Mathematik C für Informatikstudierende – Serie 7 15.12.13
Anna Schleimer, Thure Dührsen
Aufgabe 7.3 (b)

Hier ist nur die eigentliche Fixpunktiteration nebst Hauptprogramm abgedruckt (Datei `MatrixTest.c`).
Die Vektor- und Matrixoperationen sind im Archiv zu finden und nicht so ausführlich dokumentiert, denn ich hoffe, dass Datei- und Funktionsnamen sprechend genug und die im Quelltext zu findenden printf-Anweisungen hinreichend klar sind.

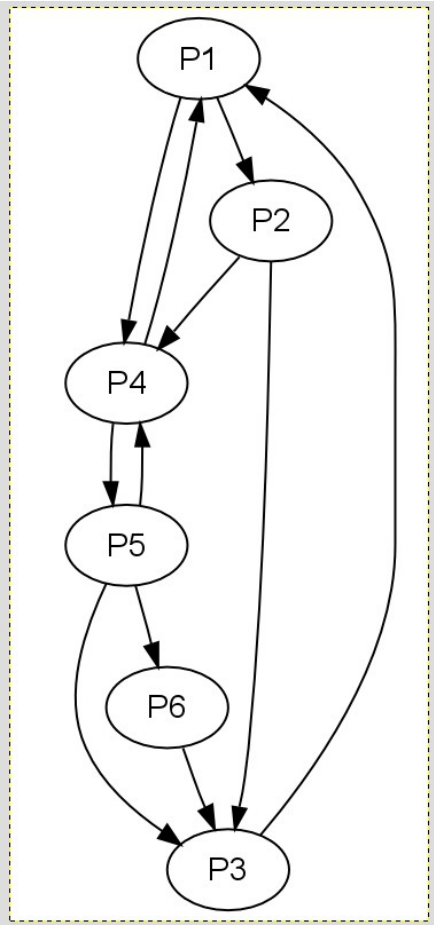
Das vollständige Archiv ist unter
https://www.informatik.uni-kiel.de/~tdu/mathec_aufgabe_7_3.tar
zu finden.

```
 1  /* Mathematik C fuer Informatikstudierende -- Serie 7 -- 15.12.2013
 2     Anna Schleimer, Thure Duehrsen
 3     Aufgabe 7.3: Vektoriteration */
 4
 5
 6  #include <stdlib.h>
 7  #include <stdio.h>
 8  #include <math.h>
 9  #include "Tuple.h"
10  #include "Vector.h"
11  #include "Matrix.h"
12
13  void PageRank_g_A (VectorOfDouble x_neu, MatrixOfDouble A,
14                     VectorOfDouble x,     double c);
15  /* Vorausdeklaration, weil der Compiler sonst meckert */
16
17  int main (int argc, char* argv[]) {
18
19    /* Einziger auf der Kommandozeile zu uebergebender Parameter ist c,
20       der Anteil der Basiswichtigkeit */
21
22    const double c = atof (argv[1]);
23    /* Hier fehlt Fehlerbehandlung */
24
25    const unsigned int   n = 6;
26
27    MatrixOfDouble       A; /* Abbildungsmatrix */
28
29    VectorOfDouble       w0; /* Startvektor */
30
31    VectorOfDouble       w_i_alt; /* Vektor zur Iteration i-1 */
32
33    VectorOfDouble       w_i_neu; /* Vektor zur Iteration i */
34
35    VectorOfDouble       diff; /* w_i_neu - w_i_alt */
36
37    double               norm_i_diff; /* Norm des Vektors diff */
38
39    const double         eps = 0.0001;
40
41    char                 filename[] = "linkmatrix.txt";
42                         /* Speicherort der Abbildungsmatrix */
43
44    int                  i = -1; /* Zaehler */
45
46
47
```

```
48
49      printf ("\nFixpunktiteration zu c = %0.3f\n\n", c);
50
51      A = MatrixOfDouble_LoadFromFile (filename);
52
53      printf ("Linkmatrix A =\n\n"); MatrixOfDouble_Print (A);
54      printf ("\n\n");
55
56      w0 = VectorOfDouble_Create (6); /* w0 \in double^6 */
57
58      /* w0 soll der erste Standardbasisvektor sein */
59      for (i = 1; i<=6; i++) {
60
61        if (i == 1)
62          VectorOfDouble_SetEntry_1 (w0, i, 1);
63        else
64          VectorOfDouble_SetEntry_1 (w0, i, 0);
65
66      }
67
68      printf ("w0 = "); VectorOfDouble_Print (w0); printf ("\n\n");
69
70      w_i_neu = VectorOfDouble_Copy (w0);
71      /* jetzt ist w_i_neu auch der erste Standardbasisvektor, und wir
72         koennen ihn durch die Iterationen schicken.  w_i_neu bekommt das
73         Ergebnis jeder Iteration zugewiesen
74      */
75
76      w_i_alt = VectorOfDouble_Create (n);
77      /* bewahrt die alte Iteration auf */
78
79      /* Es ist
80       */
81
82      diff = VectorOfDouble_Create (n);
83      /* bewahrt die Differenz zweier aufeinanderfolgender Iterationen auf */
84
85      i = 0;
86      do {
87
88        i++;
89
90        /* w_i_alt <--- w_i_neu --- alte Iteration retten */
91        VectorOfDouble_CopyValues(w_i_alt, w_i_neu);
92
93        /* Eine Iteration durchfuehren: w_i_neu <--- g_A(w_i_neu) */
94        PageRank_g_A(w_i_neu, A, w_i_alt, c);
95
96        /* Norm des Differenzvektors bestimmen */
97        /* diff = w_i_neu - w_i_alt */
98        VectorOfDouble_Sub (diff, w_i_neu, w_i_alt);
99        norm_i_diff = VectorOfDouble_Norm (diff);
100
101       printf ("w_%3d = ",i); VectorOfDouble_Print (w_i_neu); printf ("\n");
102
103       } while (norm_i_diff >= eps);
104
105     printf ("\n");
106
107     return 0;
108
109   }
110
```

```
111  void PageRank_g_A (VectorOfDouble x_neu, MatrixOfDouble A,
112                     VectorOfDouble x,     double c) {
113
114    /* Fuehrt einen Iterationsschritt der PageRank-Iteration durch. */
115
116    unsigned int n = x.length; /* Laenge aller beteiligten Vektoren */
117
118    VectorOfDouble basiswichtigkeit;
119    /* Vektor der Basiswichtigkeit (VL 13.12.13) */
120
121    unsigned int i; /* Eine Zaehlvariable kann man immer mal gebrauchen */
122
123    if (c <= 0 || c >= 1) {
124
125      printf ("Dieses c tut mir weh!\n");
126      exit (1);
127
128    }
129
130    /* Hier muss noch mehr Fehlerbehandlung hin */
131
132    /* benoetigte Vektoren erzeugen */
133    basiswichtigkeit = VectorOfDouble_Create (n);
134
135    /* basiswichtigkeit mit Einsen vorbesetzen */
136    for (i = 1; i <= n; i++) {
137
138      VectorOfDouble_SetEntry_1 (basiswichtigkeit, i, 1);
139
140    }
141
142    /* und skalieren */
143    VectorOfDouble_Scale (basiswichtigkeit, c/n);
144
145    MatrixVectorMult_inPlace (x_neu, A, x); /* x_neu <--- A * x;
146                                    ermittelt die eigentliche
147                                                Wichtigkeit, hat aber noch
148                                    Fixpunkt 0 */
149
150    VectorOfDouble_Scale (x_neu, 1-c);
151    /* x_neu <--- (1-c) * x_neu; vermindert die eben berechnete
152       Wichtigkeit ein wenig, damit die Basiswichtigkeit dazuaddiert
153       werden kann */
154
155    VectorOfDouble_Add (x_neu, x_neu, basiswichtigkeit);
156    /* x_neu <--- x_neu + basiswichtigkeit */
157
158  }
```

Graph der Webseiten:



Programmausgabe (Drei separate Aufrufe: `./MatrixTest.out 0.1; ./MatrixTest.out 0.5; ./MatrixTest.out 0.9`):

```
Fixpunktiteration zu c = 0.100

Linkmatrix A =

[ 0.00000000,   0.00000000,   0.85000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.42500000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]
[ 0.00000000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.85000000]
[ 0.42500000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]


w0 = [ 1.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]

w_  1 = [ 0.01666667,   0.39916667,   0.01666667,   0.39916667,   0.01666667,   0.01666667]
w_  2 = [ 0.18209792,   0.02304167,   0.18634792,   0.17997292,   0.16934792,   0.02091667]
w_  3 = [ 0.22806246,   0.08631912,   0.08466507,   0.13831628,   0.08550631,   0.05985038]
w_  4 = [ 0.13434142,   0.10390056,   0.11727338,   0.15872173,   0.06957264,   0.03847077]
w_  5 = [ 0.16709187,   0.06805226,   0.10357980,   0.12553525,   0.07737773,   0.03440769]
w_  6 = [ 0.14392244,   0.08057931,   0.08874986,   0.12634062,   0.06468390,   0.03639799]
w_  7 = [ 0.13288560,   0.07171700,   0.09182711,   0.11903298,   0.06499195,   0.03316106]
w_  8 = [ 0.13244452,   0.06749541,   0.08603958,   0.11150011,   0.06219678,   0.03323961]
w_  9 = [ 0.12513574,   0.06732669,   0.08377214,   0.10900387,   0.05931546,   0.03252685]
w_ 10 = [ 0.12244634,   0.06453109,   0.08242761,   0.10540899,   0.05836065,   0.03179211]
w_ 11 = [ 0.12004272,   0.06350239,   0.08055273,   0.10306749,   0.05698560,   0.03154863]
w_ 12 = [ 0.11771282,   0.06258301,   0.07962236,   0.10140400,   0.05608998,   0.03119800]
w_ 13 = [ 0.11636480,   0.06169182,   0.07877408,   0.09993277,   0.05545370,   0.03096961]
w_ 14 = [ 0.11515312,   0.06117620,   0.07809623,   0.09891402,   0.05489095,   0.03080736]
w_ 15 = [ 0.11424490,   0.06071274,   0.07763139,   0.09810983,   0.05450128,   0.03066386]
w_ 16 = [ 0.11358169,   0.06036534,   0.07724497,   0.09748579,   0.05419368,   0.03056449]
w_ 17 = [ 0.11304738,   0.06011166,   0.07695763,   0.09702079,   0.05395498,   0.03048605]
w_ 18 = [ 0.11264971,   0.05990729,   0.07673973,   0.09665852,   0.05377712,   0.03042519]
```

```
w_ 19 = [ 0.11234444,   0.05975518,   0.07656964,   0.09638288,   0.05363855,   0.03037983]
w_ 20 = [ 0.11210889,   0.05963842,   0.07644142,   0.09617260,   0.05353312,   0.03034450]
w_ 21 = [ 0.11193038,   0.05954832,   0.07634285,   0.09601096,   0.05345269,   0.03031761]
w_ 22 = [ 0.11179314,   0.05948004,   0.07626731,   0.09588770,   0.05339086,   0.03029710]
w_ 23 = [ 0.11168820,   0.05942754,   0.07620973,   0.09579332,   0.05334371,   0.03028134]
w_ 24 = [ 0.11160806,   0.05938740,   0.07616557,   0.09572109,   0.05330761,   0.03026931]
w_ 25 = [ 0.11154664,   0.05935675,   0.07613181,   0.09566587,   0.05327998,   0.03026011]


Fixpunktiteration zu c = 0.500

Linkmatrix A =

[ 0.00000000,   0.00000000,   0.85000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.42500000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]
[ 0.00000000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.85000000]
[ 0.42500000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]


w0 = [ 1.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]

w_  1 = [ 0.08333333,   0.29583333,   0.08333333,   0.29583333,   0.08333333,   0.08333333]
w_  2 = [ 0.18161458,   0.10104167,   0.19342014,   0.17571181,   0.14619792,   0.09513889]
w_  3 = [ 0.20287565,   0.12192643,   0.16595009,   0.16410916,   0.12067209,   0.10404470]
w_  4 = [ 0.18873532,   0.12644441,   0.17055691,   0.16944899,   0.11820653,   0.10042855]
w_  5 = [ 0.19182793,   0.12343959,   0.16963083,   0.16705495,   0.11934124,   0.10007926]
w_  6 = [ 0.19092561,   0.12409677,   0.16900461,   0.16723436,   0.11883251,   0.10024001]
w_  7 = [ 0.19069759,   0.12390503,   0.16914051,   0.16711019,   0.11887063,   0.10016794]
w_  8 = [ 0.19072896,   0.12385657,   0.16907453,   0.16702640,   0.11884425,   0.10017334]
w_  9 = [ 0.19068312,   0.12386324,   0.16906279,   0.16701903,   0.11882644,   0.10016960]


Fixpunktiteration zu c = 0.900

Linkmatrix A =

[ 0.00000000,   0.00000000,   0.85000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.42500000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]
[ 0.00000000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.85000000]
[ 0.42500000,   0.42500000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.42500000,   0.00000000,   0.00000000]
[ 0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.28333333,   0.00000000]


w0 = [ 1.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000,   0.00000000]

w_  1 = [ 0.15000000,   0.19250000,   0.15000000,   0.19250000,   0.15000000,   0.15000000]
w_  2 = [ 0.17093125,   0.15637500,   0.17518125,   0.16880625,   0.15818125,   0.15425000]
w_  3 = [ 0.17206467,   0.15726458,   0.17423899,   0.16839232,   0.15717427,   0.15448180]
w_  4 = [ 0.17196699,   0.15731275,   0.17426797,   0.16844976,   0.15715667,   0.15445327]
w_  5 = [ 0.17197189,   0.15730860,   0.17426709,   0.16844716,   0.15715911,   0.15445277]
```

Diskussion.

Fällt, auch mangels Zeit, kurz aus: Es wird keinerlei Aussage über den Inhalt der Webseiten getroffen; der Algorithmus arbeitet allein aufgrund der Verlinkung der Seiten. Der Parameter c hat erheblichen Einfluss auf das Ranking; es wäre schön zu wissen, woran das genau liegt. Ich halte daher den hier vorgestellten Algorithmus ohne Kenntnis der Bedeutung der Seiten und ohne tiefgehendes Wissen über c und α, in seiner Aussagekraft für begrenzt.