Schubert Julian, Gruppe 4
Philipp Wahl, Gruppe 4

Abgabe zum 9. Übungsblatt (AGT 21)

Aufgabe 1:

a + b) Die Spitze des Turmgraphen s muss eine andere Farbe haben als die beiden Knoten mit denen sie Verbunden ist (u und v). Da diese beiden Knoten u und v ebenfalls miteinander Verbunden sind, muss $f(s) \neq f(u) \neq f(v)$ gelten. Die letzen beiden Knoten im Turm (x und y) sind beide jeweils mit u und v verbunden und müssen damit eine andere Färbung als u und v haben. Da wir nur drei Farben zur Auswahl haben müssen x und y die selbe Farbe wie s haben. Die Knoten x und y dürfen die selbe Farbe haben da es keine Kante $xy \in E$ gibt. Wir hängen nun zwei Turmgraphen aneinander indem wir den Knoten x_1 vom Turmgraphen 1 mit dem Knoten y_2 vom zweiten Turmgraphen vereinigen. Dies verletzt die Färbeigenschaft nicht, da x_1 und y_2 wie oben gezeigt die selbe Farbe haben müssen. Somit haben auch die Spitzen der beiden Türme die gleiche Farbe. Nach diesem Schema können wir eine beliebig Lange Katte an Turmgraphen bilden. Hierbei haben y_1 und x_n immer die selbe Farbe. Wir können die Kette schließen indem wir y_1 und x_n vereinigen und erhalten somit eine Valide Färbung für einen Sterngraphen der aus n Türmen besteht wobei Alle Knoten mit Grad 2 die selbe Färbung haben.

c) Wir sollen zeigen, dass wenn es eine effiziente Lösung für 3COL4 geben würde, wir dann auch 3COL effizient lösen könnten.
Wir müssen für jeden Knoten $v \in V$ mit Grad $deg[v] \geq 5$ den Graphen G so in Teilgraphen aufteilen, dass für jeden Teilgraphen G' $\forall v \in V' : deg[v] \geq 4$ gilt.
Wir zeigen zunächst das es egal ist, wie wir diese Teilmengen wählen:
Alle Knoten die in G die selbe Färbung haben, bilden in G eine unabhängige Menge. Sind zwei Knoten in G unabhängig, so sind sie auch im Teilgraphen G' unabhängig. Es ist also nicht möglich, dass wir eine valide Färbung nicht erkennen weil wir schlechte Teilmengen gewählt haben. Allerdings besteht die Möglichkeit, dass es für einen Teilgraphen mehrere mögliche Färbungen gibt wovon jedoch nur eine in G zulässig ist. Wenn sich 3COL4 effizient lösen lässt, dann können wir in den Teilgraphen G' ebenfalls alle möglichen Lösungen effizient bestimmen und uns merken.
Wir beginnen also damit, den Graphen G in so Teilgraphen G' zu zerlegen sodass kein Knoten in einem der G' einen Grad größer als 4 hat. Dann bestimmen wir für jeden dieser Teilgraphen alle Lösungen.
Nun beginnen wir bei einem Knoten v der in G einen Grad größer als 4 hat. Falls es keinen solchen Knoten gibt wählen wir einfach eine der validen Färbungen und sind fertig. Falls dieser Knoten mehrere mögliche Färbungen hat wählen wir davon eine, die keine Färbung in einer der Teilgraphen verletzt die den Knoten v enthalten. Verschwinden dadurch mögliche Färbungen passen wir die noch verbleibenden Möglichkeiten für die anderen Knoten an und wiederholen das so lange, bis sich keine Färbung mehr ändert. Dann wählen wir den nächsten

Knoten mit Grad größer als 4 in G aus und wiederholen das vorgehen. Dies tun wir so lange, bis alle Knoten mit Grad 4 fest zugeteilt sind und wiederholen das Verfahren dann noch für alle übrigen Knoten.
Hierdurch sind wir in der Lage 3COL effizient zu lösen, falls es eine effiziente Lösung für 3COL4 gibt. Da 3COL aber nicht effizient Lösbar ist, kann es keine effiziente Lösung für 3COL4 geben.

Aufgabe 2:

a) Im schlimmsten Fall müssen wir alle Nachbarn von v unterschiedlich färben und anders gefärbt als v. Mehr Farben können nicht benötigt werden da sonst eine Farbe unbenutzt bleibt
Angenommen wir haben einen Knoten $v \in V$ mit $deg[v] = k$, wobei v der Knoten mit dem größten Grad aus V ist. Im extremfall sind alle Nachbarn von v untereinander Verbunden ($N(v)$ ist eine Clique in G). Dann müssen alle Knoten in $N(v)$ eine unterschiedliche Farbe bekommen, und v selbst darf keine dieser Farben bekommen. Insgesamt werden also $|N(v)| + 1 = deg[v] + 1 = max_{v \in v} deg[v] + 1$ viele Farben benötigt. Die Ungleichung gilt.

b) Wir wählen den selben Graphen wie in Teilaufgabe a, entfernen aber n - delta viele Kanten zwischen den Knoten (wobei wir nie Kanten zwischen)

c) Wir minimieren:

$$max_{v \in V}(f(v))$$

Wobei $f(v)$ die Farbe ist, die der Knoten v zugeteilt bekommt. Wir minimieren also die Zahl der Färbungen. Nebenbedingung:

$$\forall (u, v) \in E : f(u) \neq f(v)$$

Wir lösen mit unserem Programm: Die chormatische Zahl des gegeben Graphen ist 3.

e) Ansatz: Für jeden Knoten für jede Farbe speichern: hat Farbe k Ja = 1, Nein = 0.
Dann summe über die zuweisungen minimieren mit bedinung das f von beiden knoten unterschiedlich

Aufgabe 3:

Wir führen zunächst das im Satz von Dirac beschriebene Vorgehen aus um im Graphen $G = (V, E)$ einen Knoten v_1 zu finden, der simplizial in $G[V] = G$ ist. Dann Entfernen wir den Knoten v_1 und führen das Vorgehen für den resultierenen Graphen aus. Das wiederholen wir bis nur 1 Knoten über bleibt, wir haben so V iterationen. Wir müssen den Graphen wieder in Teilmengen U N(U) und W aufteilen, das geht in O(V). Dann können wir für jeden Knoten in W in $O(V^2)$ prüfen ob er simplizial ist (indem wir für jeden Nachbarn schauen ob alle Nachbarn passen) und ihn zurück geben. Da maximal $O(V)$ knoten in W liegen geht das insgesamt in $O(V^3)$. Die Bedingungen formulieren wir als LP.

1 COde:

task.mod:

```
int numNodes = ...;
int numEdges = ...;

// Represents an edge in the graphe
tuple Edge {
    int node1;
    int node2;
}

// Loads all edges present in the graph
{Edge} edges = ...;

// Stores the solution for all nodes
dvar int+ nodeSolution[1..numNodes];

// Minimize the sum of all nodes
minimize max (i in 1..numNodes) nodeSolution[i];

// Each edge needs to be adjacent to at least one node with a value of 1
subject to {
    forall (edge in edges) {
        // Both nodes edge different color
        nodeSolution[edge.node1] != nodeSolution[edge.node2];
    }

}


// Printing solution
execute {
    for(var i = 1; i <= numNodes; i++) {
        writeln("Knoten ", i, ": ", nodeSolution[i]);

    }
}
```

task.dat:

```
numNodes = 8;
numEdges = 15;

edges = {
    <1, 2>, <1, 3>, <1, 8>,
    <2, 3>, <2, 4>, <2, 5>,
    <3, 4>, <3, 5>,
    <4, 6>, <4, 7>,
    <5, 6>, <5, 7>,
    <6, 7>, <6, 8>,
```

<7, 8>};