

Übung 2 – Algorithmen II

Sebastian Lamm, Demian Hespe – lamm@kit.edu, hespe@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS18.php

Institut für Theoretische Informatik - Algorithmik II

```
    result = current_weight;
    return true;
}

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

- Ganzzahlige *Priority Queues*
 - *Bucket Queues*
 - *Radix Heaps*
- *Average case* Analyse am Beispiel MST

Spezielle *Priority Queues*

monoton, ganzzahlig

Bedingungen

- positive, **ganzzahlige** Schlüssel
- neue/geänderte Schlüssel $\geq$ minimaler Schlüssel *min*
- **maximales** Schlüsselinkrement *C*
(bezogen auf minimalen Schlüssel *min*)

Spezielle *Priority Queues*

monoton, ganzzahlig

Warum das Ganze?

- spezialisierte Datenstruktur → schneller
- Dijkstras Algorithmus

Idee

- speichere Schlüssel in *Buckets* statt *Bäumen*

Spezielle *Priority Queues*

Dijkstras Algorithmus

Laufzeit Dijkstras Algorithmus

■ $T_{Dijkstra} = O(m \cdot T_{decreaseKey} + n \cdot (T_{deleteMin} + T_{insert}))$

amortisierte Laufzeiten

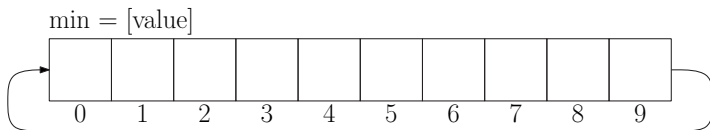
$O(\cdot)$	<i>Bin. Heap</i>	<i>Fib. Heap</i>	<i>Bkt. Queue</i>	<i>Radix Heap</i>
$T_{decreaseKey}$	$\log n$	1	1	1
T_{insert}	$\log n$	1	1	$\log C$
$T_{deleteMin}$	$\log n$	$\log n$	C	$\log C$
$T_{Dijkstra}$	$(m + n) \log n$	$m + n \log n$	$m + nC$	$m + n \log C$

Aufbau:

- zirkuläre Liste aus Buckets $B[\cdot]$
- Variable mit minimalem Schlüssel min
(der in die Datenstruktur aufgenommen werden kann)
- min : letzter deleteMin-Schlüssel, initialisiert mit 0

gegeben:

- max. Schlüsselinkrement $C \longrightarrow \# \text{ Buckets } |B| = C + 1$



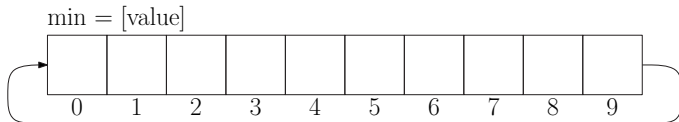
$$C = 9 \longrightarrow |B| = C + 1 = 10$$

Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

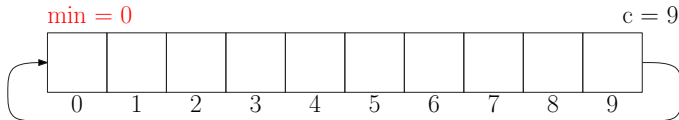


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

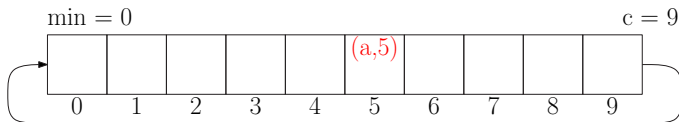


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

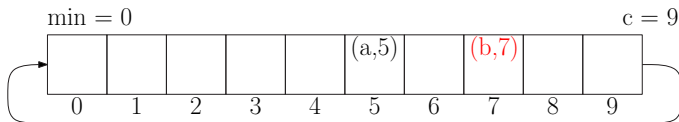


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

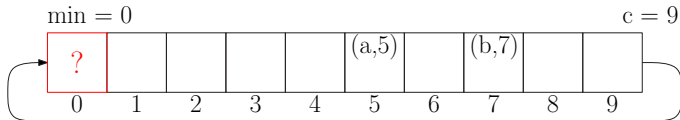


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

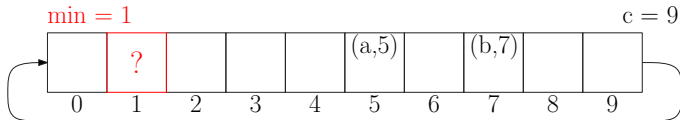


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

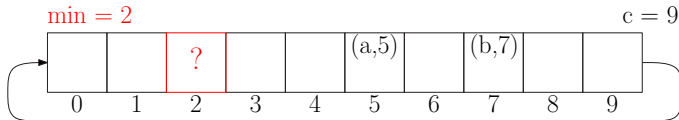


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

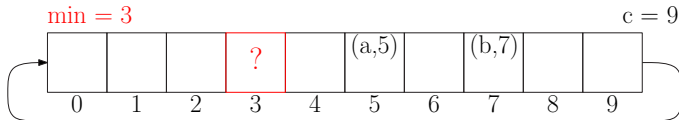


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

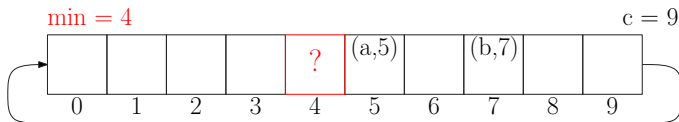


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

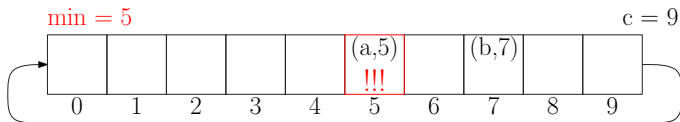


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

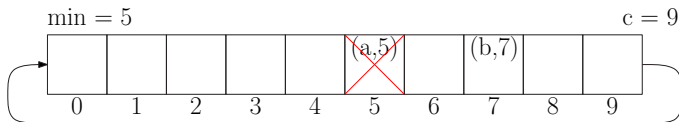


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

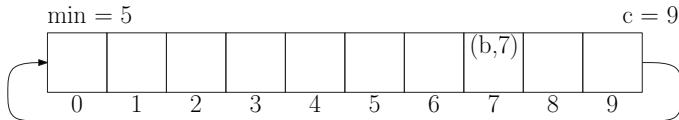


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

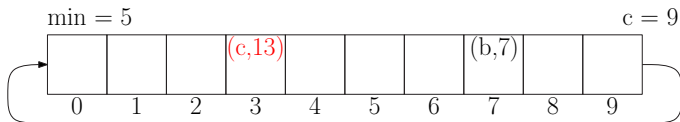


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

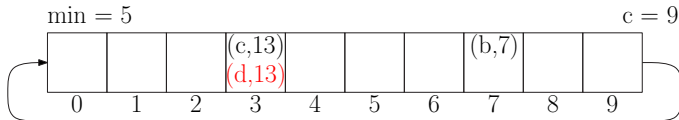


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

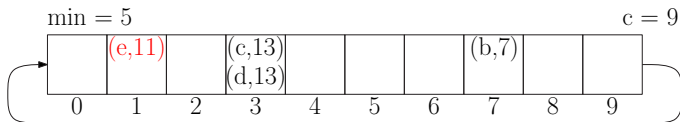


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[\min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

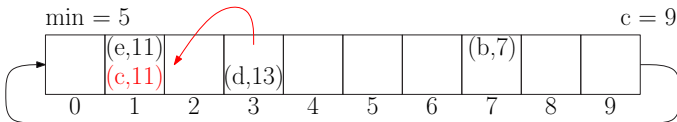


Ablauf:

insert:	Einfügen in Bucket $B[key \bmod (C + 1)]$	$O(1)$
deleteMin:	Entfernen aus Bucket $B[min \bmod (C + 1)]$ (bzw. aus erstem nicht-leeren Folgebucket)	$O(C)$
decreaseKey:	Verschieben von altem in neuen Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



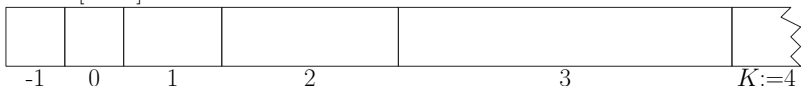
Aufbau:

- Liste aus **logarithmischer Anzahl** Buckets $B[\cdot]$
- Variable mit minimalem Schlüssel min
(der in die Datenstruktur aufgenommen werden kann)

gegeben:

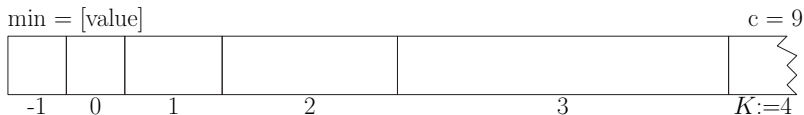
- max. Schlüsselinkrement $C \longrightarrow \# \text{ Buckets } |B| = K + 2$
 $= (\lfloor \log C \rfloor + 1) + 2$

$min = [value]$

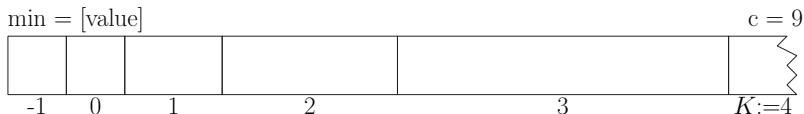


$$C = 9 \longrightarrow K + 2 = (\lfloor \log C \rfloor + 1) + 2 = (3 + 1) + 2 = 6$$

Radix Heaps

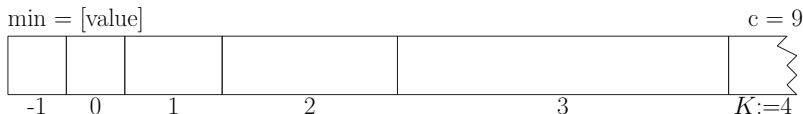


Welche Schlüssel kommen in welches Bucket?



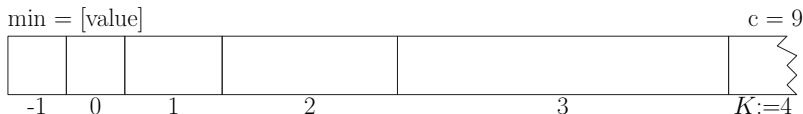
Welche Schlüssel kommen in welches Bucket?

- Bucket i : hält Elemente mit $i = \min(\text{msd}(\text{min}, \text{key}), K)$
- Bucket K ist *overflow* Bucket
 - höchstwertiges unterschiedliche Bit ist i
(bzw. -1 wenn $\text{key} = \text{min}$)
 - $\max(1, 2^i)$ verschiedene Schlüssel
 - leer, wenn Bit i von min gesetzt



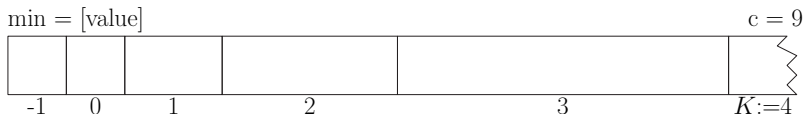
Welche Schlüssel kommen in welches Bucket?

- Bucket i : hält Elemente mit $i = \min(\text{msd}(\min, \text{key}), K)$
- Bucket K ist *overflow* Bucket
 - höchstwertiges unterschiedliche Bit ist i
(bzw. -1 wenn $\text{key} = \min$)
 - $\max(1, 2^i)$ verschiedene Schlüssel
 - leer, wenn Bit i von $\min$ gesetzt



Welche Schlüssel kommen in welches Bucket?

- Bucket i : hält Elemente mit $i = \min(\text{msd}(\text{min}, \text{key}), K)$
- Bucket K ist *overflow* Bucket
 - höchstwertiges unterschiedliche Bit ist i
(bzw. -1 wenn $\text{key} = \text{min}$)
 - $\max(1, 2^i)$ verschiedene Schlüssel
 - leer, wenn Bit i von min gesetzt



Welche Schlüssel kommen in welches Bucket?

Bucket i : hält Elemente mit $i = \min(\text{msd}(\text{min}, \text{key}), K)$

→ Bucket K ist *overflow* Bucket

→ höchstwertiges unterschiedliche Bit ist i

(bzw. -1 wenn $\text{key} = \text{min}$)

→ $\max(1, 2^i)$ verschiedene Schlüssel

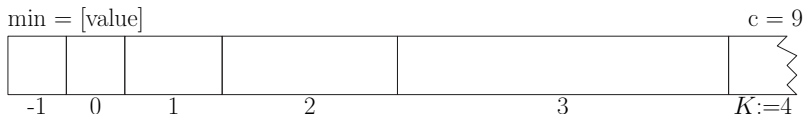
→ leer, wenn Bit i von min gesetzt

Beispiele

$\text{min} := 00001000$ (08)

$\text{msd}(00001000, 01001000) = 6$

$\text{msd}(00001000, 00001010) = 1$



Welche Schlüssel kommen in welches Bucket?

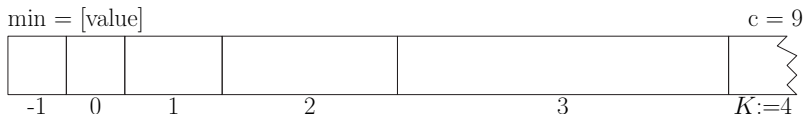
- Bucket i : hält Elemente mit $i = \min(\text{msd}(\text{min}, \text{key}), K)$
- Bucket K ist *overflow* Bucket
 - höchstwertiges unterschiedliche Bit ist i
(bzw. -1 wenn $\text{key} = \text{min}$)
 - max(1, 2^i) verschiedene Schlüssel
 - leer, wenn Bit i von min gesetzt

Beispiele

$\text{min} := 00001000$ (08)

$\text{msd}(00001000, 01*****) = 6 \rightarrow 2^6 = 64$ Werte

$\text{msd}(00001000, 0000101*) = 1 \rightarrow 2^1 = 2$ Werte



Welche Schlüssel kommen in welches Bucket?

Bucket i : hält Elemente mit $i = \min(\text{msd}(\text{min}, \text{key}), K)$

→ Bucket K ist *overflow* Bucket

→ höchstwertiges unterschiedliche Bit ist i

(bzw. -1 wenn $\text{key} = \text{min}$)

→ $\max(1, 2^i)$ verschiedene Schlüssel

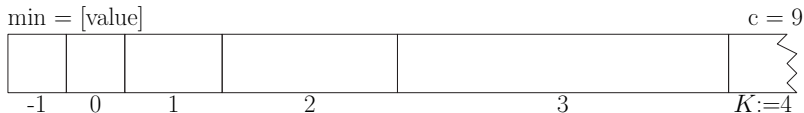
→ **leer, wenn Bit i von min gesetzt**

Beispiele

$\text{min} := 00001000$ (08)

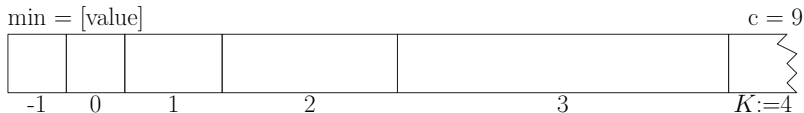
$\text{msd}(0000\mathbf{1}000, 0000\mathbf{0}110) = 3 \rightarrow$ kann nicht passieren, da $6 < 8$

$\text{msd}(0000\mathbf{1}000, 0000\mathbf{0}100) = 3 \rightarrow$ kann nicht passieren, da $4 < 8$



Welche Schlüssel kommen in welches Bucket?

	Bucket $B[\cdot]$					
<i>min</i>	-1	0	1	2	3	4
1000 (08)	8	9	10..11	12..15	-	16..8+C
1010 (10)	10	11	-	12..15	-	16..10+C
1111 (15)	15	-	-	-	-	16..15+C
1000100 (68)	68	69	70..71	-	72..79 ?	80..68+C ?



Welche Schlüssel kommen in welches Bucket?

	Bucket $B[\cdot]$					
<i>min</i>	-1	0	1	2	3	4
1000 (08)	8	9	10..11	12..15	-	16..8+C
1010 (10)	10	11	-	12..15	-	16..10+C
1111 (15)	15	-	-	-	-	16..15+C
1000100 (68)	68	69	70..71	-	72..77 !	- !

Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

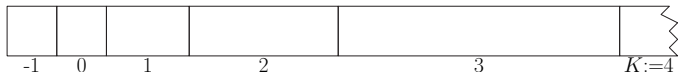
■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0000



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0000



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

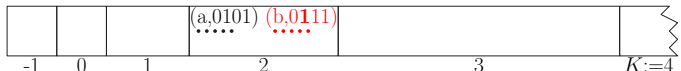
■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0000



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

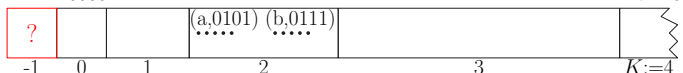
■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0000

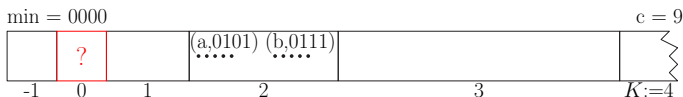


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	$\text{min} = \text{minimaler Schlüssel (aus Bucket } i\text{)},$ Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

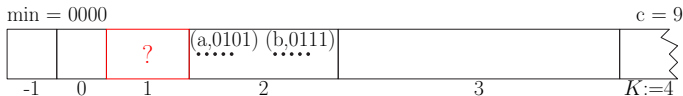
■ deleteMin()

■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

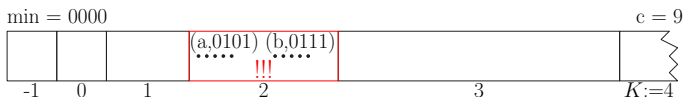


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

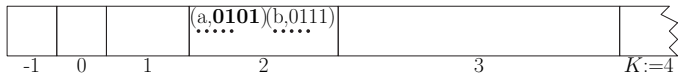
■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0101

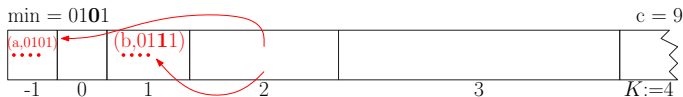


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

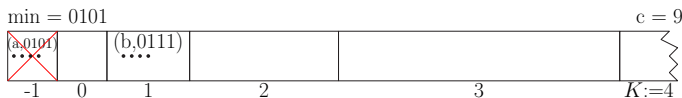


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

■ insert(a,5)

■ insert(b,7)

■ deleteMin()

■ insert(c,13)

■ insert(d,13)

■ insert(e,11)

■ decreaseKey(c,11)

min = 0101

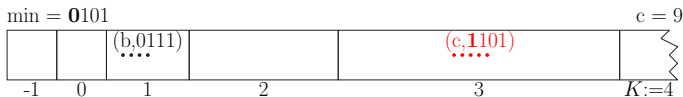


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

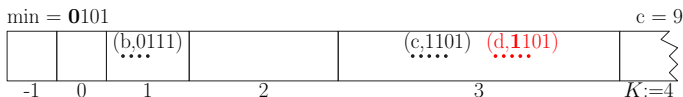


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

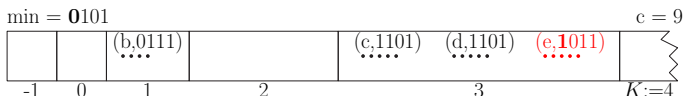


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)

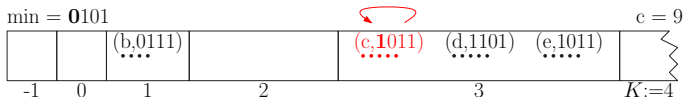


Ablauf und Laufzeiten (amortisiert):

insert:	Einfügen in Bucket $B[\min(\text{msd}(\text{min}, \text{key}), K)]$	$O(\log C)$
deleteMin:	min = minimaler Schlüssel (aus Bucket i), Elemente aus Bucket i verschieben, Entfernen aus Bucket $B[-1]$	$O(\log C)$
decreaseKey:	Verschieben von altem in neues Bucket	$O(1)$

Beispiel:

- insert(a,5)
- insert(b,7)
- deleteMin()
- insert(c,13)
- insert(d,13)
- insert(e,11)
- decreaseKey(c,11)



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Radix Heaps

Beispiel (fortgesetzt):

■ `insert(f,14)`

■ `deleteMin()`

■ `deleteMin()`

■ `insert(g,20)`

■ `deleteMin()`

■ `insert(h,18)`

■ `deleteMin()`

■ `decreaseKey(g,16)`

■ `deleteMin()`

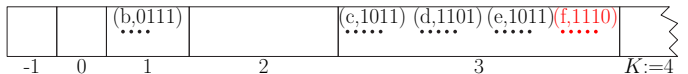
■ `deleteMin()`

■ `insert(i,24)`

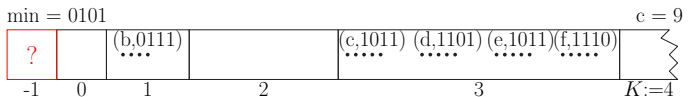
■ `insert(j,22)`

■ `decreaseKey(i,16)`

min = 0101



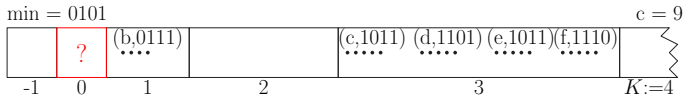
- decreaseKey(i, 16)



Radix Heaps

Beispiel (fortgesetzt):

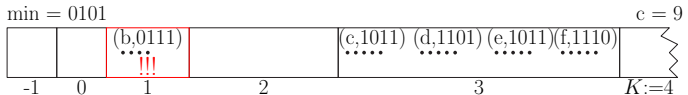
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

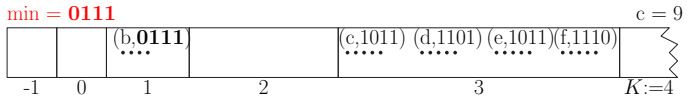
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

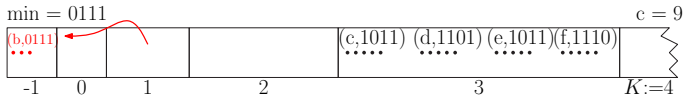
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

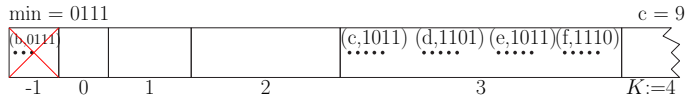
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

■ `insert(f,14)`

■ `deleteMin()`

■ `deleteMin()`

■ `insert(g,20)`

■ `deleteMin()`

■ `insert(h,18)`

■ `deleteMin()`

■ `decreaseKey(g,16)`

■ `deleteMin()`

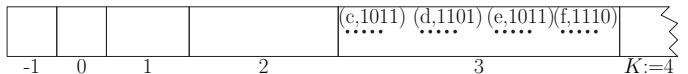
■ `deleteMin()`

■ `insert(i,24)`

■ `insert(j,22)`

■ `decreaseKey(i,16)`

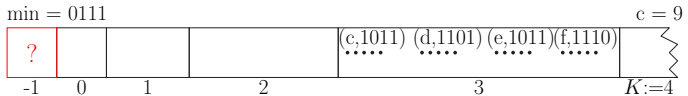
min = 0111



Radix Heaps

Beispiel (fortgesetzt):

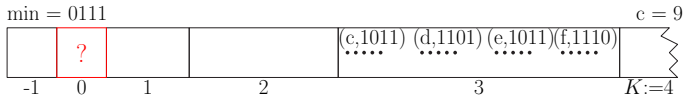
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

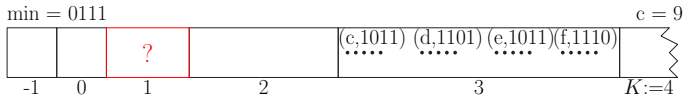
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

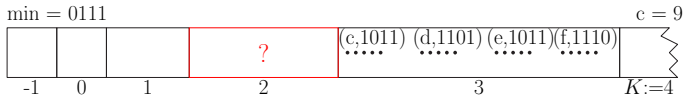
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

Beispiel (fortgesetzt):

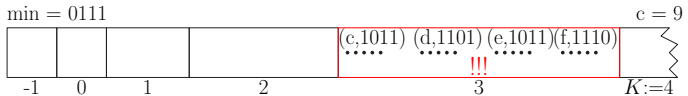
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



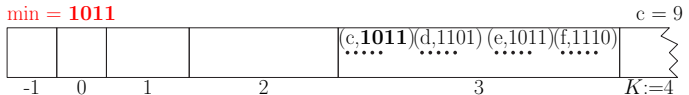
Radix Heaps

Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`

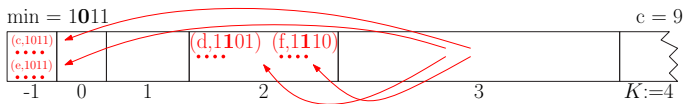


- decreaseKey(i, 16)



Beispiel (fortgesetzt):

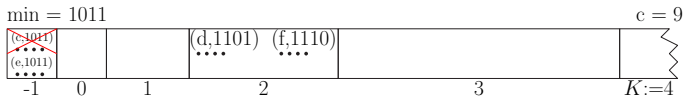
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Radix Heaps

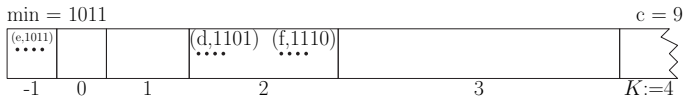
Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Beispiel (fortgesetzt):

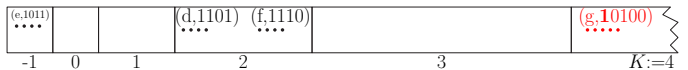
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

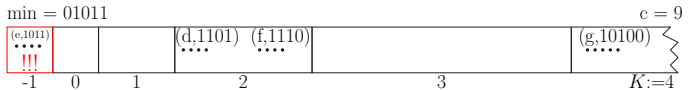
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01011



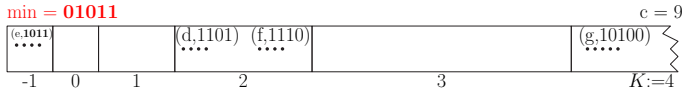
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

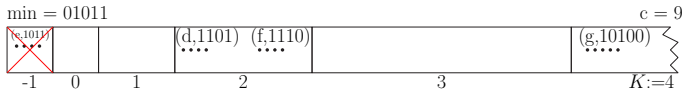
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Radix Heaps

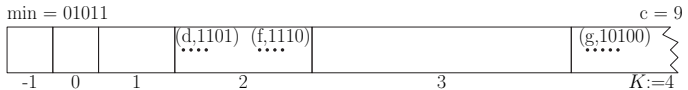
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



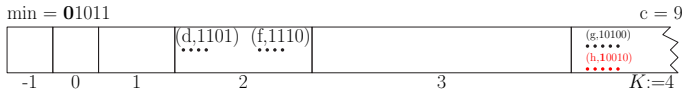
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

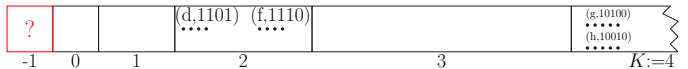


Radix Heaps

Beispiel (fortgesetzt):

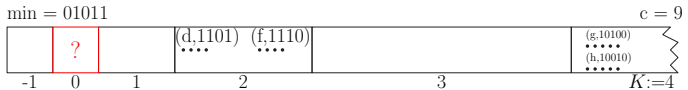
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01011



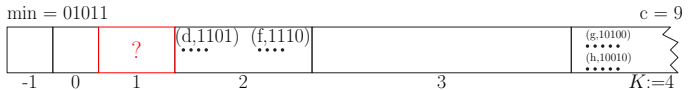
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

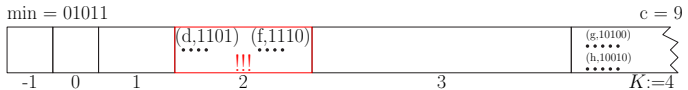
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- **deleteMin()**
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Radix Heaps

Beispiel (fortgesetzt):

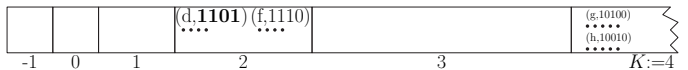
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01101



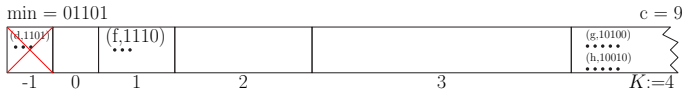
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



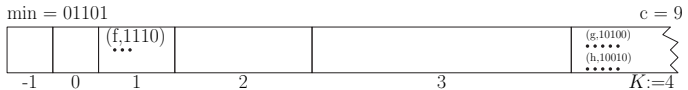
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- **deleteMin()**
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

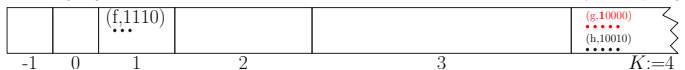
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- **deleteMin()**
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

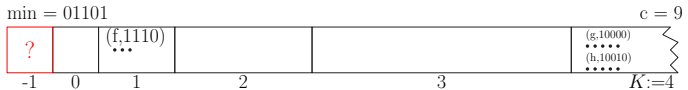
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01101



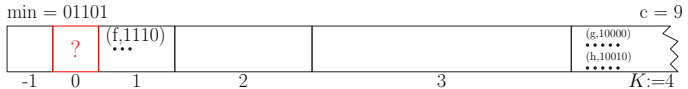
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



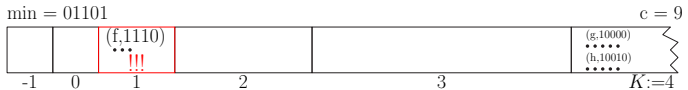
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



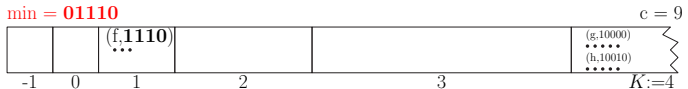
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



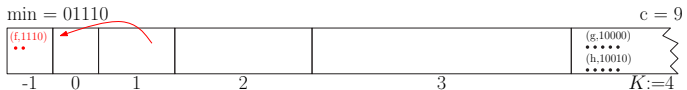
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

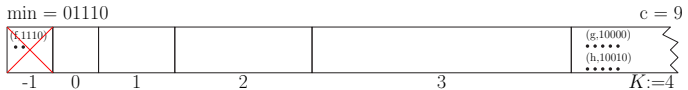
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Radix Heaps

Beispiel (fortgesetzt):

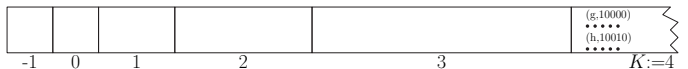
- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`

min = 01110



Beispiel (fortgesetzt):

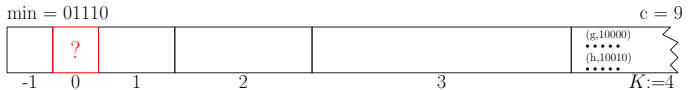
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- **deleteMin()**
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01110



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- **deleteMin()**
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- **deleteMin()**
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01110



Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01110

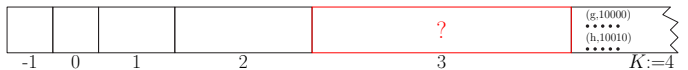


Radix Heaps

Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`

min = 01110



Beispiel (fortgesetzt):

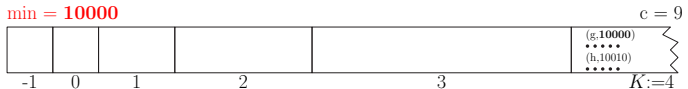
- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- **deleteMin()**
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)

min = 01110



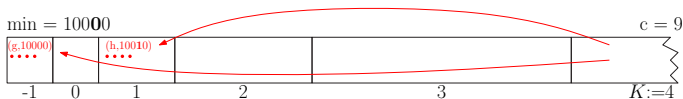
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



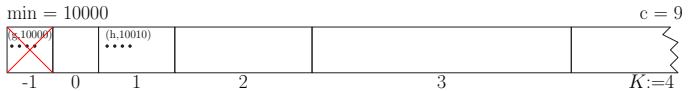
Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



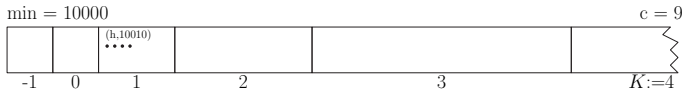
Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



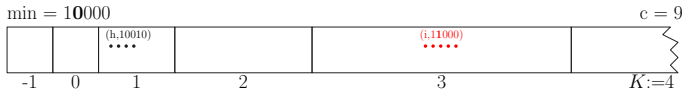
Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



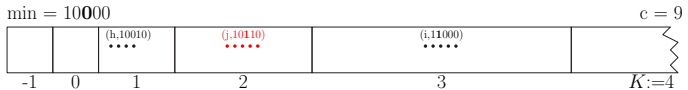
Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



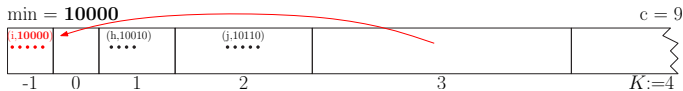
Beispiel (fortgesetzt):

- insert(f,14)
- deleteMin()
- deleteMin()
- insert(g,20)
- deleteMin()
- insert(h,18)
- deleteMin()
- decreaseKey(g,16)
- deleteMin()
- deleteMin()
- insert(i,24)
- insert(j,22)
- decreaseKey(i,16)



Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Beispiel (fortgesetzt):

- `insert(f,14)`
- `deleteMin()`
- `deleteMin()`
- `insert(g,20)`
- `deleteMin()`
- `insert(h,18)`
- `deleteMin()`
- `decreaseKey(g,16)`
- `deleteMin()`
- `deleteMin()`
- `insert(i,24)`
- `insert(j,22)`
- `decreaseKey(i,16)`



Warum komplizierte Formel $\min(\text{msd}(\text{min}, \text{key}), K)$?

- viel anschaulicher wäre doch

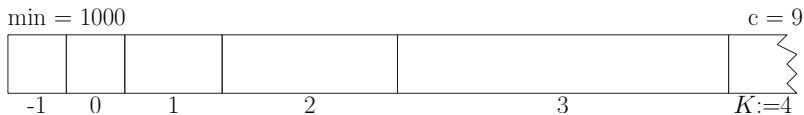
$$i = \lceil \log(\text{key} - \text{min}) \rceil$$

- okay, Anzahl Verschiebungen **erhöht sich nicht** ...
- ... aber, jede Änderung von *min* ändert **potentiell alle Buckets**
→ **schlechtere Laufzeit!**

Beispiel

$\text{min} := 01000 \text{ (08)} \xrightarrow{\text{deleteMin}} \text{min} := 01010 \text{ (10)}$

	Bucket $B[\cdot]$					
<i>min</i>	-1	0	1	2	3	4
1000 (08)	8	9	10	11..12	13..16	17..8+C
1010 (10)	10	11	12	13..14	15..18	19..10+C



Änderung von min $\rightarrow$ Umverteilung eines Bucket genügt

$min := 01000$ (08)

	Bucket $B[.]$					
min	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16

$\min = 1000$
 $c = 9$


Änderung von $\min \rightarrow$ Umverteilung eines Bucket genügt

$\min := 01000$ (08) $\xrightarrow{\text{deleteMin}}$ $\min := 01001$ (09)

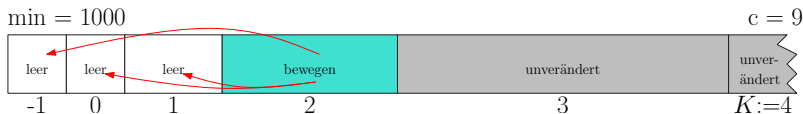
	Bucket $B[\cdot]$					
$\min$	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16
01001 (09)	9	-	10..11	12..15	-	≥ 16



Änderung von min $\rightarrow$ Umverteilung eines Bucket genügt

$min := 01000 (08) \xrightarrow{\text{deleteMin}} min := 0101 * (10..11)$

	Bucket $B[\cdot]$					
min	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16
01010 (10)	10	11	-	12..15	-	≥ 16
01011 (11)	11	-	-	12..15	-	≥ 16

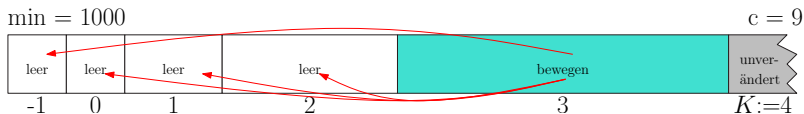


Änderung von min $\rightarrow$ Umverteilung eines Bucket genügt

$min := 01000 (08) \xrightarrow{\text{deleteMin}} min := 011 ** (12..15)$

	Bucket $B[.]$					
min	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16

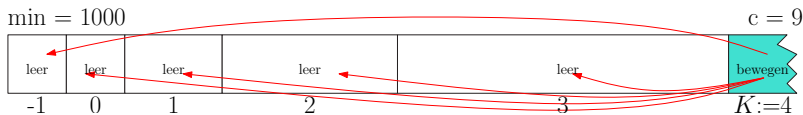
01100 (12)	12	13	14..15	-	-	≥ 16
01101 (13)	13	-	14..15	-	-	≥ 16
01110 (14)	14	15	-	-	-	≥ 16
01111 (15)	15	-	-	-	-	≥ 16



Änderung von min $\rightarrow$ Umverteilung eines Bucket genügt

$min := 01000 (08) \xrightarrow{\text{deleteMin}} min := -, \text{ Bucket 3 ist leer}$

	Bucket $B[.]$					
min	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16



Änderung von min $\rightarrow$ Umverteilung eines Bucket genügt

$min := 01000$ (08) $\xrightarrow{\text{deleteMin}}$ $min := 1****$ (16..)

	Bucket $B[.]$					
min	-1	0	1	2	3	4
01000 (08)	8	9	10..11	12..15	-	≥ 16

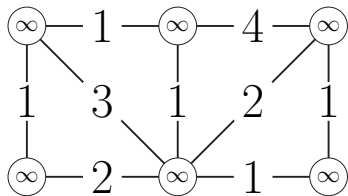
10000 (16)	16	17	-	-	-	-
10001 (17)	17	-	-	-	-	-

Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von **Jarnik-Prim**

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

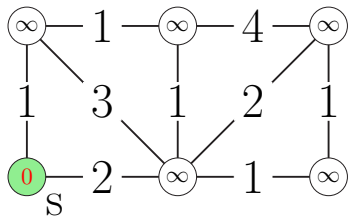


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von **Jarnik-Prim**

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

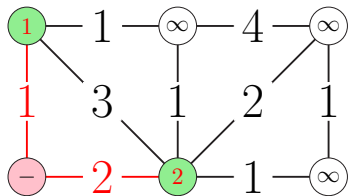


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von **Jarnik-Prim**

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

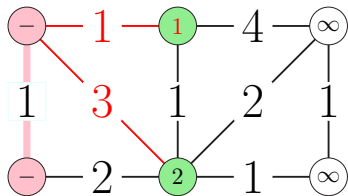


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von *Jarnik-Prim*

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

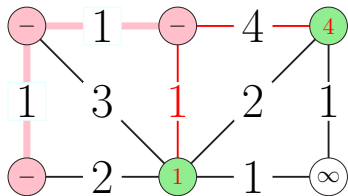


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von *Jarnik-Prim*

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

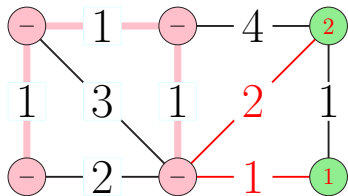


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von *Jarnik-Prim*

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

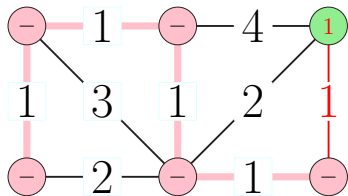


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von *Jarnik-Prim*

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!

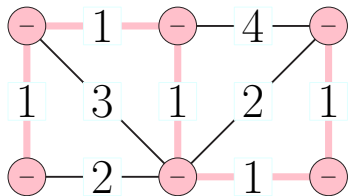


Motivation

- Algorithmus zur Berechnung minimaler Spannbäume (MSTs)
(siehe Algorithmen I: Algorithmus von Jarnik-Prim)
- Beweis **analog** zu Dijkstras Algorithmus

Kurze Wiederholung von **Jarnik-Prim**

- Wähle beliebigen Startknoten s
 - Füge iterativ Knoten v zu MST mit kleinstem Abstand $d[v]$
- Verwalte vorläufige Abstände $d[\cdot]$ in *Priority Queue*!



Durchschnittliche Laufzeit Jarnik Prim Algorithmus

- **Annahme:** Gewichte $1, \dots, m$ sind Kanten zufällig zugeordnet.
- $T_{Prim} = O(m + n \cdot (T_{insert} + T_{deleteMin} + T_{insertMST}) + x \cdot T_{decreaseKey})$

Durchschnittliche Anzahl decreaseKey Operationen

■ betrachte Knoten v :

decreaseKey(v), wenn Knoten u zu MST
hinzugefügt wird mit $c(u, v) < d[v]$

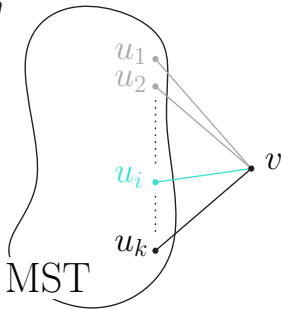
→ Reihenfolge $u_1, \dots, u_k$

→ $c(u_i, v) < \min_{j < i} c(u_j, v)$

■ Wie oft tritt dies auf?

→ Erwartungswert M_k für Anzahl Präfixminima

→ $\mathbb{E}(M_k) = H_k$ (k 'te harmonische Zahl)



Zufallsvariablen

- M_k : Anzahl Präfixminima einer Folge von k Zahlen
- I_i : $I_i = 1$ gdw. i -te Zahl Präfixminimum (**Indikator**) $\rightarrow \mathbb{P}[I_i = 1] = \frac{1}{i}$

$$\begin{aligned}\mathbb{E}(M_k) &= \mathbb{E}\left(\sum_{i=1}^k I_i\right) = \sum_{i=1}^k \mathbb{E}(I_i) \\ &= \sum_{i=1}^k \frac{1}{i} = H_k\end{aligned}$$

Intuition

- Wahrscheinlichkeit, dass i -tes Element Präfixminimum ist, ist $\frac{1}{i}$, da nur ein Minimum in den ersten i Elementen existiert.

Durchschnittliche Anzahl decreaseKey Operationen (weiter)

- erwartet H_k Prefixminima
 - Erstes Minimum $\rightarrow \text{insert}(v)$
 - $H_k - 1$ Minima $\rightarrow \text{decreaseKey}(v)$
(Bemerkung: $H_k - 1 < \ln k = \ln \text{grad}(v)$)

- Gesamt:

$$\begin{aligned} x &:= \sum_{v \in V} (H_k - 1) = \sum_{v \in V} (H_{\text{grad}(v)} - 1) \\ &< \sum_{v \in V} \ln \text{grad}(v) \stackrel{\text{konkav}}{\leq} n \ln \frac{2m}{n} = O(n \ln \frac{m}{n}) \end{aligned}$$

Legende

	Kante mit Gewicht 4		Zeiger auf Vorgänger (in Vorwärtsrichtung)
	Kante mit Gewicht 4, wird gerade in Vorwärtsrichtung betrachtet		
	Kante mit Gewicht 4, wird gerade in Rückwärtsrichtung betrachtet		Zeiger auf Vorgänger (in Rückwärtsrichtung)
	Kante mit reduziertem Gewicht 1		
	neu eingefügte Kante mit Gewicht 4		
	Knoten v mit oberer Schranke für Distanz vom Start $d[v] = \infty$ und Potential $pot[v] = 2$		neu eingefügter Knoten
	<u>erreichter</u> Knoten v mit oberer Schranke für Distanz vom Start $d[v] = 2$ (in <i>Priority Queue</i>)		
	<u>erreichter</u> Knoten v mit oberer Schranke für Distanz vom Start $d[v] = 1$ (in <i>Priority Queue</i>) (obere Schranke gerade verkleinert)		
	<u>gescannt</u> Knoten v mit exakter Distanz vom Start $d[v] = 1$ (aus <i>Priority Queue</i> entfernt)		
	Knoten in Vorwärtsrichtung <u>gescannt</u> mit exakter Distanz vom Start $d_{fwd}[v] = 1$, in Rückwärtsrichtung <u>erreicht</u> mit oberer Schranke für Distanz zum Ziel $d_{bwd}[v] = 3$		
	Knoten in Vorwärtsrichtung <u>gescannt</u> mit $d_{fwd}[v] = 1$, in Rückwärtsrichtung <u>erreicht</u> mit $d_{bwd}[v] = 2$ (obere Schranke in Rückwärtsrichtung gerade verkleinert)		
	aktueller Treffpunkt von Vorwärts- und Rückwärtssuche mit minimalem $d_{fwd}[v] + d_{bwd}[v] = 3$		
	<u>erreichter</u> Knoten v mit oberer Schranke für Distanz vom Start $d[v] = 2$ (in <i>Priority Queue</i>) (Schlüssel in <i>Priority Queue</i> ist $d[v] + pot[v] = 3 + 2$)		

Ende!



Feierabend!