

# 2021\_t2\_A

## ⚠ Disclaimer

Alles was hier drinnen steht kann Fehler enthalten! Falls dir etwas auffällt melde dich gerne auf Discord bei mir ([@xmozz](#))

## A1: P und NP, Spezialfälle

Stoff: [9. Polynominalzeitreduktionen](#) und [10. NP-Vollständigkeit Spezialfälle](#)

a)

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

- eine Reduktion von INDEPENDENT SET nach A in Zeit  $n$ ,
- eine Reduktion von A nach B in Zeit  $n$ ,
- eine Reduktion von A nach C in Zeit  $n^3$ ,
- eine Reduktion von B nach C in Zeit  $5^n$ ,
- eine Reduktion von C nach 3-COLOR in Zeit  $n^2$ ,

(i)

(i) Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von A auf 3-COLOR in O-Notation an, die sich aus diesen Annahmen ableiten lässt und begründen Sie Ihre Antwort.

$$\begin{array}{c} A \rightarrow C \\ C \rightarrow 3Color \end{array}$$

$$\implies O((n^3)^2) = O(n^6)$$

(ii)

(ii) Nehmen Sie jetzt zusätzlich an, dass bei der Reduktion von A nach C die erzeugten Instanzen für das Problem C maximal  $2n$  groß sind. Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von A auf 3-COLOR in O-Notation an, die sich aus diesen Annahmen ableiten lässt und begründen Sie Ihre Antwort.

$$O(n^3 + (2n)^2) = O(n^3)$$

(iii)

- (iii) Welche der Probleme A, B, C, INDEPENDENT SET, 3-COLOR sind garantiert NP-schwer?

A und C, weil:

$$\begin{aligned} A &\rightarrow C \\ C &\rightarrow NP - \text{Schwer} \end{aligned}$$

**(iv)**

- (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) Es gibt eine Polynomialzeitreduktion von B auf 3-COLOR.

Wahr  Falsch  unbestimmt

(Q2) Wenn es einen Polynomialzeit-Algorithmus für B gibt, dann kann auch A in Polynomialzeit gelöst werden.

Wahr  Falsch  unbestimmt

**b)**

- 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 jedes Problem in NP auf 3-Färbbarkeit reduziert werden.

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

Fehler des Studierenden:

Er verwechselt die Eigenschaft, dass ein Problem NP-vollständig ist (also dass **alle** Probleme in NP auf es reduziert werden können), mit der Eigenschaft, dass es selbst in P liegt.

## A2: Branch-and-Bound

Stoff: 11. Branch and Bound

a)

- 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	5	7	3
Wert	40	25	42	12
Verhältnis	10	5	6	4

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.

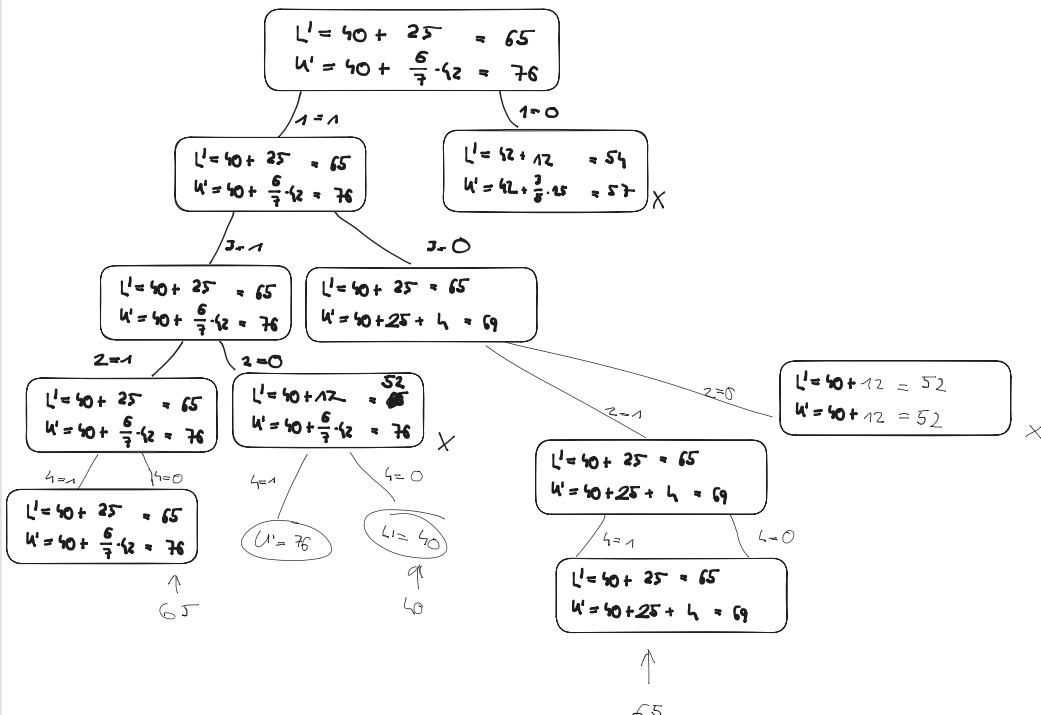
Reihenfolge:

1, 3, 2, 4

b)

- 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.



c)

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)

- Die Reihenfolge der Auswahl des nächsten Teilproblems ist für die Korrektheit von Bedeutung.
- 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.
- Depth-first und Best-first können gemeinsam eingesetzt werden, um die Vorteile zu kombinieren.

d)

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

- Ja
- Nein

Nein sie wäre  $3^n$

## A3: Dynamisches Programmieren

### Stoff: 12. Dynamische Programmierung

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.

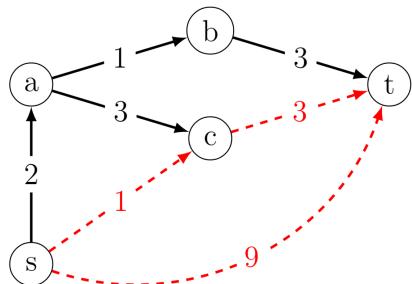
In dieser Aufgabe wird das Problem EXTENDED SHORTEST PATH mittels dynamischer Programmierung durch eine Erweiterung des aus der Vorlesung bekannten Bellman-Ford Algorithmus gelöst.

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)

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



(i)

- (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	$\infty$	$\infty$	4	4	4
b	$\infty$	3	3	3	3
c	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
s	$\infty$	$\infty$	6	6	6

Good	0	1	2	3	4
t	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
a	$\infty$	$\infty$	6	6	6
b	$\infty$	$\infty$	6	6	6
c	$\infty$	3	2	3	3
s	$\infty$	9	7	7	7

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

Das ist  $s \rightarrow a \rightarrow c \rightarrow_{red} t$

b)

- 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  $v \in V$  do
     $\text{NoRed}[0, v] \leftarrow \infty$ 
     $\text{NoRed}[0, t] \leftarrow 0$ 
for  $i \leftarrow 1$  to  $n - 1$ 
    foreach  $v \in V$  do
         $\text{NoRed}[i, v] \leftarrow \text{NoRed}[i - 1, v]$ 
        foreach schwarze Kante  $(v, w) \in E_s$  do
             $\text{NoRed}[i, v] \leftarrow \min\{\text{NoRed}[i, v], c_{vw} + \text{NoRed}[i - 1, w]\}$ 
    foreach  $v \in V$  do
         $\text{Good}[0, v] \leftarrow \infty$ 
    for  $i \leftarrow 1$  to  $n - 1$ 
        foreach  $v \in V$  do
             $\text{Good}[i, v] \leftarrow \text{Good}[i - 1, v]$ 
        foreach schwarze Kante  $(v, w) \in E_s$  do
            (i)  $\text{Good}[i, v] \leftarrow \min\{\text{Good}[i, v], c_{vw} + \text{Good}[i - 1, w]\}$ 
        foreach rote Kante  $(v, w) \in E_r$  do
            (ii)  $\text{Good}[i, v] \leftarrow \min\{\text{Good}[i, v], c_{vw} + \text{NoRed}[i - 1, w]\}$ 

```

(i):  $\min\{\text{Good}[i, v], c_{vw} + \text{Good}[i - 1, w]\}$

(ii):  $\min\{\text{Good}[i, v], c_{vw} + \text{NoRed}[i - 1, w]\}$

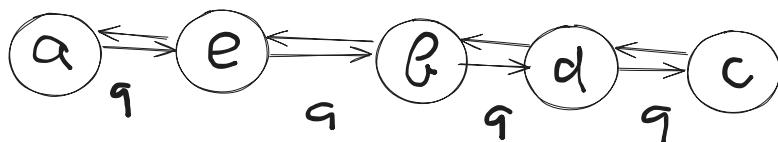
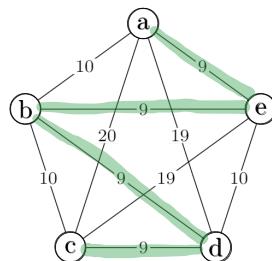
## A4: Approximationsalgorithmen

Stoff: 13. Approximation

a)

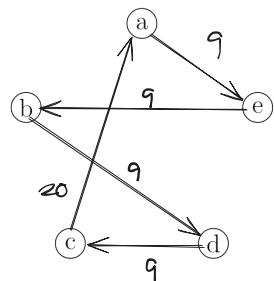
- a) (12 Punkte) Betrachten Sie folgenden symmetrischen gewichteten Graphen für ein TSP Problem und wenden Sie die Spanning-Tree-Heuristik an.

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



$$9 \cdot 8 = 72$$

- (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?



55

(iii)

- (iii) (2 Punkte) Nehmen Sie an, dass in einem Graphen die optimale Tour die Länge 2021 km hat. In welchem Wertebereich ist die Länge einer Lösung, die mittels eines 2-Approximation Algorithmus berechnet wird?

$$2021 \leq x \leq 4042$$

b)

- 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  $\text{opt}(G) + 1$  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 1.1-Approximation für Vertex Cover.

$$\frac{x+1}{x} \leq 1.1$$

$$1 + \frac{1}{x} \leq 1.1$$

$$\frac{1}{x} = 0.1$$

$$\frac{1}{x} = \frac{1}{10}$$

$$x = 10$$

c)

- c) (4 Punkte) Geben Sie einen (sehr einfachen) Algorithmus  $C$  an, der in Zeit  $O(n^{20})$  entscheidet, ob ein Graph ein Vertex Cover der Größe 18 oder kleiner besitzt und wenn es ein solches gibt, ein minimales Vertex Cover berechnet.

Für  $k = 0$  bis 18:

Für jede Menge  $S \subseteq V$  mit  $|S| = k$ :

Prüfe, ob  $S$  ein Vertex Cover ist:

Für jede Kante  $(u, v) \in E$ :

Wenn weder  $u \in S$  noch  $v \in S$ :  $\rightarrow S$  ist kein Vertex Cover

Wenn alle Kanten abgedeckt:  $\rightarrow$  gib  $S$  als minimale Lösung zurück

Gib "Nein" zurück

## A5: Heuristische Verfahren

Stoff: 14. Heuristiken und Lokale Suche

a)

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

```
ConstructVC( $G = (V, E)$ ):
 $C \leftarrow \emptyset$ 
foreach  $u \in V$  do
    if  $u \notin C$  then
        foreach  $v$  such that  $(u, v) \in E$  do
            if  $v \notin C$  then
                 $C \leftarrow C \cup \{v\}$ 
return  $C$ 
```

(i)

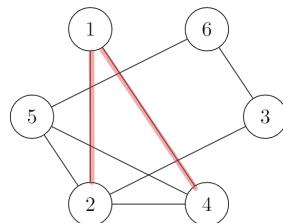
- (i) Bestimmen Sie die Laufzeit von `ConstructVC` in  $\Theta$ -Notation unter der Annahme, dass der Inputgraph  $G$  als Adjazenzliste übergeben wird.

Die Laufzeit ist  $\Theta(n + m)$ , weil:

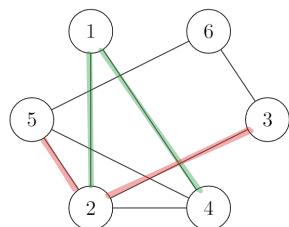
- jeder Knoten höchstens einmal verarbeitet wird,
- jede Kante höchstens zweimal betrachtet wird (einmal von jedem Endpunkt),
- und alle anderen Operationen konstantzeitlich sind.

(ii)

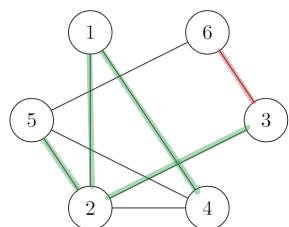
- (ii) Führen Sie `ConstructVC` mit nachfolgenden Graphen als Input aus. Durchmustern Sie dabei die Knoten in **aufsteigender** Reihenfolge und geben Sie  $C$  nach **jeder** Iteration der äußeren `foreach`-Schleife an.



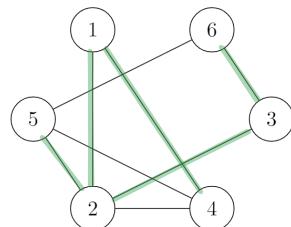
$$C = \{ \}$$



$$C = \{(1,2), (1,6)\}$$



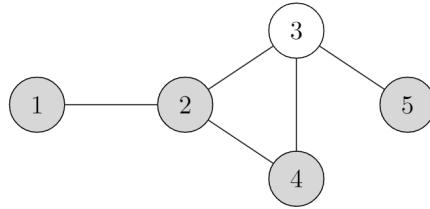
$$C = \{(1,2), (1,6), (2,3), (2,4)\}$$



$$C = \{(1,2), (1,6), (2,3), (2,4), (3,6)\}$$

b)

- b) (12 Punkte) Gegeben ist ein Graph  $G = (V, E)$  mit Vertex Cover  $S = \{1, 2, 4, 5\}$ .



- (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  $N_1(S)$ .

$$N_1(S) = \{\{2, 4, 5\}\}$$

- (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.

$$N_2(S) = \{\{1, 3, 5\}, \{1, 2, 3\}\}$$

- (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.

Nur alleine N1 zu verwenden wird zu keinem globalen Optimum führen, da man 3 braucht um ein globales Optimum zu bekommen. Da man 3 nur mit N1 niemals bekommen wird ist das unmöglich.

Mit N2 allerdings kann man 3 zur Menge hinzufügen und dann dadurch an das Optimum kommen