Aufgabe 3: Flußprobleme

In dieser Aufgabe werden zwei Algorithmen zum Finden des maximalen Durchsatzes implementiert und verglichen.

Aufgabenstellung

Implementieren Sie nun unter Verwendung der ADT Graph folgende zwei Algorithmen, **so, wie sie in der Vorlesung vorgestellt wurden** (also nicht mittels Residualnetzwerk):

1. Den Algorithmus von **Ford und Fulkerson**, so wie er in der Vorlesung vorgestellt wurde. Das Ergebnis ist nachvollziehbar (vergrößernde Wege mit dem Delta der Veränderung) in einer Datei `VerGrWege.log` auszugeben (z.B. 88<-63<-42->62<-43->61<-44<-22 mit Delta 1). Der berechnete Fluss ist unter dem Attributnamen `flow` an der Kante im Graphen zu speichern.
   Schnittstelle: `fordfulkerson:fordfulkerson(<Filename>,<Quelle>,<Senke>)`; Rückgabewert: `[<Liste der im letzten Lauf inspizierten Ecken>]` sowie ein Bild des resultierenden Graphen als *.dot und *.svg Datei;
   bzw. `fordfulkerson:fordfulkerson(<Graph>,<Quelle>, <Senke>)`. Sowie `fordfulkerson:fordfulkersonT(<Graph>,<Quelle>,<Senke>)`; mit obigem Rückgabewert, jedoch ohne weitere Ausgaben (keine log-Dateien oder io-Ausgaben) für die Zeitmessung.
2. Den Algorithmus von **Edmonds und Karp**, wobei soviel Code, wie möglich, von der Implementierung des Ford und Fulkerson Algorithmus verwendet werden soll. Das Ergebnis ist nachvollziehbar (vergrößernde Wege mit dem Delta der Veränderung) in einer Datei `VerGrWege.log` auszugeben. Der berechnete Fluss ist unter dem Attributnamen `flow` an der Kante im Graphen zu speichern.
   Schnittstelle: `edmondskarp:edmondskarp(<Filename>,<Quelle>,<Senke>)`; Rückgabewert: `[<Liste der im letzten Lauf isnpizierten Ecken]` sowie ein Bild des resultierenden Graphen als *.dot und *.svg Datei; bzw. `edmondskarp:edmondskarp(<Graph>, <Quelle>,<Senke>)`. Sowie `edmondskarp:edmondskarpT(<Graph>,<Quelle>,<Senke>)`; mit obigem Rückgabewert, jedoch ohne weitere Ausgaben (keine log-Dateien oder io-Ausgaben) für die Zeitmessung.
3. **Erweitern** Sie die ADT Graph aus Aufgabe 1 um eine Funktion `printGFF(<Graph>, <Filename>)`, die für die Graphen dieser Aufgabe die Graphen in eine dot Datei mit der Kapazität (Attribut `weight`) und dem berechneten Fluss (Attribut `flow`) ausgibt.
4. Für die Laufzeitmessungen erstellen Sie bitte Versionen der Algorithmen ohne print-Ausgabe (`fordfulkersonT` bzw. `edmondskarpT`)! Führen Sie mit diesen Versionen aussagenkräftige Messungen durch (**Einheit: ms!**), mit der Sie die Laufzeitkomplexität der einzelnen Algorithmen nachweisen können. **Im Entwurf** sind für die Messungen bereits die Parameter der einzelnen Messungen anzugeben: welches Ziel (z.B. Laufzeitkomplexität) soll geprüft werden und welche Parameter (z.B. Anzahl Ecken, Grad der Ecken, Gewichte an den Kanten etc) werden in welchem Versuch mit welchen Erwartungen verändert. **Die Messungen sind ins Referat zu integrieren,** indem die Messungen dort dokumetiert werden (Versuchsaufbau, Resultate, Interpretation, geforderte Nachweise etc.). Sofern Ihre Messungen die erwartete Komplexität **nicht bestätigen**, ist die Implementierung zu verbessern oder ausführlich herzuleiten, warum Ihre Implementierung eine andere Laufzeitkomplexität hat.

Auf der Hauptseite zur Vorlesung sind Graphen vorgegeben. Die Attributwerte stellen die Kapazitäten der Kante dar. Die Graphen 01 (18119->80999), 02 (1->22), 03 (11->44), 04 (11->19) und 06 (22->88) sind zum Test der Korrektheit zu verwenden (in der Klammer stehen

Quelle->Senke). Führen Sie mit Ihren Algorithmen eine **Zeitmessung** durch (aufg3test.beam, Aufruf `aufg3test:zeitmessung()`) und **geben die damit erzeugte Datei (sowie die erzeugten log-Dateien) mit ab**. Für diesen Test werden die folgenden Graphen (im gleichen Ordner wie die beam Dateien) benötigt: testaufg3.graph und graph_06.graph. Führen Sie bitte den Test test3GRAPH durch (Aufruf: test3GRAPH:testFKT()) und geben die Bildschirmausgabe mit ab. Sie benötigen dazu graph_02.graph, graph_03.graph, graph_04.graph, graph_06.graph und graph_de.graph. Darüber hinaus sind für die Zeitmessung (**Einheit: ms!**) Graphen mittels dem Tool gengraph zu generieren und zu verwenden. (ausschließlich gerichtet!) . Zur Zeitmessung können Sie folgendes Tool verwenden: zeit3m.beam.

Die zugehörigen Dateien heißen `fordfulkerson.erl` und `edmondskarp.erl`. Weitere Dateien neben `adtgraph.erl` und `util.erl` sind nicht zulässig.

Abnahme

Der Entwurf für diese Aufgabe ist in das Referat zu integrieren.

Am Tag vor dem Referatstermin ist bis spätestens **20:00 Uhr** (Achtung: Prüfungsrelevanter Termin!) die Abgabe des Referates zu erfolgen (schriftliche Ausarbeitung, Code, *.log Dateien etc., Präsentation). Nachreichungen sind nicht möglich.

Am Tag des Referates besteht **während aller Vortragstermine Anwesenheitspflicht.**. Die Nutzung von Rechnern während der Vorträge ist für die Zuhörer*innen untersagt.

Beachten Sie die Regularien zum Referat!

Graphentheoretische Konzepte und Algorithmen

Aufgabe 3

Algorithmus von Ford und Fulkerson

*Algorithmus von **Ford und Fulkerson**: Gegeben sei ein schwach zusammenhängender, schlichter Digraph G = (V,E), eine Kapazitätsfunktion c und ein Fluss f.*

1. *(Initialisierung) Weise allen Kanten $f(e_{ij})$ als einen (initialen) Wert zu, der die Nebenbedingungen erfüllt, d.h. falls $f(e_{ij}) > 0$ muss er von q zu s durchfließen können. Markiere q mit (undefiniert, ∞).*
2. *(Inspektion und Markierung)*
   (a) Falls alle markierten Ecken inspiziert wurden, gehe nach 4.
   (b) Wähle eine beliebige markierte, aber noch nicht inspizierte Ecke v_i und inspiziere sie wie folgt (Berechnung des Inkrements)
   - *(Vorwärtskante) Für jede Kante $e_{ij} \in O(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ij}) < c(e_{ij})$ markiere v_j mit $(+v_i, \delta_j)$, wobei δ_j die kleinere der beiden Zahlen $c(e_{ij}) - f(e_{ij})$ und δ_i ist.*
   - *(Rückwärtskante) Für jede Kante $e_{ji} \in I(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ji}) > 0$ markiere v_j mit $(-v_i, \delta_j)$, wobei δ_j die kleinere der beiden Zahlen $f(e_{ji})$ und δ_i ist.*

   (c) Falls s markiert ist, gehe zu 3., sonst zu 2.(a).
3. *…*

Algorithmus von Ford und Fulkerson

2. …

3. (Vergrößerung der Flussstärke) Bei s beginnend lässt sich anhand der Markierungen der gefundene vergrößernde Weg bis zur Ecke q rückwärts durchlaufen. Für jede Vorwärtskante wird $f(e_{ij})$ um δ_s erhöht, und für jede Rückwärtskante wird $f(e_{ji})$ um δ_s vermindert. Anschließend werden bei allen Ecken mit Ausnahme von q die Markierungen entfernt. Gehe zu 2.

4. Es gibt keinen vergrößernden Weg. Der jetzige Wert von d ist optimal. Ein Schnitt A(X,X) mit c(X,X) = d wird gebildet von genau denjenigen Kanten, bei denen entweder die Anfangsecke oder die Endecke inspiziert ist.

Wenn jede Vergrößerung der Flussstärke d durch einen vergrößernden Weg minimaler Kantenanzahl erfolgt, dann sind höchstens $O(|E| \cdot |V|)$ vergrößernde Wege zu berechnen, bis d seinen Maximalwert erreicht hat.

Schnittstellen

```
fordfulkerson:fordfulkerson(<Filename>,<Quelle>,<Senke>);
```

Rückgabewert:

```
[<Liste der im letzten Lauf inspizierten Ecken>]
```

sowie ein Bild des resultierenden Graphen als *.dot und *.svg Datei;

Für zB Laufzeitmessung:

```
fordfulkerson:fordfulkerson(<Graph>,<Quelle>,<Senke>);
fordfulkerson:fordfulkersonT(<Graph>,<Quelle>,<Senke>);
```

Benötigte Dateien: *fordfulkerson.erl* und *adtgraph.erl* sowie *util.erl*

Algorithmus von Edmonds und Karp

*Algorithmus von **Edmonds und Karp**: Gegeben sei ein schwach zusammenhängender, schlichter Digraph G = (V,E), eine Kapazitätsfunktion c und ein Fluss f.*

1. *(Initialisierung) Weise allen Kanten $f(e_{ij})$ als einen (initialen) Wert zu, der die Nebenbedingungen erfüllt, d.h. falls $f(e_{ij}) > 0$ muss er von q zu s durchfließen können. Markiere q mit (undefiniert, ∞) und füge q in die Schlange BS ein.*
2. *(Inspektion und Markierung)*
   (a) Falls alle markierten Ecken inspiziert wurden (BS ist leer), gehe nach 4.
   (b) Wähle die nächste markierte, aber noch nicht inspizierte Ecke v_i in BS und inspiziere sie wie folgt (Berechnung des Inkrements)
   - *(Vorwärtskante) Für jede Kante $e_{ij} \in O(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ij}) < c(e_{ij})$ markiere v_j mit $(+v_i, \delta_j)$, wobei δ_j die kleinere der beiden Zahlen $c(e_{ij}) - f(e_{ij})$ und δ_i ist.*
   - *(Rückwärtskante) Für jede Kante $e_{ji} \in I(v_i)$ mit unmarkierter Ecke v_j und $f(e_{ji}) > 0$ markiere v_j mit $(-v_i, \delta_j)$, wobei δ_j die kleinere der beiden Zahlen $f(e_{ji})$ und δ_i ist.*

   (c) Falls s markiert ist, gehe zu 3., sonst zu 2.(a).
3. *…*

Schnittstellen

```
edmondskarp:edmondskarp(<Filename>,<Quelle>,<Senke>);
```

Rückgabewert:

```
[<Liste der im letzten Lauf inspizierten Ecken>]
```

sowie ein Bild des resultierenden Graphen als *.dot und *.svg Datei;

Für zB Laufzeitmessung:

```
edmondskarp:edmondskarp(<Graph>,<Quelle>,<Senke>);
edmondskarp:edmondskarpT(<Graph>,<Quelle>,<Senke>);
```

Benötigte Dateien: *edmondskarp.erl* und *adtgraph.erl* sowie *util.erl*

Durchzuführende Tests

```
3> aufg3test:zeitmessung().
Graph mit 876 Ecken und 5267 Kanten aus 'testaufg3.graph' importiert.
        fordfulkerson
..........
        edmondskarp
..........
        >>--<<
Graph mit 10 Ecken und
        fordfulkerson
        edmondskarp
        >>--<<
ok
4>
```

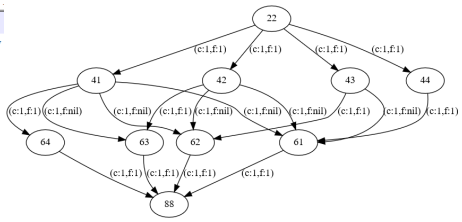
fofuerg214297.svg
edkaerg230299.svg
edkaerg230299.dot
edkaerg917294.dot
fofuerg214297.dot
results323287.log
VerGrWege.log
fofuerg917289.dot

Benötigt: testaufg3.graph
graph_06.graph

VerGrWege.log

```
876<-1 mit Delta 8
876<-318<-270<-269<-268<-267<-266<-178<-1 mit Delta 7
876<-875<-213<-212<-178<-1 mit Delta 3
876<-687<-686<-685<-499<-2<-1 mit Delta 7
876<-542<-250<-2<-1 mit Delta 3
876<-758<-757<-754<-746<-1 mit Delta 7
876<-546<-545<-451<-450<-449<-1 mit Delta 9
876<-542<-513<-328<-327<-1 mit Delta 7
876<-875<-604<-535<-534<-1 mit Delta 4
876<-542<-513<-512<-213->875<-604<-535<-534<-1 mit Delt
876<-758<-757<-756<-740<-580<-495<-1 mit Delta 7
876<-758<-589<-476<-329<-328<-327<-1 mit Delta 6
```

```
fordfulkerson: 1015ms, im Durchschnitt: 92
    Minimaler Schnitt A [{1,q,1,infinitely
{449,fw,1,14},
{534,bw,728,3},
{535,bw,604,2},
{604,bw,606,2},
{605,fw,604,2},
{606,bw,607,2},
{607,bw,608,2},
{608,bw,609,2}
```

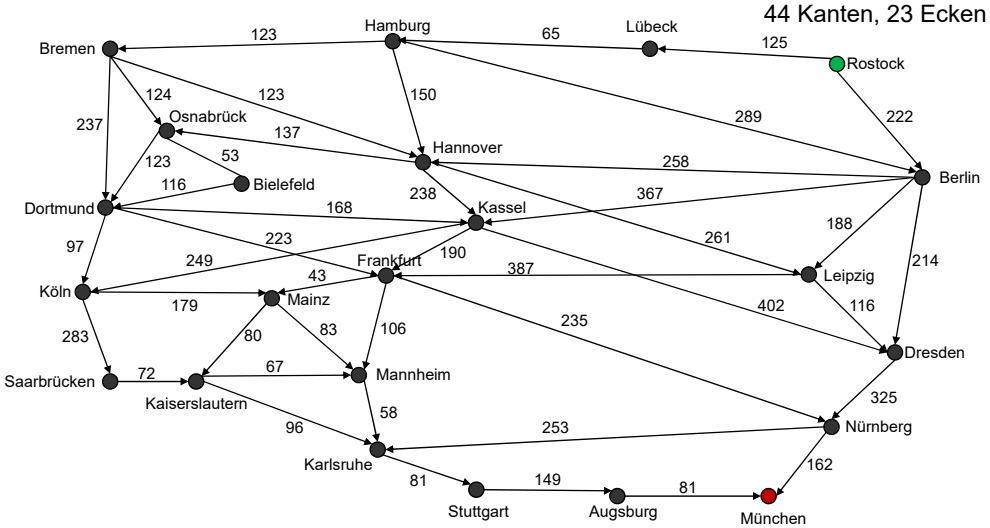


Graphentheoretische Konzepte und Algorithmen

Beispielgraphen

Gerichtete Beispielgraphen

graph_de.graph
44 Kanten, 23 Ecken

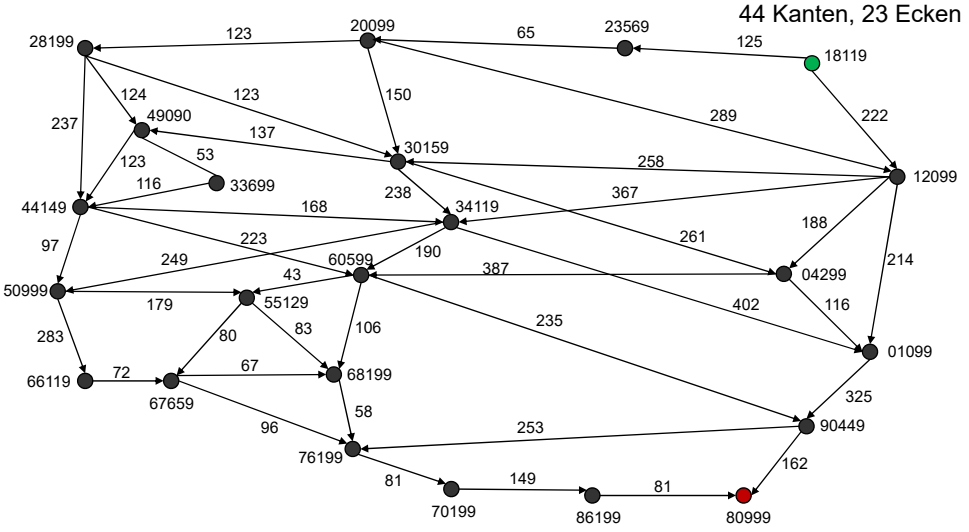


Informatik

Hochschule für angewandte Wissenschaften Hamburg

Gerichtete Beispielgraphen

graph_de.graph
44 Kanten, 23 Ecken

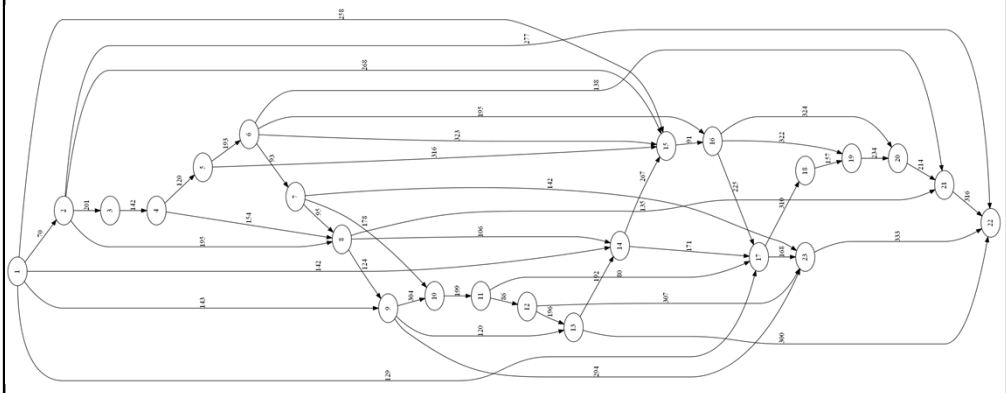


Informatik

Hochschule für angewandte Wissenschaften Hamburg

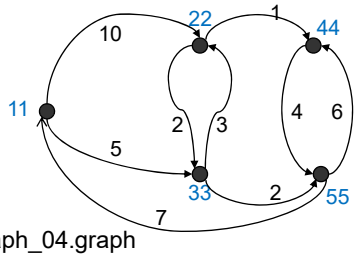
Gerichtete Beispielgraphen

graph_02.graph
47 Kanten, 23 Ecken

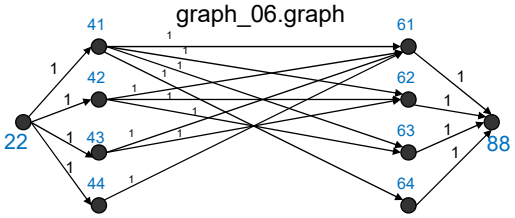


Gerichtete Beispielgraphen

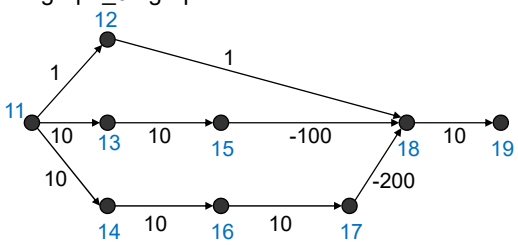
graph_03.graph



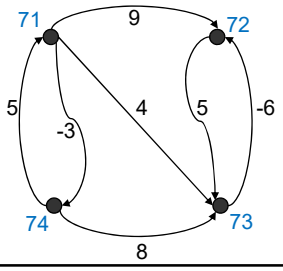
graph_06.graph



graph_04.graph

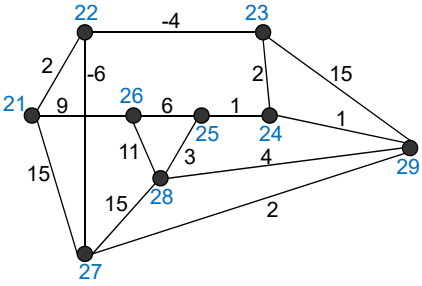


graph_05.graph

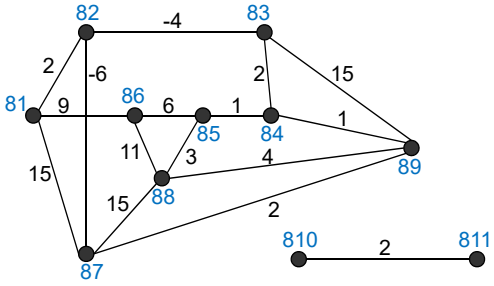


Ungerichtete Beispielgraphen

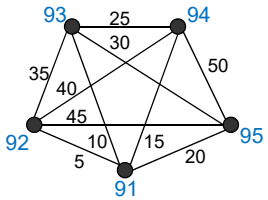
graph_07.graph



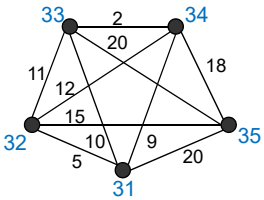
graph_08.graph



graph_09.graph

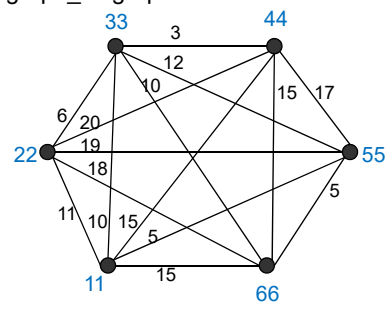


graph_10.graph



Ungerichtete Beispielgraphen

graph_11.graph



aktueller Pfad: [11, 55, 11] mit Kosten 10
aktueller Pfad: [11, 66, 55, 11] mit Kosten 25
aktueller Pfad: [11, 33, 66, 55, 11] mit Kosten 30
aktueller Pfad: [11, 44, 33, 66, 55, 11] mit Kosten 38
aktueller Pfad: [11, 44, 33, 22, 66, 55, 11] mit Kosten 52
Startecke 11 mit Kosten 52 und Kreis [11, 44, 33, 22, 66, 55, 11]
Startecke 22 mit Kosten 45 und Kreis [22, 11, 55, 66, 44, 33, 22]
Startecke 33 mit Kosten 45 und Kreis [33, 22, 11, 55, 66, 44, 33]
Startecke 44 mit Kosten 45 und Kreis [44, 66, 55, 11, 22, 33, 44]
Startecke 55 mit Kosten 52 und Kreis [55, 11, 44, 33, 22, 66, 55]
Startecke 66 mit Kosten 45 und Kreis [66, 44, 33, 22, 11, 55, 66]

Kosten: **45**(1-mal), z.B. [55, 11, 22, 33, 44, 66, 55]
Kosten: **52**(1-mal), z.B. [55, 11, 44, 33, 22, 66, 55]
Kosten: 54(1-mal), z.B. [55, 11, 22, 44, 33, 66, 55]
Kosten: 57(1-mal), z.B. [55, 44, 33, 22, 11, 66, 55]
Kosten: 61(3-mal), z.B. [55, 11, 33, 22, 44, 66, 55]
Kosten: 63(4-mal), z.B. [55, 22, 11, 33, 44, 66, 55]
Kosten: 64(8-mal), z.B. [55, 33, 22, 11, 44, 66, 55]
Kosten: 66(1-mal), z.B. [55, 33, 44, 22, 11, 66, 55]
Kosten: 70(4-mal), z.B. [55, 22, 33, 11, 44, 66, 55]
Kosten: 71(3-mal), z.B. [55, 44, 11, 33, 22, 66, 55]
Kosten: 72(2-mal), z.B. [55, 22, 44, 33, 11, 66, 55]
Kosten: 73(8-mal), z.B. [55, 33, 11, 22, 44, 66, 55]
Kosten: 75(2-mal), z.B. [55, 33, 44, 66, 11, 22, 55]
Kosten: 76(1-mal), z.B. [55, 44, 66, 11, 22, 33, 55]
Kosten: 79(2-mal), z.B. [55, 22, 44, 11, 33, 66, 55]
Kosten: 80(3-mal), z.B. [55, 33, 11, 44, 22, 66, 55]
Kosten: 82(6-mal), z.B. [55, 33, 44, 11, 66, 22, 55]
Kosten: 83(3-mal), z.B. [55, 44, 11, 22, 66, 33, 55]
Kosten: 85(1-mal), z.B. [55, 44, 22, 11, 66, 33, 55]
Kosten: 89(2-mal), z.B. [55, 33, 11, 44, 66, 22, 55]
Kosten: 91(2-mal), z.B. [55, 22, 44, 11, 66, 33, 55]
Kosten: 92(1-mal), z.B. [55, 44, 22, 66, 11, 33, 55]

Ungerichtete Beispielgraphen

graph_09.graph

aktueller Pfad: [91,92,91] mit Kosten 10
aktueller Pfad: [91,93,92,91] mit Kosten 50
aktueller Pfad: [91,94,93,92,91] mit Kosten 80
aktueller Pfad: [91,94,93,95,92,91] mit Kosten 120
Startecke 91 mit Kreis [91,94,93,95,92,91] und Kosten 120
Startecke 92 mit Kreis [92,94,93,95,91,92] und Kosten 120
Startecke 93 mit Kreis [93,94,92,91,95,93] und Kosten 120
Startecke 94 mit Kreis [94,93,95,92,91,94] und Kosten 120
Startecke 95 mit Kreis [95,93,94,92,91,95] und Kosten 120

Kosten: 120 (2-mal), z.B. [93,94,92,91,95,93]
Kosten: 135 (4-mal), z.B. [93,91,92,94,95,93]
Kosten: 140 (4-mal), z.B. [93,92,94,91,95,93]
Kosten: 155 (2-mal), z.B. [93,92,94,95,91,93]

graph_10.graph

aktueller Pfad: [31,32,31] mit Kosten 10
aktueller Pfad: [31,34,32,31] mit Kosten 26
aktueller Pfad: [31,34,33,32,31] mit Kosten 27
aktueller Pfad: [31,34,33,35,32,31] mit Kosten 51
Startecke 31 mit Kreis [31,34,33,35,32,31] und Kosten 51
Startecke 32 mit Kreis [32,35,33,34,31,32] und Kosten 51
Startecke 33 mit Kreis [33,35,32,31,34,33] und Kosten 51
Startecke 34 mit Kreis [34,31,32,35,33,34] und Kosten 51
Startecke 35 mit Kreis [35,34,33,31,32,35] und Kosten 50

Kosten: 50 (1-mal), z.B. [33,34,35,32,31,33]
Kosten: 51 (1-mal), z.B. [33,35,32,31,34,33]
Kosten: 56 (1-mal), z.B. [33,34,35,31,32,33]
Kosten: 57 (1-mal), z.B. [33,34,31,35,32,33]
Kosten: 59 (2-mal), z.B. [33,34,32,31,35,33]
Kosten: 63 (2-mal), z.B. [33,32,31,34,35,33]
Kosten: 65 (1-mal), z.B. [33,35,34,32,31,33]
Kosten: 66 (1-mal), z.B. [33,35,32,34,31,33]
Kosten: 71 (1-mal), z.B. [33,32,34,35,31,33]
Kosten: 72 (1-mal), z.B. [33,35,31,34,32,33]

Tourenproblem

Startecke a1 und Kosten 2157	Startecke a2 und Kosten 2127	Startecke a3 und Kosten 1934
Startecke a4 und Kosten 1985	Startecke a5 und Kosten 1894	Startecke a6 und Kosten 1985
Startecke a7 und Kosten 1780	Startecke a8 und Kosten 1907	Startecke a9 und Kosten 1851
Startecke a10 und Kosten 1895	Startecke a11 und Kosten 1938	Startecke a12 und Kosten 2009
Startecke a13 und Kosten 2050	Startecke a14 und Kosten 2091	Startecke a15 und Kosten 2192
Startecke a16 und Kosten 2038	Startecke a17 und Kosten 1765	Startecke a18 und Kosten 1884
Startecke a19 und Kosten 1783	Startecke a20 und Kosten 1924	Startecke a21 und Kosten 2048
Startecke a22 und Kosten 2224	Startecke a23 und Kosten 2031	Startecke a24 und Kosten 1977
Startecke a25 und Kosten 1784	Startecke a26 und Kosten 1993	Startecke a27 und Kosten 2294
Startecke a28 und Kosten 2038	Startecke a29 und Kosten 1832	Startecke a30 und Kosten 1863
Startecke a31 und Kosten 2094	Startecke a32 und Kosten 2160	Startecke a33 und Kosten 1998
Startecke a34 und Kosten 1938	Startecke a35 und Kosten 2196	Startecke a36 und Kosten 2214
Startecke a37 und Kosten 1901	Startecke a38 und Kosten 2053	Startecke a39 und Kosten 2016
Startecke a40 und Kosten 2291	Startecke a41 und Kosten 2092	Startecke a42 und Kosten 1908
Startecke a43 und Kosten 2283	Startecke a44 und Kosten 2066	Startecke a45 und Kosten 1972
Startecke a46 und Kosten 2024	Startecke a47 und Kosten 2031	Startecke a48 und Kosten 1918
Startecke a49 und Kosten 2356	Startecke a50 und Kosten 2119	Startecke a51 und Kosten 1901

Tourenproblem

Startecke a52 und Kosten 1997	Startecke a53 und Kosten 2202	Startecke a54 und Kosten 2095
Startecke a55 und Kosten 1935	Startecke a56 und Kosten 2242	Startecke a57 und Kosten 2039
Startecke a58 und Kosten 1933	Startecke a59 und Kosten 1957	Startecke a60 und Kosten 1889
Startecke a61 und Kosten 2032	Startecke a62 und Kosten 2012	Startecke a63 und Kosten 2388
Startecke a64 und Kosten 2040	Startecke a65 und Kosten 2053	Startecke a66 und Kosten 1927
Startecke a67 und Kosten 2232	Startecke a68 und Kosten 2362	Startecke a69 und Kosten 2150
Startecke a70 und Kosten 2362	Startecke a71 und Kosten 2006	Startecke a72 und Kosten 2130
Startecke a73 und Kosten 2179	Startecke a74 und Kosten 2141	Startecke a75 und Kosten 2133
Startecke a76 und Kosten 2180	Startecke a77 und Kosten 2004	Startecke a78 und Kosten 1836
Startecke a79 und Kosten 2129	Startecke a80 und Kosten 2376	Startecke a81 und Kosten 1790
Startecke a82 und Kosten 1790	Startecke a83 und Kosten 2021	Startecke a84 und Kosten 2003
Startecke a85 und Kosten 2109	Startecke a86 und Kosten 2006	Startecke a87 und Kosten 2251
Startecke a88 und Kosten 1899	Startecke a89 und Kosten 2096	Startecke a90 und Kosten 2092
Startecke a91 und Kosten 2021	Startecke a92 und Kosten 2130	Startecke a93 und Kosten 2029
Startecke a94 und Kosten 2082	Startecke a95 und Kosten 2082	Startecke a96 und Kosten 1776
Startecke a97 und Kosten 1675	Startecke a98 und Kosten 1976	Startecke a99 und Kosten 2388