Theoretische Grundlagen der Informatik II
Blatt 7

Markus Vieth Marvin Becker

17. Dezember 2015

Aufgabe 1

a)

Man startet mit einem Graphen $G = /V, E)$, einem Wert bestsofar $= \infty$, dem Startknoten s, einer Queue S und dem Teilproblem $P_0 = (s, s, s)$. Das Tupel ist wie folgt zu lesen : (Startknoten, Menge der Knoten zwischen Star- und Endknoten inklusive dieser, Endknoten). Zu beginn wird das Teilproblem P_0 der Queue hinzugefügt.
In jedem Schritt des Algorithmus, wird ein Teilproblem aus der Queue genommen. Dieses Teilproblem (a, W, b) wird mit einer Kante $(b, x) \in E$ mit $x \in V \setminus W$ und erhalten so ein Teilproblem $(a, W \cup x, x)$. Dies wird für jede Kante von b wiederholt. Man erhält so die Teilprobleme P_i mit $i \in 1, \ldots, n$. //Für jedes Teilproblem P_i wird wie folgt vorgegangen.
Ist nun $W \cup x = V$, so wird die Länge des Pfades berechnet. Ist diese Länge kleiner als das bisherige bestsofar, dann wird bestsofar auf die Länge dieses Pfades gesetzt und der Pfad selbst ist die bisher beste Lösung.
Ist der Pfad noch nicht komplett, so wird die untere Schranke $c(W \cup x) + c(a) + c(x) + c(T)$[1] berechnet. Ist diese kleiner, als das bisherige bestsofar, so wird das neue Teilproblem P_i der Queue S hinzugefügt. Diese Schritte werden solange wiederholt, wie sich Elemente in der Queue S befinden.

b)

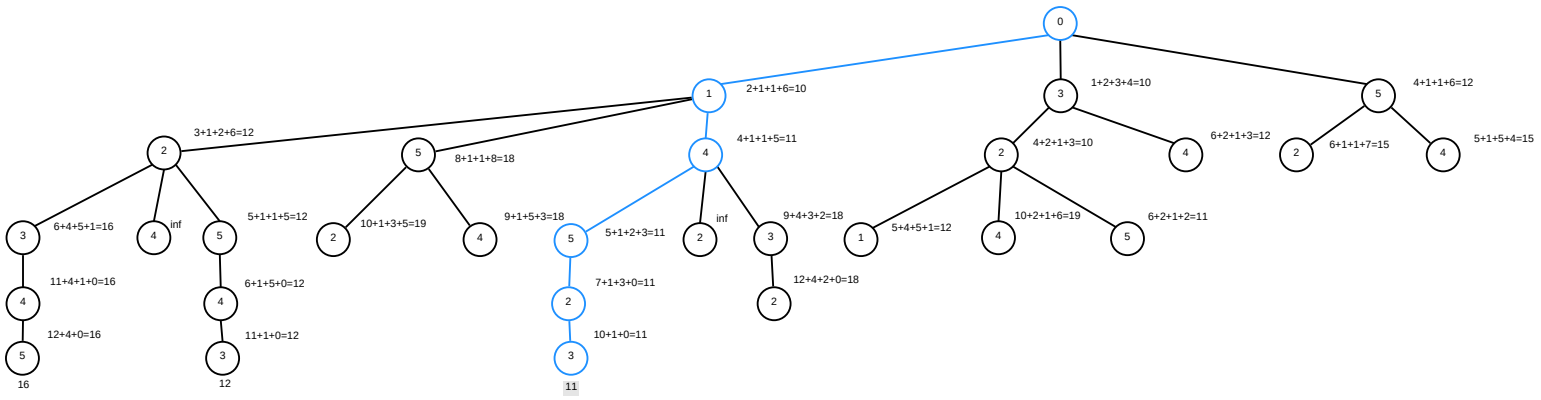

Abbildung 1: Diagramm zu Aufgabe 1

Die Reihenfolge der Summanden entspricht jener aus der Formel aus Aufgabe 1 a).

c)

Wie in 1b) ersichtlich, ist die kürzeste Tour $(0, 1, 4, 5, 2, 3, 0)$ mit den Kosten 11.

[1] siehe Vorlesung 04 Folie 11

Aufgabe 2

i												
4	0	0	0	50	50	50	50	90	90	90	90	
3	0	0	0	0	40	40	40	40	40	50	70	
2	0	0	0	0	40	40	40	40	40	50	50	
1	0	0	0	0	0	10	10	10	10	10	10	
0	0	0	0	0	0	0	0	0	0	0	0	
	0	1	2	3	4	5	6	7	8	9	10	W

$\Rightarrow$ Wir erhalten eine optimale Lösung für $W = 10$ mit den Items 2 und 4.

Aufgabe 3

Backtracking für SAT: Der NP-Algorithmus für SAT (hier $f(\Phi) \rightarrow false, true$) ist true, falls eine erfüllende Belegung $(x_1, x_2, \ldots, x_n)$ für Φ existiert.
Um eine solche Belegung zu finden wird der Backtrack-Algorithmus angewandt:

Wiederhole folgende Schritte bis keine Klauseln mehr vorhanden sind:

Wähle eine Variable x aus, die in Φ vorkommt:
Angenommen es existiert eine erfüllende Belegung für Φ mit $x = 0$
$x = 0$:

- Entferne alle Klauseln in denen $\overline{x}$ auftritt
- Entferne alle Literale x aus den übrigen Klauseln
- Prüfe ob Klauseln ohne Literal existieren, falls ja ist die entstandene Formel nicht erfüllbar, springe zu $x = 1$
- $f(\Phi)$ sollte nun true ergeben, falls nicht dann ist $x = 0$ nicht Teil der erfüllenden Belegung

$x = 1$:
$\Rightarrow$: Änderungen aus den vorherigen Schritten werden zurückgenommen

- Entferne alle Klauseln in denen x auftritt
- Entferne alle Literale $\overline{x}$ aus den übrigen Klauseln
- Prüfe ob Klauseln ohne Literal existieren, falls ja existiert keine erfüllende Belegung
- $f(\Phi)$ sollte nun true ergeben, falls nicht dann existiert keine erfüllende Belegung