Übungsblatt 1

186.813 VU Algorithmen und Datenstrukturen 2 VU 3.0

25. September 2013

Aufgabe 1 *Wir betrachten das 0/1 Rucksack-Problem, das sich für den Weihnachtsmann mit vier verschiedenen Geschenken stellt. Die folgende Tabelle gibt jeweils für jedes der Geschenke G_1, G_2, G_3 und G_4 sein Gewicht und seinen Wert an. Der Rucksack des Weihnachtsmanns kann ein Gewicht von 100 Einheiten transportieren.*

Geschenk	G_1	G_2	G_3	G_4
Gewicht g_i	*32*	*16*	*21*	*50*
Wert w_i	*8*	*2*	*7*	*10*

Sei $F = \langle F_1, ..., F_4 \rangle$ die Folge der vier Gegenstände, absteigend sortiert nach dem Wert des Quotienten w_i/g_i.

In einem Branch-and-Bound-Algorithmus wird das Problem rekursiv in Teilprobleme zerlegt, indem man jeweils in der durch F festgelegten Reihenfolge die Werte der Ent- scheidungsvariablen x_i der Gegenstände fixiert.

Dadurch ergibt sich ein Branch-and-Bound-Baum. Für jeden Knoten v des Baums werden eine obere und eine untere Schranke für die im Teilbaum mit Wurzel v enthaltene beste Lösung bestimmt. Seien nun die Variablen x_1 bis x_k für $1 \leq k \leq 4$ fixiert, wobei x_i für $1 \leq i \leq k$ den Wert 1 hat, falls Geschenk F_i eingepackt wird, und 0 sonst.

Die untere Schranke erhält man, indem man alle Gegenstände, deren Variable noch nicht festgelegt ist, in der durch F gegebenen Reihenfolge durchläuft und den aktuellen Gegenstand einpackt, falls noch Platz im Rucksack ist. Man erhält so eine gültige Lösung für das Problem, deren Wert die untere Schranke ist.

Eine obere Schranke wird berechnet, indem man ebenfalls alle Gegenstände, deren Variable noch nicht festgelegt ist, in der durch F gegebenen Reihenfolge durchläuft. Man packt aber nun alle Gegenstände ein, bis man zu dem ersten Gegenstand F_i kommt, der nicht mehr in den Rucksack passt. Sei r die noch freie Kapazität des Rucksacks. Dann zählt man $r \cdot w_i/g_i$ noch zu dem Wert der Gegenstände im Rucksack dazu, d.h., dieser letzte Gegenstand wird "anteilsmäßig" eingepackt. Alle verbleibenden Gegenstände werden ignoriert.

Zeichnen Sie einen möglichen Branch-and-Bound-Baum für das Problem und schreiben Sie zu jedem Knoten die Werte der unteren und oberen Schranken sowie als wievielter der Knoten abgearbeitet wurde. Nehmen Sie an, dass beim Fixieren einer neuen Variable immer zuerst das

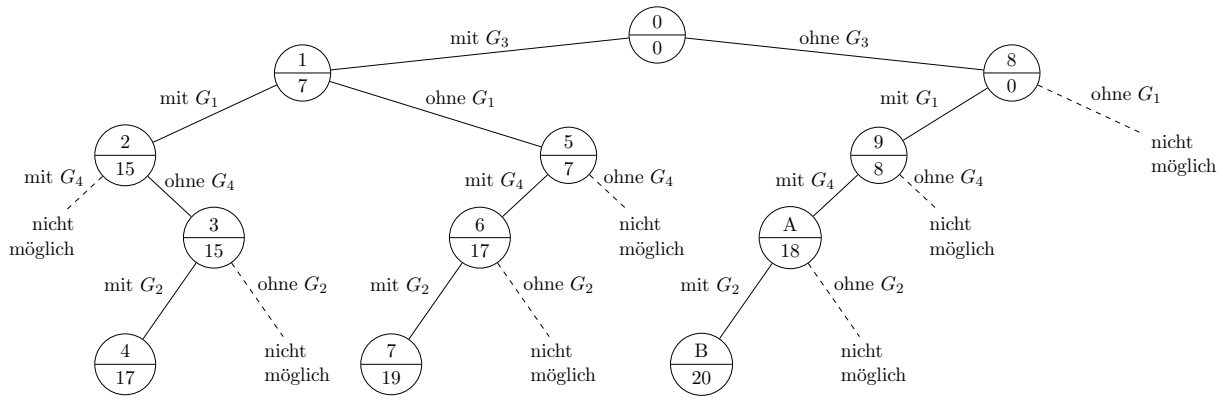
Kind betrachtet wird, bei dem die Variable auf 1 gesetzt wird, und dass Sie den Baum mittels Tiefensuche durchmustern. Achten Sie darauf, dass Sie nur jene Teile des Baums zeichnen, die zur Berechnung der optimalen Lösung nötig sind.

Lösung Als erstes berechne ich mir den Quotienten für jeden der Gegenstände $G_1 - G_n$ und sortiere die Gegenstände absteigend nach Nutzen:

Geschenk	G_3	G_1	G_4	G_2
Gewicht g_i	21	32	50	16
Wert w_i	7	8	10	2
Nutzen n_i	$0.\dot{3}$	0.25	0.2	0.125

Anschließend beginne ich mit dem Branch-and-Bound-Verfahren.
Beachte, dass bei den Knoten 1 und 2 bei der Berechnung des Upper-Limits der Gegenstand G_4 nur teilweise reinpasst, das bedeutet, wir müssen ihn Anteilsmäßig hinzufügen. Der Anteil wird berechnet durch $r \cdot w_i/g_i$.

Auf das konkrete Beispiel angewendet bedeutet das: $21 + g(G_1) = 21 + 32 = 53 \Rightarrow r = 47$
$w_4 = 10$ und $g_4 = 50 \Rightarrow r \cdot w_i/g_i = 47 \cdot 10/50 = 47 \cdot 0.2 = 9.4$.



Knoten	selektiert	unselektiert	g_{ges}	w_{ges}	Lower-Lim.	Upper-Lim.
1	$\{G_3\}$	$\{\}$	21	7	$7 + G_1 + G_4(teilw.) = 24.4$	$7 + G_1 + G_2 = 17$
2	$\{G_3, G_1\}$	$\{\}$	53	15	$15 + G_4(teilw.) = 24.4$	$15 + G_2 = 17$
3	$\{G_3, G_1\}$	$\{G_4\}$	53	15	$15 + G_2 = 17$	$15 + G_2 = 17$
4	$\{G_3, G_1, G_2\}$	$\{G_4\}$	69	17	17	17
5	$\{G_3\}$	$\{G_1\}$	21	7	$7 + G_4 + G_2 = 19$	$7 + G_4 + G_2 = 19$
6	$\{G_3, G_4\}$	$\{G_1\}$	71	17	$17 + G_2 = 19$	$17 + G_2 = 19$
7	$\{G_3, G_4, G_2\}$	$\{G_1\}$	88	19	19	19
8	$\{\}$	$\{G_3\}$	0	0	$G_1 + G_4 + G_2 = 20$	$G_1 + G_4 + G_2 = 20$
9	$\{G_1\}$	$\{G_3\}$	32	8	$8 + G_4 + G_2 = 20$	$8 + G_4 + G_2 = 20$
A	$\{G_1, G_4\}$	$\{G_3\}$	82	18	$18 + G_2 = 20$	$18 + G_2 = 20$
B	$\{G_1, G_4, G_2\}$	$\{G_3\}$	98	20	20	20

Aufgabe 2 *Branch-and-Bound:*
In welchen Situationen bzw. unter welchen Voraussetzungen kann beim Branch-and- Bound Algorithmus ein Teilproblem aus der Problemliste entfernt werden?

Lösung Bei Minimierung:

i) Wenn bei einem Teilproblem das Lower-Limit größer als das globale Upper-Limit ist.

ii) Wenn bei einem Teilproblem das Lower-Limit gleich dem Upper-Limit ist, denn dann ist die Lösung optimal.

Bei Maximierung:

i) Wenn bei einem Teilproblem das globale Lower-Limit größer als das Upper-Limit ist.

ii) Wenn bei einem Teilproblem das Lower-Limit gleich dem Upper-Limit ist, denn dann ist die Lösung optimal.

iii) Die Summe der Gewichte darf die Kapazität nicht übersteigen.

Aufgabe 3 *Betrachten Sie den Branch-and-Bound-Algorithmus für das asymmetrische Traveling-Salesman-Problem aus dem Skriptum bzw. der Vorlesung und die folgende Distanzmatrix:*

$$D = \begin{pmatrix} \infty & 7 & 3 & 2 \\ 4 & \infty & 3 & 5 \\ 8 & 3 & \infty & 2 \\ 5 & 2 & 3 & \infty \end{pmatrix}$$

(a) Führen Sie zuerst Zeilen- und danach Spaltenreduktion durch. Geben Sie die vollständige resultierende Matrix nach beiden Schritten an. Welche untere Schranke L für die Tourlänge können Sie aus den Reduktionen ableiten?

(b) Zeichnen Sie den Nulldigraphen nach den obigen Reduktionen. Welche Schlussfolgerung können Sie aus diesem Nulldigraphen ziehen?

Lösung

(a) Zeilenreduktion: (in jeder Zeile das Minimum suchen und dieses von jedem Wert in dieser Zeile subtrahieren)

$$D = \left(\begin{array}{cccc|c} \infty & 5 & 1 & 0 & 2 \\ 1 & \infty & 0 & 2 & 3 \\ 6 & 1 & \infty & 0 & 2 \\ 3 & 0 & 1 & \infty & 2 \end{array}\right)$$

$$\Rightarrow L_{Zeilen} = 2 + 3 + 2 + 2 = 9$$

(b) Spaltenreduktion: (in jeder Spalte das Minimum suchen und dieses von jedem Wert in dieser Spalte subtrahieren)

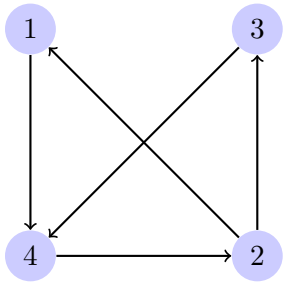
$$D = \left(\begin{array}{cccc} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 5 & 1 & \infty & 0 \\ 2 & 0 & 1 & \infty \\ \hline 1 & 0 & 0 & 0 \end{array}\right)$$

$$\Rightarrow L = L_{Zeilen} + 1 + 0 + 0 + 0 = 10$$

(c) Matrix nach Zeilen- und Spaltenreduktion:

$$D = \begin{pmatrix} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 5 & 1 & \infty & 0 \\ 2 & 0 & 1 & \infty \end{pmatrix}$$

(d) Nulldigraph:



Hier sieht man, dass eine Rundtour nicht möglich ist, da es keine Möglichkeit gibt, in einem Zug alle Knoten genau einmal zu besuchen. (Eulertour) ⇒ Branching würde das Problem lösen.

Aufgabe 4 *Setzen Sie den Branch-and-Bound-Algorithmus aus der vorigen Aufgabe fort.*

(a) Führen Sie den Branching-Schritt aus. Wählen Sie Kante $(3, 4)$ *zum Definieren der neuen Teilprobleme. Geben Sie die Matrizen der neuen Teilprobleme an.*

(b) Führen Sie auf den Matrizen der neuen Teilprobleme wiederum Zeilen- und Spaltenreduktion durch und geben Sie die resultierenden Matrizen an. Nennen Sie die neuen unteren Schranken für die Tourlänge.

(c) Zeichnen Sie für beide Teilprobleme den Nulldigraphen nach den obigen Reduktionen. Welche Schlussfolgerungen können Sie aus diesen Nulldigraphen ziehen?

Lösung Matrix nach Zeilen- und Spaltenreduktion aus vorigem Beispiel:

$$D = \begin{pmatrix} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 5 & 1 & \infty & 0 \\ 2 & 0 & 1 & \infty \end{pmatrix}$$

(i) Teilproblem mit Kante $(3,4)$: (Spalte 3 und Zeile 4 aus reduzuierter Matrix entfernen, außerdem muss die Gegenüberliegende Kante (4,3) ∞ werden.)

$$D = \begin{pmatrix} \infty & 5 & 1 \\ 0 & \infty & 0 \\ 2 & 0 & \infty \end{pmatrix}$$

(a) Zeilenreduktion: (in jeder Zeile das Minimum suchen und dieses von jedem Wert in dieser Zeile subtrahieren)

$$D = \left(\begin{array}{ccc|c} \infty & 4 & 0 & 1 \\ 0 & \infty & 0 & 0 \\ 2 & 0 & \infty & 0 \end{array}\right)$$

$$\Rightarrow L_{Zeilen} = L_{vorher} + 1 + 0 + 0 = 11$$

(b) Spaltenreduktion: (in jeder Spalte das Minimum suchen und dieses von jedem Wert in dieser Spalte subtrahieren)

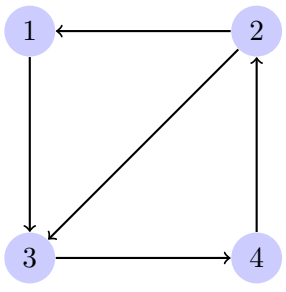
$$D = \left(\begin{array}{ccc} \infty & 4 & 0 \\ 0 & \infty & 0 \\ 2 & 0 & \infty \\ \hline 0 & 0 & 0 \end{array}\right)$$

$$\Rightarrow L = L_{Zeilen} + 0 + 0 + 0 = 11$$

(c) Matrix nach Zeilen- und Spaltenreduktion:

$$D = \begin{pmatrix} \infty & 4 & 0 \\ 0 & \infty & 0 \\ 2 & 0 & \infty \end{pmatrix}$$

(d) Nulldigraph:



Hier ist ein Rundkurs möglich, $\Rightarrow$ Zumindest mit den Kosten von $L = 11$ kann man alle Orte einmal besuchen.

(ii) Teilproblem ohne Kante $(3, 4)$: (Kante (3,4) muss ∞ werden.)

$$D = \begin{pmatrix} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 5 & 1 & \infty & \infty \\ 2 & 0 & 1 & \infty \end{pmatrix}$$

(a) Zeilenreduktion: (in jeder Zeile das Minimum suchen und dieses von jedem Wert in dieser Zeile subtrahieren)

$$D = \left(\begin{array}{cccc|c} \infty & 5 & 1 & 0 & 0 \\ 0 & \infty & 0 & 2 & 0 \\ 4 & 0 & \infty & \infty & 1 \\ 2 & 0 & 1 & \infty & 0 \end{array}\right)$$

$$\Rightarrow L_{Zeilen} = L_{vorher} + 0 + 0 + 1 + 0 = 11$$

(b) Spaltenreduktion: (in jeder Spalte das Minimum suchen und dieses von jedem Wert in dieser Spalte subtrahieren)

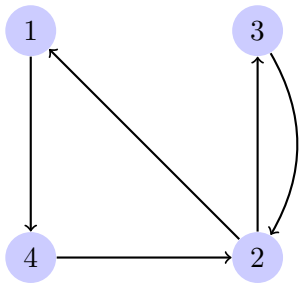
$$D = \left(\begin{array}{cccc} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 4 & 0 & \infty & \infty \\ 2 & 0 & 1 & \infty \\ \hline 0 & 0 & 0 & 0 \end{array}\right)$$

$$\Rightarrow L = L_{Zeilen} + 0 + 0 + 0 + 0 = 11$$

(c) Matrix nach Zeilen- und Spaltenreduktion: (sieht genauso aus, wie vorher, da sie nicht reduziert werden konnte)

$$D = \begin{pmatrix} \infty & 5 & 1 & 0 \\ 0 & \infty & 0 & 2 \\ 4 & 0 & \infty & \infty \\ 2 & 0 & 1 & \infty \end{pmatrix}$$

(d) Nulldigraph:



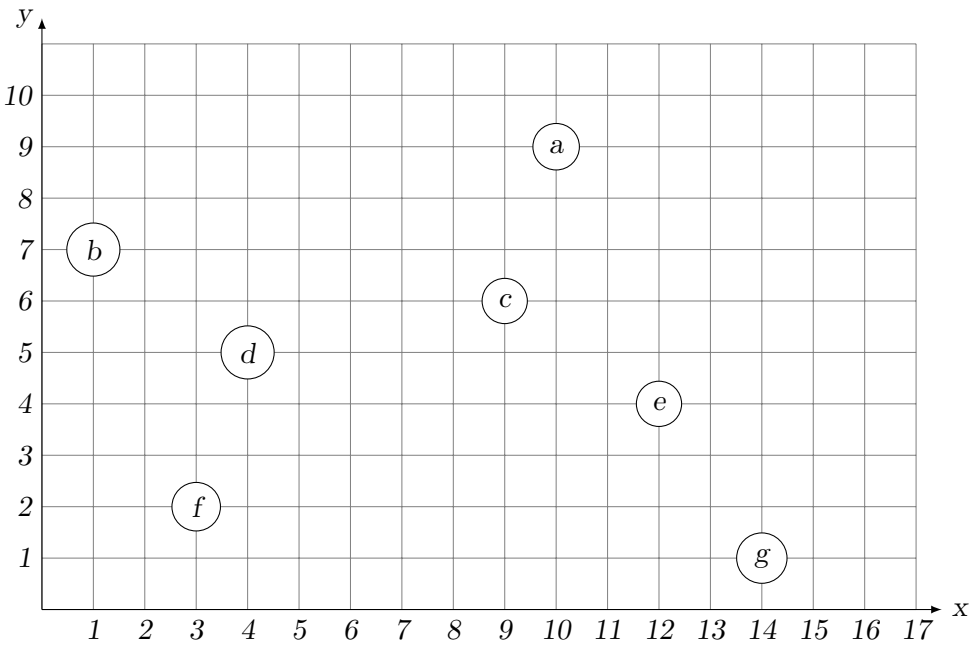
Hier sieht man, dass eine Rundtour nicht möglich ist, da es keine Möglichkeit gibt, in einem Zug alle Knoten genau einmal zu besuchen. (Eulertour) $\Rightarrow$ weiteres Branching würde das Problem lösen.

Das bedeutet, dass die Lösung des Problems nicht $L = 11$ sein kann, somit muss sie mindestes um 1 größer sein (also $L \geq 12$), dieses Problem wurde aber bereits gelöst

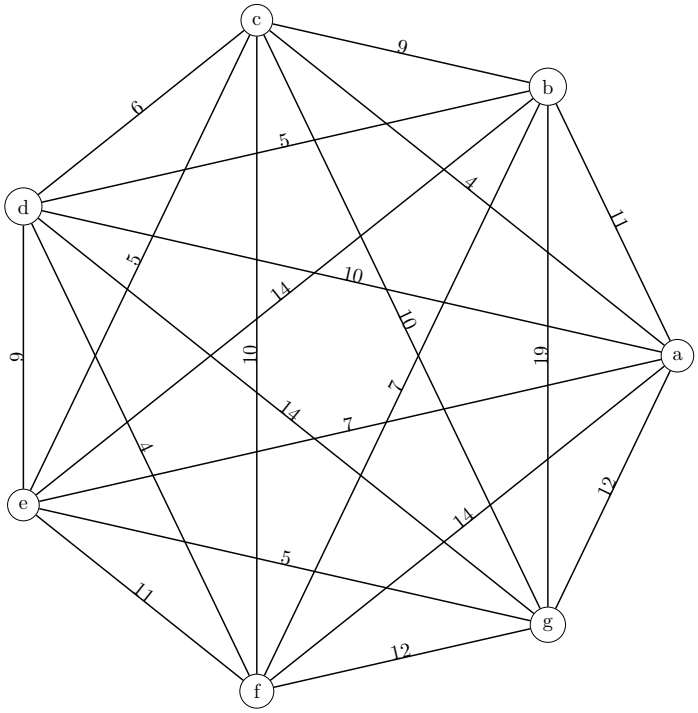
durch die Kante (3,4) und hier kam bereits $L = 11$ heraus, das bedeutet, wir sind fertig.

Aufgabe 5 *Das folgende Diagramm stellt die Knoten eines vollständigen, ungerichteten Graphen dar; die Gewichte der einzelnen Kanten entsprechen jeweils der Manhattan Distanz zwischen den beiden adjazenten Knoten. Wenden Sie die Spanning-Tree-Heuristik an, um eine möglichst gute Lösung zum Traveling Salesman Problem (TSP) in diesem Graphen zu finden. Beschreiben und veranschaulichen Sie dabei alle Schritte.*

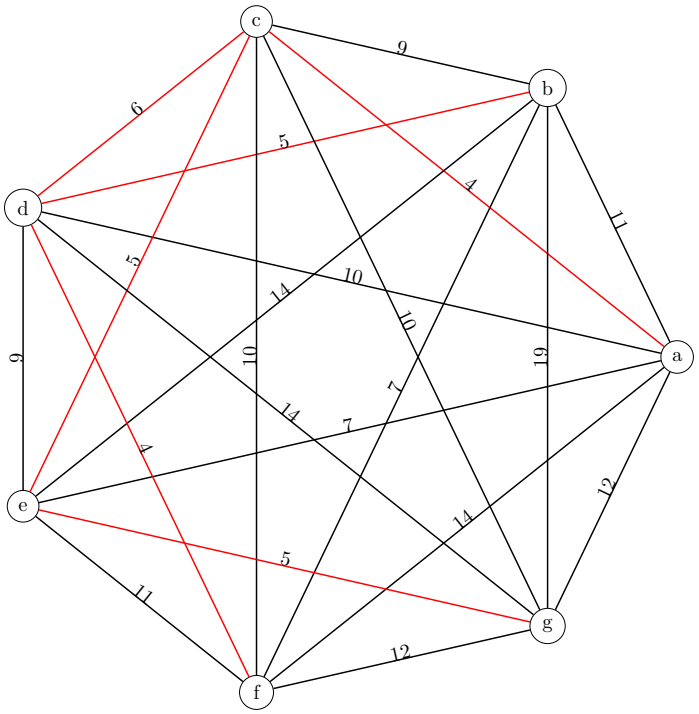
Kann die Wahl vom Startknoten das Endergebnis beeinflussen?



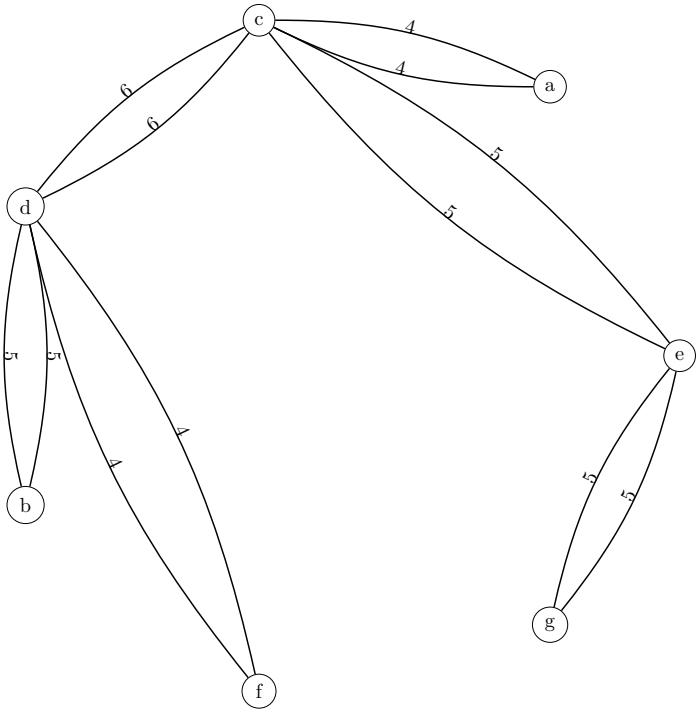
Lösung



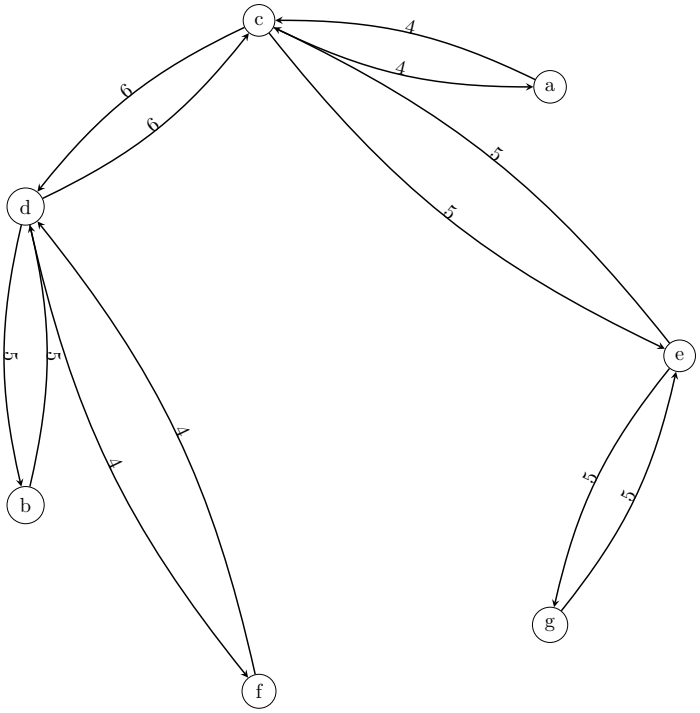
i) Bestimmen von miminalem aufspannenden Baum B von K_n. (Algorithmus von Prim, Startknoten ist a)
Der Reihe nach werden immer nur die Billigsten Kanten genommen, bis alle Knoten erreicht wurden.



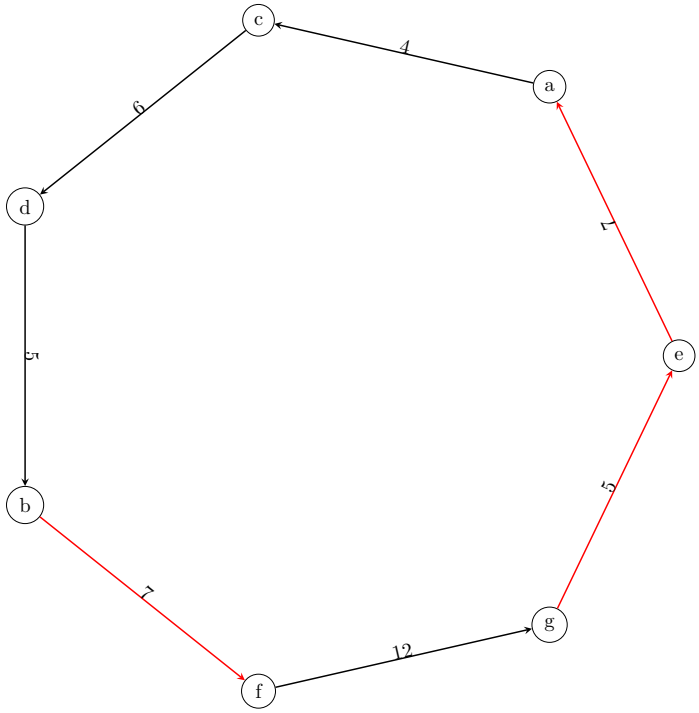
ii) Jede Kante im MST verdoppeln:



iii) Orientierung: (Gerichtete Kanten)



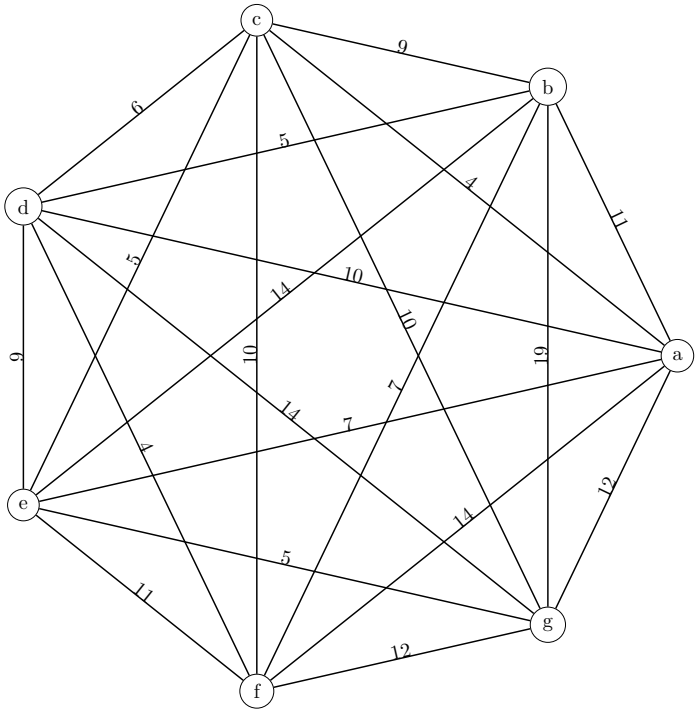
iv) Wähle Startknoten (a) und folge den Kanten des verdoppelten Spannbaumes im Sinne des Eulerkreises. Im folgenden Graph sind alle roten Kanten die, die nicht in dem verdoppelten Graphen von oben drin waren und die einfach eingefügt wurden.



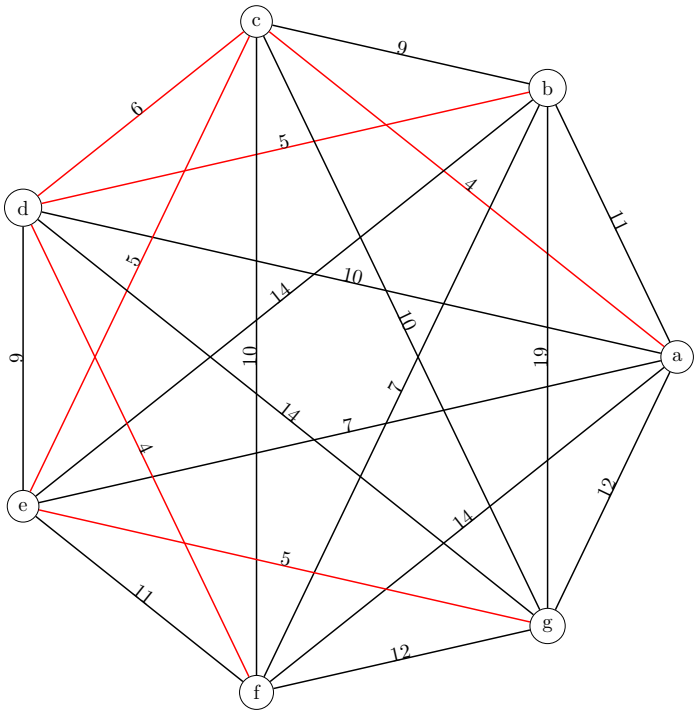
Das bedeutet in Summe betragen die Kosten für den gesamten Eulerkreis 46.

Aufgabe 6 *Wenden Sie nun die Christophides-Heuristik auf den Graphen aus der vorigen Aufgabe an, um einen möglichst kurzen Rundweg zu finden. Beschreiben und veranschaulichen Sie dabei wieder alle notwendigen Schritte.*

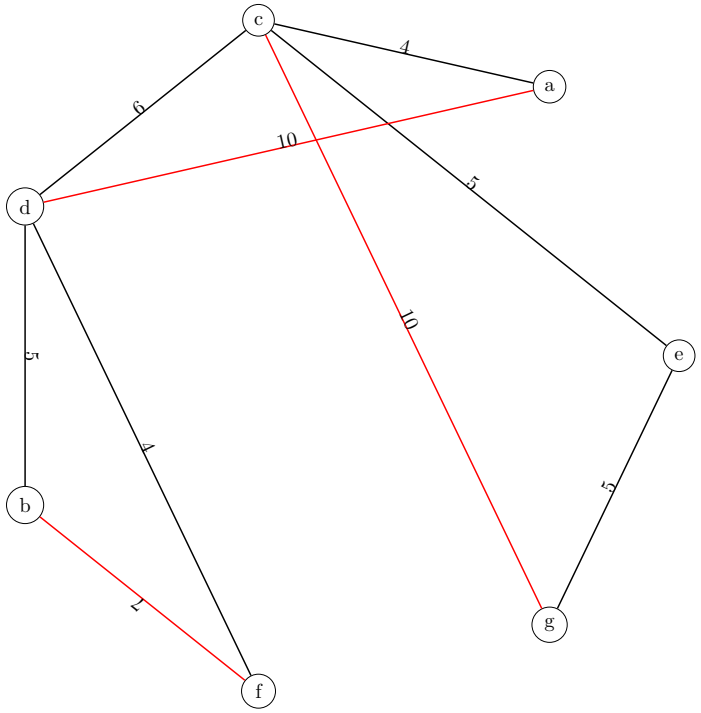
Lösung



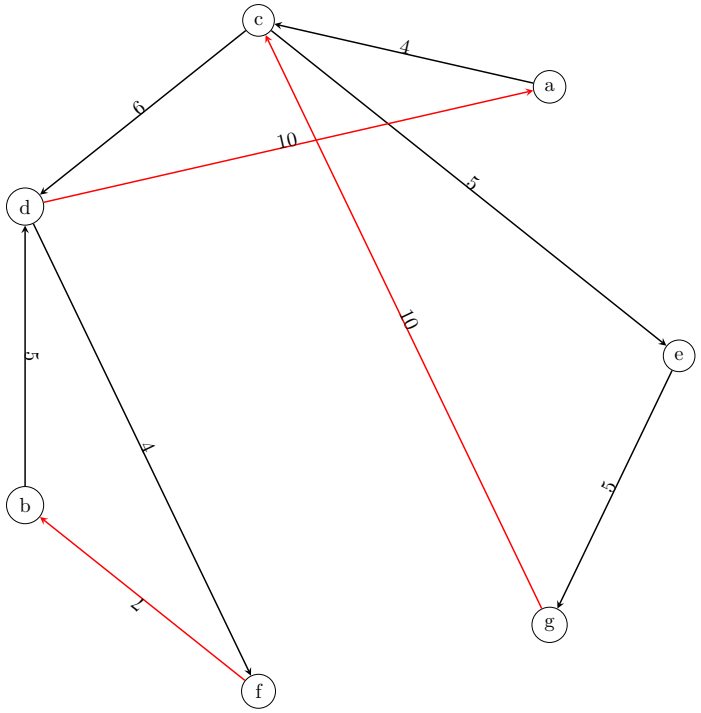
i) Bestimmen von miminalem aufspannenden Baum B von K_n. (Algorithmus von Prim, Startknoten ist a)
Der Reihe nach werden immer nur die Billigsten Kanten genommen, bis alle Knoten erreicht wurden.



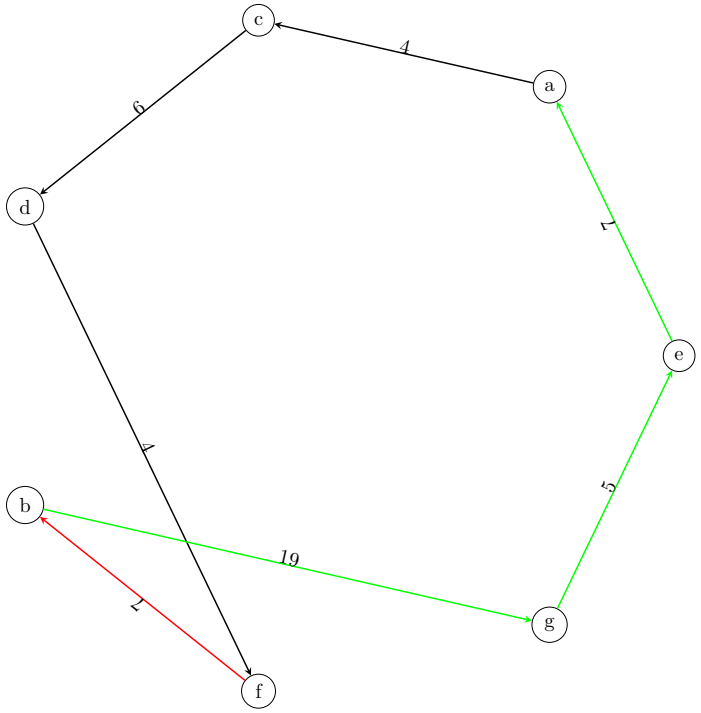
ii) Suche ein perfektes Matching im Graphen für die Knoten mit ungeradem Grad.



iii) Orientierung (gerichtete Graphen)



iv) Suche eine Eulertour beginnend mit a.



Die Summe davon ist 52.

Aufgabe 7 *Entwerfen Sie einen ausführlichen Pseudocode für die $Nearest - Neighbor - Heuristik(NN)$ für das symmetrische TSP mit der folgenden Funktionsweise:*
Gegeben sei ein vollständiger gewichteter Graph $G = (V, E)$. Von einem beliebigen Startknoten ausgehend wird eine inzidente Kante mit minimalem Gewicht zu einem nächsten noch nicht besuchten Knoten gewählt. Dieses Prinzip wird sukzessive fortgesetzt, bis alle Knoten zu einem Hamiltonschen Pfad zusammengefasst wurden. Dieser wird letztlich durch eine letzte Kante zu einem Kreis geschlossen.
Geben Sie die Laufzeit der NN-Heuristik in Θ-Notation an. Was können Sie über die Approximationsgüte der NN-Heuristik aussagen?

Lösung **Algorithmus** NNH(G, s):

Eingabe: Graph $G = (V, E)$; Knoten $s \in V$
Ausgabe: alle Knoten G, die von s aus erreichbar sind

1: Initialisierung: setze $markiert[v] = 0$ für alle v;
2: Initialisierung: leere Kanten-Queue Q (hier kommen die gewählten Kanten hinein)
3: $next = s$;
4: **solange** $next \neq NULL$ **{**
5: $v = next$;
6: $markiert[v] = 1$;
7: $min_e = NULL$;

```
8:    min_g = NULL;
9:    next = NULL;
      // suche die billigste Kante zu einem noch nicht besuchten Knoten
10:   für alle Knoten w aus N(v) {
11:     falls markiert[w] == 0 dann {
12:       e = edge(v, w);  // bekomme die Kante
13:       g = weight(e);  // bekomme das Gewicht der Kante
14:       falls min_e ≠ NULL ODER min_g > g dann {
15:         min_g = g;
16:         min_e = e;
17:         next = w;
18:       }
19:     }
20:   }
21:   falls min_e ≠ NULL dann {
22:     Q.push(min_e);
23:   } sonst {
24:     Q.push(edge(v, s));
25:   }
26: }
```

Mein Algorithmus hat eine Laufzeit von $\Theta(|V|^2)$.
Die Approximationsgüte ist nicht gut definierbar (∞), denn es gibt Graphen, die ein immer schlechteres Ergebnis erzielen. (Es werden weiters auch nicht der Start- und der Endknoten betrachtet. Somit sind zwischen Start- und Endknoten schier unendlich große Distanzen erlaubt.)

Aufgabe 8 *Betrachten Sie die Nearest-Neighbor-Heuristik aus der vorigen Aufgabe.*
Eine gängige Verbesserung ist die All Nearest-Neighbor-Heuristik (ANN), die iterativ jeden Knoten des Graphen als Startknoten für die NN-Heuristik benutzt, alle resultierenden Hamiltonschen Kreise vergleicht, und am Ende eine beste Lösung nimmt. Dieses Verfahren schneidet in der Praxis ublicherweise deutlich besser als die NN-Heuristik ab.
Welche Laufzeit besitzt die ANN-Heuristik in Θ-Notation? Was können Sie über die Approximationsgüte der ANN-Heuristik aussagen?

Lösung Dieser Algorithmus hätte eine Laufzeit von $\Theta(|V|^3)$. Hier ist das Problem mit Start- und Endknoten nicht mehr, jedoch ist es trotzdem nicht gut definierbar(∞), denn es gibt Graphen, die ein immer schlechteres Ergebnis erzielen. Außerdem ist das Problem bei der NN-Heuristik nicht gut definierbar, somit ist das Problem bei ANN auch nicht gut definierbar.

Aufgabe 9 *Beim Miller-Rabin-Primzahltest wird eine Funktion "$Zeuge(a, n)$" benutzt, die für zwei positive ganze Zahlen a und n prüft, ob $a^{n-1} \equiv 1$ mod n gilt. Führen Sie den Algorithmus für a = 3 und n = 12 durch. Geben Sie für jeden Schleifendurchlauf den Wert Ihrer Variablen an. Was sagt das Ergebnis aus?*

Lösung Hier nochmal der Algorithmus:
Algorithmus Zeuge(a, n):

```
1: b = (b_{k-1}, ..., b_0) =Binärdarstellung von n − 1;
2: d = 1;  // c = 0
3: für i = k − 1, ..., 0 {
4:    d = (d ∗ d) mod n;  // c = 2c
5:    falls b_i == 1 dann {
6:       d = (d ∗ a) mod n;  // c = c + 1
7:    }
8: }
9: falls d ≠ 1 dann {
10:   Retourniere true;  // a ist Zeige für "n ist nicht prim"
11: } sonst {
12:   Retourniere false;  // a ist kein Zeuge
13: }
```

Im folgenden brauchen wir somit die Binärdarstellung von $n-1$: $n - 1 = 12 - 1 = 11_d = 1011_b$

1. Durchlauf:
   $d = 1 * 1 \bmod 12 = 1$
   und da die 1. Stelle der Binärdarstellung 1 ist, gilt außerdem noch:
   $d = 1 * 3 \bmod 12 = 3$

2. Durchlauf:
   $d = 3 * 3 \bmod 12 = 9$
   und da die 2. Stelle der Binärdarstellung 0 ist, sind wir hier fertig.

3. Durchlauf:
   $d = 9 * 9 \bmod 12 = 9$
   und da die 3. Stelle der Binärdarstellung 1 ist, gilt außerdem noch:
   $d = 9 * 3 \bmod 12 = 3$

4. Durchlauf:
   $d = 3 * 3 \bmod 12 = 9$
   und da die letzte Stelle der Binärdarstellung 1 ist, gilt außerdem noch:
   $d = 9 * 3 \bmod 12 = 3$

d wurde im letzten Schritt auf 3 gesetzt. $d = 3 \neq 1 \Rightarrow a$ ist ein Zeuge, dass n nicht prim ist.

Aufgabe 10 *Gegeben sei eine anfangs leere perfekte Skipliste S und folgende Elemente, die in der vorgegebenen Reihenfolge eingefügt werden sollen:*

Schlüssel	*13*	*2*	*27*	*19*	*12*	*4*	*28*	*11*	*5*

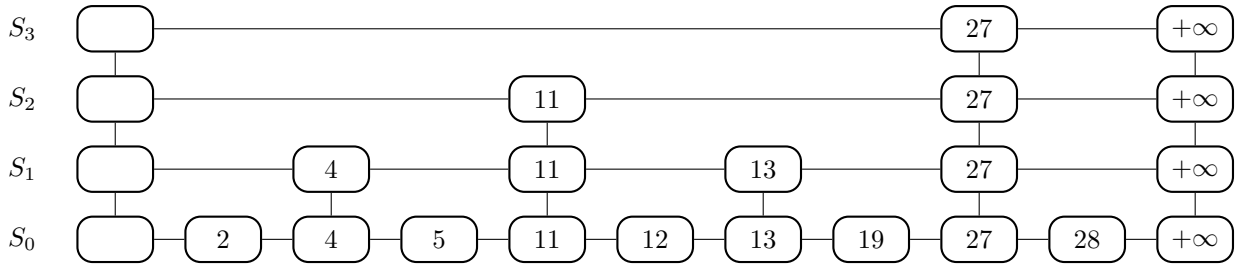
(a) Zeichnen Sie die Skipliste S, nachdem alle Elemente eingefügt wurden.

(b) Suchen Sie in der endgültigen Skipliste S nach der Zahl 7. Wie viele Schlüsselvergleiche waren hierfür notwendig? (Vergleiche mit ∞ zählen mit.)

(c) Suchen Sie in der endgültigen Skipliste S nach der Zahl 13. Wie viele Schlüsselvergleiche waren hierfür notwendig? (Vergleiche mit ∞ zählen mit.)

Lösung

(a) Skipliste gezeichnet:



(b) Wir vergleichen der Reihe nach mit den Schlüsseln $27, 11, 4, 11, 5, 11$ und kommen drauf, dass das Element 7 nicht in der Liste ist. (Es waren 6 Schlüsselvergleiche notwendig.)

(c) Wir vergleichen der Reihe nach mit den Schlüsseln $27, 11, 27, 13, 12, 13$ und kommen drauf, dass das Element 13 nicht in der Liste ist. (Es waren 6 Schlüsselvergleiche notwendig.)
