Karlsruher Institut für Technologie

Lösung zum Übungsblatt 2

Abgabe bis 20. November 2024

Aufgabe 1: Weighted Vertex Cover — 5 Punkte

Gegeben sei ein ungerichteter Graph G mit Kostenfunktion $c : V \to \mathbb{N}$ auf den Knoten. Beim Weighted Vertex Cover-Problem ist ein Vertex Cover von G mit minimalen Gesamtkosten gesucht. In dieser Aufgabe betrachten wir nur Bäume. Gib ein dynamisches Programm an, das für einen Baum die minimalen Kosten eines Vertex Covers bestimmt. Gib außerdem die Laufzeit für die Berechnung des DPs an.

Lösung 1:

Wir wurzeln den gegebenen Baum T zunächst beliebig. Den Teilbaum unter Knoten v nennen wir T_v. Für jeden Knoten v möchten wir die minimalen Kosten eines Vertex Covers von T_v bestimmen. Dabei unterscheiden wir die Fälle, ob v im Vertex Cover enthalten ist oder nicht. Im DP speichern wir also für jeden Knoten v die beiden Werte $\mathrm{vc}[v,0]$ (kostenminimales VC ohne v in T_v) und $\mathrm{vc}[v,1]$ (kostenminimales VC mit v in T_v). Die minimalen Kosten eines VC von T lassen sich dann an der Wurzel r als $\min\{\mathrm{vc}[r,0], \mathrm{vc}[r,1]\}$ ablesen.

Die Werte berechnen wir, beginnend bei den Blättern, wie folgt: Für ein Blatt v ist $\mathrm{vc}[v,0] = \infty$ und $\mathrm{vc}[v,1] = c(v)$. Für einen inneren Knoten v gilt: Wählen wir v *nicht* in einem VC in T_v, dann müssen die Kanten von v zu den Kindern von v durch die Kinder abgedeckt werden. Das heißt, in einem VC ohne v sind alle Kinder von v enthalten. Wählen wir andererseits v in einem VC in T_v, dann sind alle Kanten von v zu den Kindern von v bereits durch v abgedeckt. Wir können also für jedes Kind u von v ein kostenminimales VC für T_u wählen (mit oder ohne u). Damit gilt

$$\mathrm{vc}[v,0] = \sum_{u \in \mathrm{child}(v)} \mathrm{vc}[u,1]$$

und

$$\mathrm{vc}[v,1] = c(v) + \sum_{u \in \mathrm{child}(v)} \min_{x \in \{0,1\}} \mathrm{vc}[u,x]$$

. Die Bestimmung des DP geht in Linearzeit, da jede Kante nur konstant oft betrachtet wird.

Aufgabe 2: 3-Hitting Set — 10 Punkte

Eine Instanz von 3-Hitting Set besteht aus einer Familie von Mengen $\{S_1, \ldots, S_n\}$ mit $|S_i| \leq 3$ über einer Grundmenge U (also $S_i \subseteq U$). Ziel ist es zu entscheiden, ob es eine Menge $H \subseteq U$ mit $|H| \leq k$

gibt, sodass $H \cap S_i \neq \emptyset$ für alle S_i. Gib einen beschränkten Suchbaum an, der einen Algorithmus mit Laufzeit $2{,}562^k \cdot n^{O(1)}$ für dieses Problem liefert.

Lösung 2:

3-Hitting Set lässt sich mithilfe eines Suchbaumes mit einem dominierenden Verzweigungsvektor von (1,2,2,2,2) lösen. Hierfür definieren wir folgende Reduktionsregeln und Verzweigungsregeln.

I. Reduktionsregel: falls eine ein-elementige Menge existiert, nehme ihr Element in die Lösung H auf.

II. Reduktionsregel: falls eine Menge existiert, die keine Elemente mit einer anderen Menge gemeinsam hat, nehme ein beliebiges Element dieser Menge in H auf.

III. Falls eine zwei-elementige Menge existiert, wähle eines der beiden Elemente. Dies resultiert in einem Verzweigungsvektor von (1,1).

IV. Falls zwei Mengen S_a, S_b mit $|S_a \cap S_b| = 2$ existieren, wähle entweder eines der beiden gemeinsamen Elemente oder die beiden verschiedenen Elemente. Dies resultiert in einem Verzweigungsvektor von (1,1,2).

V. Falls zwei Mengen S_a, S_b mit $|S_a \cap S_b| = 1$ existieren, wähle entweder das gemeinsame Element oder eine Kombination der jeweils zwei verschiedenen Elemente. Dies resultiert in einem Verzweigungsvektor von (1,2,2,2,2).

Nach Anwendung der ersten drei Regeln sind nur noch drei-elementige Mengen übrig, die sich Elemente teilen. Die letzten zwei Regeln behandeln dann die verbleibenden Fälle. Der Verzweigungsvektor (1,2,2,2,2) resultiert in einer Baumgröße von $2{,}562^k$. Also lässt sich 3-Hitting Set in $2{,}562^k \cdot n^{O(1)}$ entscheiden.

Aufgabe 3: d-Hitting Set **10 Punkte**

Eine Instanz von d-Hitting Set besteht aus einer Familie von Mengen $\{S_1, \ldots, S_n\}$ mit $|S_i| \leq d$ über einer Grundmenge U (also $S_i \subseteq U$). Ziel ist es zu entscheiden, ob es eine Menge $H \subseteq U$ mit $|H| \leq k$ gibt, sodass $H \cap S_i \neq \emptyset$ für alle S_i. Gib einen Algorithmus mit Laufzeit $(d - 0{,}658)^k \cdot n^{O(1)}$ für diese Problem an.

Hinweis: Benutze iterative Kompression, um zunächst einen Algorithmus mit Laufzeit $2{,}342^k \cdot n^{O(1)}$ für 3-Hitting Set zu erhalten.

Lösung 3:

Wir definieren d-Hitting Set Compression und Disjoint d-Hitting Set analog wie in der Vorlesung gezeigt. Das heißt, bei Disjoint d-Hitting Set ist eine Familie an Mengen $\{S_1, \ldots, S_n\}$

mit $|S_i| \leq d$ über einer Grundmenge U sowie ein Hitting Set H der Größe $k+1$ gegeben. Es soll entschieden werden, ob es ein Hitting Set X der Größe k gibt, sodass $X \cap H = \emptyset$. Ähnlich ist bei d-Hitting Set Compression eine d-Hitting Set Instanz mit einer Lösung H der Größe $k+1$ gegeben und es soll entschieden werden, ob es eine Lösung X der Größe k gibt.

Wir stellen fest, dass wir das Kompressionsproblem lösen können, indem wir das Disjoint Problem auf jeder möglichen Teilmenge der gegebenen Lösung H ausführen. Außerdem läst sich das Grundproblem lösen, indem die Instanz auf eine k-elementige Teilmenge des Universums eingeschränkt wird. Die Instanz kann dann iterativ um ein Element erweitert werden, das einer Lösung der Größe k hinzugefügt wird, woraufhin das Kompressionsproblem gelöst werden muss.

Wir zeigen nun per Induktion über d, dass d-Hitting Set in $O((d-0.658)^k \cdot n^{c+d})$ Zeit gelöst werden kann. Als Induktionsanfang betrachten wir $d = 2$. Hier besteht jede Menge aus zwei Elementen, also ist das Problem äquivalent zu Vertex Cover (Mengen sind Kanten, die Grundmenge ist die Knotenmenge). Mit dem Algorithmus aus der Vorlesung können wir 2-Hitting Set in $O(1.342^k \cdot n^c)$ lösen.

Wir zeigen nun, dass auch $(d+1)$-Hitting Set in $O((d+1-0{,}658)^k \cdot n^{c+d+1})$ Zeit lösbar ist, falls wir d-Hitting Set in $O((d-0.658)^k \cdot n^{c+d})$ Zeit lösen können. Wir betrachten hierfür das Disjoint Problem, d.h. wir nehmen an eine Lösung H der Größe $k+1$ zu erhalten und eine disjunkte Lösung X der Größe k zu suchen. H schneidet alle Mengen in mindestens einem Element, welches nicht mehr gewählt werden darf. Das heißt, dass für jede Menge nur noch $d-1$ Elemente gewählt werden dürfen. Somit ist Disjoint $d+1$-Hitting Set äquivalent zu d-Hitting Set, welches nach Induktionsvoraussetzung in $O((d-0.658)^k \cdot n^{c+d})$ Zeit gelöst werden kann. Nach Vorlesung ergibt das eine Laufzeit von $O((k+1)(d+1-0{,}658)^k \cdot n^{c+d})$ für $(d+1)$-Hitting Set. Wenn wir $(k+1)$ als n abschätzen, dann ergibt sich daraus $O((d+1-0{,}658)^k \cdot n^{c+d+1})$. Da d als konstant angenommen wird, ist das die Laufzeit, die zu zeigen war.

Aufgabe 4: Vertex Cover mit beschränkten Suchbäumen 5 Punkte + 2 Bonus-Punkte

In dieser Aufgabe sollst du in einer Programmiersprache deiner Wahl ein Programm implementieren, das einen Graphen einliest und dazu ein minimales Vertex Cover berechnet. Nutze dazu die Verfahren zu beschränkten Suchbäumen aus Vorlesung 3. Du darfst aber gerne auch weitere Regeln implementieren und Optimierungen machen.

Beschreibe in der PDF-Abgabe, welche Verfahren du implementiert hast und was du noch weiter optimiert hast. Gib außerdem in der PDF-Abgabe für jeden Graphen an, welche Größe dein gefundenes Vertex Cover hat. Gib zusätzlich den Quellcode sowie deine gefundenen Lösungen (im unten beschriebenen Format) als eine ZIP-Datei ab.

Für jedes minimale Vertex Cover kannst du einen Punkt bekommen.

Dateiformat Eingabe: In der ersten Zeile stehen n und m, die Anzahl von Knoten und Kanten des Graphen. In den nächsten m Zeilen sind jeweils zwei Knoten angegeben, die durch eine ungerichtete Kante verbunden sind. Dabei sind die Knoten von 1 bis n durchnummeriert.

Dateiformat Ausgabe: In der ersten Zeile steht die Lösungsgröße k. In den folgenden k Zeilen steht jeweils die ID eines Knotens, der Teil des Covers ist.

Hinweis: Mithilfe der Datei `validator.py` kannst du verifizieren, ob ein Vertex Cover gültig ist. Minimalität wird dabei nicht geprüft. Bitte prüfe selbst vor der Abgabe, ob deine gefundenen Vertex Cover gültig sind.

Lösung 4:

Graph	**Minimales Vertex Cover**
Graph 0	35
Graph 1	119
Graph 2	102
Graph 3	81
Graph 4	72
Graph 5	80
Graph 6	112