Karlsruher Institut für Technologie — Prof. Dr. Jörn Müller-Quade
Institut für Theoretische Informatik — Björn Kaidel, Sebastian Schlag, Sascha Witt

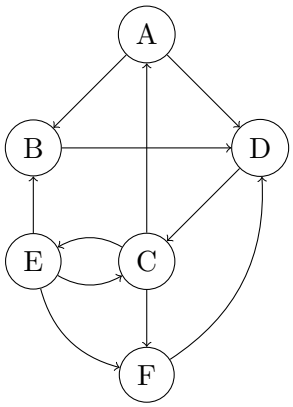
# 8. Übungsblatt zu Algorithmen I im SoSe 2017

http://crypto.iti.kit.edu/index.php?id=799
{bjoern.kaidel,sascha.witt}@kit.edu

## Musterlösungen

**Aufgabe 1** (*Breitensuche,* $2+1+1+1=5$ *Punkte*)

Gegeben sei folgender gerichteter Graph mit der Knotenmenge $\{A, B, C, D, E, F, G\}$:




In diesem Graph werde nun eine Breitensuche durchgeführt, und zwar ausgehend vom Knoten $A$. Wählen Sie die Knoten bei mehreren Möglichkeiten jeweils in alphabetischer Reihenfolge aus.

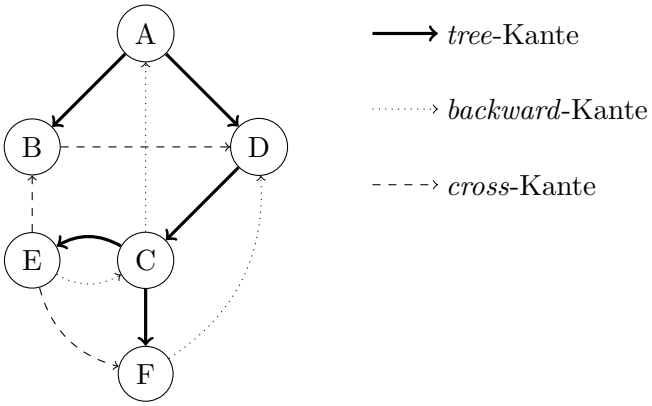
a) Zählen Sie die Knoten des Graphen in einer Reihenfolge auf, in der diese von einer Breitensuche jeweils zum ersten Mal berührt werden. Geben Sie außerdem alle Kanten an, die bezüglich dieser Breitensuche *tree*-Kanten sind.

b) Sind im Graph bzgl. dieser Breitensuche irgendwelche *backward*-Kanten vorhanden? Wenn ja, geben Sie diese an.

c) Sind im Graph bzgl. dieser Breitensuche irgendwelche *cross*-Kanten vorhanden? Wenn ja, geben Sie diese an.

d) Sind im Graph bzgl. dieser Breitensuche irgendwelche *forward*-Kanten vorhanden? Wenn ja, geben Sie diese an.

**Musterlösung:**



a) Die Knoten werden von der Breitensuche in der Reihenfolge A, B, D, C, E, F besucht. Die *tree*-Kanten sind (A,B), (A,D), (C,E), (C,F) und (D,C).

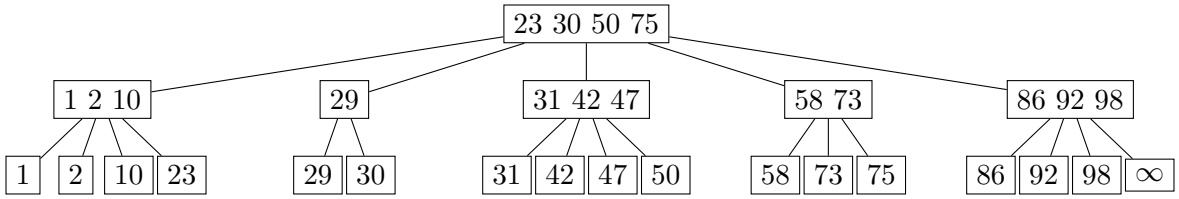
b) Ja, es gibt *backward*-Kanten. Diese sind (C,A),(E,C) und (F,D).

c) Ja, es gibt *cross*-Kanten. Diese sind (B,D), (E,B) und (E,F).

d) Bei einer Breitensuche können keine *forward*-Kanten auftreten.

**Aufgabe 2** $((a,b)$-*Bäume,* $3+3=6$ *Punkte*)

Führen Sie auf dem folgenden $(2,5)$-Baum die geforderten Operationen in der angegebenen Reihenfolge durch. Zeichnen Sie den Endzustand des Baums nach jeder der angegebenen Operationen.

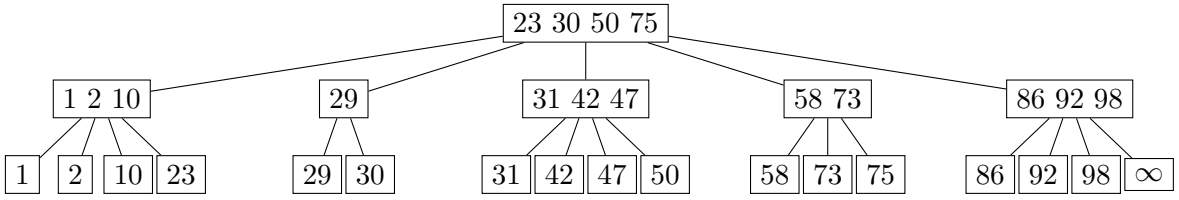



a) Einfügen von 93, Einfügen von 95, Einfügen von 28.

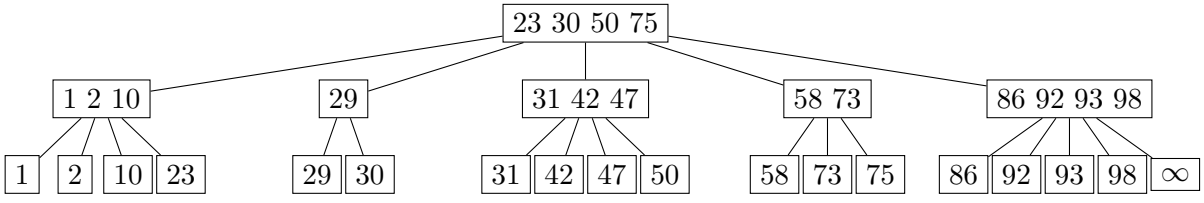
b) Beginnen Sie wieder *beim Ausgangszustand*. Löschen von 50, Löschen von 75, Löschen von 73.

**Musterlösung:**

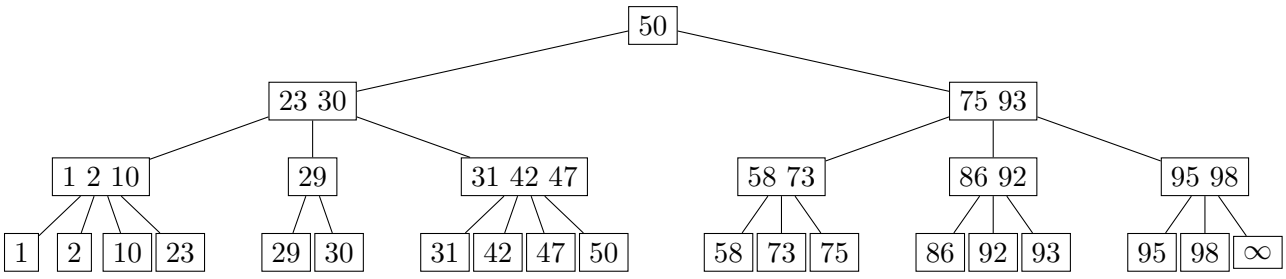
a) Originalzustand



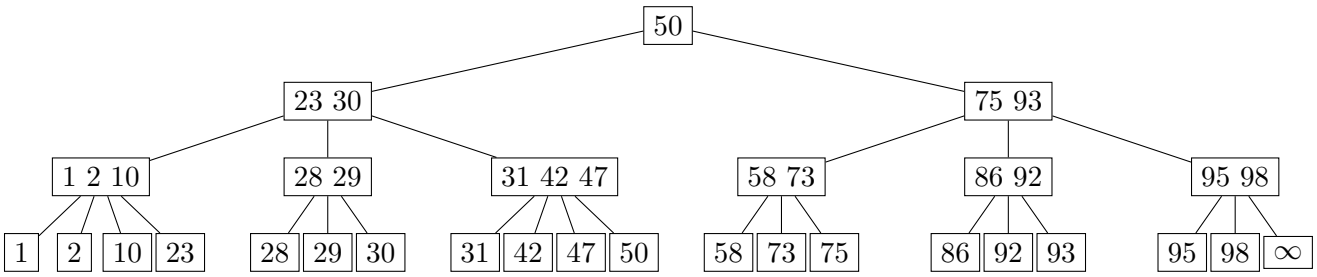

nach Einfügen von 93



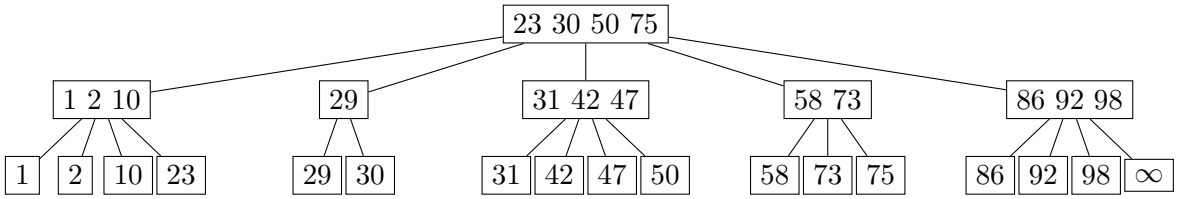

nach Einfügen von 95



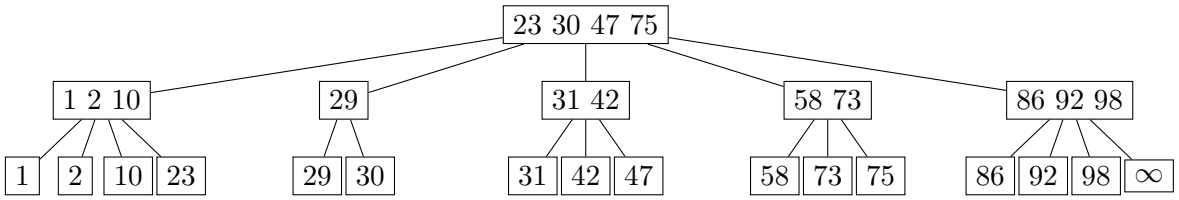
nach Einfügen von 28



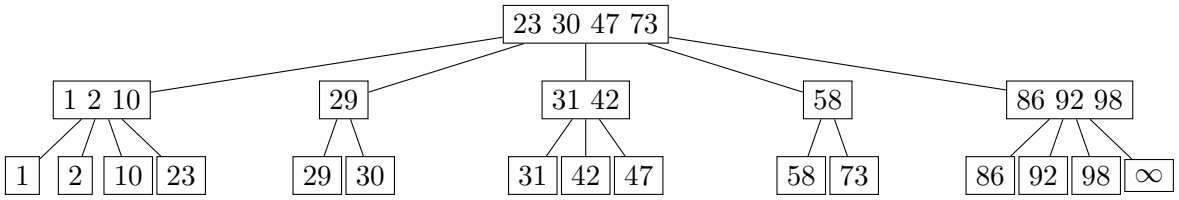

b) Originalzustand



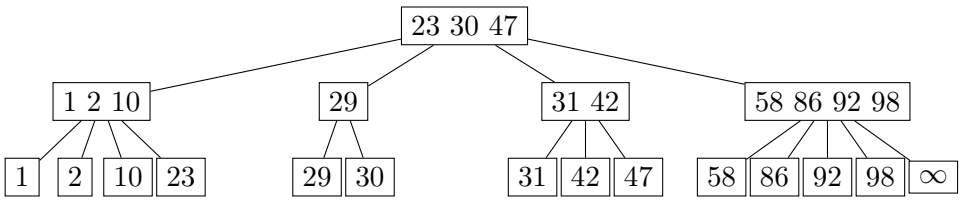

nach Löschen von 50




nach Löschen von 75




nach Löschen von 73




**Aufgabe 3** (*Zweifärbung,* $1 + 4 + 2 = 7$ *Punkte*)

Es sei $G = (V = \{1, ..., n\}, E)$ ($n \in \mathbb{N}$) ein ungerichteter und zusammenhängender Graph. $G$ ist *zweifärbbar*, wenn es eine Abbildung $f : V \to \{0, 1\}$ gibt, so dass $\forall\{u, v\} \in E : f(u) \neq f(v)$. Es dürfen also keine zwei benachbarten Knoten die gleiche „Farbe“ 0 bzw. 1 haben.

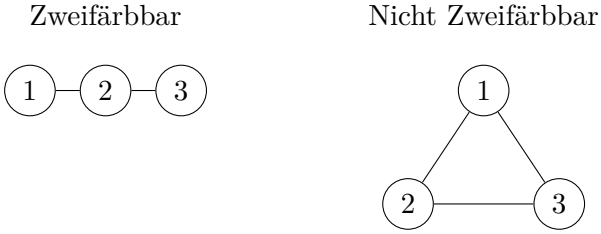
a) Geben Sie einen zweifärbbaren und einen nicht zweifärbbaren Graphen mit je 3 Knoten an.

b) Zeigen Sie: $G$ ist zweifärbbar $\Leftrightarrow$ $G$ enthält keinen Zyklus ungerader Länge.

c) Geben Sie einen Algorithmus im Pseudocode an, der eine Zweifärbung für $G$ berechnet. Existiert eine Zweifärbung, so soll diese in einem Array $f$ gespeichert werden, sodass $f[i]$ die Farbe des Knotens $i \in V$ angibt. Existiert keine Zweifärbbung, so soll der Algorithmus $\perp$ ausgeben. Die Laufzeit des Algorithmus soll in $\mathcal{O}(|V| + |E|)$ sein. Argumentieren Sie, dass Ihr Algorithmus auch wirklich diese Laufzeit erreicht.

**Musterlösung:**

a) Die folgenden Graphen sind Beispiele für (nicht) zweifärbbare Graphen mit 3 Knoten:




b) **$\Rightarrow$:** Ein Zyklus ungerader Länge kann nicht zweigefärbt werden. Wir beweisen dies: Sei $\langle v, ..., u, v\rangle$ ein beliebiger Zyklus ungerader Länge. Dann habe o.B.d.A. der Knoten $v$ die Farbe 0, der nächste Knoten die Farbe 1, der übernächste Knoten die Farbe 0 etc. Da der Zyklus ungerade Länge hat, ist $u$ über eine gerade Anzahl von Kanten erreichbar und muss somit die Farbe 0 haben. Somit kann der Zyklus nicht zweigefärbt werden, da bereits $v$ die Farbe 0 hatte und $u$ und $v$ über eine Kante verbunden sind. Es müssen in einem Zyklus ungerader Länge also zwangsläufig zwei benachbarte Knoten die gleiche Farbe erhalten. Da $G$ zweifärbbar ist, kann $G$ also keinen Zyklus ungerader Länge enthalten.

**$\Leftarrow$:** Wir führen ausgehend vom Knoten $1 \in V$ eine Breitensuche aus. Diese liefert uns ein Array $d$ der Abstände der Knoten in $V$ im Breitensuchbaum zum Knoten 1. Wir färben nun alle Knoten $i \in V$ mit ungeradem $d[i]$ mit der „Farbe“ 1 und alle anderen mit 0. Wir müssen nun zeigen, dass dies eine gültige Zweifärbung ist.

Angenommen, es handelt sich bei der obigen Färbung nicht um eine gültige Zweifärbung. D.h. es existiert eine Kante $(x, y) \in E$, sodass $x$ und $y$ die gleiche Farbe zugewiesen wurde. Man beachte nun, dass entweder $d[x]$ und $d[y]$ identisch sind oder sich nur um den Faktor 1 unterscheiden, da $x$ und $y$ über eine Kante verbunden sind und der Graph ungerichtet ist.

Wir betrachten nun die Pfade von 1 zu $x$ und $y$. Es sei $w$ der letzte Knoten, den beide Pfade gemeinsam haben. Falls $w$ gleich $x$ oder $y$ ist, so sind die Pfade unterschiedlich lang und unterscheiden sich nach obiger Überlegung nur um den Faktor 1. Damit wären $x$ und $y$ aber unterschiedlich gefärbt worden.

Es muss also ein Zyklus zwischen $w, x$ und $y$ vorliegen, der aus dem Pfad von $w$ zu $x$ im Breitensuchbaum, dem Pfad von $w$ zu $y$ im Breitensuchbaum und der Kante $\{x, y\}$ besteht. Die Länge dieses Zyklus ist somit

$$\ell := (d[x] - d[w]) + (d[y] - d[w]) + 1 = d[x] + d[y] - 2 \cdot d[w] + 1.$$

Da $G$ keine ungeraden Zyklen enthält, muss $\ell$ gerade sein, woraus $d[x] \neq d[y]$ folgt. Hieraus folgt wiederum, dass $x$ und $y$ von obigem Algorithmus unterschiedlich gefärbt worden sein müssen.

Insgesamt folgt also, dass die berechnete Zweifärbbung gültig und $G$ somit zweifärbbar ist.

c) Der Pseudocode könnte folgendermaßen aussehen:

```
Function TwoColoring(G = (V = {1, ..., n}, E)) :
    f[1...n] : Array of {0, 1}
    (parent, d) := bfs(V[1], G)                    -- BFS vom ersten Knoten aus

    for i := 1 to n do                             -- Farben zuweisen
        f[i] := d[i] mod 2
    for {u, v} ∈ E do                              -- Überprüfen, ob die Färbung gültig ist
        if f[u] = f[v] then
            return ⊥
    return f
```

Die Breitensuche ist in $\mathcal{O}(|V| + |E|)$ möglich und die folgenden Schleifen liegen in $\mathcal{O}(|V|)$ bzw. $\mathcal{O}(|E|)$, woraus sich eine Gesamtlaufzeit von $\mathcal{O}(|V| + |E|)$ ergibt.