Karlsruher Institut für Technologie Prof. Dr. Jörn Müller-Quade
Institut für Theoretische Informatik Björn Kaidel, Sebastian Schlag, Sascha Witt

11. Tutorenblatt zu Algorithmen I im SoSe 2017

http://crypto.iti.kit.edu/index.php?id=799
{bjoern.kaidel,sascha.witt}@kit.edu

Aktuelle Inhalte der Vorlesung

- Greedy-Algorithmen
- (I)LPs
- Dynamische Programmierung
- Branch-and-Bound

Diejenigen von euch, die die Union-Find-Datenstruktur in der letzten Woche noch nicht besprochen haben, können das dann diese Woche tun.
Zu Greedy-Algorithmen muss vermutlich nicht mehr viel getan werden, Dijkstra und die MST-Algorithmen sind da ja bereits gute Beispiele.

1 (M)(I)LPs

In der Vorlesung wurden bereits zwei lineare Programme gezeigt. Es ist aber vielleicht sinnvoll, auch ein MILP nochmal gesehen zu haben.
Als einfaches Beispiel könntet ihr sowas zeigen wie in der Nachklausur vom letzten Sommersemester (Aufgabe 5 hier: `https://crypto.iti.kit.edu/fileadmin/User/Lectures/Algorithmen_SS16/loesungsvorschlag_02.pdf`).
Ansonsten: Leider kennen die Studenten quasi noch keine NP-vollständigen Probleme. Daher müsstet ihr ein solches Problem vorher auch kurz einführen. Unser Vorschlag wäre *Vertex Cover* (da es auch auf dem Übungsblatt und in der Übung drankommt – aber nicht als ILP):

Vertex Cover Gegeben ein Graph $G = (V, E)$, finde eine minimale Menge $V' \subseteq V$, sodass für jedes $(u, v) \in E$ gilt, dass $\{u, v\} \cap V' \neq \emptyset$.
Eine ILP-Formulierung für Vertex Cover sieht so aus: Für jeden Knoten $v \in V$ wird eine Variable i_v mit dem Wertebereich $\{0, 1\}$ eingeführt. Diese soll anzeigen, ob v in V' enthalten ist oder nicht. Die Zielfunktion ist also:

$$\min \sum_{v \in V} i_v$$

Die Bedingung, dass alle Kanten abgedeckt sein müssen, lässt sich schreiben als:

$$\forall (u, v) \in E : i_v + i_u \geq 1$$

Somit ergeben sich n Variablen und m Nebenbedingungen.
Ein weiteres Problem, dass intuitiv ist und sich schön als ILP formulieren lässt, ist das *Travelling Salesman Problem*. Dieses wurde auch schon kurz in der letzten Übung vorgestellt (+ Näherungslösung mit MST-Algorithmen). Eine ILP-Formulierung dieses Problems findet ihr z.B. unter
https://en.wikipedia.org/wiki/Travelling_salesman_problem#Integer_linear_programming_formulation

2 Dynamische Programmierung

Aufgabe 1 (*Rucksackproblem,* $2+2$ *Punkte*)

Gegeben seien $n = 5$ Gegenstände mit Gewichten $w_1 = 40,\ w_2 = 18,\ w_3 = 20,\ w_4 = 15,\ w_5 = 10$ und jeweiligen Profiten $p_1 = 90,\ p_2 = 45,\ p_3 = 52,\ p_4 = 42,\ p_5 = 35$. Gesucht ist eine Teilmenge der Gegenstände, die insgesamt das Gewicht $M = 50$ nicht überschreitet und die Summe der zugehörigen Profite maximiert.

a) Finden Sie diese Teilmenge mit Hilfe dynamischer Programmierung. Verwenden Sie dazu wie in der Vorlesung vorgestellt eine Tabelle (Vorlesung vom 06.07., Folie 13).

b) Finden Sie diese Teilmenge mit Hilfe von Branch-and-Bound. Verwenden Sie dazu die Schreibweise aus der Vorlesung (Vorlesung vom 11.07., Folie 12). Sortieren Sie die Gegenstände zunächst gemäß der Annahme im vorgestellten Algorithmus. Benutzen Sie als obere Schranke den maximalen Wert, der erreichbar wäre, wenn auch Bruchteile von Gegenständen eingepackt werden dürften.

Lösungsvorschlag:

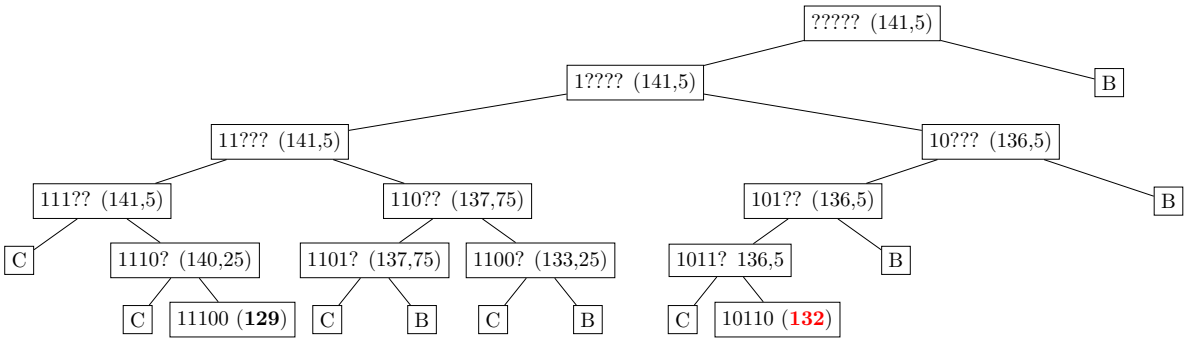
- Wir betrachten nur die Gewichte, die als Kombination auftreten können.

$i\backslash C$	10	15	18	20	25	28	30	33
0	0	0	0	0	0	0	0	0
1	0, (0)	0, (0)	0, (0)	0, (0)	0, (0)	0, (0)	0, (0)	0, (0)
2	0, (0)	0, (0)	45, (1)	45, (1)	45, (1)	45, (1)	45, (1)	45, (1)
3	0, (0)	0, (0)	45, (0)	52, (1)	52, (1)	52, (1)	52, (1)	52, (1)
4	0, (0)	42, (1)	45, (0)	52, (0)	52, (0)	52, (0)	52, (0)	87, (1)
5	35, (1)	42, (0)	45, (0)	52, (0)	77, (1)	80, (1)	87, (1)	87, (0)

$i\backslash C$	35	38	40	43	45	48	50
0	0	0	0	0	0	0	0
1	0, (0)	0, (0)	90, (1)	90, (1)	90, (1)	90, (1)	90, (1)
2	45, (1)	45, (1)	90, (0)	90, (0)	90, (0)	90, (0)	90, (0)
3	52, (1)	97, (1)	97, (1)	97, (1)	97, (1)	97, (1)	97, (1)
4	94, (1)	97, (0)	97, (0)	97, (0)	97, (0)	97, (0)	97, (0)
5	94, (0)	97, (0)	97, (0)	97, (0)	129, (1)	132, (1)	132, (1)

Damit liefert die Teilmenge $\{2, 3, 5\}$ einen maximalen Profit von 132.

- Es gilt $\frac{p_1}{w_1} = 2,25$, $\frac{p_2}{w_2} = 2,5$, $\frac{p_3}{w_3} = 2,6$, $\frac{p_4}{w_4} = 2,8$ und $\frac{p_5}{w_5} = 3,5$, wir müssen die Objekte also gerade in umgekehrter Reihenfolge betrachten. Der theoretisch maximal erreichbare Profit beträgt $p_5 + p_4 + p_3 + \frac{5}{18}p_2 = 141,5$. Wir schreiben C, wenn der Abzweig das Maximalgewicht überschreiten würde und B, wenn der Abzweig maximal ein Gewicht liefern würde, das nicht größer als das bisher entdeckte Optimalgewicht ist.



Hinweis:
Zu Dynamischer Programmierung bietet es sich z.B. an auch den Algorithmus von Floyd-Warshall vorzustellen, den ihr z.B. hier wunderbar erklärt findet:
https://de.wikipedia.org/wiki/Algorithmus_von_Floyd_und_Warshall
Die Teilproblemeigenschaft ist dabei sehr schön zu sehen, und da kommen im Zweifel bestimmt auch die Studenten drauf. ;)