Lösungsstrategien für NP-schwere Probleme
Blatt 3

Jakob Rieck 6423721
Konstantin Kobs 6414943
Thomas Maier 6319878
Tom Petersen 6359640

Abgabe zum 02.05.16

Aufgabe 1

a) Wir erstellen eine Tabelle, welche an der linken Seite die Teilmengen B_i und an der oberen Seite die Elemente $a_j \in A$ stehen hat. Nun füllen wir die Tabelle aus, indem wir in einer Zelle $z_{i,j}$ eine Eins einfügen, wenn $a_j \in B_i$; sonst fügen wir eine Null ein. Zeilenweise addieren sich die Zahlen nun zu maximal c, da in jeder Teilmenge maximal c Elemente vorkommen können. Spaltenweise addieren sich die Zahlen zu unterschiedlichen Summen. Dies nutzen wir nun. Da eine Spalten-Summe größer ist, wenn mehr Elemente dieses Element enthalten, ist es sinnvoll, dieses Element in das Hitting-Set einzufügen. Sei die größte Spalten-Summe nun in Spalte i zu finden. Gibt es gleich-große Summen, so wird die mit dem kleineren Index gewählt (die Auswahl kann auch zufällig geschehen, allerdings ist der Algorithmus somit deterministisch). Nun wird die Spalte i komplett aus der Tabelle gestrichen. Ebenfalls werden alle Zeilen, die in der Spalte i eine Eins stehen hatten, gestrichen, denn die repräsentierten Teilmengen der Zeilen sind durch die Hinzunahme des Elements a_i getroffen worden. Die Tabelle ist also um eine Spalte und eine maximale Anzahl von m Zeilen kleiner geworden. Nun werden die Spalten-Summen erneut berechnet und das Element mit der größten Spalten-Summe in H aufgenommen. Dieses Summieren und Löschen wird nun solange iteriert, bis eine der folgenden Bedingungen zutrifft:

1. Es wurden k Iterationen durchgeführt. Das Hitting-Set H enthält somit k Elemente. Da diese eine bestimmte Menge von Teilmengen B_i treffen und theoretisch alle Teilmengen treffen sollen, muss überprüft werden, ob die Tabelle nach dem Streichen des letzten Elements und seinen getroffenen Teilmengen noch Zeilen enthält. Ist dies nicht der Fall, so sind alle Teilmengen mit dem Hitting-Set getroffen worden. Enthalt die Tabelle noch Zeilen, so wurden diese Teilmengen nicht getroffen und es gibt kein Hitting-Set mit maximal k Elementen.
2. Es gibt keine Zeilen mehr, sodass keine Spalten-Summen mehr berechnet werden können. Dies bedeutet, dass es ein Hitting-Set mit weniger als k Elementen gibt.
3. Es gibt keine Spalten mehr, was bedeutet, dass sich alle Elemente bereits im Hitting-Set befinden. Somit gab es von Anfang an weniger als k Elemente in A, sodass ein Hitting-Set somit trivial ist.

   Nach dem Abbruch der Iterationen wurde entweder ein Hitting-Set der Größe k oder kleiner gefunden, oder es wurde festgestellt, dass es kein Hitting-Set mit dieser Größe geben kann.

b) Zum Aufstellen der Tabelle, sowie zum Berechnen der Spalten-Summen wird jeweils $p(n, m) = \mathcal{O}(n \cdot m)$ Zeit benötigt. Das Streichen (technisch wohl ein Null-Setzen der Zellen in Zeilen und Spalten) braucht ebenfalls $p(n, m)$ Zeit. Das Streichen führen wir insgesamt maximal $\min(n, m, k)$-mal aus, wobei dies nach oben hin mit $f(k) = k$ abgeschätzt werden kann. Wir erhalten damit eine Gesamtlaufzeit von $\mathcal{O}(f(k) \cdot p(n, m)) = \mathcal{O}(k \cdot n \cdot m)$.

c) Das *HITTING-SET*-Problem ist ein Spezialfall vom *c-HITTING-SET*-Problem, nämlich dann, wenn $c \geq n$ gilt. Dann nämlich kann jede Teilmenge B_i eine beliebige Teilmenge von A sein. Da *HITTING-SET* NP-vollständig ist, ist somit auch die Familie der schwereren Probleme, *c-HITTING-SET*, NP-vollständig.

Aufgabe 2

a) Damit alle Kanten inzident zu einem Knoten der Überdeckung C sind, muss für alle $v \in D$ gelten: entweder $v \in C$ oder $\forall u \in N(v) : u \in D$. Da die Schranke der Knotenüberdeckung $k < d(v) \forall v \in D$ ist, muss $v \in C$ gelten, da sonst alle Nachbarn von v aufgenommen werden müssten und $|N(v)| = d(v) > k$ gilt.

Folglich müssen alle Knoten $v \in D$ auch Teil einer Knotenüberdeckung sein. Gilt nun $|D| > k$, dann müsste jede mögliche Knotenüberdeckung mehr als k Knoten enthalten. Daher kann es keine solche für die Schranke k geben.

b) Wie in Teilaufgabe a) gezeigt wurde, gilt $v \in D \rightarrow v \in C$ für jede Knotenüberdeckung C. Diese Knoten kann man nun aus dem Graphen entfernen, da sie auf jeden Fall in der Knotenüberdeckung enthalten sein und daher nicht weiter betrachtet werden müssen. Entsprechendes gilt für zu ihnen inzidente Kanten, da diese auf jeden Fall überdeckt sind. Der so entstandene Graph kann isolierte Knoten enthalten, die jedoch ebenfalls nicht weiter betrachtet werden müssen, da sie in keiner möglichen Knotenüberdeckung enthalten sein müssen (alle ihre Kanten wurden schon von Knoten aus D überdeckt). Daher können sie auch entfernt werden.

   Der so entstandene Graph G' enthält also alle noch nicht überdeckten Kanten. Genau dann, wenn in ihm eine Knotenüberdeckung C' mit maximal $k' = k - |D|$ gefunden werden kann, dann existiert auch für den Graphen G eine Knotenüberdeckung $C = C' \cup D$, da durch diese wie gesehen alle Kanten überdeckt werden.

c) Damit G eine Knotenüberdeckung der Größe max. k haben kann, muss wie gezeigt eine Knotenüberdeckung von G' mit maximal k' Knoten existieren. Der maximale Grad dieser Knoten ist k bedingt durch die Konstruktion von G'. Daher können so maximal $k'+k'*k = k'*(k+1)$ Knoten inzident zu überdeckten Kanten in einer beliebigen Knotenüberdeckung sein. Enthält der Graph G' mehr als $k'*(k+1)$ Knoten, so müssen Knoten existieren, die inzident zu nicht überdeckten Kanten sind. Dann existiert keine Knotenüberdeckung von G' und folglich auch keine von G.

d) **Die Laufzeit des Preprocessings liegt in $\mathcal{O}(|G|)$:** Im 1. Schritt werden alle Knoten durchlaufen und deren Grad bestimmt ($\mathcal{O}(|G|)$ bei der Verwendung von Adjazenzlisten) und anschließend alle Knoten mit Grad größer k und inzidente Kanten gelöscht (ebenfalls einmaliges Durchlaufen und damit ebenfalls in $\mathcal{O}(|G|)$). Im 2. Schritt werden die Knoten von dem entstandenen Graphen gezählt, was offensichtlich ebenfalls in $\mathcal{O}(|G|)$ liegt. Damit ergibt sich auch eine Gesamtlaufzeit von $\mathcal{O}(|G|)$ für das Preprocessing.

   Die Laufzeit von Schritt 3 liegt in $\mathcal{O}(2^k k^2)$: In Kapitel 10.1. wurde ein Verfahren entwickelt, das in Laufzeit $\mathcal{O}(2^k n)$ eine Knotenüberdeckung für n Knoten berechnet. Da durch das Preprocessing in Schritt 2 zugesichert wird, dass weniger als $k' * (k + 1)$ Knoten im zu betrachtenden

Graphen G' existieren (also eine Knotenzahl die in $\mathcal{O}(k^2)$ liegt), kann der Algorithmus eine Knotenüberdeckung in $\mathcal{O}(2^k k^2)$ berechnen.