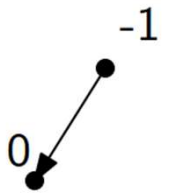
Algo-Tutorium 11

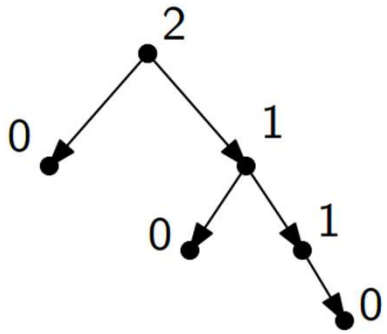
Binäre Suchbäume – Basics

- binärer Baum ($\forall v \in V: \#v.child \leq 2$)
- Ordnung zwischen den Knoten $v \in V$:
  - $key(v) \geq key(x)\ \forall x \in T_L$ (linker Teilbaum unter v)
  - $key(v) < key(x)\ \forall x \in T_R$ (rechter Teilbaum unter v)
- Höhe: zwischen $O(\log(n))$ und $O(n)$ (kann „ungünstig“ sein)

AVL-Bäume

- binärer balancierter Suchbaum



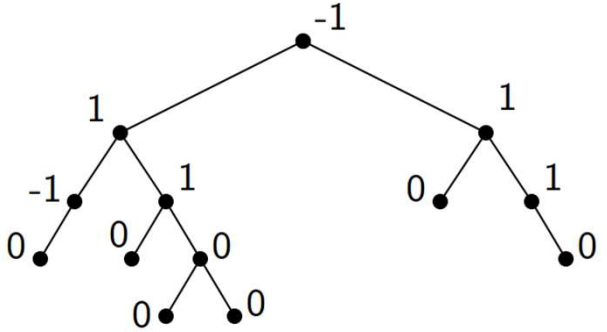


- $Balance(v) = h(T_R) - h(T_L)$

! Balance immer „von unten" berechnen

! Höhe leerer Teilbaum = -1

- im AVL-Baum gilt stets: $\forall v \in V: |bal(v)| \leq 1$
- dadurch: Höhe von $\log(n)$



Einfügen/ Löschen im AVL-Baum

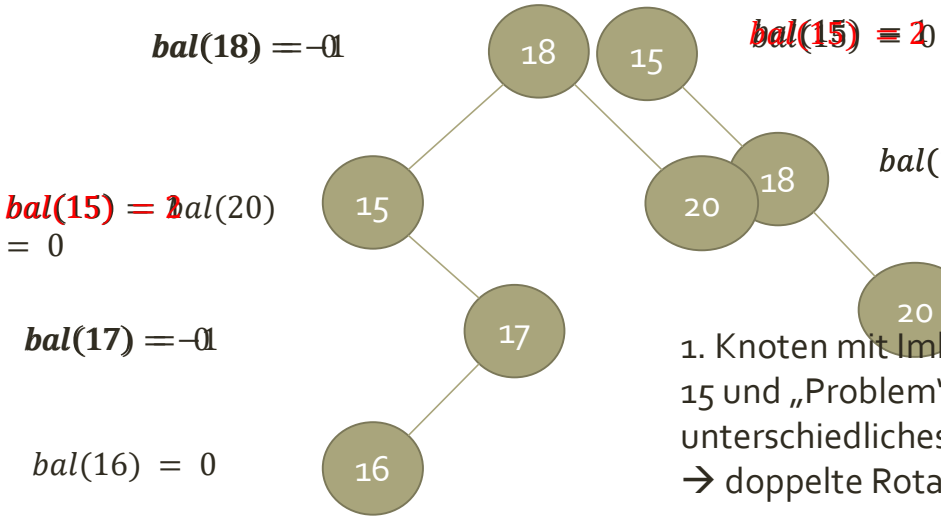
- nach beiden Operationen soll die Balance noch gelten
- falls nicht: durch Rotationen der entsprechenden Teilbäume herstellen

1. Gleiche Vorzeichen: einfache Rotation
2. Unterschiedliche Vorzeichen: Doppelrotation

Einfügen im AVL-Baum
(einfache Rotation)

Verfahren: finde richtige Stelle, füge ein, balanciere

Füge ein: 15 – 18 –20 – 17 – 16 – 13 – 9



1. Knoten mit Imbalance: 15
15 und „Problem"-Kind (18):
gleiches Vorzeichen
→ einfache Rotation

bal(18) = 1 0

bal(20) = 0

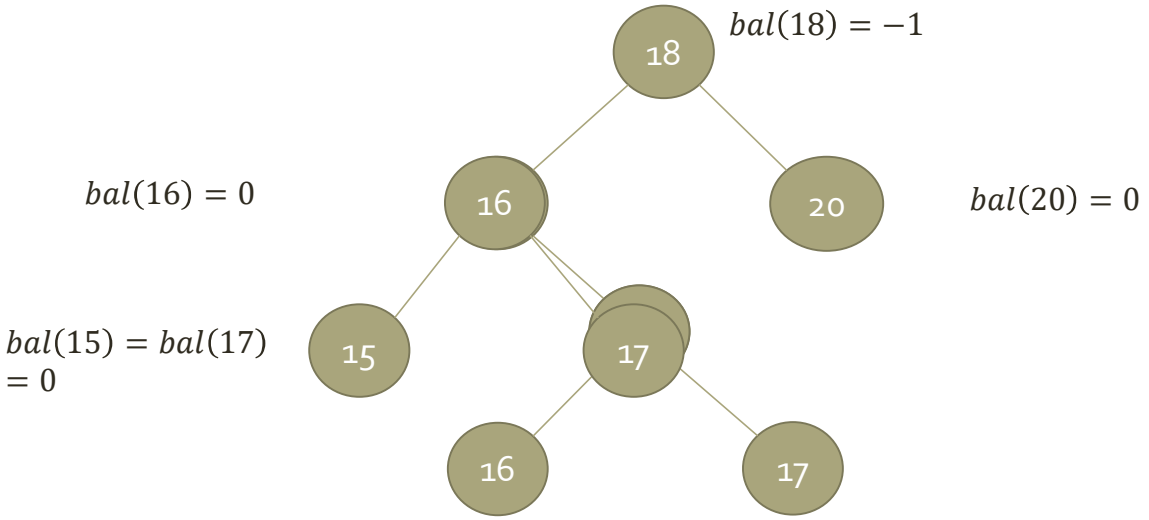
1. Knoten mit Imbalance: 15
15 und „Problem"-Kind (17):
unterschiedliches Vorzeichen
→ doppelte Rotation

$$Balance(v) = h(T_R) - h(T_L)$$

Einfügen im AVL-Baum (Doppelrotation)

Verfahren: finde richtige Stelle, füge ein, balanciere

Füge ein: 15 – 18 –20 – 17 – 16 – 13 – 9

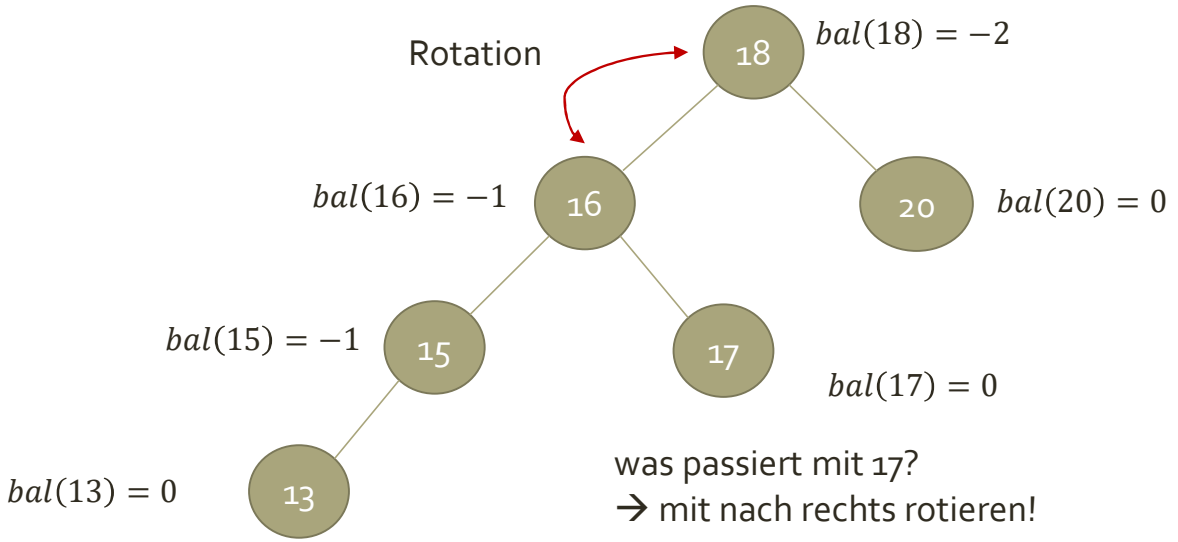


$$Balance(v) = h(T_R) - h(T_L)$$

Rotation – Kinder

Verfahren: finde richtige Stelle, füge ein, balanciere

Füge ein: 15 – 18 –20 – 17 – 16 – 13 – 9

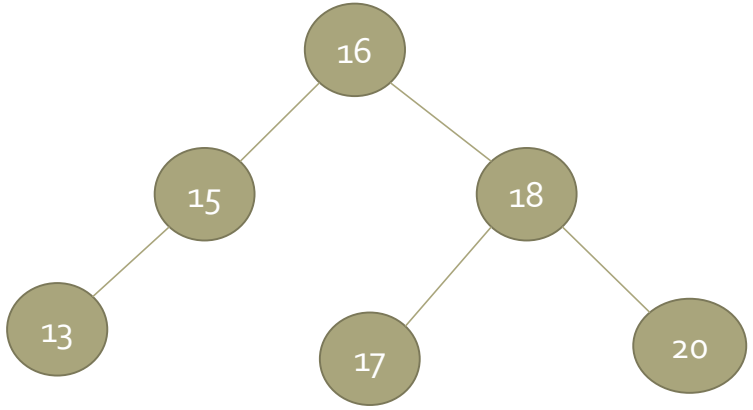


$$Balance(v) = h(T_R) - h(T_L)$$

Rotation – Kinder

Verfahren: finde richtige Stelle, füge ein, balanciere

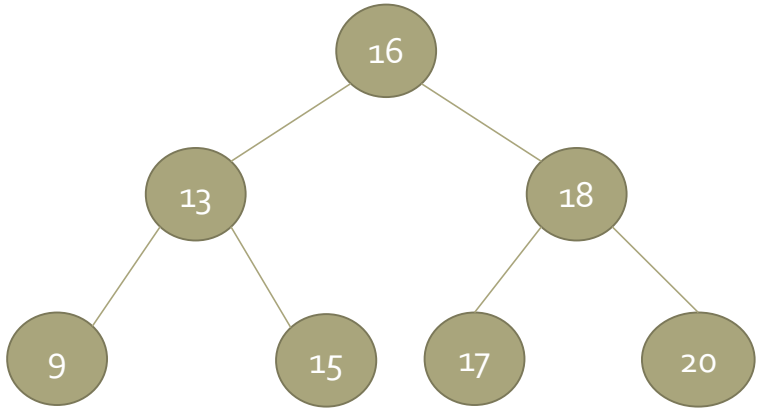
Füge ein: 15 – 18 –20 – 17 – 16 – 13 – 9



Löschen im AVL-Baum

Verfahren: finde Element & Vorgänger, replace, balanciere

Lösche: 18, 16



! Vorgängerknoten := „rechtestes Element im linken Teilbaum"

$$Balance(v) = h(T_R) - h(T_L)$$

PB

Nr.1

Aufgabe 1: AVL-Bäume

— Vorbereitung auf Aufgabe 1 und 2 des Übungsblattes —

Machen Sie sich zunächst anhand der folgenden Aufgaben noch einmal mit AVL-Bäumen vertraut. Zur Erinnerung: Die Balance $Bal(v)$ eines Knotens v in einem binären Suchbaum gibt die Differenz zwischen der Höhe des linken und des rechten Teilbaums mit Wurzel v an.

a) Fügen Sie die Zahlen 11, 17, 27, 7, 2, 13, 19 (in dieser Reihenfolge) in einen (zu Beginn leeren) AVL-Baum ein. Sie dürfen die Schritte, in denen nicht rotiert, sondern nur eingefügt wird, zusammenfassen. Anschließend löschen Sie die Zahl 17 wieder aus dem Baum.

b) Die Balance ist nicht zu verwechseln mit der Differenz zwischen den Knotenanzahlen in den entsprechenden Teilbäumen $Bal_K(v)$. Geben Sie je einen AVL-Baum an, bei dem für alle Knoten $|Bal_K(v)| \leq 1$ gilt, und einen AVL-Baum, bei dem das nicht der Fall ist.

Nr.2

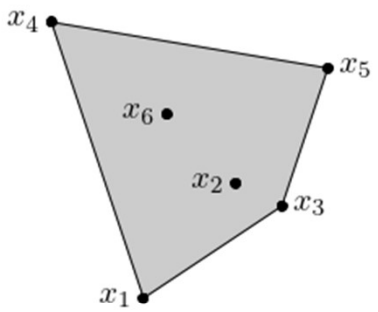
Konvexes Polygon

Konkaves Polygon

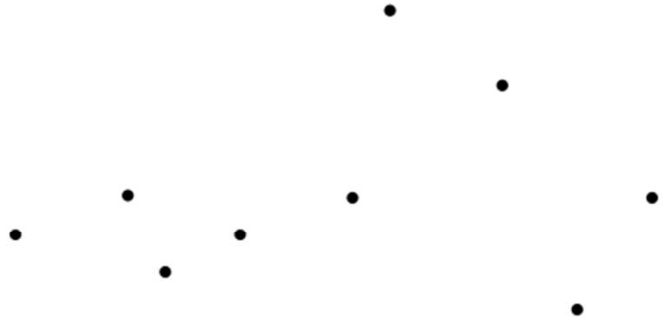
Aufgabe 2: Konvexe Hüllen

— Vorbereitung auf Aufgabe 3 des Übungsblattes —

Die konvexe Hülle $\text{conv}(\mathcal{P})$ einer Punktmenge $\mathcal{P} \subset \mathbb{R} \times \mathbb{R}$ ist das kleinste konvexe Polygon das alle Punkte in $\mathcal{P}$ enthält. Beachten Sie, dass in einem konvexen Polygon alle Innenwinkel höchstens 180° sind. Dabei lässt sich die konvexe Hülle über eine Menge von Eckpunkten $\mathcal{E} \subseteq \mathcal{P}$ definieren, die immer Teil der Punktmenge $\mathcal{P}$ sind.
Die Abbildung rechts zeigt die konvexe Hülle einer Punktmenge $\mathcal{P} = \{x_1, \ldots, x_6\}$. Beobachten Sie, dass auch hier gilt, dass $\mathcal{E} = \{x_1, x_3, x_4, x_5\} \subseteq \mathcal{P}$.



a) Berechnen Sie iterativ die konvexe Hülle der unten abgebildeten Punktmenge (diese können Sie auch von Moodle downloaden und mit Ipe editieren): Berechnen Sie zunächst die konvexe Hülle der linkesten drei Punkte und fügen dann nach x-Koordinate sortiert die weiteren Punkte ein.

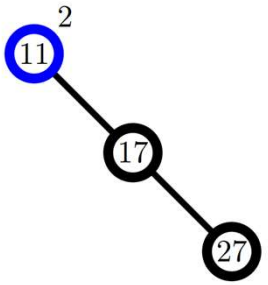


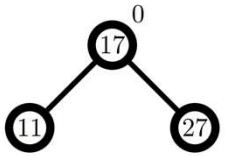
b) Diskutieren Sie mit Ihren Kommiliton_innen darüber, wie Sie die konvexe Hülle in jeder Iteration geupdatet haben. Versuchen Sie dabei u.a. folgende Fragen zu beantworten:

- Welche Segmente der konvexen Hülle haben Sie in jeder Iteration betrachtet?
- Was war anders, wenn der neue Punkt die größte/kleinste y-Koordinate hatte?
- War der neue Punkt immer Teil der konvexen Hülle?

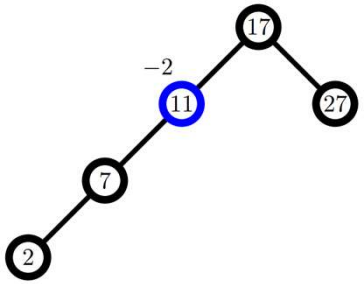
Besprechung

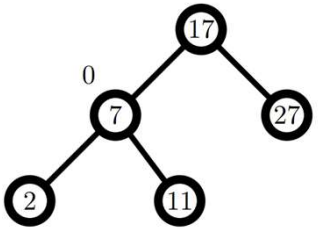
Nr.1 a) einfügen



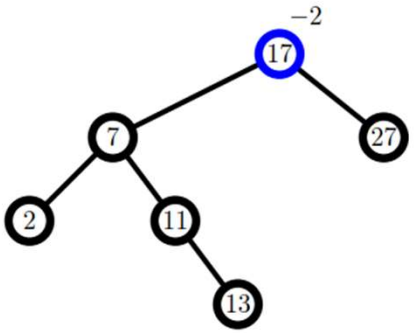


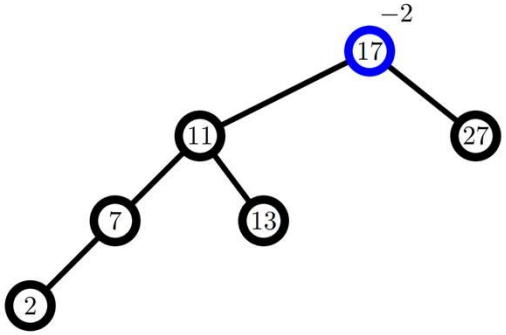
Neuer Knoten ist rechts-rechts
→ Linksrotation



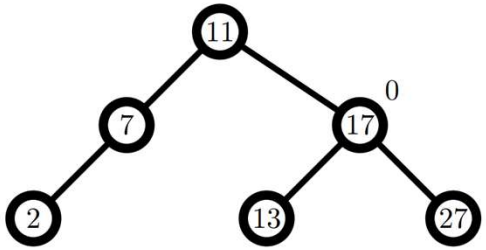


Neuer Knoten ist links-links
→ Rechtsrotation

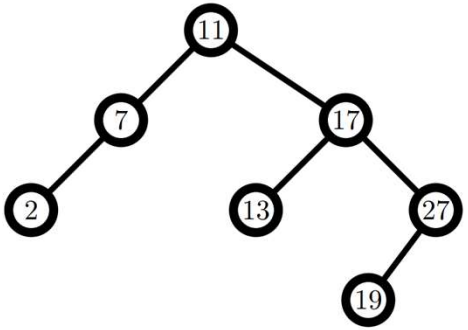




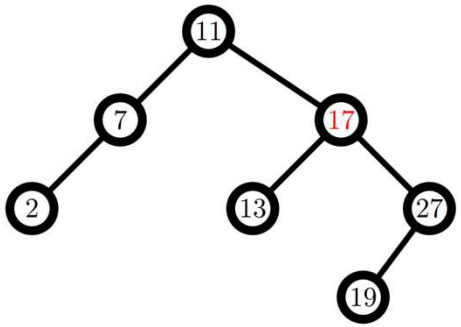
Neuer Knoten ist links-rechts
→ Doppelrotation



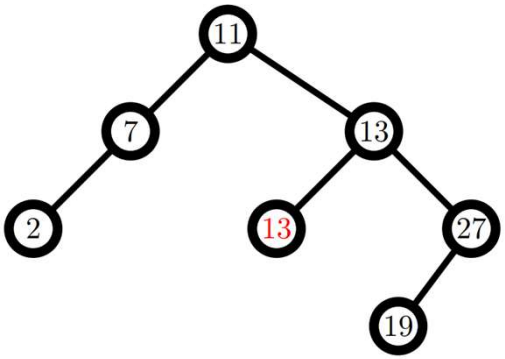
Neuer Knoten ist links-rechts
→ Doppelrotation



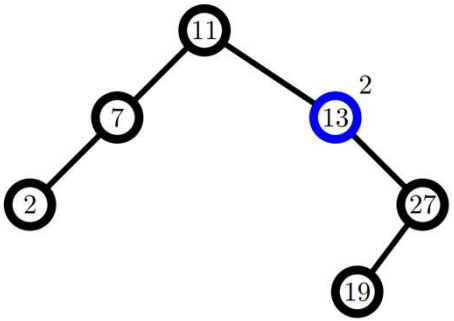
Nr.1 a) Löschen



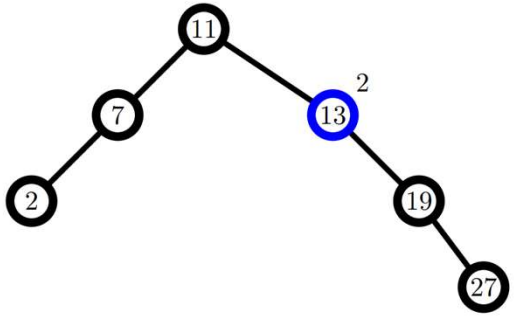
Knoten mit 17 soll gelöscht werden
→ Ersetze 17 durch 13



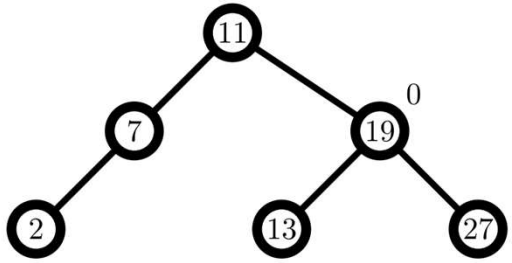
Lösche alten Knoten 13



Löschen verursacht Unbalanciertheit rechts-links
→ Doppelrotation



Löschen verursacht Unbalanciertheit rechts-links
→ Doppelrotation

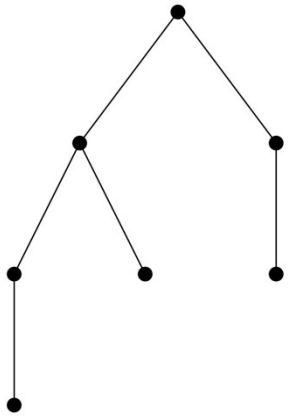


Nr.1 b)

- Differenz zwischen den Knotenanzahlen

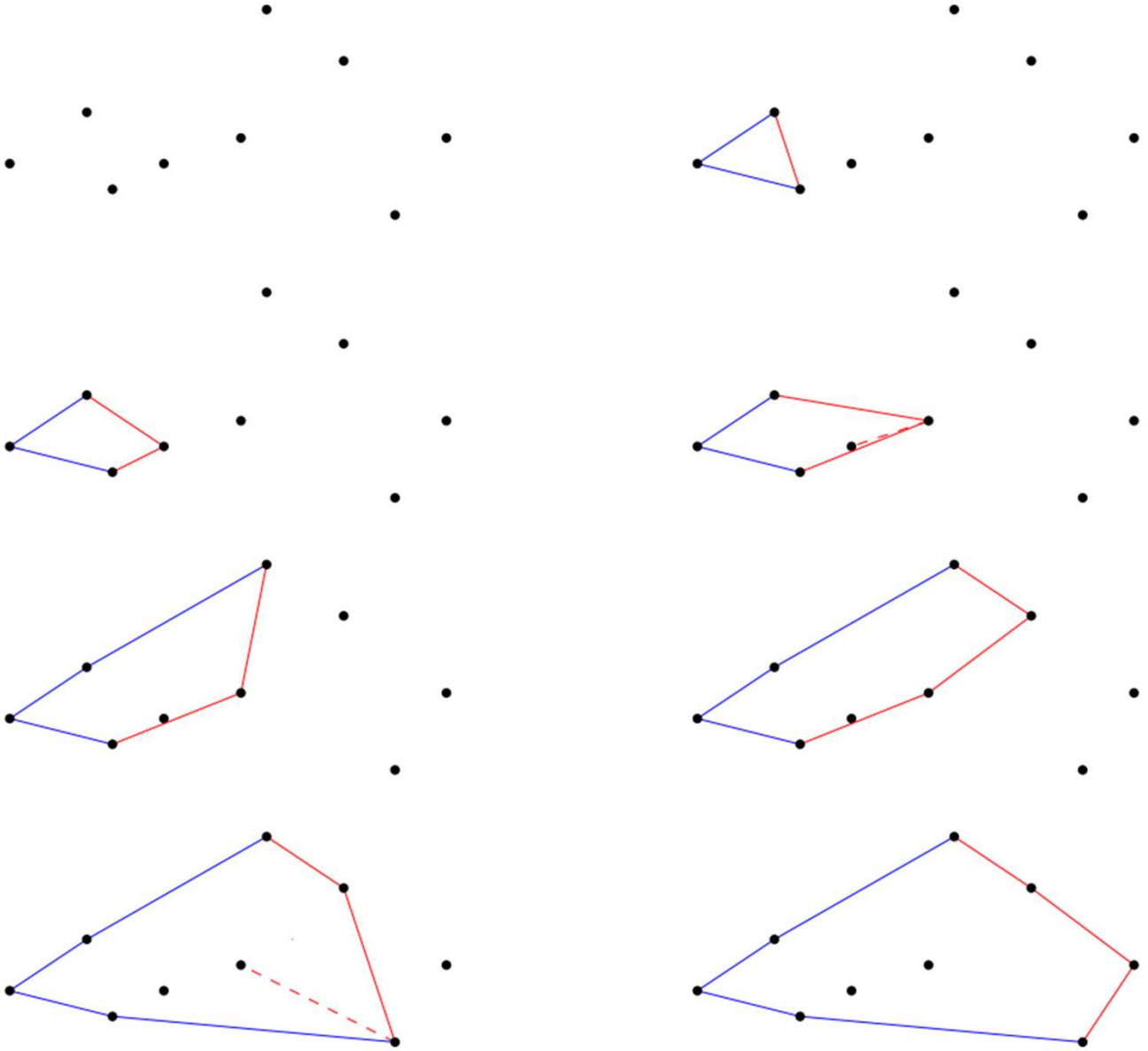
- $\forall v \in V\colon |Bal_K(v)| \leq 1$:
  vollständiger Binärbaum: $Bal_K(v) = 0\ \forall v \in V$

- $\exists v \in V\colon |Bal_K(v)| > 1$:
  Fibonacci Baum der Höhe 3:
  Wurzel r: $Bal_K(r) = 2$



Nr.2 a)

Zusätzlich betrachtete Kanten gestrichelt



Nr.2 b)

- Jede Iteration:
  - betrachte abgedeckte Konten und die beiden Nachbarn auf der konvexen Hülle
- Neuer Knoten im bereits entdeckten y-Bereich (z.B. Iteration 1 und 2):
  - Betrachte beide Knoten zwischen denen der neue liegt
  - Von dort entlang der konvexen Hülle (roter Teil) laufen bis Nachbarn identifiziert
  - Sonst: von Knoten mit größter und kleinster y-Koordinate aus suchen
- Neuer Knoten ist immer Teil der konvexen Hülle, denn
  - Er liegt rechts von allen anderen und kann nur so eingeschlossen werden