TU Wien
Institut für Logic and Computation
Algorithms and Complexity Group

186.866 Algorithmen und Datenstrukturen VU 8.0

2. Test, 2021S

30. Juni 2021

Gruppe B

Es gelten die in TUWEL verkündeten Regelungen.

Füllen Sie nachfolgende Felder des Deckblattes aus bzw. fertigen Sie ein eigenes Deckblatt mit diesen Informationen an. Fügen Sie das ausgefüllte Deckblatt als erste Seite von Ihrem PDF mit den Lösungen hinzu.

Zoom-Meeting: Breakout-Raum:

Nachname: Vorname:

Matrikelnummer: Unterschrift:

Ausweiskopie:

	B1	B2	B3	B4	B5	Summe
Erreichbare Punkte:	19	21	20	20	20	100
Erreichte Punkte:						

Viel Erfolg!

Aufgabe B1: P und NP, Spezialfälle **(19 Punkte)**

a) (15 Punkte) Seien A, B, C Ja/Nein-Probleme und n die Eingabegröße. Nehmen Sie an, es gibt

- eine Reduktion von SAT nach A in Zeit n,
- eine Reduktion von A nach C in Zeit n,
- eine Reduktion von A nach B in Zeit n^3,
- eine Reduktion von C nach B in Zeit 5^n,
- eine Reduktion von B nach HAM-CYCLE in Zeit n^2,

(i) Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von A auf HAM-CYCLE in O-Notation an, die sich aus diesen Annahmen ableiten lässt und begründen Sie Ihre Antwort.

(ii) Nehmen Sie jetzt zusätzlich an, dass bei der Reduktion von A nach B die erzeugten Instanzen für das Problem B maximal $2n$ groß sind. Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von A auf HAM-CYCLE in O-Notation an, die sich aus diesen Annahmen ableiten lässt und begründen Sie Ihre Antwort.

(iii) Welche der Probleme A, B, C, SAT, HAM-CYCLE sind garantiert NP-schwer?

(iv) Geben Sie für die folgenden Aussagen an, ob diese unter den obigen Annahmen sicher wahr (w), sicher falsch (f), oder keine Aussage möglich ist, wir bezeichnen das dann als unbestimmt (u).

(Q1) Wenn es einen Polynomialzeit-Algorithmus für C gibt, dann kann auch A in Polynomialzeit gelöst werden.

☐ Wahr ☐ Falsch ☐ unbestimmt

(Q2) Es gibt eine Polynomialzeitreduktion von C auf Ham-Cycle.

☐ Wahr ☐ Falsch ☐ unbestimmt

b) (4 Punkte) Ein Studierender liefert folgenden „Beweis“ (bestehend aus den nachfolgenden vier Aussagen), dass P = NP gilt. Markieren Sie jede Zeile mit einer **falschen** Aussage und beschreiben Sie die Fehler des Studenten in einem Satz.

(L1) ☐ Wir können in Polynomialzeit 2-Färbbarkeit lösen.

(L2) ☐ Wir können in Polynomialzeit 2-Färbbarkeit auf 3-Färbbarkeit reduzieren.

(L3) ☐ Außerdem kann man jedes Problem in NP auf 3-Färbbarkeit reduzieren.

(L4) ☐ Die vorhergehenden Aussagen zusammen implizieren, dass P = NP.

Fehler des Studierenden:

Aufgabe B2: Branch-and-Bound **(21 Punkte)**

a) (5 Punkte) Betrachten Sie die folgende Instanz des Rucksackproblems. Die Gewichte und Werte der Gegenstände sind in der Tabelle angegeben. Die Kapazität des Rucksacks beträgt $G = 10$.

Gegenstand	1	2	3	4
Gewicht	4	3	7	5
Wert	40	12	42	25
Verhältnis				

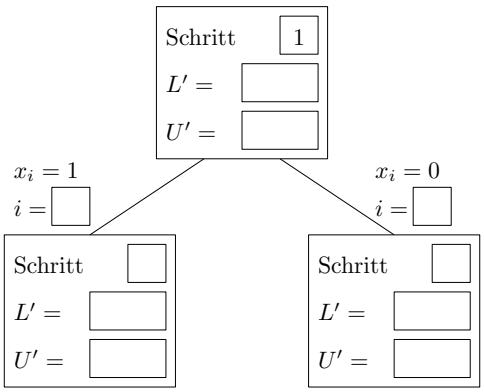
Berechnen Sie die Wert-Gewichts-Verhältnisse aller Gegenstände und tragen Sie diese in obiger Tabelle ein. Geben Sie die Reihenfolge an, in der die Gegenstände betrachtet werden, wenn Sie den Branch-and-Bound Algorithmus der Vorlesung anwenden.

Reihenfolge:

b) (10 Punkte) Wenden Sie den **verbesserten** Branch-and-Bound-Algorithmus aus der Vorlesung auf die Instanz aus Unteraufgabe a) an. Nutzen Sie dabei die **Best-first Strategie** zur Auswahl des nächsten Teilproblems. Ergänzen Sie zur Lösung der Aufgabe den untenstehenden begonnenen Branch-and-Bound Baum.

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. Erweitern Sie den Baum nach Bedarf und zeichnen Sie die zusätzlich benötigten Kanten und Knoten ein. Geben Sie abschließend die optimale Lösung und den Knoten an, in welchem diese gefunden wurde.

[siehe nächste Seite]


Schritt
1
$L' =$
$U' =$
$x_i = 1$
$i =$
Schritt
$L' =$
$U' =$
$x_i = 0$
$i =$
Schritt
$L' =$
$U' =$

c) (4 Punkte) Welche Aussagen für Branch-and-Bound Verfahren sind korrekt? Kreuzen Sie Zutreffendes an.

(alles korrekt: 4 Punkte, ein Fehler: 2 Punkte, sonst / kein Kreuz: 0 Punkte)

☐ Eine korrekte Lösung kann nur gefunden werden, wenn alle Teilprobleme betrachtet werden.

☐ Die Wahl der Heuristiken für U' und L' ist entscheidend für die Effizienz.

☐ Die Reihenfolge der Auswahl des nächsten Teilproblems ist für die Korrektheit von Bedeutung.

☐ Depth-first und Best-first können gemeinsam eingesetzt werden, um die Vorteile zu kombinieren.

d) (2 Punkte) Angenommen die Laufzeit für das Auswerten eines Teilproblems ist vernachlässigbar, jedes Teilproblem erzeugt vier neue Teilprobleme und n ist die maximale Suchtiefe. Ist die worst-case Laufzeit des Branch-and-Bound Verfahrens in $O(2^n)$?

☐ Ja ☐ Nein

Falls nein, dann geben Sie die worst-case Laufzeit an: ☐

Aufgabe B3: Dynamisches Programmieren **(20 Punkte)**

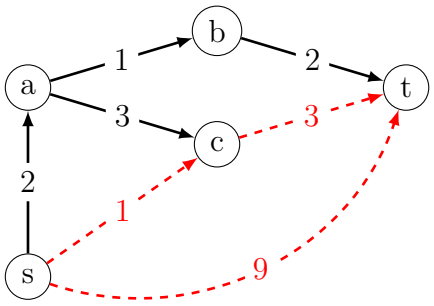
Wir definieren folgendes EXTENDED SHORTEST PATH Problem: Gegeben ist ein gerichteter Graph $G = (V, E)$ mit einem positiven Kantengewicht c_{uv} für jede Kante $(u, v) \in E$ und zwei speziellen Knoten s und t. Die Kantenmenge E besteht aus roten Kanten E_r und schwarzen Kanten E_s, d.h. $E = E_r \cup E_s$ und $E_r \cap E_s = \emptyset$. Ein *guter* Pfad ist ein Pfad, welcher genau eine rote Kante benutzt. Die Aufgabe ist es, den kürzesten guten Pfad von s nach t zu finden.

In dieser Aufgabe wird das Problem EXTENDED SHORTEST PATH mittels dynamischer Programmierung durch eine Erweiterung des aus der Vorlesung bekannten Bellman-Ford Algorithmus gelöst.

Dabei werden die folgenden beiden Arrays dynamisch berechnet:

- `NoRed`, welches die Längen der Pfade von allen Knoten $x \in V$ zu t beinhaltet, die keine roten Kanten enthalten.
- `Good`, welches die Längen der Pfade von allen Knoten $x \in V$ zu t beinhaltet, mit genau einer roten Kante.

a) (12 Punkte) Gegeben sei die folgende Probleminstanz. Die roten Kanten sind strichliert gezeichnet.



(i) Vervollständigen Sie die beiden Arrays `NoRed` und `Good` für diese Instanz. Genau wie in der Vorlesung, repräsentieren die Spalten 0-4 die Längen der Pfade mit der entsprechenden Anzahl an Kanten.

`NoRed`	0	1	2	3	4
t	0	0	0	0	0
a	∞	∞			
b	∞	2			
c	∞	∞			
s	∞	∞			

`Good`	0	1	2	3	4
t	∞	∞	∞	∞	∞
a	∞	∞			
b	∞	∞			
c	∞	3			
s	∞	9			

(ii) Was ist der kürzeste gute Pfad von s nach t in `Good`?

b) (8 Punkte) Nun ist ein Algorithmus für das Problem Extended Shortest Path zu finden. Schreiben Sie die korrekten Werte in die korrekten Formeln, um den Eintrag zu berechnen.

```
ExtShortPath(G,s,t):
foreach v ∈ V do
    NoRed[0, v] ← ∞
NoRed[0, t] ← 0
for i ← 1 to n − 1
    foreach v ∈ V do
        NoRed[i, v] ← NoRed[i − 1, v]
    foreach schwarze Kante (v, w) ∈ E_s do
        NoRed[i, v] ← min{NoRed[i, v], c_vw + NoRed[i − 1, w]}
foreach v ∈ V do
    Good[0, v] ← ∞
for i ← 1 to n − 1
    foreach v ∈ V do
        Good[i, v] ← Good[i − 1, v]
    foreach schwarze Kante (v, w) ∈ E_s do
        (i) Good[i, v] ← min{                    }
    foreach rote Kante (v, w) ∈ E_r do
        (ii) Good[i, v] ← min{                   }
```

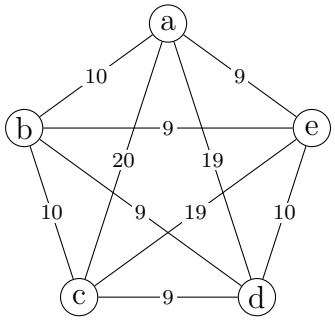
Falls mehr Platz benötigt:

(i) $\texttt{Good}[i, v] \leftarrow \min\{$

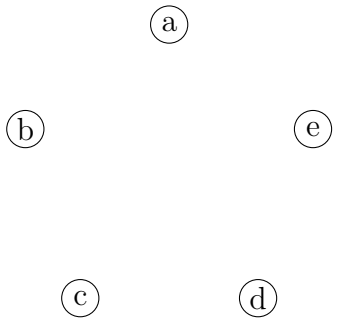
(ii) $\texttt{Good}[i, v] \leftarrow \min\{$

Aufgabe B4: Approximationsalgorithmen **(20 Punkte)**

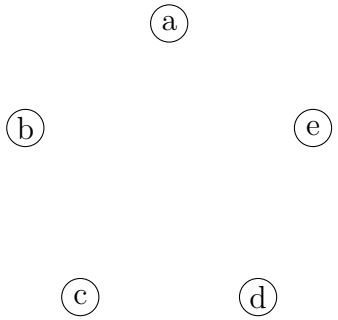
a) (12 Punkte) Betrachten Sie folgenden symmetrischen gewichteten Graphen für ein TSP Problem und wenden Sie die Spanning-Tree-Heuristik an.



(i) (5 Punkte) Zeichnen Sie eine Eulertour, die von der Spanning-Tree-Heuristik berechnet wird ein (oder geben Sie diese als Tupel an). Wie lang ist diese Eulertour?



(ii) (5 Punkte) Wandeln Sie die Eulertour in eine Lösung für das TSP Problem um und zeichnen Sie diese ein (oder geben Sie diese als Tupel an). Wie lang ist diese Tour?



(iii) (2 Punkte) Nehmen Sie an, dass in einem Graphen die optimale Tour die Länge 6 km hat. In welchem Wertebereich ist die Länge einer Lösung, die mittels eines 2-Approximation Algorithmus berechnet wird?

b) (4 Punkte) Für einen Graphen G sei $\text{opt}(G)$ die minimale Größe eines Vertex Covers. Nehmen Sie an, Sie haben einen Algorithmus B, der in Polynomialzeit ein Vertex Cover der Größe höchstens $\text{opt}(G) + 2$ berechnet. Geben Sie die kleinste Zahl x an, für die garantiert gilt: Für alle Graphen G mit $\text{opt}(G) \geq x$ berechnet Algorithmus B eine 1.2-Approximation für Vertex Cover.

c) (4 Punkte) Geben Sie einen (sehr einfachen) Algorithmus C an, der in Zeit $O(n^{21})$ entscheidet, ob ein Graph ein Vertex Cover der Größe 19 oder kleiner besitzt und wenn es ein solches gibt, ein minimales Vertex Cover berechnet.

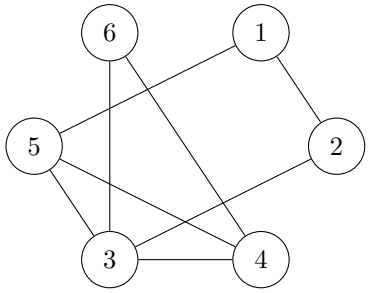
Aufgabe B5: Heuristische Verfahren **(20 Punkte)**

a) (8 Punkte) Gegeben ist eine Konstruktionsheuristik für das Vertex Cover Problem:

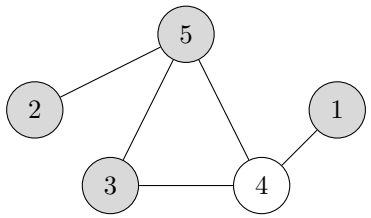
```
ConstructVC(G = (V, E)):
C ← ∅
foreach u ∈ V do
    if u ∉ C then
        foreach v such that (u, v) ∈ E do
            if v ∉ C then
                C ← C ∪ {v}
return C
```

(i) Bestimmen Sie die Laufzeit von `ConstructVC` in Θ-Notation unter der Annahme, dass der Inputgraph G als Adjazenzliste übergeben wird.

(ii) Führen Sie `ConstructVC` mit nachfolgenden Graphen als Input aus. Durchmustern Sie dabei die Knoten in **aufsteigender** Reihenfolge und geben Sie C nach **jeder** Iteration der äußeren foreach-Schleife an.



b) (12 Punkte) Gegeben ist ein Graph $G = (V, E)$ mit Vertex Cover $S = \{1, 2, 3, 5\}$.



(i) Sei N_1 die aus der Vorlesung bekannte naive Nachbarschaftsstruktur für Vertex Cover, also $C' \in N_1(C)$ wenn C' aus C durch Löschen eines einzigen Knotens erzeugt werden kann und noch immer ein Vertex Cover ist. Bestimmen Sie $N_1(S)$.

(ii) Sei N_2 die aus der Vorlesung bekannte *verbesserte* Nachbarschaftsstruktur für Vertex Cover, also $C' \in N_2(C)$ wenn $C' \in N_1(C)$ oder wenn C' durch Hinzufügen eines Knotens von $V \setminus C$ und Entfernen von zwei Knoten aus C gebildet werden kann und noch immer ein Vertex Cover ist. Geben Sie zwei beliebige Elemente aus $N_2(S) \setminus N_1(S)$ an.

(iii) Kann eine lokale Suche angewandt auf G und S mit Nachbarschaftsstruktur N_1 ein globales Optimum finden? Wie sieht es mit N_2 aus? Begründen Sie Ihre Antworten.