Technische Universität Wien
Institut für Computergraphik und Algorithmen
Algorithms and Complexity Group

186.815 Algorithmen und Datenstrukturen 2 VU 3.0

1.Übungstest SS 2016

23. Juni 2016

Machen Sie die folgenden Angaben bitte in deutlicher Blockschrift:

Nachname: | Vorname:

Matrikelnummer: | Unterschrift:

Legen Sie während der Prüfung Ihren Ausweis für Studierende vor sich auf das Pult.

Sie dürfen die Lösungen nur auf die Angabeblätter schreiben, die Sie von der Aufsicht erhalten. Es ist nicht zulässig, eventuell mitgebrachtes eigenes Papier zu verwenden. Benutzen Sie bitte dokumentenechte Schreibgeräte (keine Bleistifte!).

Die Verwendung von Taschenrechnern, Mobiltelefonen, Tablets, Digitalkameras, Skripten, Büchern, Mitschriften, Ausarbeitungen oder vergleichbaren Hilfsmitteln ist unzulässig.

	A1:	A2:	A3:	Summe:
Erreichbare Punkte:	17	20	13	50
Erreichte Punkte:				

Viel Erfolg!

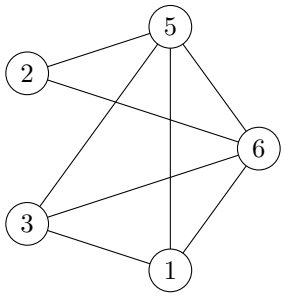
Aufgabe A1: Dynamische Programmierung **(17 Punkte)**

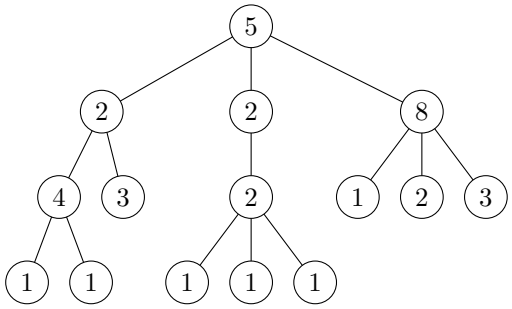
Das Minimum-Weight Vertex Cover Problem sei wie folgt definiert.

Problem (**Minimum-Weight Vertex Cover**). *Gegeben sei ein schlichter Graph* $G = (V, E)$ *mit positiven Knotengewichten* $w_v \in \mathbb{R}^+$ *für alle Knoten* $v \in V$. *Gesucht ist ein Vertex Cover* $V' \subset V$ *mit minimalem Gewicht, d.h. für jede Kante* $(u, v) \in E$ *gilt* $\{u, v\} \cap V' \neq \emptyset$ *und* $\sum_{v \in V'} w_v$ *ist minimal über alle möglichen Vertex Cover von* G.

a) (8 Punkte)

Markieren Sie in den beiden angegebenen Graphen jeweils ein Vertex Cover mit minimalem Gewicht. Die Werte in den Knoten entsprechen dabei den Knotengewichten.





b) (9 Punkte)

Betrachten Sie nun den Fall, dass der gegebene Graph ein Baum T ist. Im Folgenden ist ein Gerüst für einen Algorithmus in Pseudocode gegeben, der das Minimum-Weight Vertex Cover Problem auf Bäumen mittels dynamischer Programmierung lösen soll.

Die Struktur entspricht dem aus der Vorlesung bekannten Algorithmus zur Berechnung eines Maximum-Weight Independent Set auf Bäumen. Für jeden Knoten $u \in V$ (mit Gewicht w_u) sollen die Tabelleneinträge $M_{\text{in}}[u]$ und $M_{\text{out}}[u]$ ausgefüllt werden. Am Ende enthält $M_{\text{in}}[u]$ den besten Wert für eine Lösung des Teilbaums T_u mit u als Wurzel, so dass u Teil der Lösung ist. Äquivalent enthält $M_{out}[u]$ den besten Wert einer Lösung für T_u, aber unter der Annahme, dass u *nicht* Teil der Lösung ist.

Am Ende soll der Algorithmus das minimale Gewicht eines Vertex Cover von T retournieren. Ergänzen sie die markierten Zeilen so, dass der Algorithmus korrekt ist. Definieren Sie (falls nötig) eigene Notation.

`Minimum-Weight-Vertex-Cover-In-A-Tree`(Baum $T = (V, E)$):

Wähle eine Wurzel $r \in V$ aus
foreach *Knoten u in T in Postorder* **do**
  if *u ist ein Blatt*
    $M_{\text{in}}[u] \leftarrow$ [ ]
    $M_{\text{out}}[u] \leftarrow$ [ ]
  else `//` *u* `ist kein Blatt`
    $M_{\text{in}}[u] \leftarrow$ [ ]
    $M_{\text{out}}[u] \leftarrow$ [ ]
return [ ]

Aufgabe A2: Approximation und Polynomialzeitreduktion (20 Punkte)

Betrachten Sie erneut das Minimum-Weight Vertex Cover Problem aus Aufgabe 1.

Problem (**Minimum-Weight Vertex Cover**). *Gegeben sei ein schlichter Graph* $G = (V, E)$ *mit positiven Knotengewichten* $w_v \in \mathbb{R}^+$ *für alle Knoten* $v \in V$. *Gesucht ist ein Vertex Cover* $V' \subset V$ *mit minimalem Gewicht, d.h. für jede Kante* $(u, v) \in E$ *gilt* $\{u, v\} \cap V' \neq \emptyset$ *und* $\sum_{v \in V'} w_v$ *ist minimal über alle möglichen Vertex Cover von* G.

a) (10 Punkte)

Nehmen Sie nun zusätzlich an, dass in $G = (V, E)$ alle Knotengewichte w_v für $v \in V$ nur Werte aus der Menge $\{1, 2, 3\}$ sind. Angenommen Sie wenden den unten angegebenen Approximationsalgorithmus `Approx-Vertex-Cover` aus der Vorlesung unverändert auf den gewichteten Graphen G an. Besitzt dieser Algorithmus dann immer noch eine konstante Gütegarantie für das Minimum-Weight Vertex Cover Problem auf G? Begründen Sie Ihre Antwort und geben Sie ggf. die erzielbare Gütegarantie an.

```
Approx-Vertex-Cover(Graph G = (V, E)):
    C ← ∅
    while E ≠ ∅
    |   wähle eine beliebige Kante (u, v) ∈ E
    |   C ← C ∪ {u, v}
    |   entferne aus E alle Kanten, die inzident zu u oder v sind
    return C
```

b) (4 Punkte)

Welche der folgenden Aussagen über NP-Vollständigkeit sind korrekt?

Kreuzen Sie genau diejenigen Antworten an, die eine wahre Aussage darstellen. Bei einem Fehler (falsches oder fehlendes Kreuz) wird noch 1 Punkt vergeben, bei zwei oder mehr Fehlern werden 0 Punkte für diese Aufgabe vergeben.

- ☐ Wenn es einen polynomiellen Algorithmus für Independent Set gibt, gilt P=NP.
- ☐ Wenn $\mathsf{P} \neq \mathsf{NP}$ gilt, dann gibt es keinen polynomiellen Algorithmus für Independent Set auf Bäumen.
- ☐ Für $k = 12$ kann man in Zeit $O(n)$ bestimmen, ob ein schlichter Graph G mit n Knoten ein Vertex Cover der Größe $\leq k$ besitzt.
- ☐ Wenn $\mathsf{P} \neq \mathsf{NP}$ gilt, dann existiert kein polynomieller Algorithmus für das Problem Set Cover.

c) (4 Punkte)

Seien X, Y und Z drei Ja/Nein-Probleme in NP. Weiters seien R_{XY} ein Reduktionsalgorithmus von X auf Y mit Laufzeit $O(n^3)$ und R_{YZ} ein Reduktionsalgorithmus von Y auf Z mit Laufzeit $O(n^2)$.

Welche der folgenden Aussagen sind korrekt?

Kreuzen Sie genau diejenigen Antworten an, die eine wahre Aussage darstellen. Bei einem Fehler (falsches oder fehlendes Kreuz) wird noch 1 Punkt vergeben, bei zwei oder mehr Fehlern werden 0 Punkte für diese Aufgabe vergeben.

- ☐ Wenn Z effizient lösbar ist, dann ist auch X effizient lösbar.
- ☐ Wenn X effizient lösbar ist, dann ist auch Z effizient lösbar.
- ☐ Wenn Y NP-vollständig ist, dann ist auch X NP-vollständig.
- ☐ Wenn Y NP-vollständig ist, dann ist auch Z NP-vollständig.

d) (2 Punkte)

Seien X, Y und Z wieder drei Ja/Nein-Probleme in NP. Weiters seien R_{XY} ein Reduktionsalgorithmus von X auf Y mit Laufzeit $O(n^3)$ und R_{YZ} ein Reduktionsalgorithmus von Y auf Z mit Laufzeit $O(n^2)$.

Nehmen Sie an, Sie kennen einen Algorithmus, der eine Instanz von Z der Größe n in Zeit $O(n^{1.5})$ löst. In welcher asymptotischen Laufzeit können Sie dann eine Instanz von X der Größe n lösen?

Aufgabe A3: Branch and Bound **(13 Punkte)**

Betrachten Sie die folgende Instanz des Rucksackproblems. Die Gewichte und Werte der Gegenstände sind in der Tabelle angegeben. Die Kapazität des Rucksacks beträgt $G = 20$.

Gegenstand	A	B	C	D
Gewicht	8	7	10	3
Wert	64	49	120	18
Verhältnis				

a) (2 Punkte)

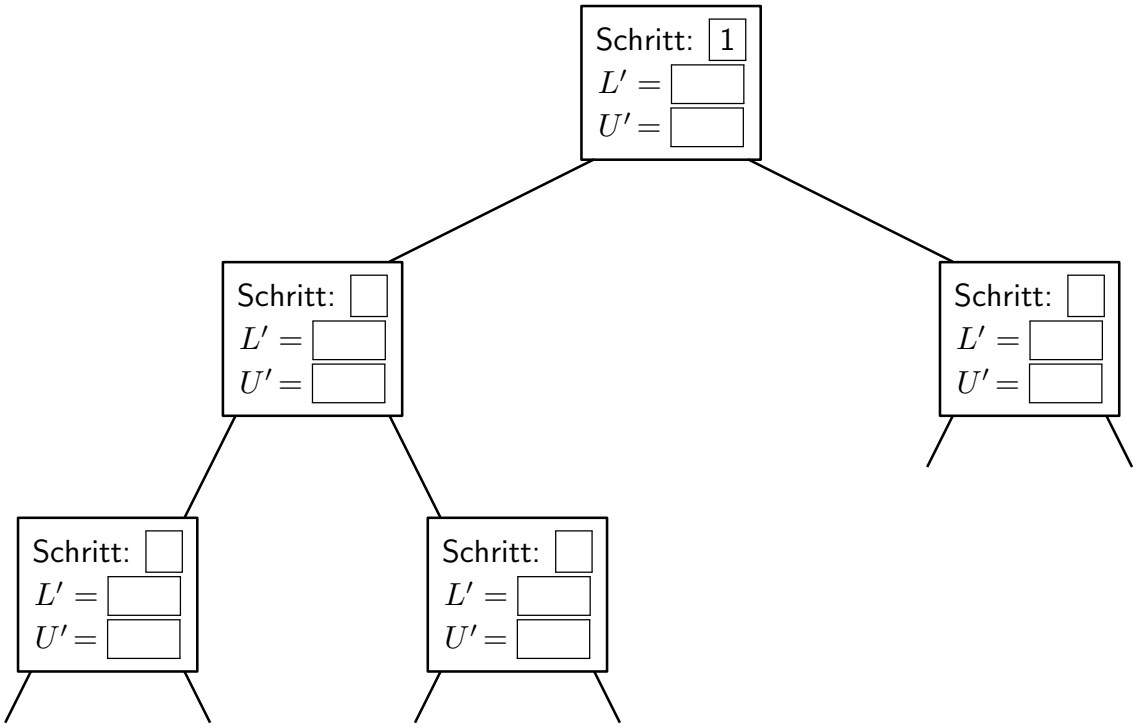
Berechnen Sie die Wert-Gewichts-Verhältnisse aller Gegenstände und tragen Sie diese in obiger Tabelle ein. Geben Sie die Reihenfolge an, in der die Gegenstände betrachtet werden, wenn Sie den Branch-and-Bound Algorithmus der Vorlesung anwenden.

b) (9 Punkte)

Wenden Sie den verbesserten Branch-and-Bound-Algorithmus[1] aus der Vorlesung auf die Instanz an. Nutzen Sie dabei die Depth-first Strategie zur Auswahl des nächsten Teilproblems. Verwenden Sie zur Lösung der Aufgabe den leeren Branch-and-Bound Baum auf der nächsten Seite.

Geben Sie in jedem Knoten an, in welchem Schritt er besucht wird und welchen Wert die zugehörigen unteren und oberen Schranken haben, bzw. markieren Sie, wenn es keine gültige Lösung geben kann. Geben Sie an den Kanten klar an, welche Branching-Entscheidung getroffen wird. Beachten Sie, dass der Baum nach Bedarf erweitert werden muss und zeichnen Sie die zusätzlich benötigten Kanten und Knoten.

[1] Zur Erinnerung: Die untere Schranke wird durch einen Greedy-Algorithmus berechnet, für die obere Schranke können Gegenstände auch partiell eingepackt werden. Die Reihenfolge, in der die Gegenstände betrachtet werden, ergibt sich aus den Wert-Gewichts-Verhältnissen (s. Aufgabe 3(a)).



c) (2 Punkte)

Geben Sie die optimale Lösung und den zugehörigen Lösungswert an.