a \neq b] \leq p$, $V[i < a \vee i > d] = p$, $V[i \in (c, d) \ c \neq d] > p$

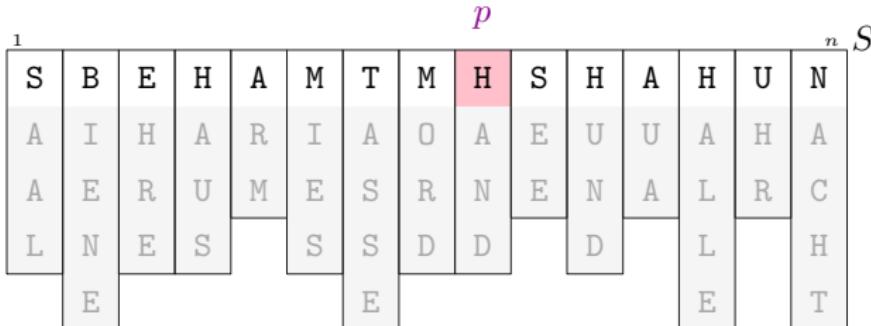
$$V \hat{=} \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|} \hline 1 & S & B & E & H & A & M & T & M & H & S & H & A & H & U & N \\ \hline A & I & H & A & R & I & A & O & A & E & U & U & A & H & A & A \\ \hline A & E & R & U & M & E & S & R & N & E & N & A & L & R & C \\ \hline L & N & E & S & & S & S & D & D & & D & & L & L & H \\ \hline E & & & & & E & & & & & & & E & & T \\ \hline n & S & & & & & & & & & & & & & & & \\ \hline \end{array}$$

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 $b > c$

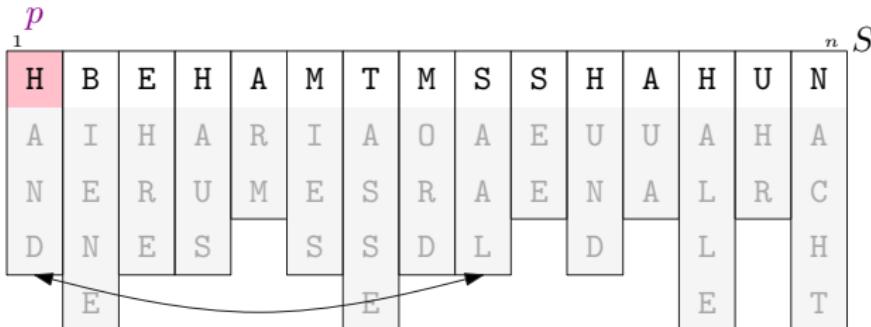


in-place Multikey Quicksort

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 $b > c$

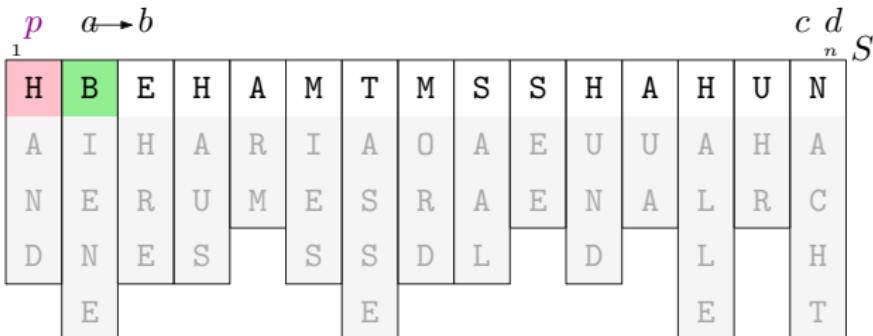
| <i>p</i> | <i>a</i> | <i>b</i> | <i>S</i> | | | | | | | | | | | | | | | | | |
|----------|----------|----------|----------|----------|--------------|----------|---|---|---|---|---|---|---|---|--|--|--|--|--|--|
| 1 | | | <i>c</i> | <i>d</i> | _n | <i>S</i> | | | | | | | | | | | | | | |
| H | B | E | 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 $b > c$

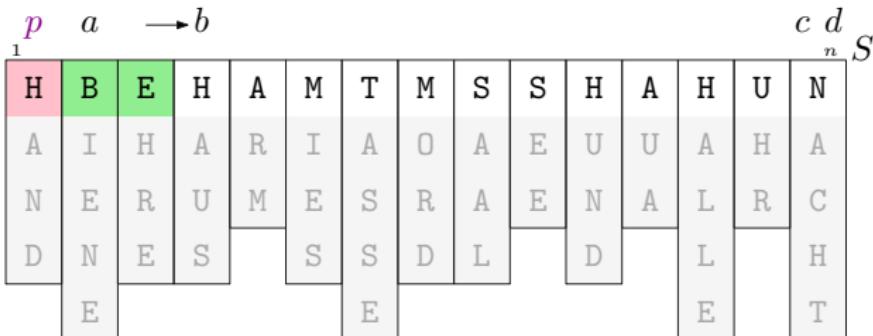


in-place Multikey Quicksort

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 $b > c$

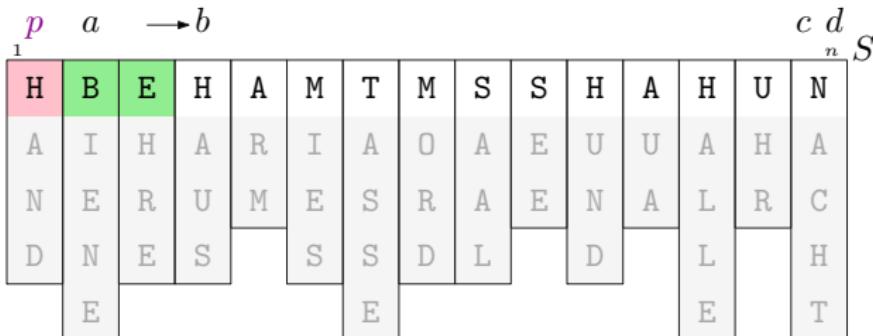


in-place Multikey Quicksort

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 $b > c$

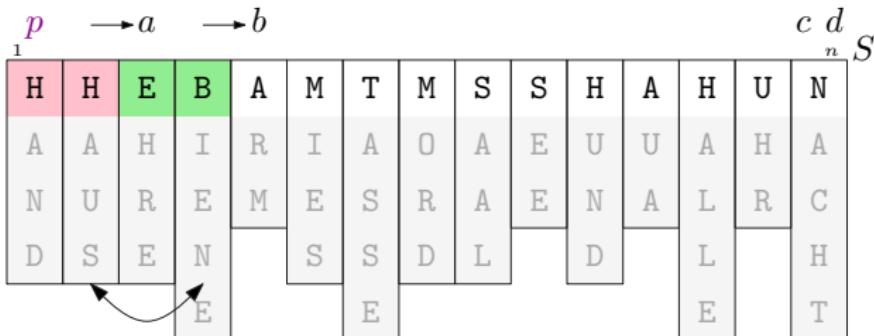


in-place Multikey Quicksort

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 | $\rightarrow b$ | | | | c | d | S |
|-----|-----|-----------------|---|---|---|-----|-----|-----|
| 1 | | | | | | n | | |
| 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 $b > c$

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

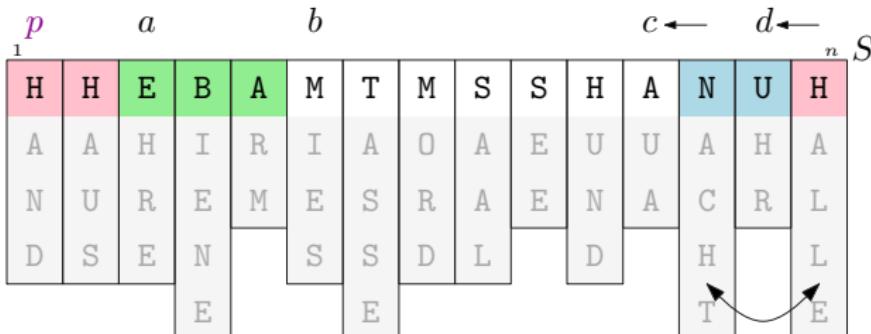
| p | a | b | | | $c \leftarrow$ | d | n | S |
|-----|-----|-----|---|---|----------------|-----|-----|-----|
| H | H | E | B | A | M | T | M | S |
| A | A | H | I | R | I | A | O | A |
| N | U | R | E | M | E | S | R | A |
| D | S | E | N | | S | S | D | L |
| | | E | | | E | | D | |
| | | | | | | |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

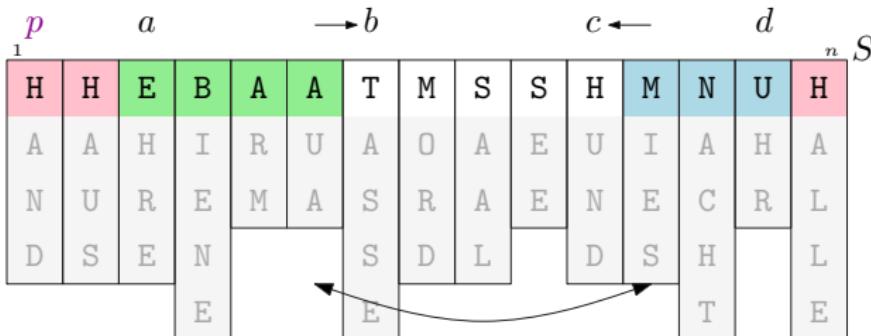


in-place Multikey Quicksort

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 $b > c$

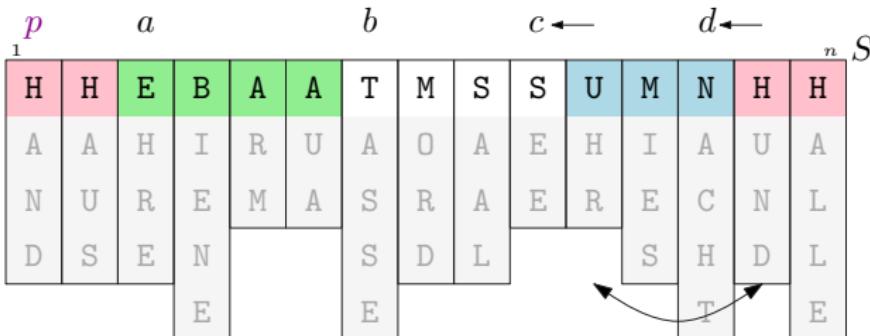


in-place Multikey Quicksort

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 $b > c$

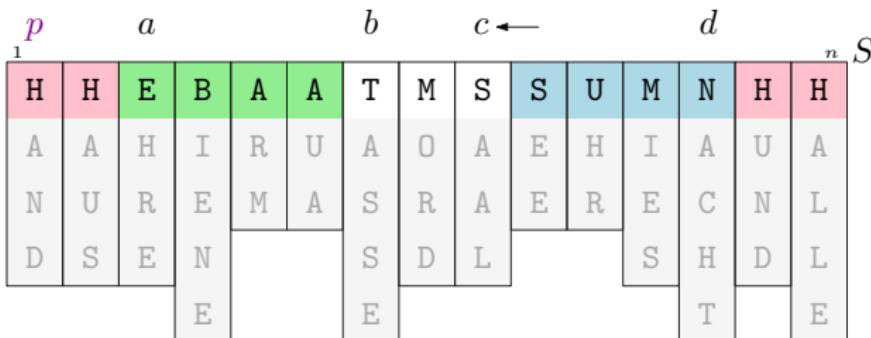


in-place Multikey Quicksort

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 $b > c$

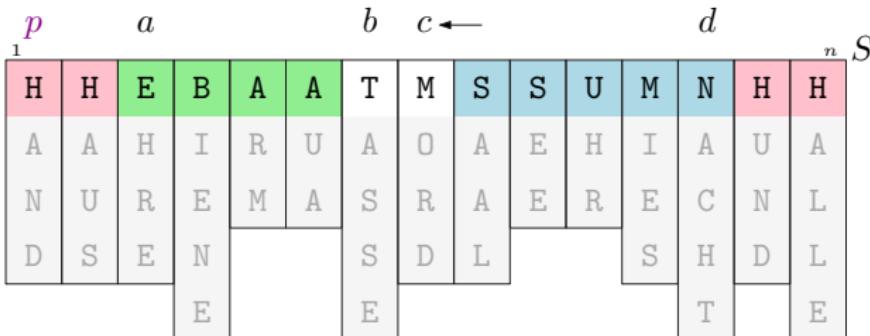


in-place Multikey Quicksort

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 $b > c$

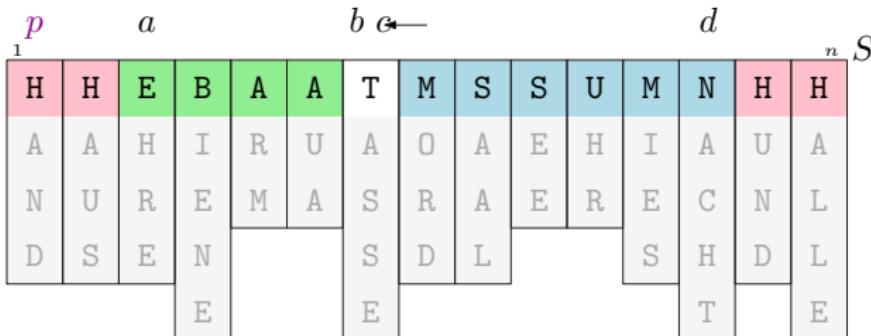


in-place Multikey Quicksort

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 $b > c$



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 |
|-----|-------------------------------|------------------|-----|-----|
| 1 | H H E B A A T M S S U M N H H | | | S |
| | 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 L E | | | |
| | 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 $[n - r + 1, n + 1)$

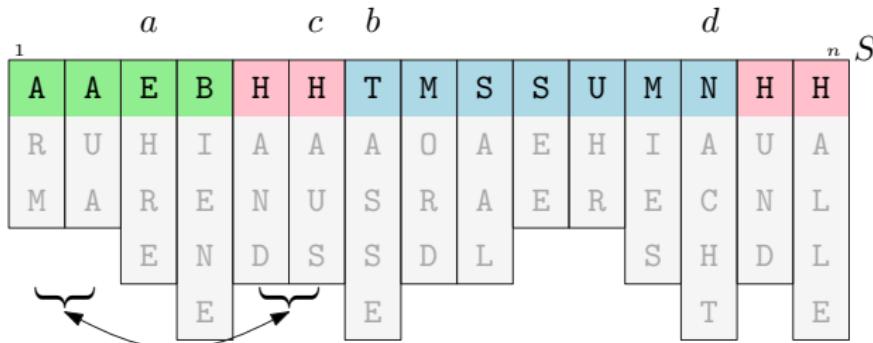
| 1 | a | | c | | b | | d | | n | | S |
|-----|-----|---|-----|---|-----|---|-----|---|-----|---|-----|
| H | H | E | B | A | A | T | M | S | S | U | M |
| A | A | H | I | R | U | A | O | A | E | H | I |
| N | U | R | E | M | A | S | R | A | E | R | E |
| D | S | E | N | | | S | D | L | | | S |
| | | 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 $[n - r + 1, n + 1)$

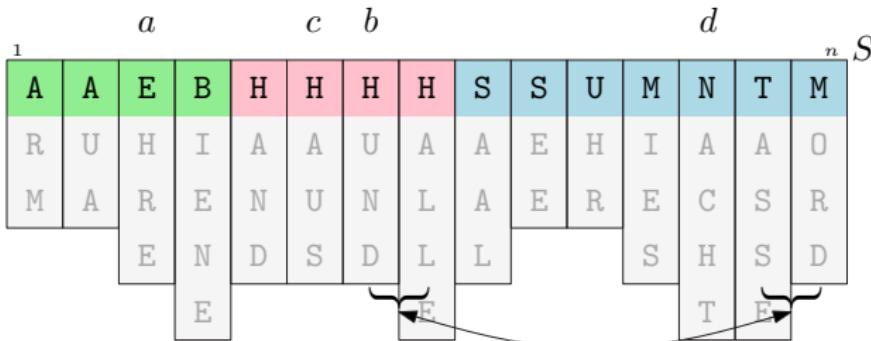
| 1 | a | | | | c | | b | | d | | | | | | n | S |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-----|---|
| A | A | E | B | H | H | T | M | S | S | U | M | N | H | H | | |
| R | U | H | I | A | A | A | O | A | E | H | I | A | U | A | | |
| M | A | R | E | N | U | S | R | A | E | R | E | C | N | L | | |
| | | | | D | S | S | D | L | | | S | H | D | L | | |
| | | | E | N | 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 $[n - r + 1, n + 1)$

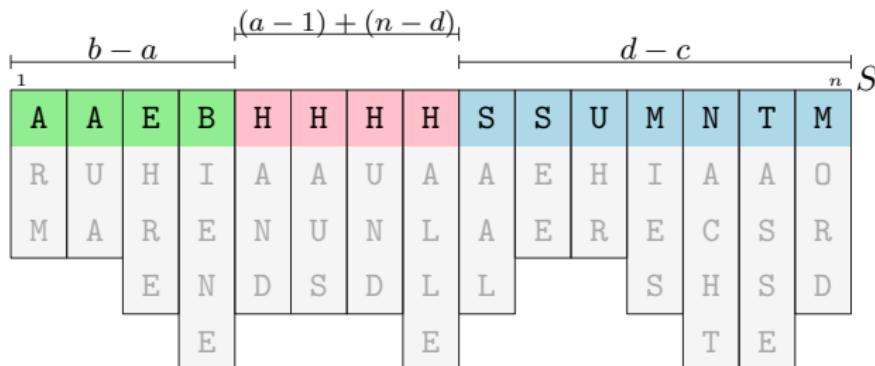


in-place Multikey Quicksort

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 $[n - r + 1, n + 1)$



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

Suffix-Arrays

Wiederholung

$T = \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ b & a & r & b & a & r & h & a & b & a & r & b & e & r & \$ \end{matrix}$

Suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

Suffix-Arrays

Wiederholung

Suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| i | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| i | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

Suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| — | | | | | | | | | | | | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| 8 | | a | b | a | r | b | e | r | | | | | | | |
| 1 | | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 2 | | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 3 | | r | b | a | r | h | a | b | a | r | b | e | r | | \$ |
| 4 | | b | a | r | h | a | b | a | r | b | e | r | | | \$ |
| 5 | | a | r | h | a | b | a | r | b | e | r | | | | \$ |
| 6 | | r | h | a | b | a | r | b | e | r | | | | | \$ |
| 7 | | h | a | b | a | r | b | e | r | | | | | | \$ |
| 9 | | b | a | r | b | e | r | | | | | | | | |
| 10 | | a | r | b | e | r | | | | | | | | | |
| 11 | | r | b | e | r | | | | | | | | | | |
| 12 | | b | e | r | | | | | | | | | | | |
| 13 | | e | r | | | | | | | | | | | | |
| 14 | | r | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| | | | | | | | | | | | | | | | |
| 8 | | a | b | a | r | b | e | r | | | | | | | \$ |
| 2 | | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 1 | | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 3 | | r | b | a | r | h | a | b | a | r | b | e | r | | \$ |
| 4 | | b | a | r | h | a | b | a | r | b | e | r | | | \$ |
| 5 | | a | r | h | a | b | a | r | b | e | r | | | | \$ |
| 6 | | r | h | a | b | a | r | b | e | r | | | | | \$ |
| 7 | | h | a | b | a | r | b | e | r | | | | | | \$ |
| 9 | | b | a | r | b | e | r | | | | | | | | |
| 10 | | a | r | b | e | r | | | | | | | | | \$ |
| 11 | | r | b | e | r | | | | | | | | | | |
| 12 | | b | e | r | | | | | | | | | | | |
| 13 | | e | r | | | | | | | | | | | | |
| 14 | | r | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| 8 | | a | b | a | r | b | e | r | | | | | | | \$ |
| 2 | | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 10 | | a | r | b | e | r | | | | | | | | | |
| 5 | | a | r | h | a | b | a | r | b | e | r | | | | \$ |
| 1 | | b | a | r | b | a | r | h | a | b | a | r | b | e | r |
| 3 | | r | b | a | r | h | a | b | a | r | b | e | r | | \$ |
| 4 | | b | a | r | h | a | b | a | r | b | e | r | | | \$ |
| 6 | | r | h | a | b | a | r | b | e | r | | | | | \$ |
| 7 | | h | a | b | a | r | b | e | r | | | | | | \$ |
| 9 | | b | a | r | b | e | r | | | | | | | | |
| 11 | | r | b | e | r | | | | | | | | | | \$ |
| 12 | | b | e | r | | | | | | | | | | | \$ |
| 13 | | e | r | | | | | | | | | | | | \$ |
| 14 | | r | | | | | | | | | | | | | \$ |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| 8 | | a | b | a | r | b | e | r | | | | | | | \$ |
| 2 | | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 10 | | a | r | b | e | r | | | | | | | | | |
| 5 | | a | r | h | a | b | a | r | b | e | r | | | | \$ |
| 1 | | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 3 | | r | b | a | r | h | a | b | a | r | b | e | r | | \$ |
| 4 | | b | a | r | h | a | b | a | r | b | e | r | | | \$ |
| 6 | | r | h | a | b | a | r | b | e | r | | | | | \$ |
| 7 | | h | a | b | a | r | b | e | r | | | | | | \$ |
| 9 | | b | a | r | b | e | r | | | | | | | | \$ |
| 11 | | r | b | e | r | | | | | | | | | | |
| 12 | | b | e | r | | | | | | | | | | | |
| 13 | | e | r | | | | | | | | | | | | |
| 14 | | r | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | \$ |
| 8 | | a | b | a | r | b | e | r | \$ | | | | | | |
| 2 | | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 10 | | a | r | b | e | r | \$ | | | | | | | | |
| 5 | | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 1 | | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 9 | | b | a | r | b | e | r | \$ | | | | | | | |
| 4 | | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 3 | | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 6 | | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 7 | | h | a | b | a | r | b | e | r | \$ | | | | | |
| 11 | | r | b | e | r | \$ | | | | | | | | | |
| 12 | | b | e | r | \$ | | | | | | | | | | |
| 13 | | e | r | \$ | | | | | | | | | | | |
| 14 | | r | \$ | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|----|----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | $T = b$ | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|----|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | $T = b$ | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |
| — | | | | | | | | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|----|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | $T = b$ | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | | | | | | | | | | | | | | | |
| 15 | \$ | | | | | | | | | | | | | | |
| 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 12 | b | e | r | \$ | | | | | | | | | | | |
| 13 | e | r | \$ | | | | | | | | | | | | |
| 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 14 | r | \$ | | | | | | | | | | | | | |
| 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 11 | r | b | e | r | \$ | | | | | | | | | | |
| 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |

Suffix-Arrays

Wiederholung

suffix-Array SA von T
indiziert alle Suffixe
in sortierter Reihenfolge
Im Beispiel $\mathcal{O}(n^3)$

| i | SA[i] | |
|----|-------|--------------------------------|
| 1 | 15 | \$ |
| 2 | 8 | a b a r b e r \$ |
| 3 | 2 | a r b a r h a b a r b e r \$ |
| 4 | 10 | a r b e r \$ |
| 5 | 5 | a r h a b a r b e r \$ |
| 6 | 1 | b a r b a r h a b a r b e r \$ |
| 7 | 9 | b a r b e r \$ |
| 8 | 4 | b a r h a b a r b e r \$ |
| 9 | 12 | b e r \$ |
| 10 | 13 | e r \$ |
| 11 | 7 | h a b a r b e r \$ |
| 12 | 14 | r \$ |
| 13 | 3 | r b a r h a b a r b e r \$ |
| 14 | 11 | r b e r \$ |
| 15 | 6 | r h a b a r b e r \$ |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

$n = \text{Textlänge}, m = \text{Pattern-Länge}$

- Naiv: $\mathcal{O}(n \cdot m)$
- KMP: $\mathcal{O}(n + m)$
- Mit Suffix-Arrays zunächst: $\mathcal{O}(m \cdot \log n)$
- Optimiert: $\mathcal{O}(m + \log n)$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende
- Ergebnis

| | | | | | | | | | | | | | | | |
|-------|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | SA[i] | | | | | | | | | | | | | | |
| 1 | 15 | \$ | | | | | | | | | | | | | |
| 2 | 8 | a | b | a | r | b | e | r | \$ | | | | | | |
| 3 | 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 4 | 10 | a | r | b | e | r | \$ | | | | | | | | |
| 5 | 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 6 | 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 7 | 9 | b | a | r | b | e | r | \$ | | | | | | | |
| 8 | 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 9 | 12 | b | e | r | \$ | | | | | | | | | | |
| 10 | 13 | e | r | \$ | | | | | | | | | | | |
| 11 | 7 | h | a | b | a | r | b | e | r | \$ | | | | | |
| 12 | 14 | r | \$ | | | | | | | | | | | | |
| 13 | 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 14 | 11 | r | b | e | r | \$ | | | | | | | | | |
| 15 | 6 | r | h | a | b | a | r | b | e | r | \$ | | | | |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

■ (SA bestimmen)

■ finde Start

■ finde Ende

■ Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche

$I = 1, r = n$

while ($I < r$) **do**

$q = \lfloor \frac{I+r}{2} \rfloor$

if ($P > T_{SA[q]..SA[q]+m-1}$)

then $I = q + 1$

else $r = q$

$s = I$

if ($P \neq T_{SA[s]..SA[s]+m-1}$)

then break

- finde Ende

- Ergebnis

| | | | | | | | | | | | | | | | |
|----------|--------------|--------------------------------|---|---|---|---|---|---|---|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| <i>i</i> | <i>SA[i]</i> | | | | | | | | | | | | | | |
| 1 | 15 | \$ | | | | | | | | | | | | | |
| 2 | 8 | a b a r b e r \$ | | | | | | | | | | | | | |
| 3 | 2 | a r b a r h a b a r b e r \$ | | | | | | | | | | | | | |
| 4 | 10 | a r b e r \$ | | | | | | | | | | | | | |
| 5 | 5 | a r h a b a r b e r \$ | | | | | | | | | | | | | |
| 6 | 1 | b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | |
| 7 | 9 | b a r b e r \$ | | | | | | | | | | | | | |
| 8 | 4 | b a r h a b a r b e r \$ | | | | | | | | | | | | | |
| 9 | 12 | b e r \$ | | | | | | | | | | | | | |
| 10 | 13 | e r \$ | | | | | | | | | | | | | |
| 11 | 7 | h a b a r b e r \$ | | | | | | | | | | | | | |
| 12 | 14 | r \$ | | | | | | | | | | | | | |
| 13 | 3 | r b a r h a b a r b e r \$ | | | | | | | | | | | | | |
| 14 | 11 | r b e r \$ | | | | | | | | | | | | | |
| 15 | 6 | r h a b a r b e r \$ | | | | | | | | | | | | | |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche

$I = 1, r = n$

while ($I < r$) **do**

$q = \lfloor \frac{I+r}{2} \rfloor$

if ($P > T_{SA[q]..SA[q]+m-1}$)

then $I = q + 1$

else $r = q$

$s = I$

if ($P \neq T_{SA[s]..SA[s]+m-1}$)

then break

- finde Ende

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---------|---|----|----|----------|----------|----------|----------|----------|----------|----------|----------|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 | | | | | | | | | | | | | | | \$ |
| 2 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 3 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 4 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 7 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| q = 8 4 | b | a | r | h | a | b | a | r | b | e | r | | | | \$ |
| 9 12 | b | e | r | \$ | | | | | | | | | | | |
| 10 13 | e | r | \$ | | | | | | | | | | | | |
| 11 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 12 14 | r | \$ | | | | | | | | | | | | | |
| 13 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 14 11 | r | b | e | r | \$ | | | | | | | | | | |
| 15 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |

$l = 1, r = 16$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche

$I = 1, r = n$

while ($I < r$) **do**

$q = \lfloor \frac{I+r}{2} \rfloor$

if ($P > T_{SA[q]..SA[q]+m-1}$)

then $I = q + 1$

else $r = q$

$s = I$

if ($P \neq T_{SA[s]..SA[s]+m-1}$)

then break

- finde Ende

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 | | | | | | | | | | | | | | | \$ |
| 2 8 | a | b | a | r | b | e | r | \$ | | | | | | | |
| 3 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| q = 4 10 | a | r | b | e | r | \$ | | | | | | | | | |
| 5 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 7 9 | b | a | r | b | e | r | \$ | | | | | | | | |
| 8 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 9 12 | b | e | r | \$ | | | | | | | | | | | |
| 10 13 | e | r | \$ | | | | | | | | | | | | |
| 11 7 | h | a | b | a | r | b | e | r | \$ | | | | | | |
| 12 14 | r | \$ | | | | | | | | | | | | | |
| 13 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 14 11 | r | b | e | r | \$ | | | | | | | | | | |
| 15 6 | r | h | a | b | a | r | b | e | r | \$ | | | | | |

$l = 1, r = 8$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche

$I = 1, r = n$

while ($I < r$) **do**

$q = \lfloor \frac{I+r}{2} \rfloor$

if ($P > T_{SA[q]..SA[q]+m-1}$) $q =$

then $I = q + 1$

else $r = q$

$s = I$

if ($P \neq T_{SA[s]..SA[s]+m-1}$) $s =$

then break

- finde Ende

- Ergebnis

| | | | | | | | | | | | | | | | |
|---------|----|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 | 15 | | | | | | | | | | | | | | |
| 2 | 8 | a | b | a | r | b | e | r | | | | | | | |
| 3 | 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 4 | 10 | a | r | b | e | r | | | | | | | | | |
| 5 | 5 | a | r | h | a | b | a | r | b | e | r | | | | |
| 6 | 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 7 | 9 | b | a | r | b | e | r | | | | | | | | |
| 8 | 4 | b | a | r | h | a | b | a | r | b | e | r | | | |
| 9 | 12 | b | e | r | | | | | | | | | | | |
| 10 | 13 | e | r | | | | | | | | | | | | |
| 11 | 7 | h | a | b | a | r | b | e | r | | | | | | |
| 12 | 14 | r | | | | | | | | | | | | | |
| 13 | 3 | r | b | a | r | h | a | b | a | r | b | e | r | | |
| 14 | 11 | r | b | e | r | | | | | | | | | | |
| 15 | 6 | r | h | a | b | a | r | b | e | r | | | | | |

$l = 5, r = 8$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche

$I = 1, r = n$

while ($I < r$) **do**

$$q = \lfloor \frac{I+r}{2} \rfloor$$

if ($P > T_{SA[q]..SA[q]+m-1}$)

then $I = q + 1$

else $r = q$

$s = I$

if ($P \neq T_{SA[s]..SA[s]+m-1}$)

then break

- finde Ende

- Ergebnis

| | | | | | | | | | | | | | | | |
|------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

$$l = 5, r = 6$$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start binäre Suche
 $I = 1, r = n$
- while ($I < r$) do
 - $q = \lfloor \frac{I+r}{2} \rfloor$
 - if ($P > T_{SA[q]..SA[q]+m-1}$)
 - then $I = q + 1$
 - else $r = q$
 - $s = I$
 - if ($P \neq T_{SA[s]..SA[s]+m-1}$)
 - then break
- finde Ende
- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |

$$l = 6, r = 6$$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende binäre Suche

$l = s, r = n$

while ($l < r$) **do**

$q = \lceil \frac{l+r}{2} \rceil$

if ($P = T_{SA[q]..SA[q]+m-1}$)

then $l = q$

else $r = q - 1$

$t = l$

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| q=10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |

$l = 5, r = 15$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende binäre Suche

$l = s, r = n$

while ($l < r$) **do**

$$q = \lceil \frac{l+r}{2} \rceil$$

if ($P = T_{SA[q]..SA[q]+m-1}$)

then $l = q$

else $r = q - 1$

$t = l$

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----------------|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | $l = 5, r = 9$ |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende binäre Suche

$l = s, r = n$

while ($l < r$) **do**

$$q = \lceil \frac{l+r}{2} \rceil$$

if ($P = T_{SA[q]..SA[q]+m-1}$) $q =$

then $l = q$

else $r = q - 1$

$t = l$

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|---|---|---|----------|---|----------|---|---|---|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 9 12 b e r \$ | | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |

$$l = 7, r = 9$$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende binäre Suche

$l = s, r = n$

while ($l < r$) **do**

$$q = \lceil \frac{l+r}{2} \rceil$$

if ($P = T_{SA[q]..SA[q]+m-1}$)

then $l = q$

else $r = q - 1$

$t = l$

- Ergebnis

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------------------------|-------------|---|---|---|---|---|---|---|---|----|----|----|----|----|----------------|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 \$ | | | | | | | | | | | | | | | |
| 2 8 a b a r b e r \$ | | | | | | | | | | | | | | | |
| 3 2 a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 4 10 a r b e r \$ | | | | | | | | | | | | | | | |
| 5 5 a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 6 1 b a r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 7 9 b a r b e r \$ | | | | | | | | | | | | | | | |
| 8 4 b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| $q = 9$ | 12 b e r \$ | | | | | | | | | | | | | | |
| 10 13 e r \$ | | | | | | | | | | | | | | | |
| 11 7 h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 12 14 r \$ | | | | | | | | | | | | | | | |
| 13 3 r b a r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| 14 11 r b e r \$ | | | | | | | | | | | | | | | |
| 15 6 r h a b a r b e r \$ | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | $l = 8, r = 9$ |

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende binäre Suche

$l = s, r = n$

while ($l < r$) **do**

$$q = \lceil \frac{l+r}{2} \rceil$$

if ($P = T_{SA[q]..SA[q]+m-1}$)

then $l = q$

else $r = q - 1$

$t = l$

- Ergebnis

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| i SA[i] | | | | | | | | | | | | | | | |
| 1 15 | | | | | | | | | | | | | | | \$ |
| 2 8 | a | b | a | r | b | a | r | b | e | r | \$ | | | | |
| 3 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 4 10 | a | r | b | e | r | | | | | | | | | | \$ |
| 5 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | | |
| 6 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 7 9 | b | a | r | b | e | r | | | | | | | | | \$ |
| 8 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 9 12 | b | e | r | | | | | | | | | | | | \$ |
| 10 13 | e | r | | | | | | | | | | | | | |
| 11 7 | h | a | b | a | r | b | e | r | | | | | | | \$ |
| 12 14 | r | | | | | | | | | | | | | | \$ |
| 13 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 14 11 | r | b | e | r | | | | | | | | | | | \$ |
| 15 6 | r | h | a | b | a | r | b | e | r | | | | | | \$ |

$$l = 8, r = 8$$

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{bar}$

- (SA bestimmen)
- finde Start
- finde Ende
- Ergebnis
 - $t - s + 1$
counting query
 - $\{SA[s], \dots, SA[t]\}$
reporting query

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| i | SA[i] | | | | | | | | | | | | | | |
| 1 | 15 | \$ | | | | | | | | | | | | | |
| 2 | 8 | a | b | a | r | b | e | r | \$ | | | | | | |
| 3 | 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| 4 | 10 | a | r | b | e | r | \$ | | | | | | | | |
| 5 | 5 | a | r | h | a | b | a | r | b | e | r | \$ | | | |
| 6 | 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | \$ |
| 7 | 9 | b | a | r | b | e | r | \$ | | | | | | | |
| q = 8 | 4 | b | a | r | h | a | b | a | r | b | e | r | \$ | | |
| 9 | 12 | b | e | r | \$ | | | | | | | | | | |
| 10 | 13 | e | r | \$ | | | | | | | | | | | |
| 11 | 7 | h | a | b | a | r | b | e | r | \$ | | | | | |
| 12 | 14 | r | \$ | | | | | | | | | | | | |
| 13 | 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 14 | 11 | r | b | e | r | \$ | | | | | | | | | |
| 15 | 6 | r | h | a | b | a | r | b | e | r | \$ | | | | |

$$s = 6, t = 8$$

Suche mit Suffix-Arrays

Zusammenfassung

- Verlagerung des Aufwands von Anfrage in Vorverarbeitung
 - einmal Suffix-Array generieren in $\mathcal{O}(n)$,
 - danach **Anfragen in $\mathcal{O}(m \log n)$** möglich, statt in $\mathcal{O}(mn)$

gut, wenn auf einem Text viele Anfragen stattfinden
- Ausnutzung der Eigenschaften des Suffix-Arrays
 - jeder Substring ist Präfix eines Suffix
 - **alle Substrings liegen "sortiert" vor**
 - mögliche Ausnahme: Substring ist Präfix von Substring

das Suffix-Array indiziert alle Suffixe in sortierter Reihenfolge

Definition:

- LCP[i]: Länge des längsten gemeinsamen Präfixes von je zwei lexikographisch benachbarten Suffixen $A[SA[i-1] \dots n]$ und $A[SA[i] \dots n]$

Erweiterung auf beliebige Suffixe

- LCP[i][j]: Länge des längsten gemeinsamen Präfix beliebiger lexikographischer Suffixe $A[SA[i] \dots n]$ und $A[SA[j] \dots n]$
- Konstruktion: $\mathcal{O}(n \log n)$ Zeit und Platz
- Zugriff: $\mathcal{O}(1)$

Schnelle Suche mit Suffix-Arrays

Erster Ansatz

Suche: $P = \text{bar}$

- Ziel: kein wiederholtes Vergleichen von Zeichen aus P
- Nutze LCP-Array um Suche zu beschleunigen
- Starte Suche bei mlr
 - $I := \text{LCP}(L, P)$
 - $r := \text{LCP}(R, P)$
 - $mlr := \min(I, r)$
 - Update von I, r , keine Neuberechnung
- Oft $\mathcal{O}(m + \log n)$
- Worst case $\mathcal{O}(m \log n)$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---------|----|----|---|---|---|---|---|---|---|----|----|----|----|----|----|----|
| $T =$ | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| i SA[i] | | | | | | | | | | | | | | | | |
| 1 | 15 | | | | | | | | | | | | | | \$ | |
| 2 | 8 | a | b | a | r | b | e | r | | | | | | | \$ | |
| 3 | 2 | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| 4 | 10 | a | r | b | e | r | | | | | | | | | \$ | |
| 5 | 5 | a | r | h | a | b | a | r | b | e | r | | | | \$ | |
| $L=6$ | 1 | b | a | r | b | a | r | h | a | b | a | r | b | e | r | \$ |
| $q=7$ | 9 | b | a | r | b | e | r | | | | | | | | | |
| | 8 | 4 | b | a | r | h | a | b | a | r | b | e | r | | \$ | |
| $R=9$ | 12 | b | e | r | | | | | | | | | | | | |
| | 10 | 13 | e | r | | | | | | | | | | | | |
| | 11 | 7 | h | a | b | a | r | b | e | r | | | | | | |
| | 12 | 14 | r | | | | | | | | | | | | | |
| | 13 | 3 | r | b | a | r | h | a | b | a | r | b | e | r | \$ | |
| | 14 | 11 | r | b | e | r | | | | | | | | | | |
| | 15 | 6 | r | h | a | b | a | r | b | e | r | | | | | |

$$L = 6, R = 9$$

$$l = 3, r = 1$$

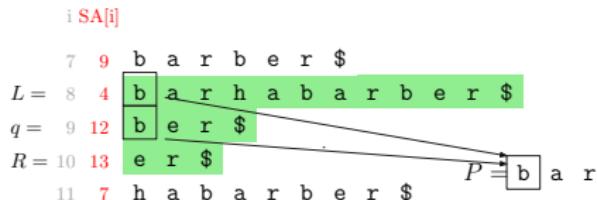
$$mlr := 1 = \min(l, r)$$

Schnelle Suche mit Suffix-Arrays

Redundante Vergleiche

Problem

- Falls $l \neq r \rightarrow$ wiederholtes Vergleichen



Definition

- Vergleich eines Zeichens aus P ist **redundant**, falls das Zeichen vorher schon einmal überprüft wurde.

Ziel

- Beschränke redundante Vergleiche auf $\mathcal{O}(1)$ pro Iteration
- Vergleiche bei $\max(l, r)$ beginnen

Suche mit Suffix-Arrays

Ablauf

Ansatz

- **if** ($I = r$)
start at mlr
Update I, r, L, R
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)
 $L := q + 1$
Update I
- **if** ($I > r \wedge \text{LCP}[L, q] < I$)
 $R := q$
 $r := \text{LCP}[L, q]$
- **if** ($I > r \wedge \text{LCP}[L, q] = I$)
start at I

Suche mit Suffix-Arrays

Ablauf

b a r b a r h a b a r b e r ...

Suche: $P \equiv \text{barberac}$

- **if** ($I = r$)
start at mlr
Update I, r, L, R
 - **if** ($I > r \wedge \text{LCP}[L, q] > I$) b a r b e r a b a ...
 - **if** ($I > r \wedge \text{LCP}[L, q] < I$)
 - **if** ($I > r \wedge \text{LCP}[L, q] = I$) b a r b e r a b c ...
b a r b e r a c c ...
b a r b e r c b c \$

b a r b i

$$l=4, r=4$$

Suche mit Suffix-Arrays

Ablauf

$L = \boxed{\text{b a r b}} \text{ a r h a b a r b e r ...}$

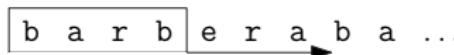
Suche: $P = \text{barberac}$

- **if** ($I = r$)

start at m/r

Update I, r, L, R

- **if** ($I > r \wedge \text{LCP}[L, q] > I$) $q =$

 $\boxed{\text{b a r b}} \rightarrow \text{e r a b a ...}$

- **if** ($I > r \wedge \text{LCP}[L, q] < I$)

$\text{b a r b e r a b b ...}$

- **if** ($I > r \wedge \text{LCP}[L, q] = I$)

$\begin{array}{c} \text{b a r b e r a b c ...} \\ \text{b a r b e r a c c \$} \\ \text{b a r b e r c b c ...} \end{array} \quad \text{LCP}[L, q] = 4$

$R = \boxed{\text{b a r b i}}$ $l = 4, r = 4$

Suche mit Suffix-Arrays

Ablauf

b a r b a r h a b a r b e r ...

Suche: $P = \text{barberac}$

- **if** ($I = r$)
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)
- **if** ($I > r \wedge \text{LCP}[L, q] < I$)
- **if** ($I > r \wedge \text{LCP}[L, q] = I$) $L =$

b a r b e r a b a ...
b a r b e r a b b ...

b a r b e r a b c ...
b a r b e r a c c \\$
b a r b e r c b c ...
 $R =$ b a r b i $l = 7, r = 4$

Suche mit Suffix-Arrays

Ablauf

b a r b a r h a b a r b e r ...

Suche: $P = \text{barberac}$

- **if** ($I = r$)
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)

$$L := q + 1$$

Update I

- **if** ($I > r \wedge \text{LCP}[L, q] < I$) $L =$
- **if** ($I > r \wedge \text{LCP}[L, q] = I$)

b a r b e r a b a ...
b a r b e r a b b ...

$q =$ b a r b e r a b c ...

b a r b e r a c c \$

b a r b e r c b c ... $\text{LCP}[L, q] = 8$

$R =$ b a r b i

$$l = 7, r = 4$$

Suche mit Suffix-Arrays

Ablauf

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-----|
| b | a | r | b | a | r | h | a | b | a | r | b | e | r | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-----|

Suche: $P = \text{barberac}$

- **if** ($I = r$)
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)
- **if** ($I > r \wedge \text{LCP}[L, q] < I$)

$R := q$

$r := \text{LCP}[L, q]$

b a r b e r a b a ...

b a r b e r a b b ...

- **if** ($I > r \wedge \text{LCP}[L, q] = I$)

| | |
|-------|-------------------------|
| $L =$ | b a r b e r a b c ... |
| $q =$ | b a r b e r a c c \$ |
| $R =$ | b a r b e r c b c ... |

$\text{LCP}[L, q] = 6$

$l = 8, r = 4$

Suche mit Suffix-Arrays

Ablauf

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-----|
| b | a | r | b | a | r | h | a | b | a | r | b | e | r | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|-----|

Suche: $P = \text{barberac}$

- **if** ($I = r$)
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)
- **if** ($I > r \wedge \text{LCP}[L, q] < I$)

$R := q$

$r := \text{LCP}[L, q]$

b a r b e r a b a ...

b a r b e r a b b ...

- **if** ($I > r \wedge \text{LCP}[L, q] = I$)

b a r b e r a b c ...

b a r b e r a c c \$

b a r b e r c b c ...

b a r b i

$l = 8, r = 6$

Suche mit Suffix-Arrays

Ablauf

b a r b a r h a b a r b e r ...

Suche: $P = \text{barberac}$

- **if** ($l = r$)
- **if** ($l > r \wedge \text{LCP}[L, q] > l$)
- **if** ($l > r \wedge \text{LCP}[L, q] < l$)
- **if** ($l > r \wedge \text{LCP}[L, q] = l$)

b a r b e r a b a ...
b a r b e r a b b ...

$L = q =$ b a r b e r a b c ...
 $R =$ b a r b e r a c c \$
b a r b e r c b c ...
b a r b i

$\text{LCP}[L, q] = 10$
 $l = 8, r = 6$

Suche mit Suffix-Arrays

Ablauf

b a r b a r h a b a r b e r ...

Suche: $P = \text{barberac}$

- **if** ($I = r$)
 - **if** ($I > r \wedge \text{LCP}[L, q] > I$)
 - **if** ($I > r \wedge \text{LCP}[L, q] < I$)
 - **if** ($I > r \wedge \text{LCP}[L, q] = I$)
start at I
- b a r b e r a b a ...
b a r b e r a b b ...

b a r b e r a b c ...
 $L = R = \boxed{\text{b a r b e r a c}} \text{ c } \$$
b a r b e r c b c ...
b a r b i

Suche mit Suffix-Arrays

Ablauf

Suche: $P = \text{barberac}$

- **if** ($I = r$)
- **if** ($I > r \wedge \text{LCP}[L, q] > I$)
- **if** ($I > r \wedge \text{LCP}[L, q] < I$)
- **if** ($I > r \wedge \text{LCP}[L, q] = I$)

Laufzeit

- LCP + SA: $\mathcal{O}(m + \log n)$ Vergleiche
- Beweisidee
 - I, r werden nur größer
 - Anzahl an redundanten Vergleichen pro Rekursion konstant

Suche mit Suffix-Arrays

Zusammenfassung

- Verlagerung des Aufwands von Anfrage in Vorverarbeitung
 - einmal Suffix-Array generieren in $\mathcal{O}(n)$,
 - danach **Anfragen in $\mathcal{O}(m \log n)$** möglich, statt in $\mathcal{O}(m + n)$

gut, wenn auf einem Text viele Anfragen stattfinden
- Verhindern redundanter Vergleiche
 - einmal Suffix-Array generieren in $\mathcal{O}(n)$,
 - einmal LCP-Array generieren in $\mathcal{O}(n)$,
 - einmal erweitertes LCP-Array generieren in $\mathcal{O}(n \log n)$,
 - danach **Anfrage in $\mathcal{O}(m + \log n)$**

Ende!



Feierabend!