Schubert Julian, Gruppe 4
Philipp Wahl, Gruppe 4

Abgabe zum 5. Übungsblatt (AGT 21)

Aufgabe 1:

a) Wir betrachten zunächst alle Blätter im Baum. gibt es ein perfektes Matching, so sind alle Knoten (also auch Blätter) teil dieses Matchings (bzw es gibt Kanten zu diesen Knoten). Es muss also, wenn es ein perfektes Matching gibt, jede Kante die einen Elternknoten mit einem Blatt verbindet im Matching enthalten sein (sonst wäre das Blatt ja nicht Teil des Matchings). Das bedeutet aber auch, dass eine Kante, die einen Knoten v mit einem Knoten u verbindet, wobei der Knoten u Elternknoten eines Blattes b ist, nicht im Matching enthalten sein darf (da sonst zwei Kanten den Start/Endpunkt u hätten). Da v aber auch im Matching enthalten sein muss, kann man nun diesen Knoten als Blatt betrachten (man lässt den Teilbaum unter selbigem Knoten einfach wegfallen). Gibt es ein perfektes Matching, kann als einzige Kante wieder nur die Kante die u mit seinem Elternknoten verbindet gewählt werden. Gibt es im Baum also ein perfektes Matching, muss dieses Eindeutig sein da es immer nur eine Möglichkeit gibt einen Knoten zum Matching hinzuzufügen.

b) Wir benutzen folgenden Algorighmus

```
BerechneMatching(G)
    blätter = new Queue matching = ∅
    foreach v ∈ V do Alle Blätter zur Liste hinzufügen
        if v is Leaf then Wenn der aktuelle Knoten v ein Blatt ist, wird er
          zur Liste hinzugefügt
            blätter.Enqueue(v)
    while not blätter.Empty() do Matching für alle Blätter bilden
        curr = blätter.Dequeue()
        parent = curr.parent
        // Kante vom Parent zum aktuellen Knoten zum matching
          hinzufügen
        matching = matching ∪ {curr, parent}
        // Parent vom Parent wird nun zur Blätter-Liste hinzugefügt
        blätter.Enqueue(parent.parent)
    return matching
```

Laufzeit: Das finden aller initialen Blätter im Graphen benötigt $\mathcal{O}$(V) (da wir über alle Knoten iterieren und für jeden Knoten prüfen ob er ein Blatt ist).
Dann wird jeder Knoten maximal ein mal in der zweiten While-Schleife behandelt (es wird jeder Zweite Knoten als curr ausgewählt). Die Operationen in der zweiten Schleife haben jeweils Eine Laufzeit von $\mathcal{O}(1)$. Die zweite Schleife hat somit also einen Laufzeit von $\mathcal{O}$(V). Insgesamt ergibt sich so für unseren Algorighmus eine Laufzeit von $\mathcal{O}$(V) (Da $\mathcal{O}$(V + V) = $\mathcal{O}$(V))

Korrektheit: Wir haben in Aufgabe a gezeigt, dass es für ein perfektes Matching immer nur eine Möglichkeit gibt einen Knoten hinzuzufügen, da unser algorithmus nach genau dieser Vorgehensweise arbeitet, ist unser algorithmus für perfekte Matchings korrekt.
Im falle eines nicht-perfekten-matchings findet unser Algorithmus ein größtes Matching. Dies ist gegeben da wir so viele Blätter wie möglich auswählen (und somit auch so viele Knoten wie möglich).

c) Wir betrachten jeden Knoten sowie alle Kanten die zu diesem Knoten führen (bzw auch diese die von dem Knoten weg führen). Wir können uns nun für jeden dieser Knoten die Kanten nach Gewicht sortieren (Das geht in $\mathcal{O}$(E log E) da wir jede Kante maximal zwei mal behandeln und das einfügen in die Liste an die richtige Stelle in $\mathcal{O}$(log E) funktioniert). Wir wählen dann die Kanten wie folgt aus: Ist eine Kante für zwei Knoten direkt die beste Kante für beide Knoten so wählen wir diese Kante. Sonst nehmen wir einfach die beste Übereinstimmende Kante. Das sorgt dafür, das jeder Knoten die beste Kante für sich wählt, außer wenn wir dadurch eine bessere Kombination kaputt machen würden, dann wählen wir die bessere Kombination. Dieses Vorgehen dauert $\mathcal{O}(E^2)$, insgesamt haben wir also eine Laufzeit von $\mathcal{O}(E^2)$

Aufgabe 2:

a) Wenn es drei Knoten gibt muss gelten: Der Knoten v_2 muss vom Knoten v_1 aus erreichbar sein (dies ist gegeben da der Graph stark zusammenhängend ist). Ebenso muss v_3 vom Knoten v_2 erreichbar sein und v_1 vom Knoten v_4. Es gibt also einen Kreis der Länge größer gleich 3, der von v_1 (über k andere Knoten) nach v_2, von dort aus nach v_3 und zum Schluss wieder nach v_1 führt.

b) Wir gehen alle Knoten im Kreis durch der Entfernt werden kann ohne das die Bedingung verletzt wird, das können wir machen bis nur noch 3 Übrig. Der Knoten muss existieren weil wegen Tunier und andere Bedingung

c)