

# Algorithmische Graphentheorie

## 5. Übungsblatt

Jasper Gude      Pia Röttgers

25. Mai 2026

/ 25

### Aufgabe 1 – Paarungen (Matchings) in Bäumen

- a) Ein Baum ist ein kreisfreier, zusammenhängender Graph. Ein Baum besitzt ein perfektes Matching genau dann, wenn die Anzahl der Knoten gerade ist und jedes Blatt keine Geschwisterknoten hat.

Die Anzahl der Knoten muss gerade sein, da in einem perfekten Matching jeder Knoten mit einem anderen gematcht wird.

Jedes Blatt muss Einzelblatt sein, da der Elternknoten nur mit einem Blatt gematcht werden kann.

Also ist ein perfektes Matching im Baum eindeutig, da jedes Blatt mit seinem Elternknoten gematcht werden muss. Diese Kante  $uv$  ist sicher im Matching enthalten und alle zu  $u$  oder  $v$  inzidenten Kanten können aus dem Baum gelöscht werden. So entsteht ein neues Blatt, für das die selbe Regel gilt.

/ 3

- b) Der Algorithmus 1 berechnet ein größtes Matching.

Der Algorithmus ist korrekt, da der Baum  $T$  entweder ein perfektes Matching enthält und somit ein perfektes Matching berechnet (Siehe Punkt a)), oder I dont fucking know.

Die Laufzeit für INITIALIZE liegt in  $\mathcal{O}(V)$ , da sich der Baum in linearer Zeit augmentieren lässt und es  $|V| - 1$  viele Kanten gibt.

Die Laufzeit für MATCHATREEREC liegt in  $\mathcal{O}(V)$ . Der Algorithmus wird genau einmal für jeden Knoten aufgerufen. Wenn eine Kante zum Matching hinzugefügt wird, werden die Knoten und Kanten aus dem Baum

### Algorithmus 1: Größtes Matching in Bäumen

```

MATCHATEE( $T = (V, E)$ )
    // Den ungerichteten Baum zu einem gerichteten Baum mit parent und
    // children pro Knoten und einer Wurzel root augmentieren.
    INITIALIZE( $T$ )
    return MATCHATEEREC( $T, T.root$ )

MATCHATEEREC( $T, r$ )
    // Rekursiv für die Kinder aufrufen
     $m \leftarrow \emptyset$ 
    foreach  $v \in r.children$  do
         $m \leftarrow m \cup \text{MATCHATEEREC}(T, v)$ 
    // Wenn Knoten Blatt ist, dann die Kante zum Elternknoten hinzufügen.
    if  $r \neq \text{nil}$  and  $\deg(r) = 1$  then
        // Die Knoten und inzidente Kanten aus dem Baum entfernen.
         $T \leftarrow T \setminus \{r.parent, r.parent.children\}$ 
         $m \leftarrow m \cup \{r, r.parent\}$ 
    return  $m$ 

```

gelöscht die nicht mehr zum Matching hinzugefügt werden können und somit nicht mehr bearbeitet werden.

Also liegt auch MATCHATEE in  $\mathcal{O}(V)$ .

/ 5

- c) Wenn eine Kante  $rv$  zu einem Matching hinzugefügt wird, kann keine zu  $v$  inzidente Kante hinzugefügt werden. Das müssen wir auch bei unserem dynamischen Programm beachten.

$$\text{OPT}(r) = \max_{v \in r.children} \left\{ rv + \sum_{i \in v.children} \text{OPT}(i) + \sum_{j \in r.children \setminus v} \text{OPT}(j) \right. \\ \left. \sum_{k \in r.children} \text{OPT}(k) \right\}$$

Der Algorithmus hält sich an die Eigenschaften eines Matchings, also ist er korrekt.

Der Algorithmus probiert alle Kindknoten von  $r$  aus. Geht man mit  $r$  im Baum von unten nach oben sind das also höchstens so viele Iterationen wie, Knoten im Baum existieren (ohne die Blätter). Also läuft das Dynamische Programm in  $\mathcal{O}(V)$ .

/ 5

## Aufgabe 2 – Hamiltonkreise

/ 1

a)

b)

|     |
|-----|
| / 2 |
|-----|

c)

|     |
|-----|
| / 1 |
|-----|

d)

|     |
|-----|
| / 1 |
|-----|

e)

|     |
|-----|
| / 1 |
|-----|

f)

|     |
|-----|
| / 1 |
|-----|

**Aufgabe 3 – Perfekte Matchings in bipartiten Graphen**

|     |
|-----|
| / 5 |
|-----|