Stefan Fischer
Benjamin Neidhardt
Merle Kammer

1	2	3	4	Σ

Übungsblatt Nr. 3

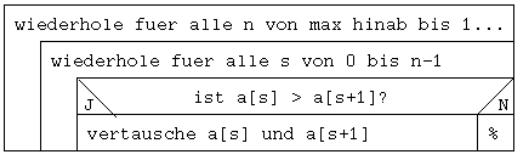
(Abgabetermin 11.05.2017)

Aufgabe 1

a)

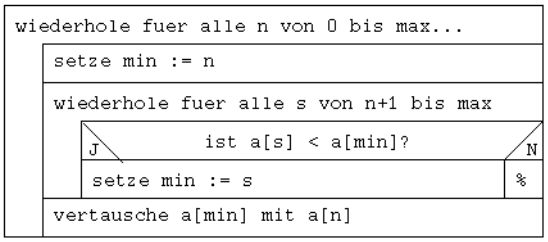
Bubblesort:
Beim Durchlaufen des Arrays werden die jeweiligen Nachbarelemente miteinander verglichen und ggf. ausgetauscht. Am Ende des ersten Laufs befindet sich das größte Element am Feldende. Dieser Vorgang wird nun erneut gestartet, allerdings diesmal ohne das letzte Element. Danach hat man das zweitgrößte Element gefunden. Der ganze Vorgang wird nun so oft wiederholt, bis das gesamte Array sortiert ist.

Pseudocode



Beschreibung Minimumsort:
Laufe ab Index 1 über die gesamte Folge und merke Dir den kleinsten Wert. Vertausche diesen Wert mit dem, der in der Folge an Stelle 2 steht. Somit erhältst du eine Folge, die bis Index 1 sortiert ist. Dann laufe ab Index 2 über diese Folge und merke Dir wieder den kleinsten Wert. Diesen vertauschst Du jetzt mit dem an Index 2 stehenden Wert. Die so entstandene neue Folge ist nun bereits bis Index 2 sortiert. Wiederhole diesen Vorgang bis Index (n-1). Nach diesen (n-1) Durchla?ufen erha?ltst Du eine vollsta?ndig sortierte Folge, und der Algorithmus ist beendet.

Pseudocode



b)

Bubblesort vorsortierte Liste:
Bei einer bereits sortierten Liste wird Bubblesort die Liste nur einmal durchgehen, um festzustellen, dass die Liste bereits sortiert ist, weil keine benachbarten Elemente vertauscht werden mussten. Daher benötigt Bubblesort $0(n)$ Schritte, um eine bereits sortierte Liste zu bearbeiten.
Insgesamt werden also (n - 1) Vergleiche und keine Vertauschungen vorgenommen.

Minimum Sort vorsortiere Liste:
Bei einer vorsortieren Liste muss MinSort mindestens einmal alle Zahlen vergleichen. .Daher benötigt auch MinSort $O(n)$ Schritte, um eine bereits sortierte Liste zu bearbeiten.
Insgesamt werden also (n - 1) Vergleiche und keine Vertauschungen vorgenommen.

c + d)

Bubblesort:
Bei absteigender Sortiertung einer Liste wird in jedem Durchlauf der inneren Schleife das erste Element bis zum n - 1 Element durchgetauscht (also n -1 viele Vertausch-Operationen) und der Wert von n um 1 reduziert. Da dies die maximale Anzahl an Vertausch-Operationen pro Iteration und die maximale Anzahl an Iterationen liefert, ist dies der Worst-Case mit der Laufzeit $\Theta(n^2)$.

Insgesamt führt Bubble Sort also höchstens:

$\sum_{i=1}^{n-1}(n-i) = \sum_{j=1}^{n-1} j = \frac{n \cdot (n-1)}{2}$

Vergleiche und Vertauschungen durch.

Beispiel Liste absteigend sortiert:

5 4 3 2 1

4 5 3 2 1
4 3 5 2 1
4 3 2 5 1
4 3 2 1 5
3 4 2 1 5
3 2 4 1 5
3 2 1 4 5
2 3 1 4 5
2 1 3 4 5
2 1 3 4 5

1 2 3 4 5

Berechnung der max Vergleiche:
$\frac{5 \cdot (5-1)}{2}$ =10 Vergleiche

Berechnung der max Vertauschungen:
$\frac{5 \cdot (5-1)}{2}$ = 10 Vertauschungen

Minimum Sort:
Beim Minimum Sort spielt der Inhalt des Eingabe-Arrays keine Rolle für die Anzahl an Vertausch-Operationen, da diese auf jeden Fall genau einmal in jeder Iteration der äußeren Schleife ausgefuührt werden und die Anzahl der Iterationen nur von der La?nge des Eingabe- Arrays abhängt.
In der i-ten Phase benötigt MinSort n- i Vergleiche, aber nur eine Vertauschung.

Insgesamt führt MinSort also höchstens:

$(n-1)$

Vergleiche und Vertauschungen durch.

Beispiel Liste absteigend sortiert:
5 4 3 2 1

1 4 3 2 5
1 2 3 4 5

1 2 3 4 5

Berechnung der max Vergleiche:
$(n-1) = 5-1 = 4$ Vergleiche

Berechnung der max Vertauschungen:
$(n-1) = 5-1 = 4$ Vertauschungen

d)

Aufgabe 2

Aufgabe 3

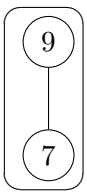
a)

$A = \{9, 7, 21, 14, 88, 23, 10, 26, 13\}$

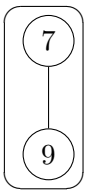
Bilden des Heaps:

Einfügen von 9

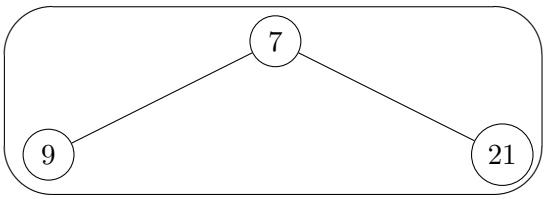
Einfügen von 7



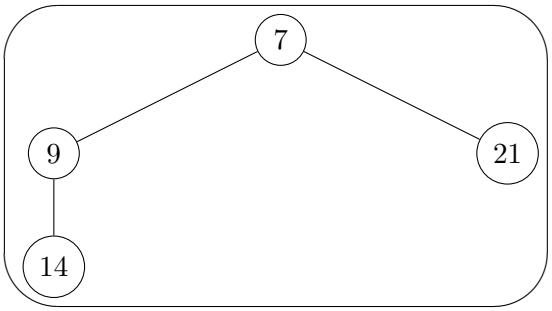
Wiederherstellen des Heap mit Beachtung des Kriteriums



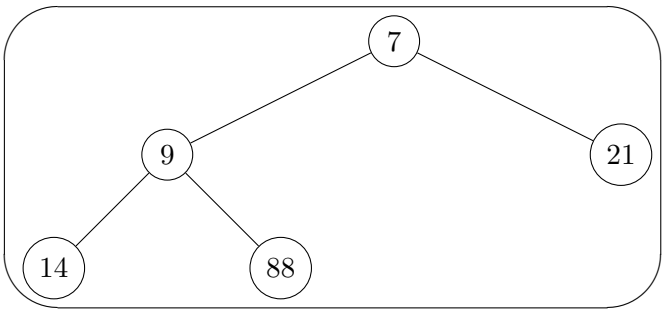
Einfügen von 21



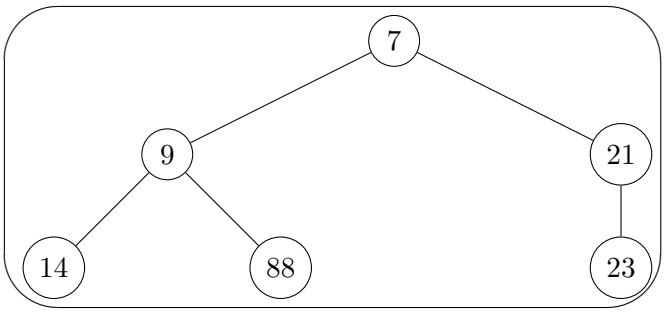
Einfügen von 14



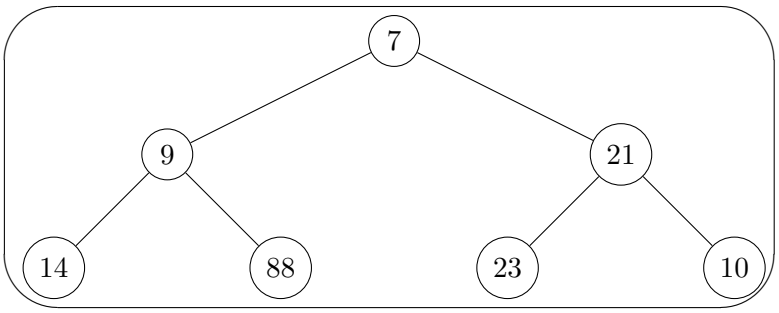
Einfügen von 88



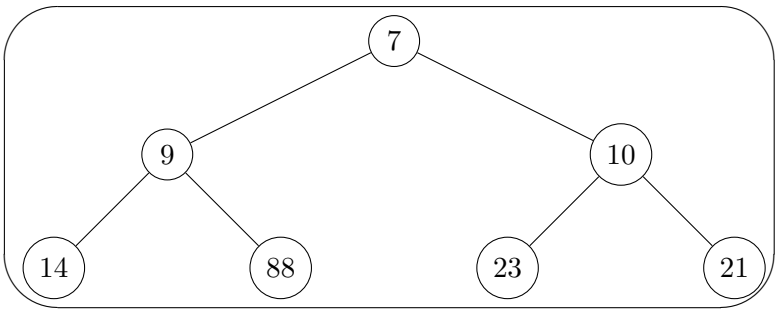
Einfügen von 23



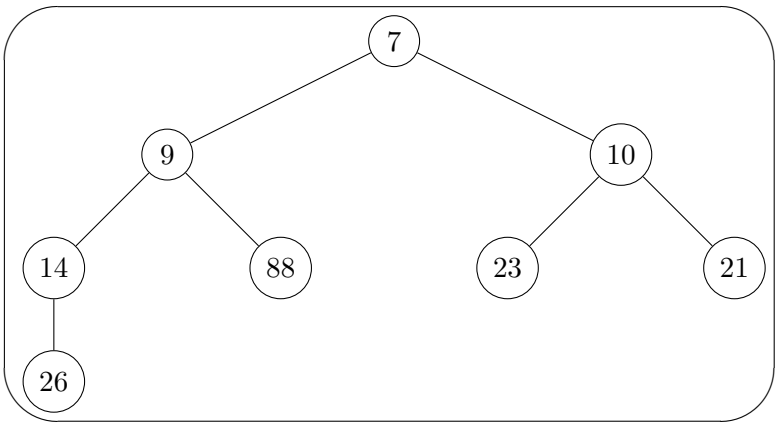
Einfügen von 10



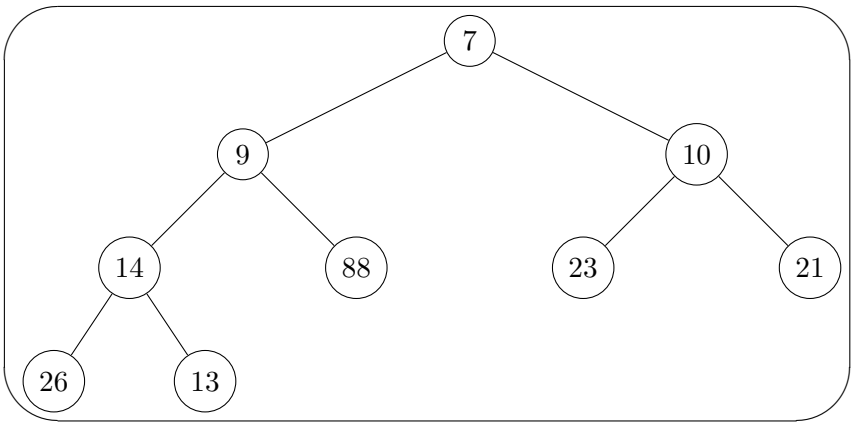
Wiederherstellen des Heap mit Beachtung des Kriteriums



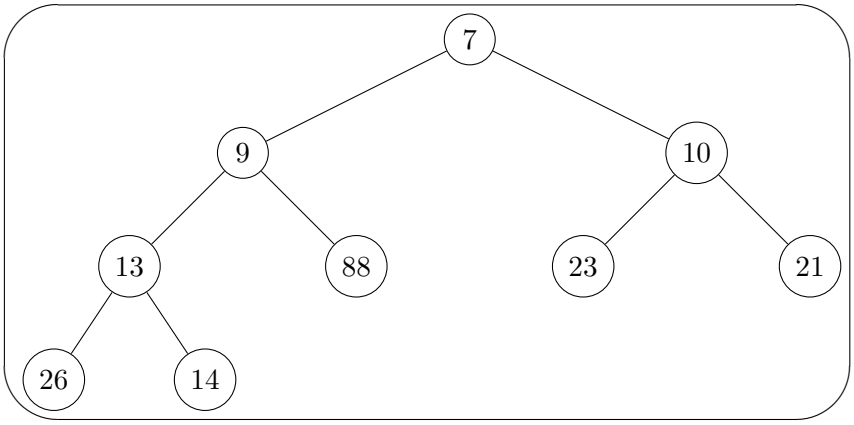
Einfügen von 26



Einfügen von 13

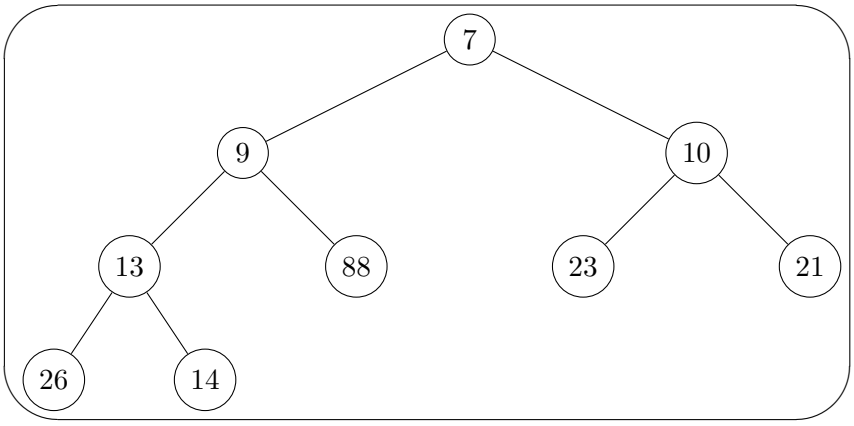


Wiederherstellen des Heap mit Beachtung des Kriteriums

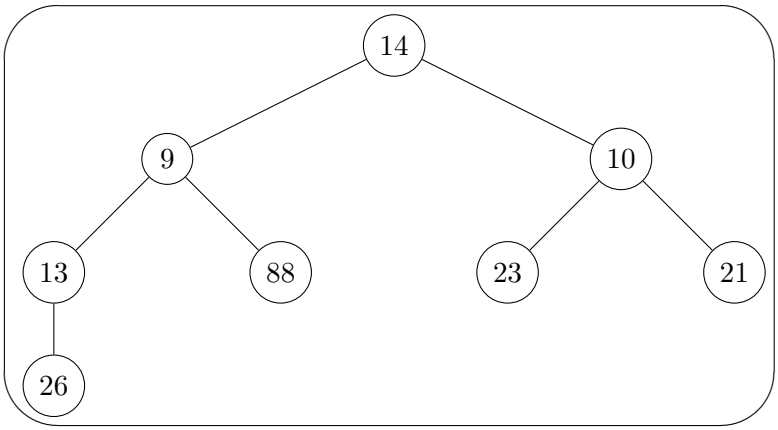


b)

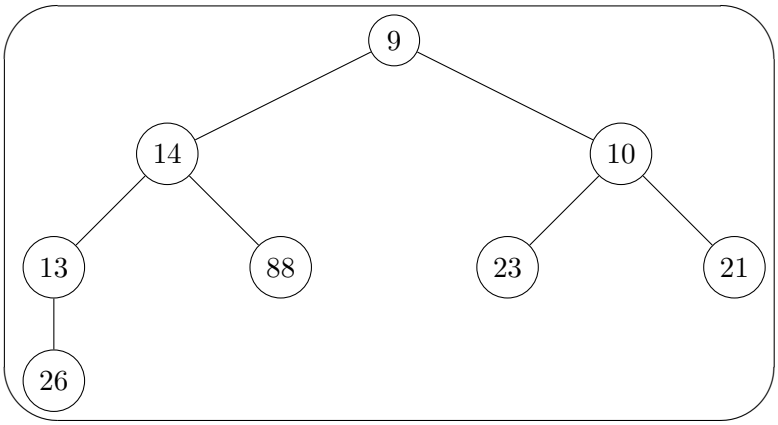
EXTRACTMIN Operation ausführen und Heap-Eigenschaft wiederherstellen:



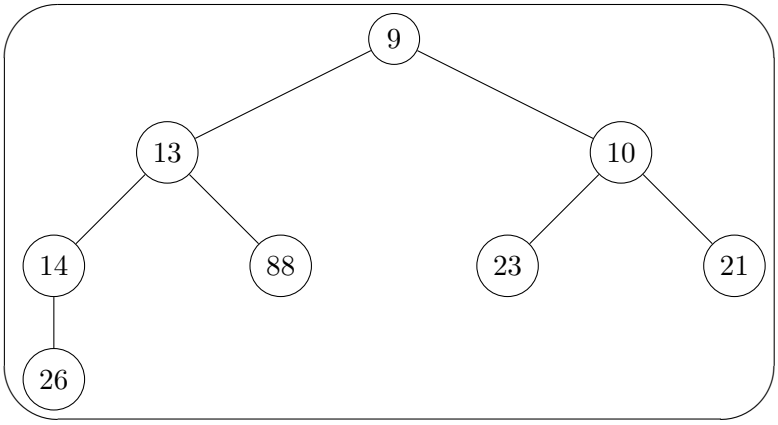
Entnehme das Minimum d.h. die Wurzel des Baumes und füge 14 als neue Wurzel hinzu:



Vergleiche 14 mit beiden Kindelementen und tausche mit kleinerem, also der 9:



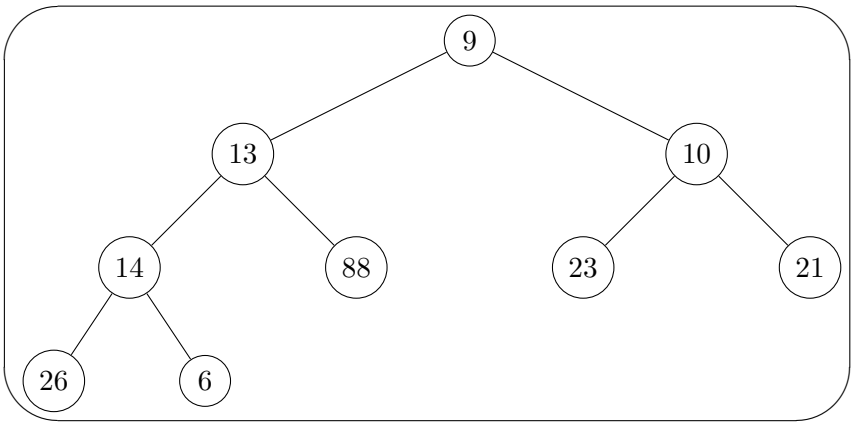
Vergleiche 14 wieder mit beiden Kindelementen, falls kleiner, tausche mit dem kleinsten, in dem Fall der 13:



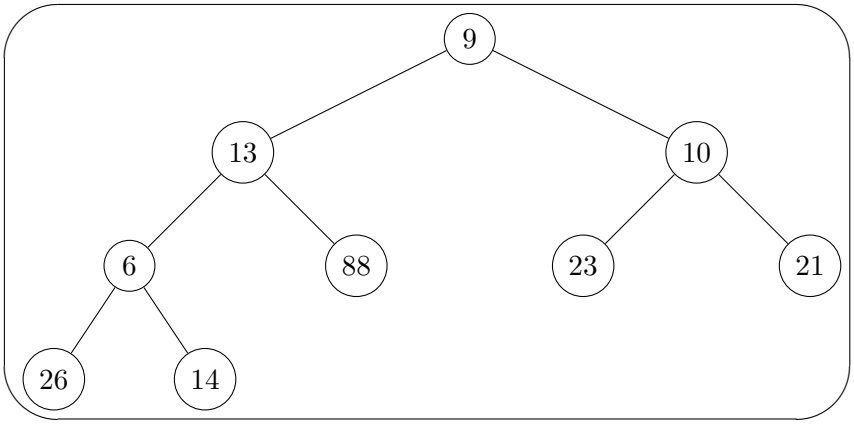
Dies ist der finale Heap, da die Heap-Eigenschaft nun komplett wiederhergestellt ist.

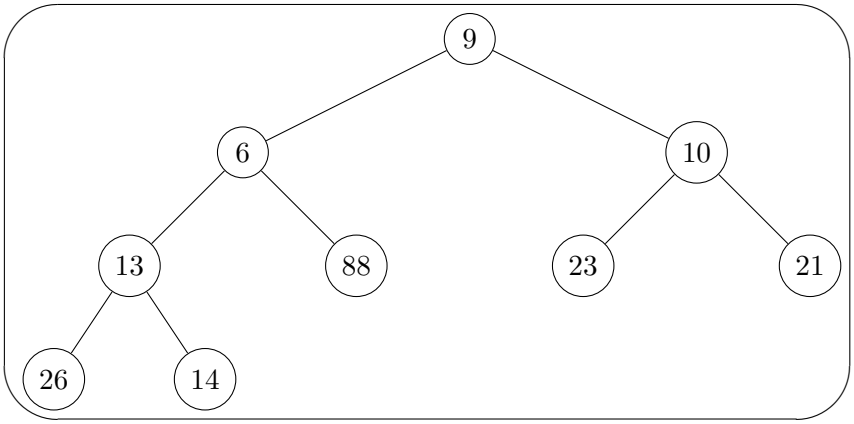
c)

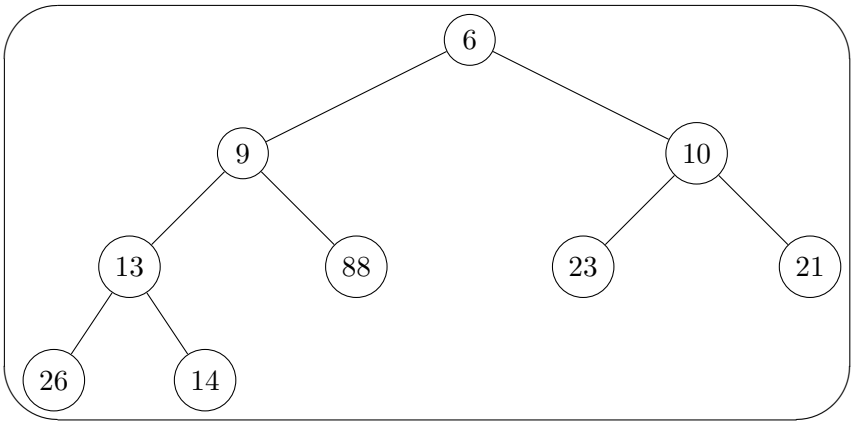
Neues Element 6 zu Heap hinzufügen:



Heap-Eigenschaft nach und nach wiederherstellen:







d)

Schritte in O-Notation um maximales Element aus dem Heap zu löschen?
→ man vergleicht alle Blätter, aber nicht die Elternknoten, da diese nach der Heap-Eigenschaft nicht das größte Element sein können

Schritt 1: Das Maximum finden → $[\frac{n}{2}]$ mit n Anzahl der Elemente d.h. O($[\frac{n}{2}]$)

Schritt 2: Maximum an unterstes Blatt,, welches am weitesten rechts steht → O(1)

Schritt 3: Heap-Eigenschaft wiederherstellen: max O($\log n$), da Höhe von Baum ausgewogen

Schritt 4: Maximum aus Baum entfernen (Heap-Eigenschaft bleibt durch Schritt 1 immer erhalten)

→ $[\frac{n}{2}] + 1 + \log n + 1 = [\frac{n}{2}] + \log n + 2 \rightarrow$ O(n)

Aufgabe 4

$\Omega = \{1, 2, 3, 4, 5, 6\}$

a)

(a) Die Wahrscheinlichkeit für das Ereignis $A = \{2\}$ ist: $P(A) = \frac{|A|}{|\Omega|} = \frac{1}{6}$

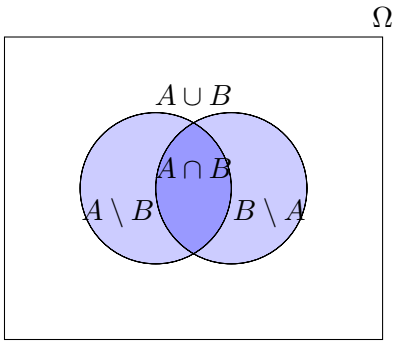
(b) Die Wahrscheinlichkeit für das Ereignis $A = \{2, 4, 6\}$ ist: $P(A) = \frac{|A|}{|\Omega|} = \frac{3}{6}$

b)

Zu zeige: Falls $A \cap B = \emptyset$, dann gilt $P(A \cap B) = P(A) + P(B)$
(1)

$$
\begin{aligned}
P(A \cup B) &\overset{(*)}{=} P((A \setminus B) \dot\cup (A \cap B) \dot\cup (B \setminus A)) \\
&= P(A \setminus B) + P(A \cap B) + P(B \setminus A) \quad \leftarrow \sigma\text{-additivität} \\
&= \underbrace{P(A \setminus B) + P(A \cap B)}_{=P(A)} + \underbrace{P(B \setminus A) + P(A \cap B)}_{=P(B)} - P(A \cap B) \\
&= P(A) + P(B) + P(A \cap B) \\
&\text{für } A \cap B = \emptyset \text{ gilt somit} \\
&= P(A) + P(B) \qquad \square
\end{aligned}
$$

(*):
$A \cup B = (A \setminus B) \dot\cup (A \cap B) \dot\cup (B \setminus A)$



Für $A \cap B \neq \emptyset$ gilt $P(A \cup B) \leq P(A) + P(B)$ denn:
$P(A \cup B) = P(A) + P(B) - P(A \cap B) \leftarrow$ siehe Beweis (1)
$\Rightarrow P(A \cup B) \leq P(A) + P(B)$

c)

$\Omega = \{1, 2, 3, 4, 5, 6\}^3$
Das Ereignis, dass Älle drei Würfel ein Auge zeigen ist $A = \{(1, 1, 1)\}$. Die Wahrscheinlichkeit für dieses Ereignis ist: $P(A) = P(\{(1, 1, 1)\}) = \frac{|\{(1,1,1)\}|}{|\Omega|} = \frac{1}{6^3} = \frac{1}{216}$

d)

$\Omega = \{1, 2, 3, 4, 5, 6\}^2$

(a)

$$\begin{aligned}[X = 4] &= \{(x, y) \in \Omega \mid X(x, y) = 4\} \\ &= \{(4, 4), (4, 4), (4, 5), (5, 4), (4, 6), (6, 4)\}\end{aligned}$$

Es werden zwei Würfel geworfen. Das Ereignis $[X = 4]$ tritt ein, wenn einer der Würfel eine 4 zeigt und die Augenzahl des anderen Würfels ≥ 4 ist. Das heißt das Minimum der gewürfelten Augenzahlen muss 4 sein, damit das Ereignis $[X = 4]$ eintrifft.

(b) $P([X = 4]) = P(\{(4, 4), (4, 4), (4, 5), (5, 4), (4, 6), (6, 4)\}) = \frac{|\{(4,4),(4,4),(4,5),(5,4),(4,6),(6,4)\}|}{|\Omega|} = \frac{6}{6^2} = \frac{6}{36} = \frac{1}{6}$
Unter Annahme der Gleichverteilung der Ereignisse ist $P([X = 4]) = \frac{1}{6}$.