

2.4 Suchen

2.4.1 Selektion

2.4.2 Hashing



2.4 Suchen

2.4.1 Selektion

2.4.1.1 Definitionen

2.4.1.2 Selektion durch Partitionierung

2.4.1.3 Median der Mediane


2.4.2 Hashing




k-Selektion

- Selektions-Problem
 - gegeben:
 - n (verschiedene) Zahlen $A[1..n]$
 - eine Zahl k mit $1 \leq k \leq n$
 - gesucht: k -kleinstes Element von A , d.h. dasjenige Element $x \in A$ mit

$$\#\{ i : A[i] < x \} = k-1$$




Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer




Man will nicht nach bestimmten Schlüsseln suchen, sondern Fragen beantworten wie „was ist das größte/kleinste Element“, etc.

k-Selektion

- Man nennt das k -kleinste Element
 - für $k=1$ das Minimum
 - für $k=\lfloor (n+1)/2 \rfloor, k=\lceil (n+1)/2 \rceil$ den unteren, oberen Median
 - für $k=n$ das Maximum



Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer



Suche nach Median ist in naiver Implementierung in $O(n^2)$. Hierfür werden wir eine intelligentere Implementierung finden.

k-Selektion: Beispiel

$A[1..10] = 41, 76, 32, 62, 21, 52, 19, 83, 0, 91$

Minimum: 0

Maximum: 91

Unterer Median: 41, denn

$$\#\{i : A[i] < 41\} = \{3, 5, 7, 9\} = 4 = \lfloor (10+1)/2 \rfloor - 1$$

Oberer Median: 52, denn

$$\#\{i : A[i] < 52\} = \{1, 3, 5, 7, 9\} = 5 = \lceil (10+1)/2 \rceil - 1$$



Selektion durch Sortieren

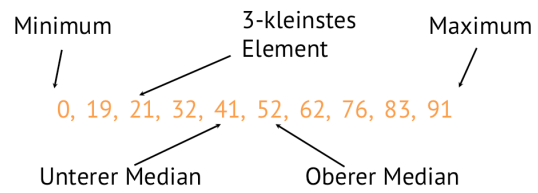
- `SortSelect(A,k)`
 `sort(A)`
 `return A[k]`
- Aufwand = Aufwand für Sortierung
 $T(n) = O(n \times \log n)$



k-Selektion: Beispiel

$A[1..10] = 41, 76, 32, 62, 21, 52, 19, 83, 0, 91$

Sortieren liefert:



2.4 Suchen

2.4.1 Selektion

2.4.1.1 Definitionen

2.4.1.2 Selektion durch Partitionierung

2.4.1.3 Median der Mediane

2.4.2 Hashing



Selektion durch Partitionierung

- PartitionSelect(A[1..n], l, r, k) Berechne das k-
kleinste Element
von A[l..r]
 if l = r then
 return A[l]
 m ← Partition(A, l, r) Precondition:
 i ← m - l + 1 1 ≤ k ≤ r - l + 1
 if k = i then Initialer Aufruf
 return A[m] mit l=1, r=n
 elseif k < i then
 return PartitionSelect(A, l, m-1, k)
 else
 return PartitionSelect(A, m+1, r, k-i)



Selektion durch Partitionierung

- Aufwand
 - Worst case (vgl. Quick-Sort)
 $T(n) = O(n^2)$
 - Average case (ohne Beweis)
 $T(n) = O(n)$
 - anschaulich: $n/2 + n/4 + \dots \leq n$



Avg. Case: nach jedem Schritt muss nur noch etwa die Hälfte der Elemente angeschaut werden

Beispiel

$k = 6$

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

11

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Beispiel

$k = 6$

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

Pivot = 4

2	9	8	3	7	5	1	6	4
↑								↑
i								j

12

Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	9	8	3	7	5	1	6	4
↑								↑
i								j

13

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 RWTH AACHEN
 UNIVERSITY

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	9	8	3	7	5	1	6	4
↑								↑
i								j

14

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 RWTH AACHEN
 UNIVERSITY

Beispiel

$k = 6$

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

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

↑

↑

i

j

15

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

RWTH AACHEN UNIVERSITY

Beispiel

$k = 6$

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

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

↑

↑

i

j

16

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

RWTH AACHEN UNIVERSITY

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9																											
2	9	8	3	7	5	1	6	4																											
<table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">9</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px; color: orange;">4</td> </tr> <tr> <td style="text-align: center;">↑</td><td></td><td></td><td></td><td></td><td></td><td style="text-align: center;">↑</td><td></td><td></td> </tr> <tr> <td style="text-align: center;">i</td><td></td><td></td><td></td><td></td><td></td><td style="text-align: center;">j</td><td></td><td></td> </tr> </table>									2	1	8	3	7	5	9	6	4	↑						↑			i						j		
2	1	8	3	7	5	9	6	4																											
↑						↑																													
i						j																													

17 Datenstrukturen und Algorithmen Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER UNIVERSITÄT

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9																											
2	9	8	3	7	5	1	6	4																											
<table style="margin-left: auto; margin-right: auto; border-collapse: collapse;"> <tr> <td style="border: 1px solid black; padding: 2px;">2</td><td style="border: 1px solid black; padding: 2px;">1</td><td style="border: 1px solid black; padding: 2px;">8</td><td style="border: 1px solid black; padding: 2px;">3</td><td style="border: 1px solid black; padding: 2px;">7</td><td style="border: 1px solid black; padding: 2px;">5</td><td style="border: 1px solid black; padding: 2px;">9</td><td style="border: 1px solid black; padding: 2px;">6</td><td style="border: 1px solid black; padding: 2px; color: orange;">4</td> </tr> <tr> <td style="text-align: center;">↑</td><td></td><td></td><td></td><td></td><td></td><td style="text-align: center;">↑</td><td></td><td></td> </tr> <tr> <td style="text-align: center;">i</td><td></td><td></td><td></td><td></td><td></td><td style="text-align: center;">j</td><td></td><td></td> </tr> </table>									2	1	8	3	7	5	9	6	4	↑						↑			i						j		
2	1	8	3	7	5	9	6	4																											
↑						↑																													
i						j																													

18 Datenstrukturen und Algorithmen Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer
 FRIEDRICH-ALEXANDER UNIVERSITÄT

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	8	3	7	5	9	6	4
		↑		↑				
		i		j				

19 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	8	3	7	5	9	6	4
		↑		↑				
		i		j				

20 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	8	3	7	5	9	6	4
	↑	↑						
	i	j						

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	3	8	7	5	9	6	4
	↑	↑						
	i	j						

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	3	8	7	5	9	6	4
↑								
j								
↑								
i								

23

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	3	8	7	5	9	6	4
↑			↑					
j			i					

24

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	8	3	7	5	9	6	4
	↑	↑						
	j	i						

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

1	2	3	4	5	6	7	8	9
2	9	8	3	7	5	1	6	4
2	1	3	8	7	5	9	6	4
	↑	↑						
	j	i						

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

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

↑ ↑
j i

27 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

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

↑
 $m = 4, i = m - l + 1 = 4$

28 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

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

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

$k = 6$

$k = 2$

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

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

Pivot = 8

31
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
---	---	---	---	---

↑
i

↑
j

32
Datenstrukturen und Algorithmen
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
↑				↑
i				j

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRITZ-HAGEN UNIVERSITY

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
↑				↑
i				j

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

FRITZ-HAGEN UNIVERSITY

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
	↑		↑	
	i		j	

35

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
	↑	↑		
	i	j		

36

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	9	8
---	---	---	---	---

↑ ↑

i j

37

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG

Beispiel

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

k = 6

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

k = 2

7	5	6	9	8
---	---	---	---	---

↑

j

↑

i

38

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

 FRIEDRICH-ALEXANDER
 UNIVERSITÄT
 ERLANGEN-NÜRNBERG

Beispiel

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

k = 6

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

k = 2

7	5	6	9	8
---	---	---	---	---

↑ ↑
 j i

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	9	6	8
---	---	---	---	---

↑ ↑
 j i

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	9	8
---	---	---	---	---

↑ ↑
 j i

41 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-SCHILLER
UNIVERSITÄT

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

↑ ↑
 j i

42 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer FRIEDRICH-SCHILLER
UNIVERSITÄT

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

↑
m = 8, i = m - l + 1 = 4

43

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

k = 2

7	5	6
---	---	---

44

Datenstrukturen und Algorithmen
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

k = 2

7	5	6
---	---	---

45 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

k = 2

5	6	7
---	---	---

46 **Datenstrukturen und Algorithmen**
 Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

k = 2

5	6	7
---	---	---

↑
 $m = 6, i = m - l + 1 = 2$

47 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

Beispiel

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

k = 6

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

k = 2

7	5	6	8	9
---	---	---	---	---

k = 2

5	6	7
---	---	---

Ergebnis 6

48 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer

2.4 Suchen

2.4.1 Selektion

2.4.1.1 Definitionen

2.4.1.2 Selektion durch Partitionierung

2.4.1.3 Median der Mediane

2.4.2 Hashing



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]..A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M| / 2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```



Ähnlich PartitionSelect, allerdings mit Garantie über Größe der Aufteilung

Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]...A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Basisfall der Rekursion, die magische Konstante wird später klar.



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]...A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Bilde $\lfloor n/5 \rfloor$ Gruppen von je 5 Elementen (es bleiben 0-4 Elemente übrig) und bestimme deren Median.



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]..A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋-1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Bestimme rekursiv den Median der Mediane.



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]..A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋-1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Nach Voraussetzung sind alle Elemente verschieden und damit $|S=| = \{m\} = 1$.



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]...A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Das k-te Element liegt in S<



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]...A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S=| = k then
    return m
  return Select(S>, k - |S<| - |S=|)
```

Das k-te Element liegt in S={m}



Median der Mediane

```
• Select(A, k)
  n ← |A|
  if n < 50 then
    return SortSelect(A, k)
  M ← { SortSelect(A[5i+1]..A[5i+5], 3) :
        i = 0, ..., ⌊n/5⌋ - 1 }
  m ← Select(M, ⌊|M|/2⌋)
  S< ← { A[i] : A[i] < m }
  S= ← { A[i] : A[i] = m }
  S> ← { A[i] : A[i] > m }
  if |S<| ≥ k then
    return Select(S<, k)
  if |S<| + |S|= = k then
    return m
  return Select(S>, k - |S<| - |S|=)
```

Das k-te Element ist das
(k - |S<| - |S|=) te Element aus
S>



Median der Mediane: Aufwand

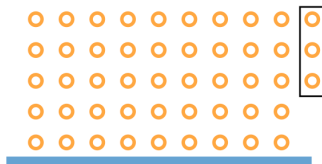
Ausgangs-Situation:



Median der Mediane: Aufwand

Anordnung in 5er Gruppen:

werden vernachlässigt



$\lfloor n/5 \rfloor$ Gruppen



Median der Mediane: Aufwand

Anordnung in 5er Gruppen:



$\lfloor n/5 \rfloor$ Gruppen



Median der Mediane: Aufwand

Anordnung in 5er Gruppen, *sortiert dargestellt*:

$\hat{o} < o$

M = Mediane der 5er Gruppen

m = Median der Mediane

$\lfloor n/5 \rfloor$ Gruppen

61 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer **FRITZ-HAGEN UNIVERSITY**

Konstanter Aufwand $O(1)$ für Mediane der 5er Gruppen
 Rekursiv auf Mediane der 5er Gruppen anwenden

Median der Mediane: Aufwand

Anordnung in 5er Gruppen, *sortiert dargestellt*:

$\hat{o} < o$

$\geq 3 \lfloor n/10 \rfloor$ Elemente

$\geq \lfloor n/10 \rfloor$ Gruppen

$\geq \lfloor n/10 \rfloor$ Gruppen

$\geq 3 \lfloor n/10 \rfloor$ Elemente

62 **Datenstrukturen und Algorithmen**
Prof. Dr. Leif Kobbelt, Thomas Stroder, Fabian Emmes, Sven Middelberg, Michael Kremer **FRITZ-HAGEN UNIVERSITY**

Median der Mediane: Aufwand

- mindestens $3\lfloor n/10 \rfloor$ Elemente sind $\leq m$
- höchstens $n - 3\lfloor n/10 \rfloor$ Elemente sind $> m$
- für $n > 50$ ist $|S_>| < n - 3\lfloor n/10 \rfloor < 3n/4$
- für $n > 50$ ist $|S_<| < n - 3\lfloor n/10 \rfloor < 3n/4$

$n=47$	$n - 3\lfloor n/10 \rfloor = 35$	$3n/4 = 35.25$
$n=48$	$n - 3\lfloor n/10 \rfloor = 36$	$3n/4 = 36.00$
$n=49$	$n - 3\lfloor n/10 \rfloor = 37$	$3n/4 = 36.75$
$n=50$	$n - 3\lfloor n/10 \rfloor = 35$	$3n/4 = 37.50$
$n=51$	$n - 3\lfloor n/10 \rfloor = 36$	$3n/4 = 38.25$



Median der Aufwand

- `Select(A, k)`
 - $n \leftarrow |A|$
 - `if` $n < 50$ `then`
 - `return` `SortSelect(A, k)` } O(1)
 - $M \leftarrow \{ \text{SortSelect}(A[5i+1]..A[5i+5], 3) : i = 0, \dots, \lfloor n/5 \rfloor - 1 \}$ } O(n)
 - $m \leftarrow \text{Select}(M, \lfloor |M| / 2 \rfloor)$ } T(n/5)
 - $S_< \leftarrow \{ A[i] : A[i] < m \}$
 - $S_= \leftarrow \{ A[i] : A[i] = m \}$ } O(n)
 - $S_> \leftarrow \{ A[i] : A[i] > m \}$
 - `if` $|S_<| \geq k$ `then`
 - `return` `Select(S_<, k)`
 - `if` $|S_<| + |S_=| = k$ `then`
 - `return` m
 - `return` `Select(S_>, k - |S_<| - |S_=|)` } $< T(3n/4)$



Median der Aufwand

- Rekurrenz

$$T(n) \leq \begin{cases} c_1 & \text{für } n < 50 \\ c_2n + T(\frac{n}{5}) + T(\frac{3n}{4}) & \text{für } n \geq 50 \end{cases}$$

- Behauptung: für $c = \max(c_1, 20c_2)$ ist
 $T(n) < cn = O(n)$



Median der Aufwand

- Induktionsanfang: für $n < 50$ okay, da $c > c_1$
- Induktionsannahme: $T(m) \leq cm$ für $m < n$
- Induktionsschluss:

$$\begin{aligned} T(n) &\leq c_2n + T(n/5) + T(3n/4) \\ &\leq c_2n + cn/5 + 3cn/4 \\ &\leq c_2n + 19cn/20 \\ &\leq cn + n(c_2 - c/20) \\ &\leq cn \end{aligned}$$

$$20c_2 \leq c \Leftrightarrow c_2 \leq c/20 \Leftrightarrow c_2 - c/20 \leq 0$$



Selektion: Zusammenfassung

- Selektion durch Sortieren: $O(n \times \log n)$
- Selektion durch Partitionieren
 - Worst case: $O(n^2)$
 - Average case: $O(n)$
- Median der Mediane: $O(n)$

