Stream Processing II

K-Buckets Histogram

- Histogramme sind graphische Darstellungen der Verteilung von numerischen Werten
- Werden durch Intervalle, die sich nicht überlappen, dargestellt
- Ein Intervall wird durch seine Grenzen und einen Zähler angegeben

Split & Merge

- Tritt beim Insert auf
- Wird durchgeführt wenn der Zähler eines Buckets größer wird als ein vorgegebener Threshold
- Teilt zu großen Bucket in zwei Buckets auf, und verschmilzt zwei nebeneinanderliegende Buckets

Merge & Split

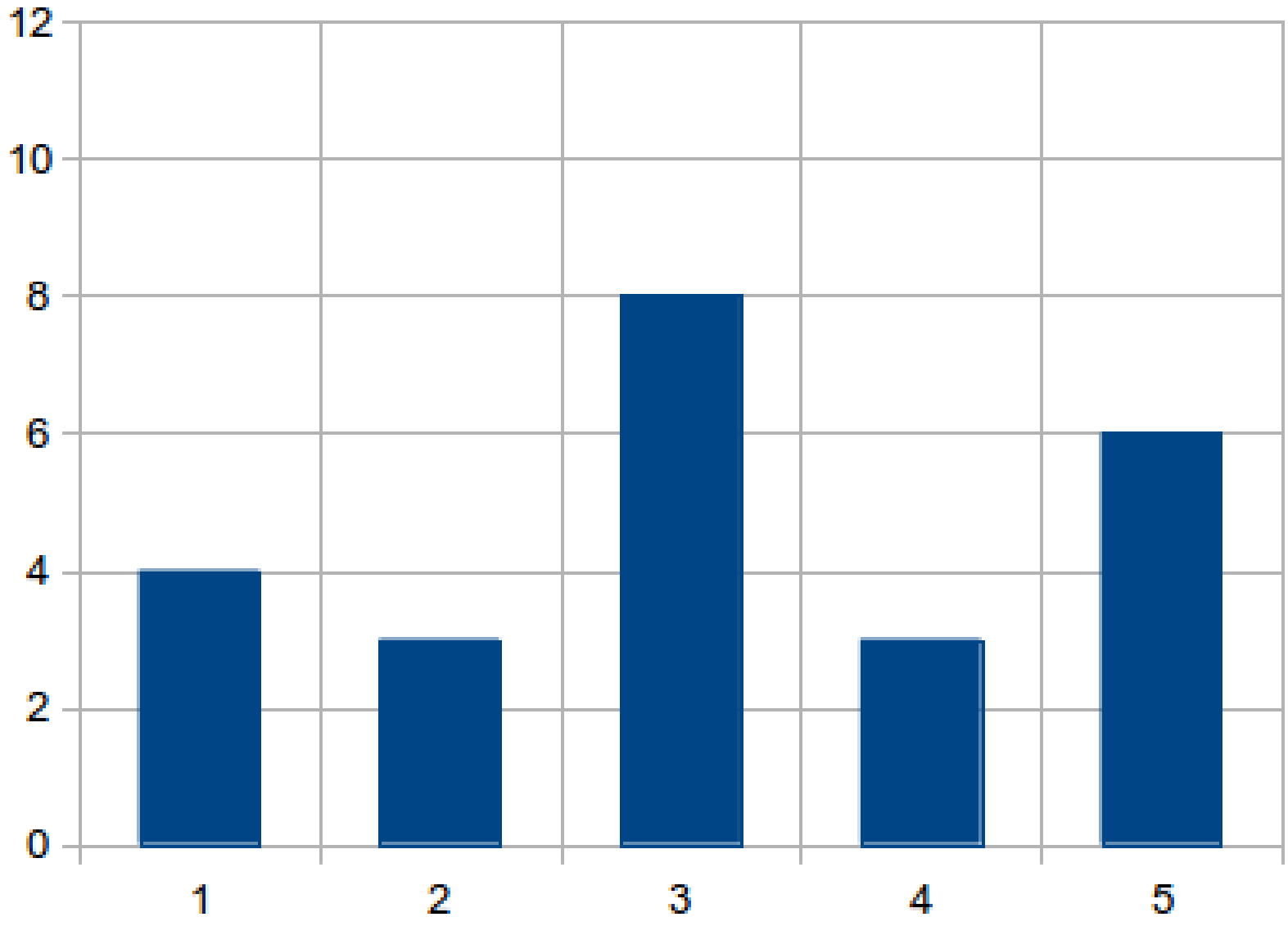
- Tritt beim Löschen auf
- Wird durchgeführt wenn der Zähler eines Buckets kleiner wird als ein vorgegebener Threshold
- Verschmilzt zu kleinen Bucket mit einem Nachbar, und teilt den Bucket mit den höchsten Zähler auf

K-Buckets Histogram

Gegeben sei das folgende Histogramm. Führen Sie den K-Buckets Algorithmus für das Einfügen aus, bis der erste Overflow auftritt und führen Sie dann Split & Merge durch. Nehmen Sie dabei folgende Regeln an:

- Das Histogramm besteht aus k=5 Buckets
- Der obere Threshold (Max) pro Bucket ist 10, der untere (Min) 2
- Bei Split & Merge tritt der Split auf,wenn ein Bucket mehr Elemente hat als 10. Der Merge wird zwischen den zwei nebeneinanderliegenden Buckets durchgeführt, die nicht am Split beteiligt waren, und die zusammengezählt am wenigsten Elemente enthalten.
- Die einzufügenden Elemente sind die Sequenz $s = (3, 1, 3, 5, 2, 3, 4, 1, 5, 3)$. Jedes Item ist nach dem Index seines Buckets benannt.

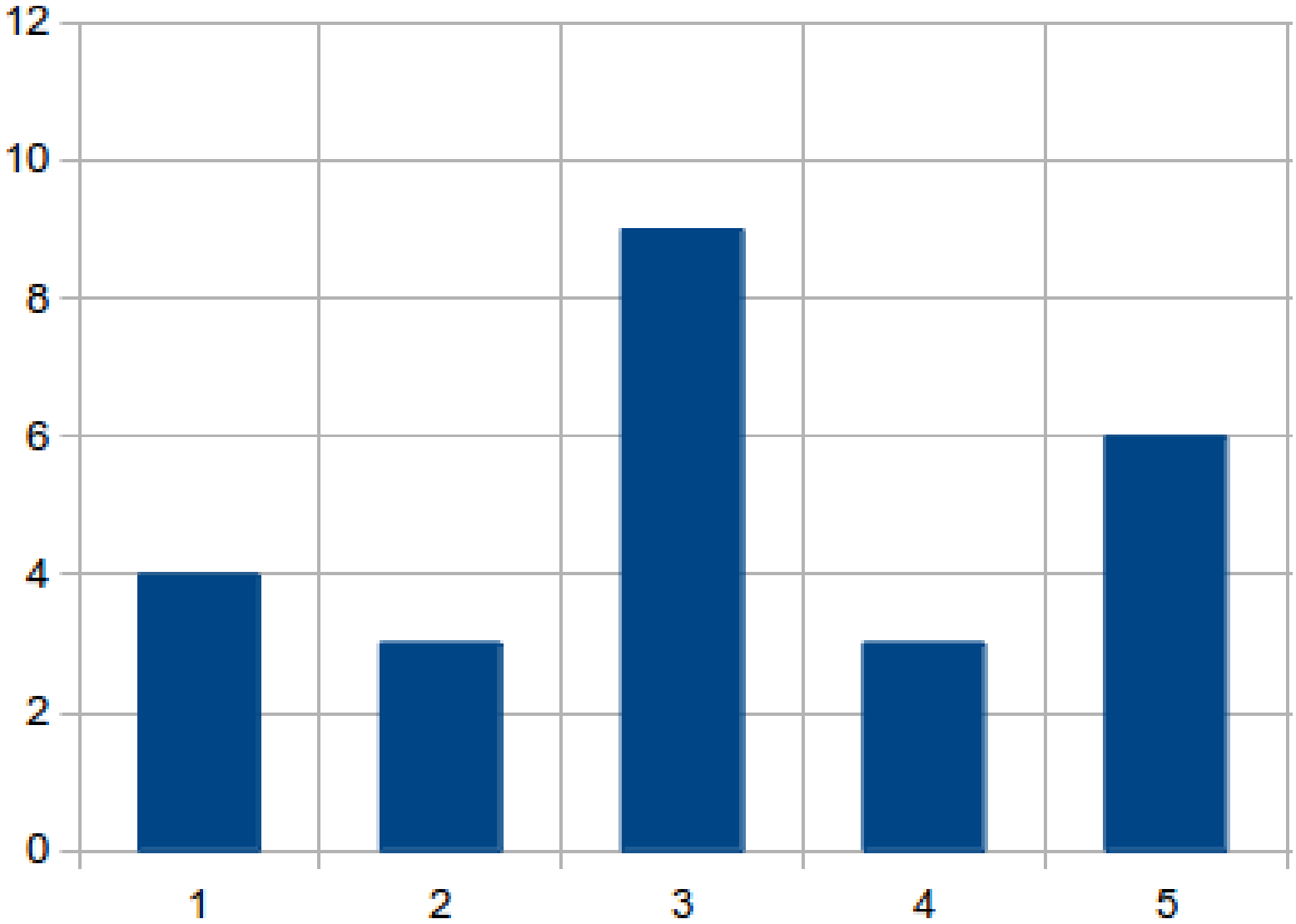
Histogramm



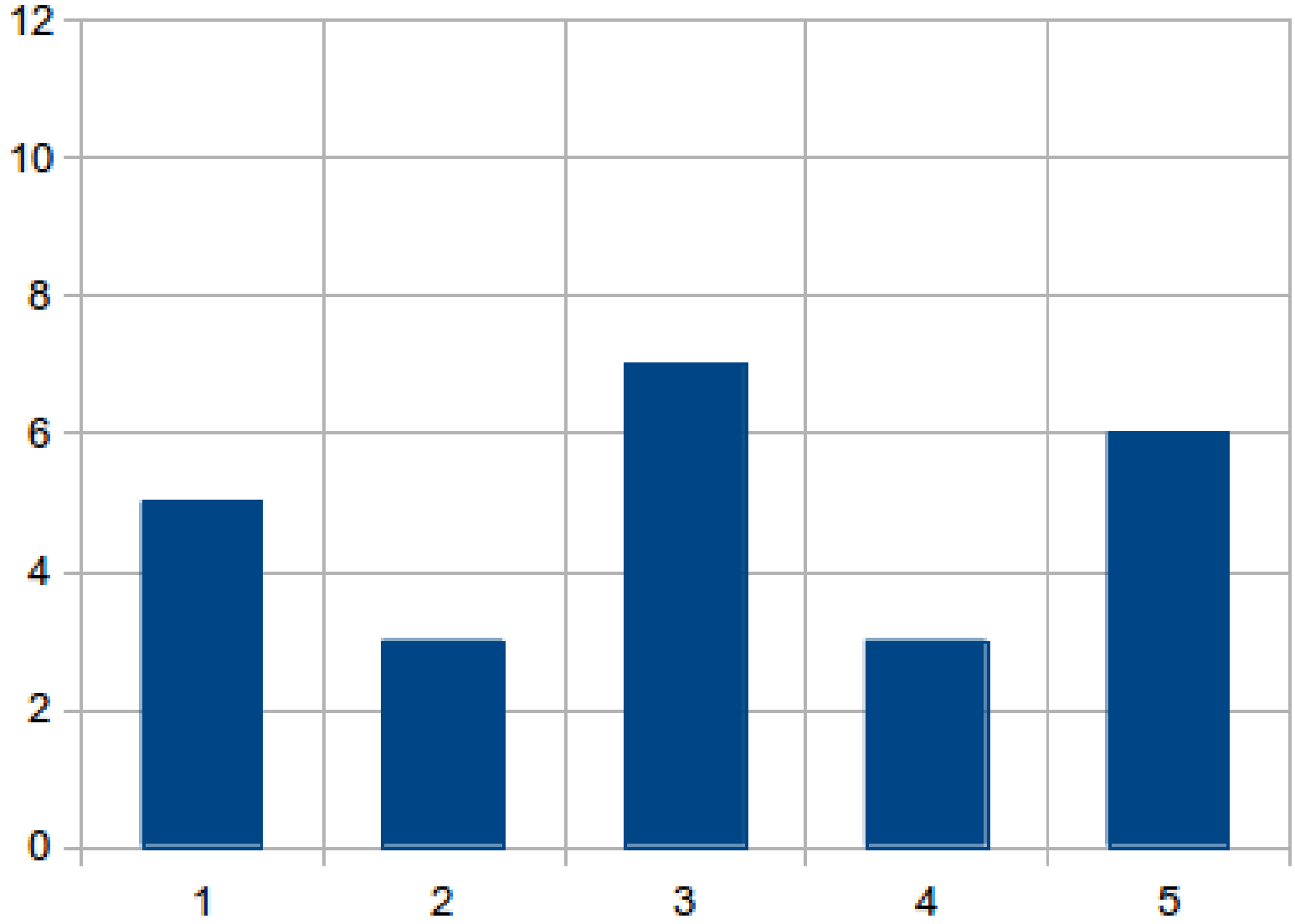
Lösung

- 3
- 1
- 3
- 5
- 2
- 3 $\rightarrow$ SPLIT(3), MERGE(1,2 $\rightarrow$ 1), new indices
- STOP

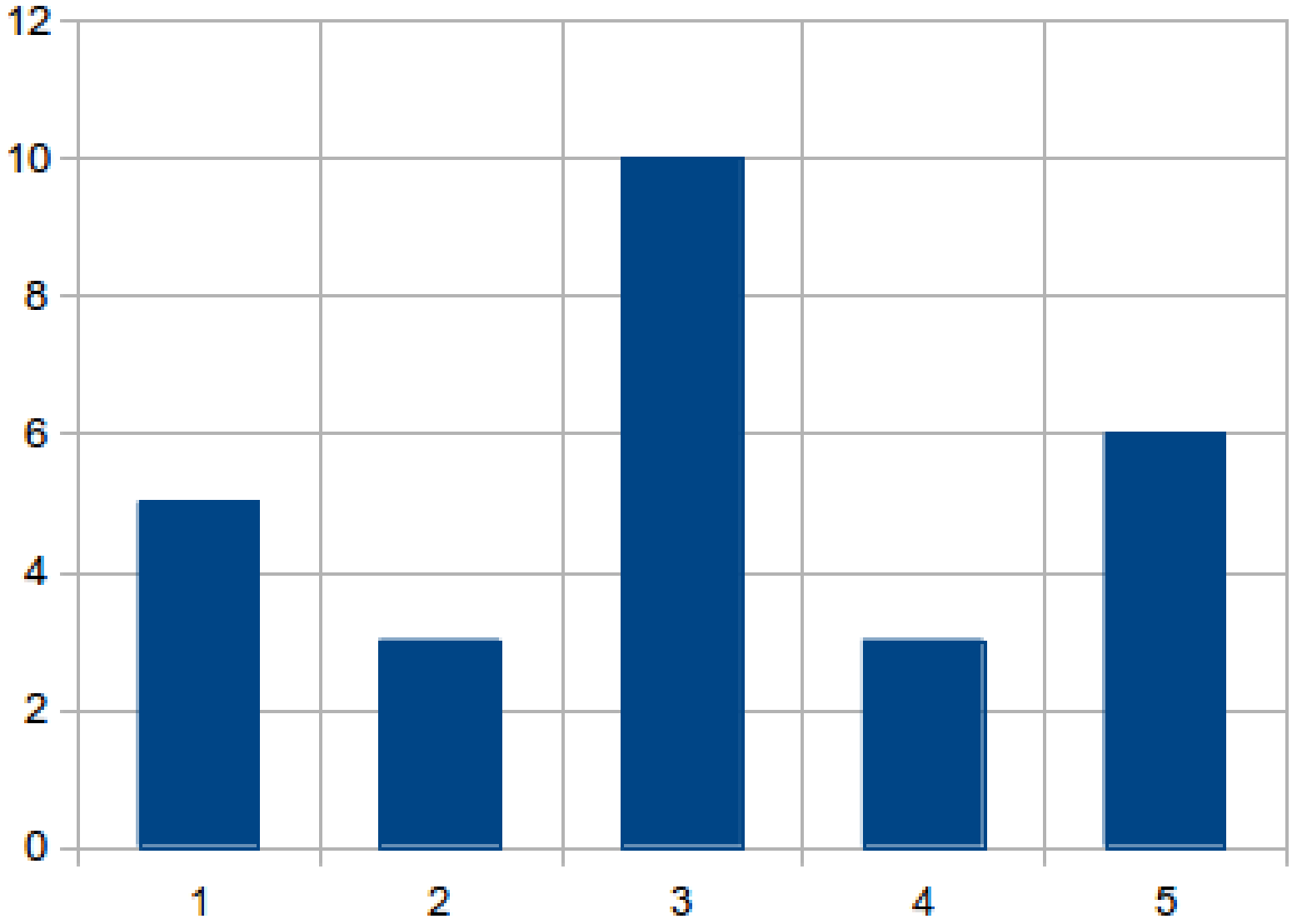
insert 3



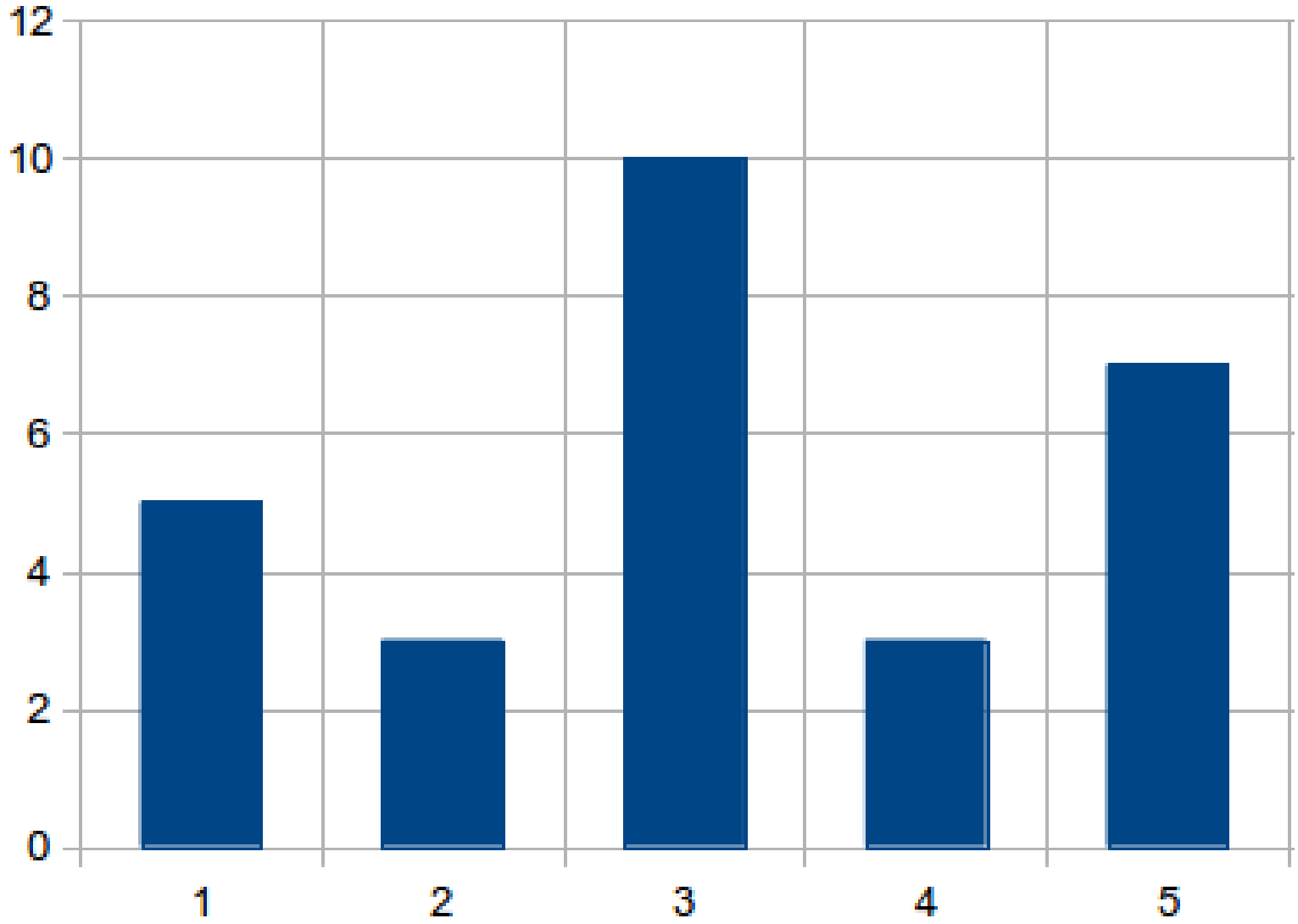
insert 1



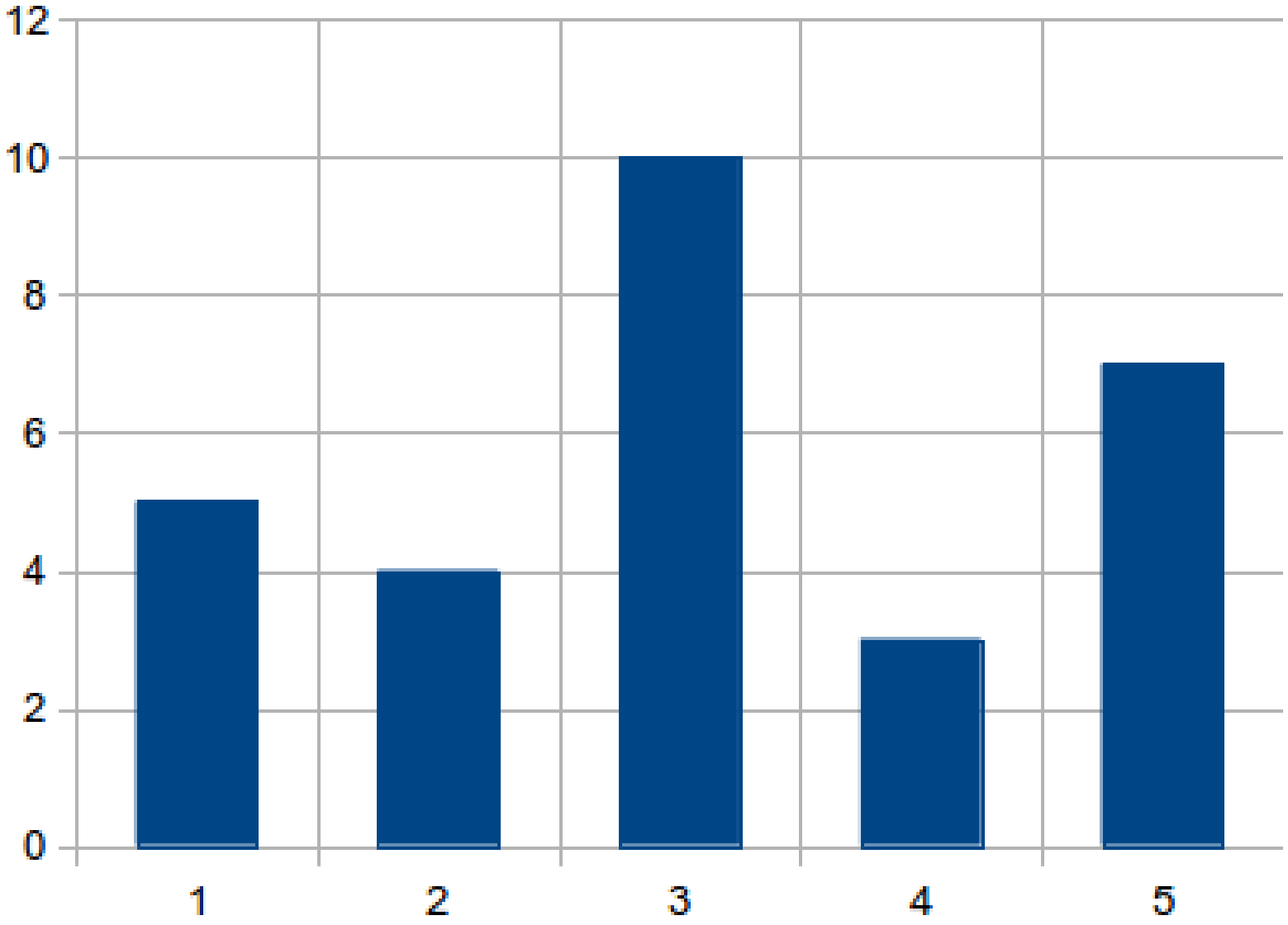
insert 3



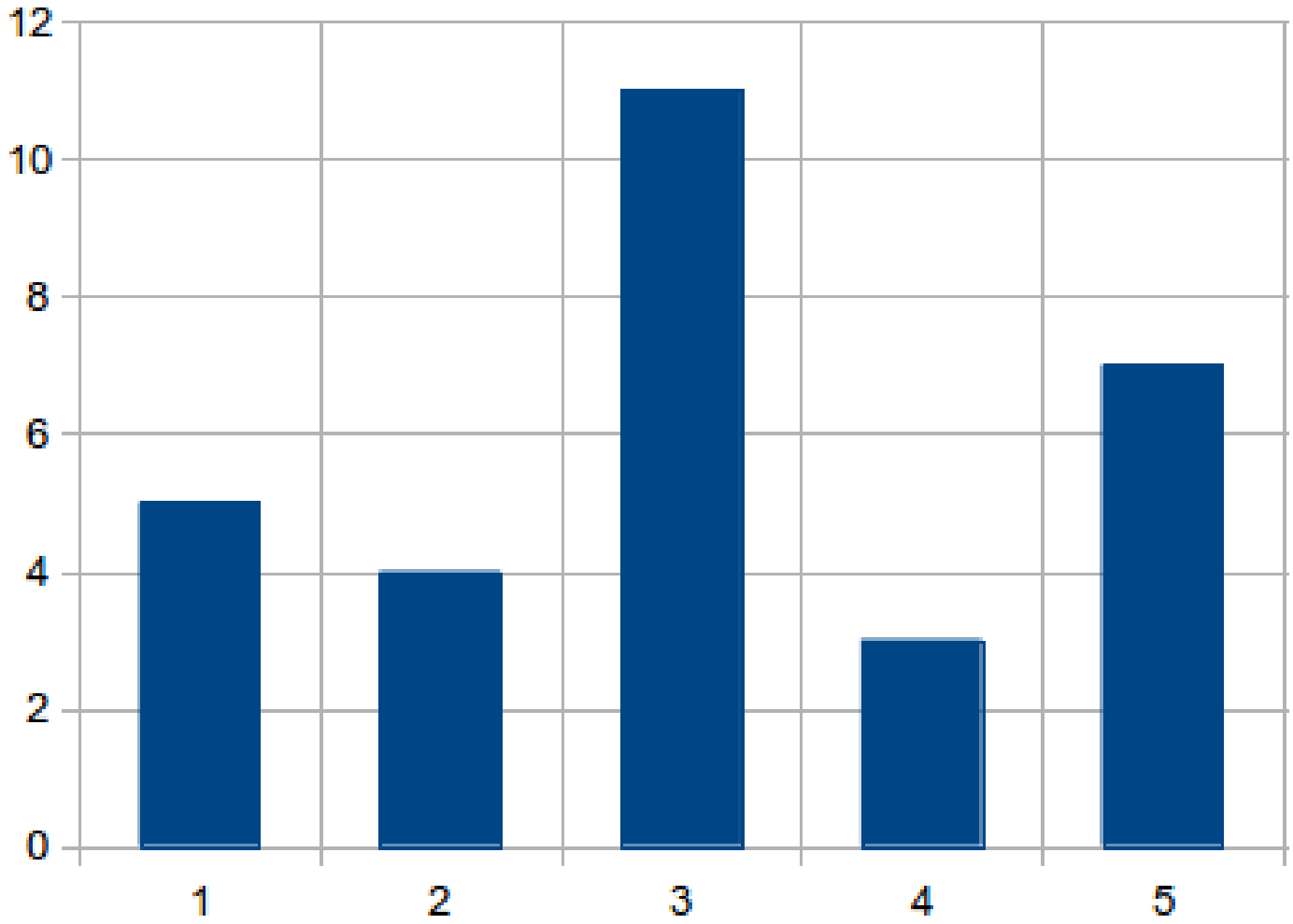
insert 5



insert 2

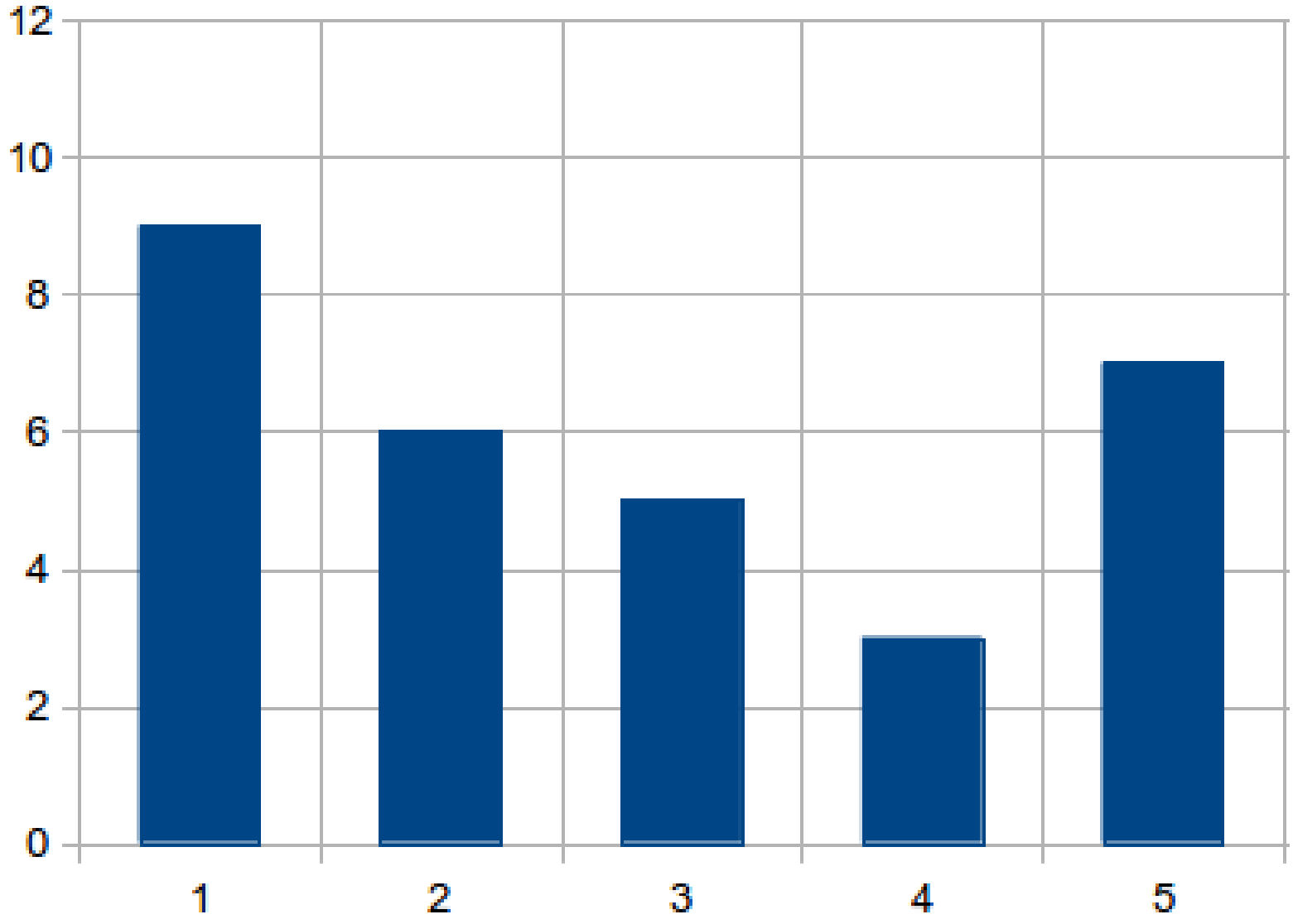


insert 3



Bucket Nummer 3 zu groß=> Split & Merge

Split & Merge



Merge & Split

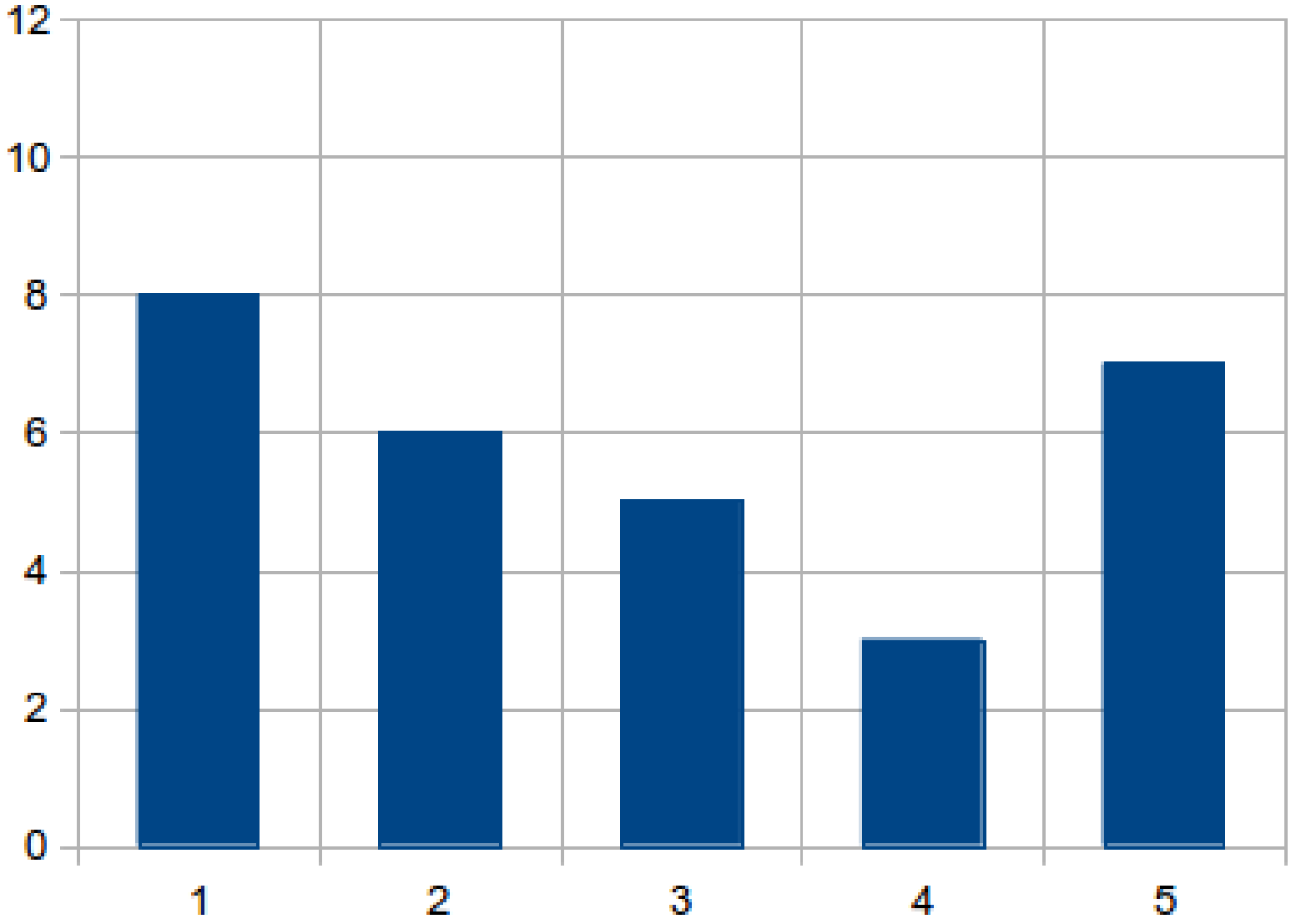
Nehmen Sie das zuletzt entstandene Histogramm als Basis um Elemente zu löschen, bis der erste Underflow auftritt. Führen Sie dann den merge & Split durch.

- Ein Merge betrifft den Bucket, in dem der Underflow auftritt, sowie dessen kleineren Nachbarn
- Die zu löschenden Items sind die Sequenz $S = (1, 3, 4, 5, 4, 3, 2, 5, 1, 2)$. Jedes Item ist nach dem Index seines Buckets benannt.

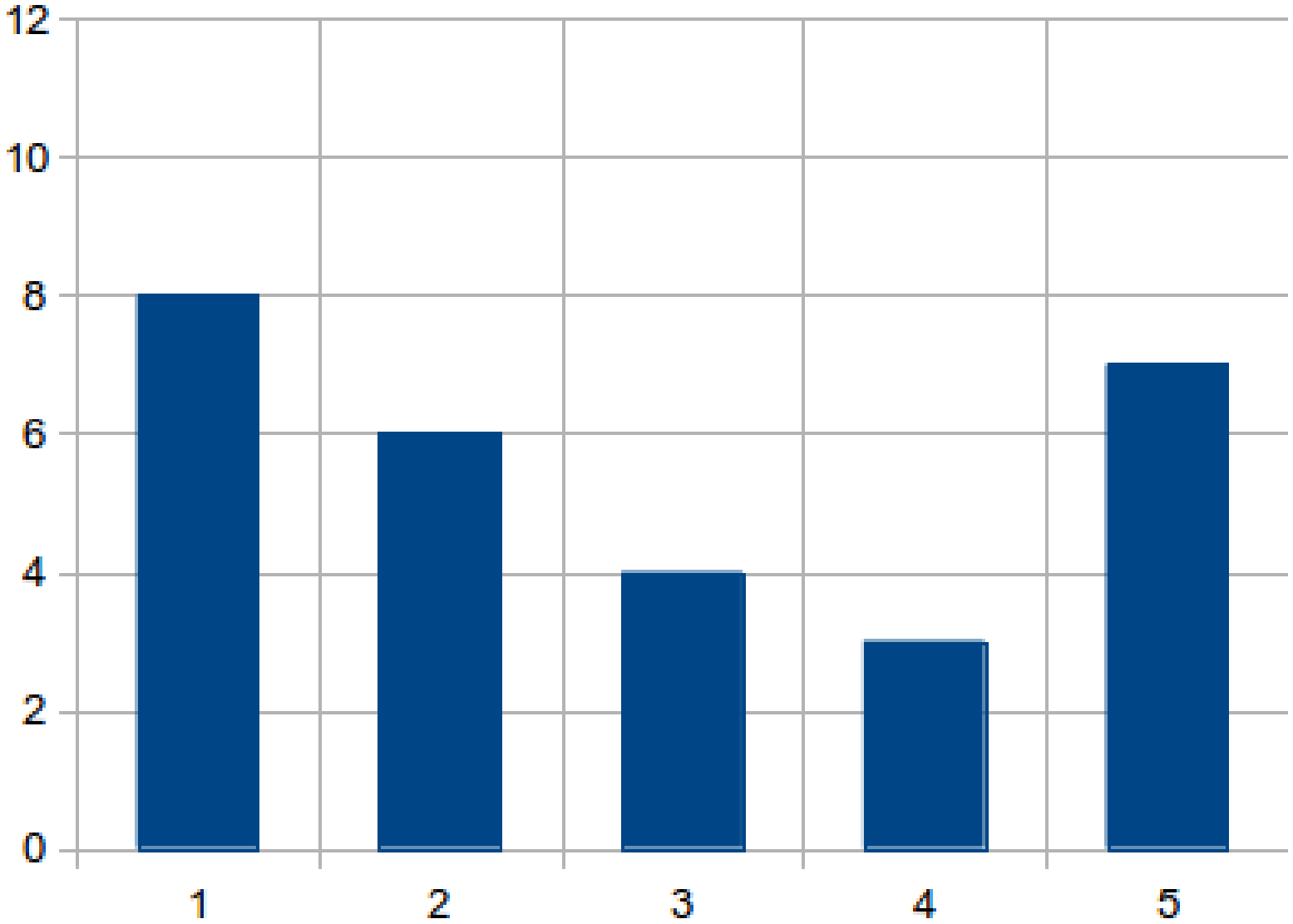
Lösung

- 1
- 3
- 4
- 5
- 4 $\rightarrow$ MERGE(3,4$\rightarrow$3), SPLIT(1), new indices
- STOP

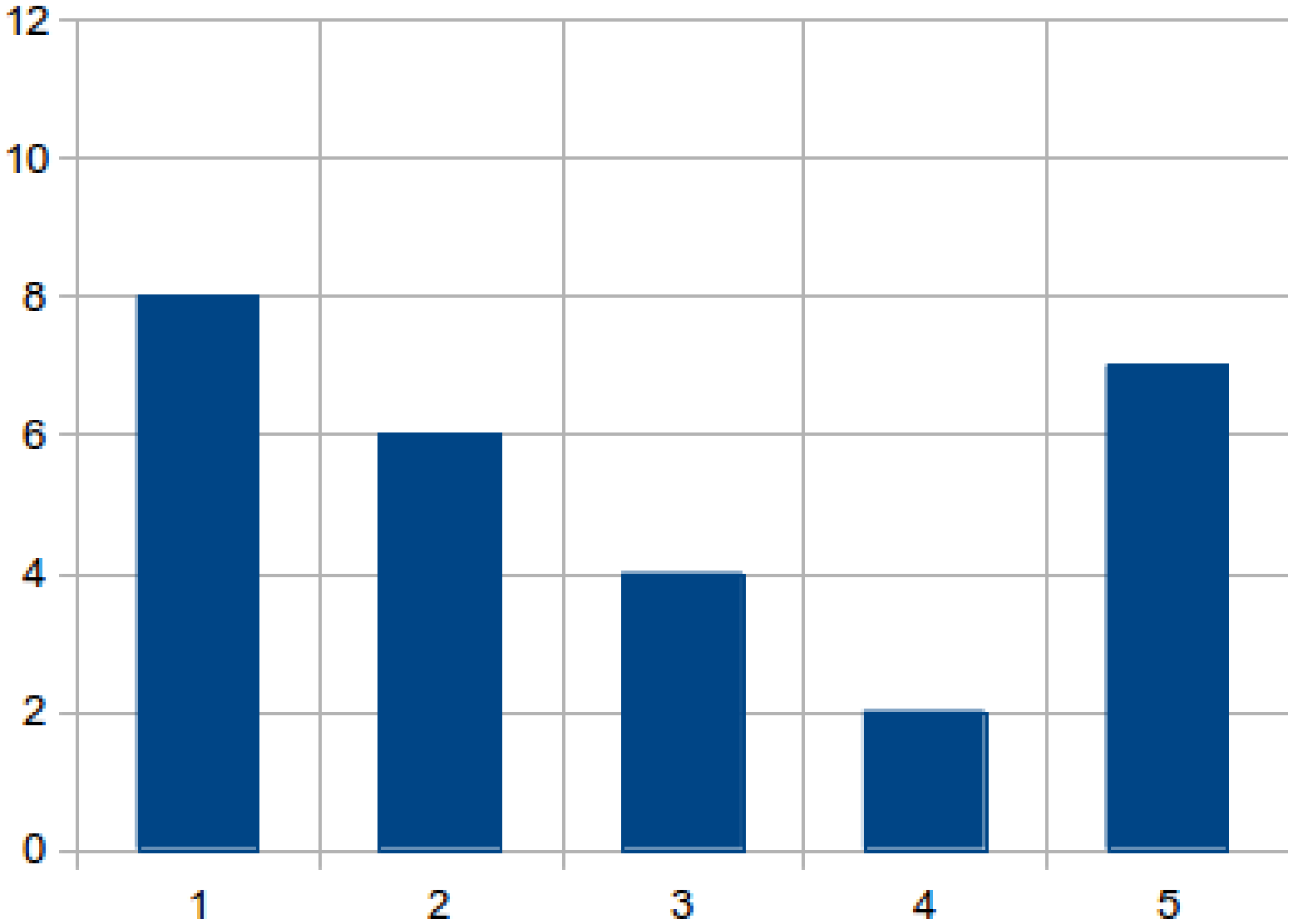
Delete 1



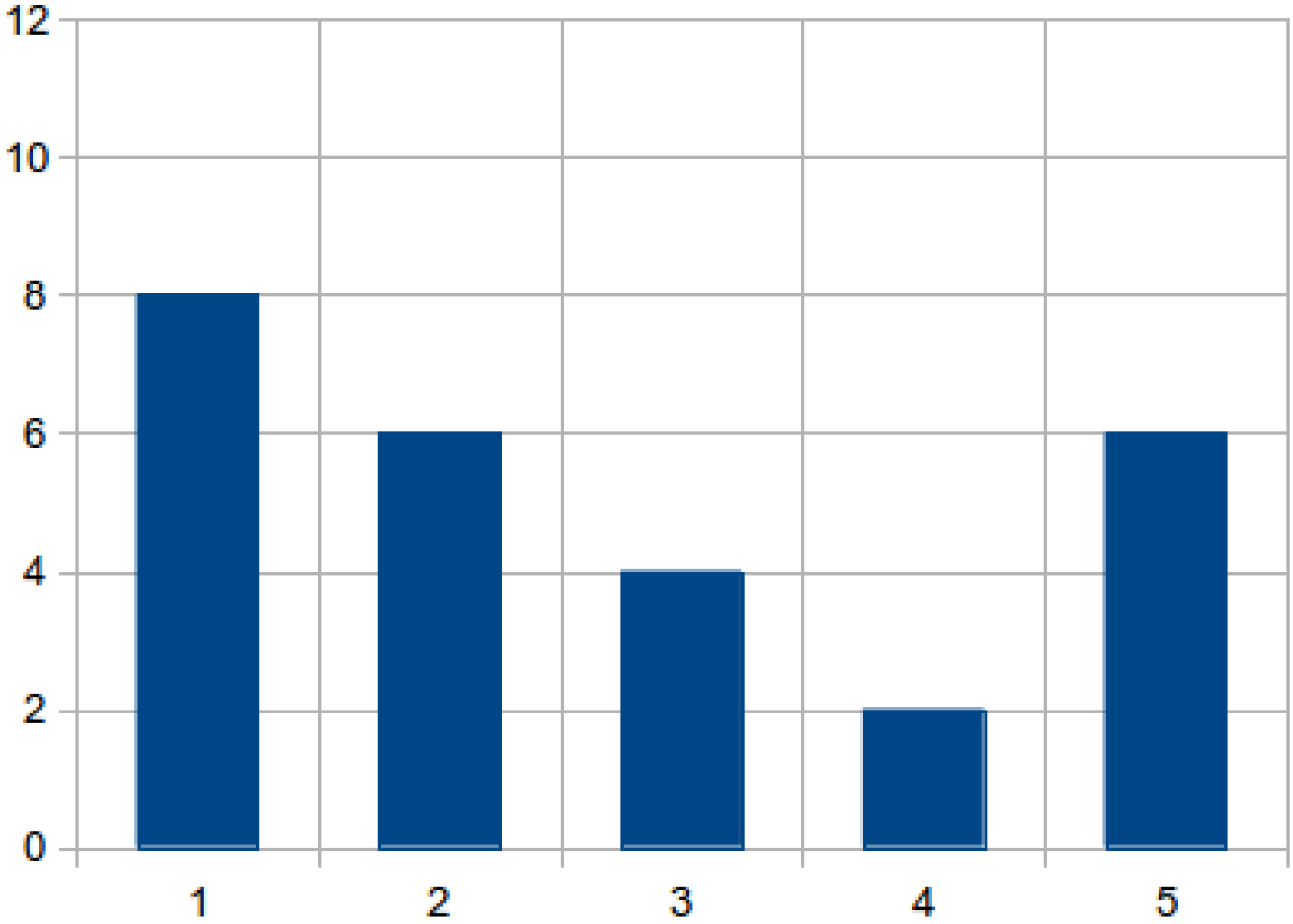
Delete 3



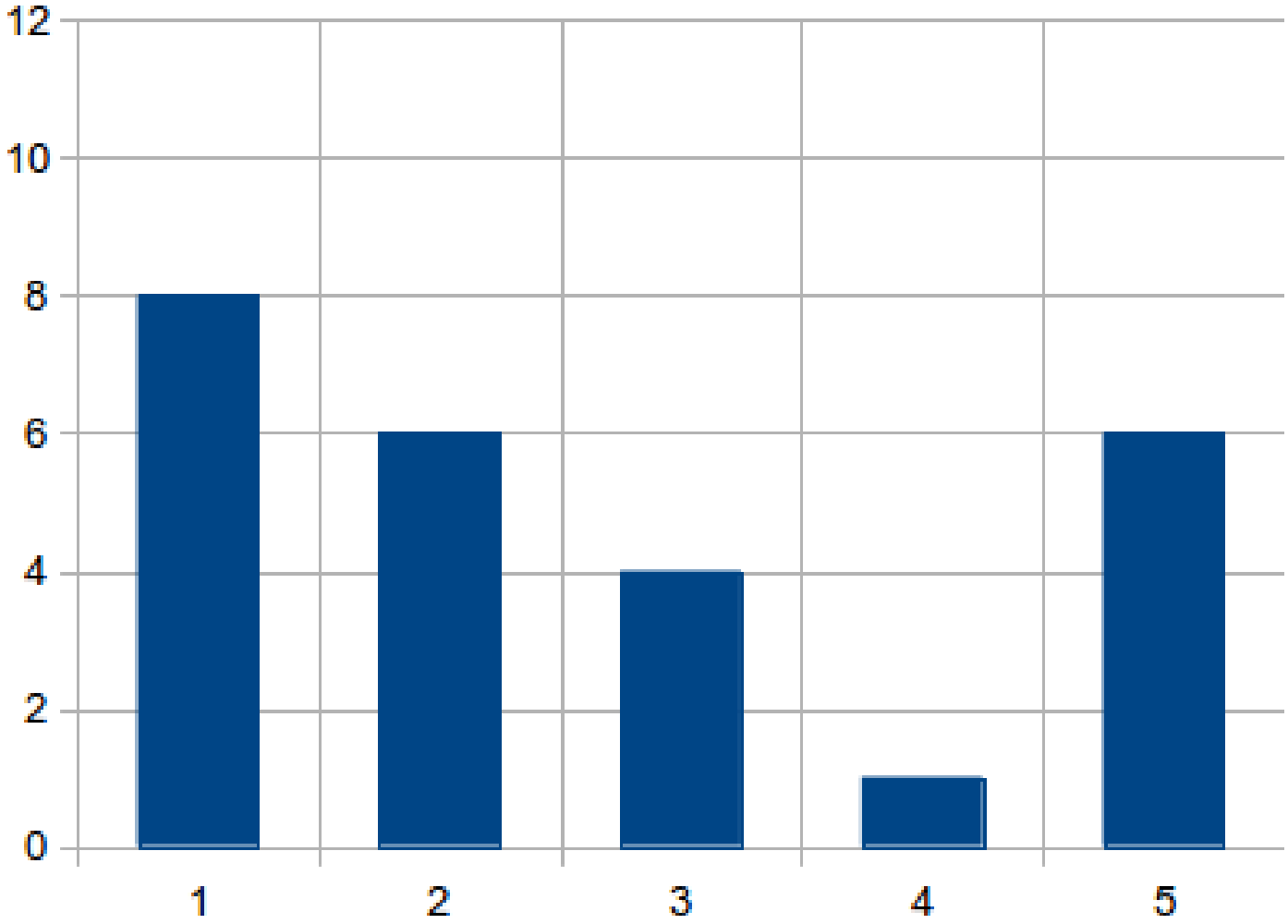
Delete 4



Delete 5

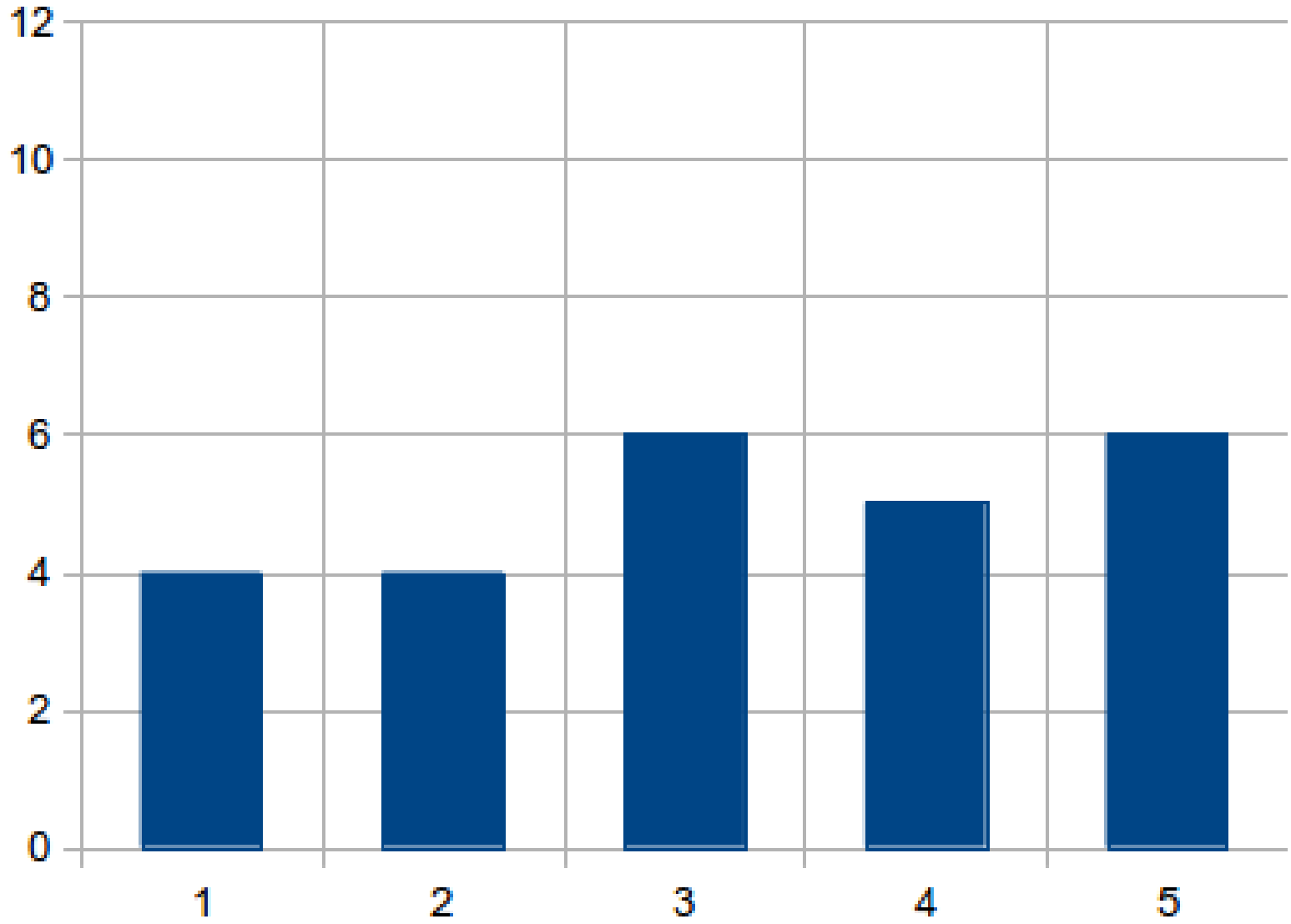


Delete 4



Bucket Nummer 4 zu klein => Merge & Split

Merge & Split



CUSUM - Change Detection

- Algorithmus um Veränderungen auf dem Datenstrom zu erkennen
- Beobachtet die kumulative Summe von Instanzen einer zufälligen Variable
- Erkennt eine Veränderung wenn der normalisierte Mittelwert der eingehenden Daten sich stark von 0 unterscheidet

CUSUM

Algorithm CUSUM
Input: data stream S, threshold param. α
begin
$G_0 := 0$
while S **do**
$x_t :=$ next instance of S
compute estimated mean ω_t
$G_t := \max(0, G_{t-1} - \omega_t + x_t)$
if $G_t > \alpha$ **then**
report change at time t
$G_t := 0$
end

Anmerkung: Dieser Algorithmus erkennt lediglich positive Changes. Um nur negative Changes zu erkennen benutzt man die Formel $G_t := -min(0, -(G_{t-1} - x_t + \omega_t)$. Im folgenden benutzen wir eine Formel, die Changes in beiden Richtungen erkennt, und zwar $G_t := (G_{t-1} - \omega_t + x_t)$

CUSUM

Gegeben sei ein Mittelwert von $\omega = 3$ und ein Grenzwert von $\alpha = 8$. Führen Sie den CUSUM-Algorithmus zur Change Detection mit der folgenden Sequenz durch:

$$s = 2, 3, 7, 4, 0, 2, 5, 6, 8, 7$$

t	$x_n - \omega$	G_n
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

CUSUM

t	$x_n - \omega$	G_n
0	-	0
1	-1	-1
2	0	-1
3	4	3
4	1	4
5	-3	1
6	-1	0
7	2	2
8	3	5
9	5	10
10	4	14

Change wird zwischen 8 und 9 erkannt

Lossy Counting Algorithm

Notation:

- Support threshold $s \in [0, 1]$
- Error threshold $\epsilon \in [0, 1]$
- $\epsilon \ll s$
- Stream S wird in Buckets der Größe $\omega = \lceil \frac{1}{\epsilon} \rceil$ aufgeteilt
- Die id des aktuellen Buckets ist $b_{curr} = \lceil \frac{N}{\omega} \rceil$
- Die bisher beobachtete tatsächliche Häufigkeit des Auftretens von einem Element e ist f_e
- Die Datenstruktur D ist eine Menge von Einträgen (e, f, Δ) mit e als Element, f als beobachtete Häufigkeit seit e in D ist und Δ als maximalen Fehler in f

LS-Algorithm

LossyCounting Algorithm (Manku et al., 2002)

Algorithm LossyCounting
Input: data stream S, error threshold ϵ
begin
  $D = \emptyset, N = 0, \omega = \left\lceil \frac{1}{\epsilon} \right\rceil$
  while S **do**
    $e_i :=$ next object from S
    $N \mathrel{+}= 1$
    $b_{curr} = \left\lceil \frac{N}{\omega} \right\rceil$
    if $e_i \in D$ **then**
      increment e_i's frequency by 1
    else
      $D.add\big((e_i, 1, b_{curr} - 1)\big)$
    whenever $D \equiv 0 \; mod \; \omega$ **do**
      foreach entry (e, f, Δ) in D **do**
        if $f + \Delta \leq b_{curr}$ **then**
          delete (e, f, Δ)
end

Algorithm LossyCounting – User request
Input: lookup table D, support threshold s
begin
  $S = \emptyset$
  foreach entry (e, f, Δ) in D **do**
    if $f \geq (s - \epsilon)N$ **then**
      add (e, f, Δ) to S
  return S
end

f is the exact frequency count of e since the entry was inserted into D

Δ is the maximum number of times e could have occurred in the first $b_{curr} - 1$ buckets

LS-Algorithm Beispiel:

- $s = 0.1$
- $\epsilon = 0.01 \Rightarrow \omega = \lceil \frac{1}{0.01} \rceil = 100$
- $S = (x, x, y, x, y, y...)$

N	e	b_{curr}	D
0	-	-	$\emptyset$
1	x	$\lceil \frac{1}{100} \rceil = \lceil 0.01 \rceil = 1$	$\{(x, 1, 0)\}$
2	x	$\lceil \frac{2}{100} \rceil = \lceil 0.02 \rceil = 1$	$\{(x, 2, 0)\}$
3	y	$\lceil \frac{3}{100} \rceil = \lceil 0.03 \rceil = 1$	$\{(x, 2, 0), (y, 1, 0)\}$
4	x	$\lceil \frac{4}{100} \rceil = \lceil 0.04 \rceil = 1$	$\{(x, 3, 0), (y, 1, 0)\}$
5	y	$\lceil \frac{5}{100} \rceil = \lceil 0.05 \rceil = 1$	$\{(x, 3, 0), (y, 2, 0\}$
6	y	$\lceil \frac{6}{100} \rceil = \lceil 0.06 \rceil = 1$	$\{(x, 3, 0), (y, 3, 0)\}$

Beweis

Beweisen Sie folgende Aussage mittels Induktion und benutzen Sie die Notation aus den Folien:

Immer wenn ein Eintrag (e, f, Δ) gelöscht wird, ist die genaue Häufigkeit $f_e \leq b_{curr}$

Induktionsanfang:

$b_{curr} = 1$

Eintrag wird nur gelöscht, wenn gilt $f = 1$, und das ist auch gleichzeitig f_e von e

$=> f_e \leq b_{curr}$

Beweis

Zu zeigen:

Immer wenn ein Eintrag (e, f, Δ) gelöscht wird, ist die genaue Häufigkeit $f_e \leq b_{curr}$

Induktionsschritt:
Angenommen, es existiert ein Eintrag (e, f, Δ) das beim Bucket $b_{curr} > 1$ gelöscht wird. Dieser Eintrag wurde eingefügt, als der Bucket $\Delta + 1$ verarbeitet wurde. Ein Eintrag für e konnte spätestens gelöscht werden, wenn der Bucket mit der id Δ voll wurde. Nach Induktion, konnte die tatsächliche Häufigkeit von e, in den Buckets 1 bis Δ während der Löschung nicht größer als Δ sein. Außerdem ist f die echte Frequenz von e, seit e eingefügt wurde. $=> f_e$ in den Buckets 1 bis $b_{current}$ ist maximal $f + \Delta$. Zusammen mit der Regel zur Löschung, dass $f + \Delta \leq b_{current}$ ist, ergibt sich $f_e \leq b_{current}$

Exponential Histograms

- Hat einen Datenstrom, der nur aus Nullen und Einsern besteht
- Zählt die Vorkommen von Einsern in einem Sliding Window der Größe N
- Jedes Bucket hat Größe und Zeitstempel
- Hat außerdem die Variablen LAST und TOTAL um die Anzahl der Elemente in dem Sliding Window zu erkennen

Exponential Histograms

Algorithm Exponential Histogram Maintenance
Input: data stream S, window size N, error param. ϵ
begin
  $TOTAL := 0$
  $LAST := 0$
  while S **do**
    $x_i := S.next$
    if $x_i == 1$ **do**
      create new bucket b_i with timestamp t_i
      $TOTAL \mathrel{+}= 1$
      while $t_l < t_i - N.length$ **do**
        $TOTAL \mathrel{-}= b_l.size$
        drop the oldest bucket b_l
        $b_l := b_{l-1}$
        $LAST := b_l.size$
      while *exist* $\lfloor 1/\epsilon \rfloor/2 + 2$ *buckets of the same size* **do**
        merge the two oldest buckets of the same size with the largest timestamp of both buckets
        if *last bucket was merged* **do**
          $LAST :=$ size of the new created last bucket
end

Exponential Histograms

Konstruieren Sie ein Exponential Histogram mit der Window Size N=8 und den Error Parameter $\epsilon = \frac{1}{2}$ für die gegebene Sequenz

$$\text{Sequence } s = 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1$$

Exponential Histograms

Zeitstempel	Buckets
1	1_1
2	$1_1\ 1_2$
4	$2_2\ 1_4$
7	$2_2\ 1_4\ 1_7$
8	$2_2\ 2_7\ 1_8$
9	$2_2\ 2_7\ 1_8\ 1_9$
10	$4_7\ 2_9\ 1_{10}$
12	$4_7\ 2_9\ 1_{10}\ 1_{12}$
13	$4_7\ 2_9\ 2_{12}\ 1_{13}$
15	$4_7\ 2_9\ 2_{12}\ 1_{13}\ 1_{15}$
16	$4_{12}\ 2_{15}\ 1_{16}$