Graphentheorie 1

Mathematik in wxMaxima www.mathematik-verstehen.de Haftendorn April 2011

0.1 Handling

0.2 Inhalt

1 Graphen erzeugen und zeichnen

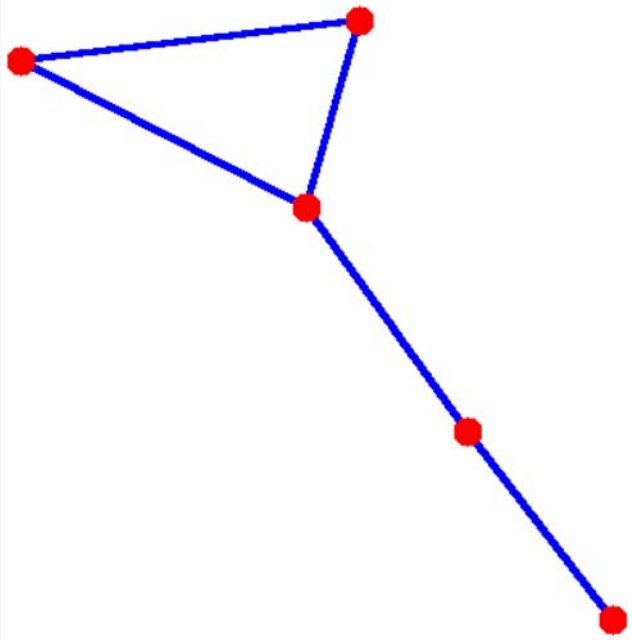
```
(%i3) load(graphs)$
      load(draw)$
```

Diese beiden Bibliotheken muss man unbedingt zu Beginn laden. (doppelt absch
In der Hilfe findet man mit dem Suchwort graphs das Kapitel 53.
Dort stehen alle Funktionen zum Erzeugen, Zeichnen und Untersuchen von Graph
Hier treffe ich eine Auswahl, die zu meinem Lehrplan (Fachwissenschaft, Gym-

```
--> my:random_graph(5,0.3);
(%o5) Structure [GRAPH]
```

Hier wird ein Zufallsgraph mit 5 ecken erzeugt, bei dem zufällig
30% der möglichen Kanten gesetzt werden

```
--> draw_graph(my,vertex_size=3,show_edges=edges(my),show_edge_width=4);
```



(%t14)

(%o14) *done*

Im Zeichenbefehl sind gleich noch einige Eigenschaften gesetzt.

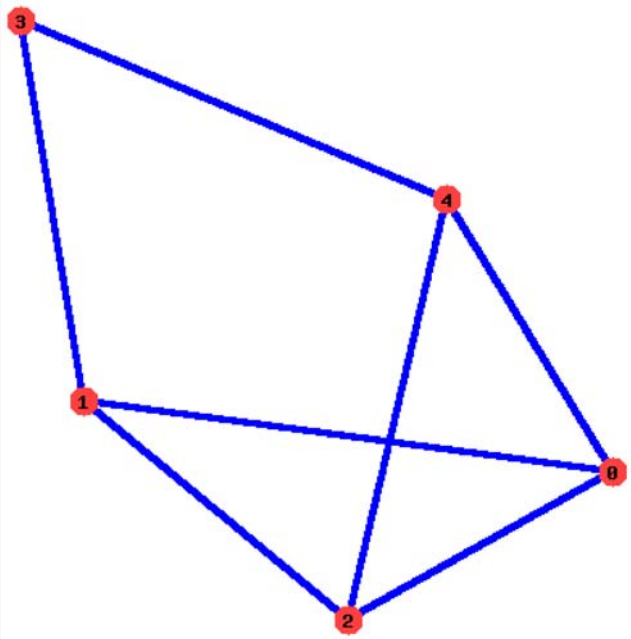
Reparaturfunktion abschicken !!!

```
(%i5) adjazenz_matrix(g):=block(local(A,BB,i,j,li),
                A:adjacency_matrix(g),
                li:matrix_size(A),
                array(BB,fixnum,li[1],li[2]),
                for i:1 thru li[2] do
                    for j:1 thru li[1] do
                        BB[i,j]:A[li[1]-i+1,li[2]-j+1],
                    endfor,
                endfor,
            genmatrix(BB,li[1],li[2])
        )$
```

1.1 Spielwiese zum Ausprobieren

```
--> sp:my:random_graph(5,0.6)$
    draw_graph(sp,vertex_size=3,show_id=true,
                  show_edges=edges(my),show_edge_width=4)$
```

(%t22)



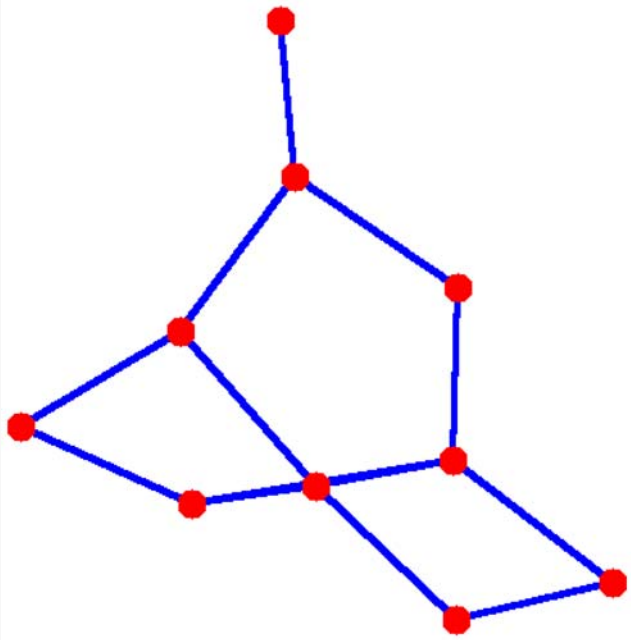
1.2 Einige Graphen zum Lernen

Achtung, bei erneuter Auswertung würden diese Graphen sich verändern!

```
--> my:random_graph(10,0.3)
```

```
--> draw_graph(my,vertex_size=3,show_edges=edges(my),show_edge_width=4);
```

(%t12)



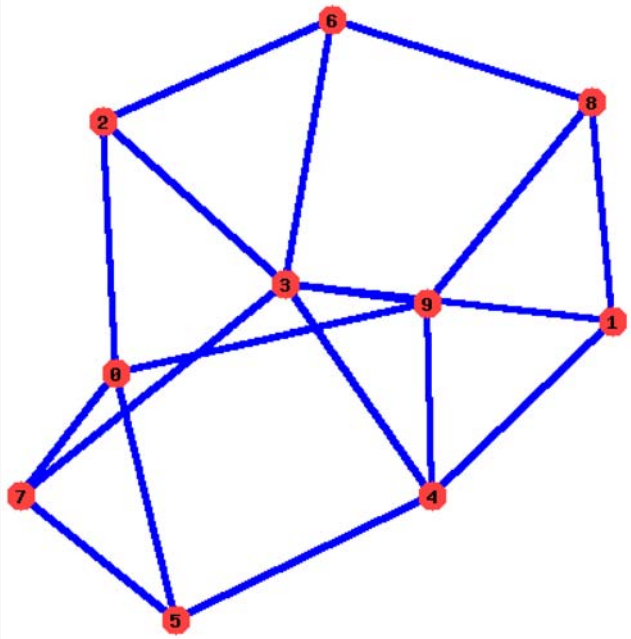
(%o12) *done*

```
--> my:random_graph(10,0.4);
```

(%o20) *Structure [GRAPH]*

```
--> draw_graph(my,vertex_size=3,show_id=true,
    show_edges=edges(my),show_edge_width=4);
```

(%t22)



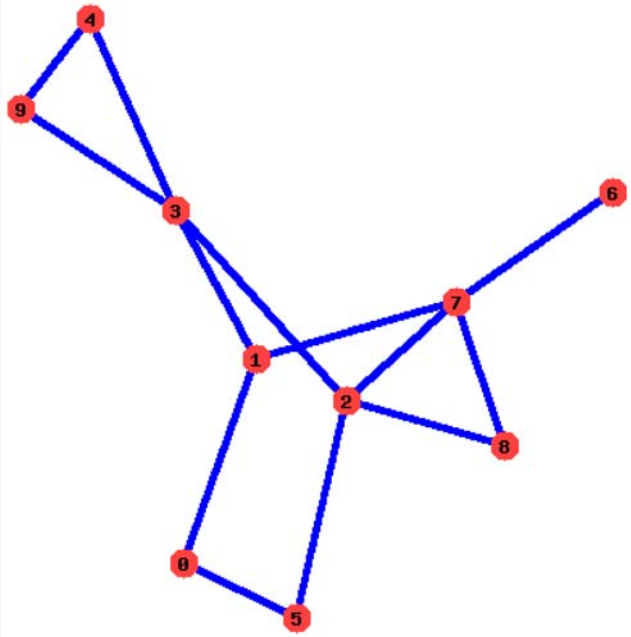
(%o22) *done*

```
--> my:random_graph(10,0.2);
```

(%o33) *Structure [GRAPH]*

```
--> draw_graph(my,vertex_size=3,show_id=true,
    show_edges=edges(my),show_edge_width=4);
```

(%t34)



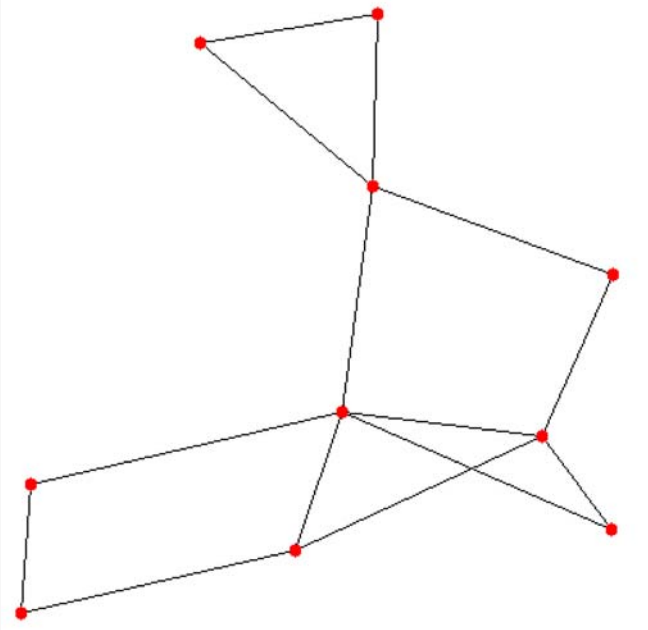
(%o34) *done*

```
--> my:random_graph(10,0.3);
```

(%o11) *Structure [GRAPH]*

```
--> draw_graph(my);
```

(%t16)

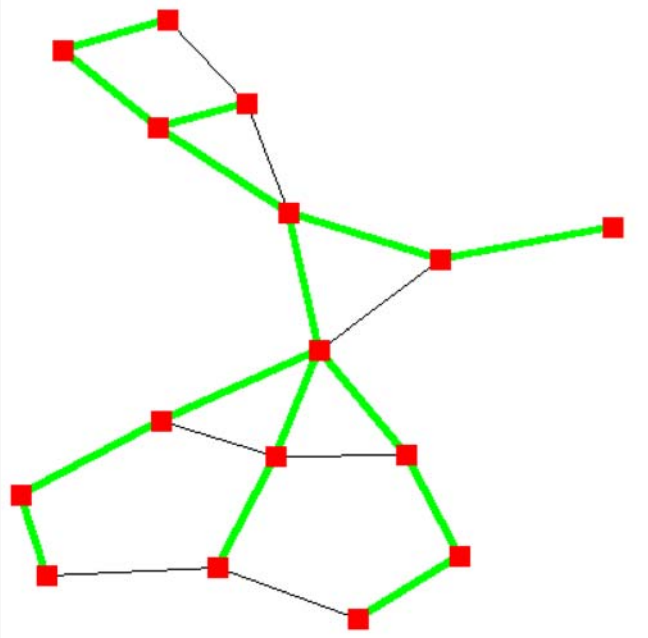


(%o16) *done*

Von dem nachfolgenden Eintrag aus der Hilfe habe ich die Optionen abgeguckt.

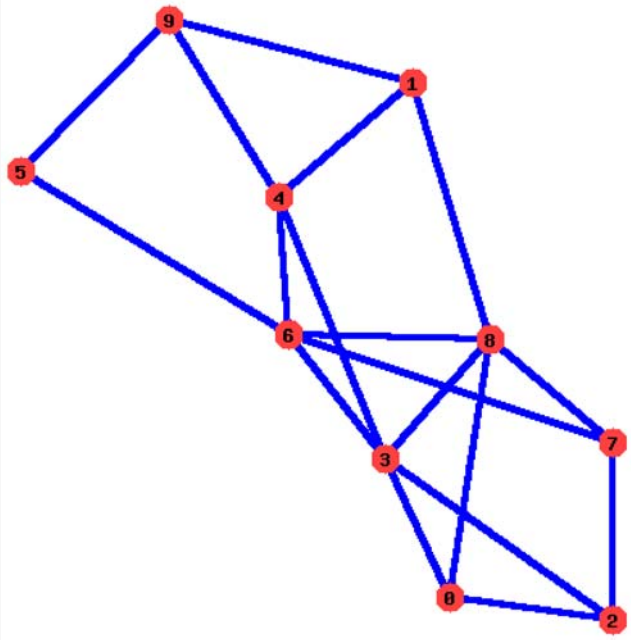
```
--> g:create_graph(16,
        [
         [0,1],[1,3],[2,3],[0,2],[3,4],[2,4],
         [5,6],[6,4],[4,7],[6,7],[7,8],[7,10],[7,11],
     [8,10],[11,10],[8,9],[11,12],[9,15],[12,13],
         [10,14],[15,14],[13,14]
        ])$
     t:minimum_spanning_tree(g)$
    draw_graph(g,
        show_edges=edges(t), show_edge_width=4,
        show_edge_color=green,
        vertex_type=filled_square, vertex_size=2
        )$
```

(%t54)



```
--> my:random_graph(10,0.4);
```
(%o42) *Structure [GRAPH]*

```
--> draw_graph(my,vertex_size=3,show_id=true,
    show_edges=edges(my),show_edge_width=4);
```
(%t43)



(%o43) *done*

1.3 Reparatur der Anzeige der Adjazenzmatrix und Erzeugen mit einer Adjazenzmatix

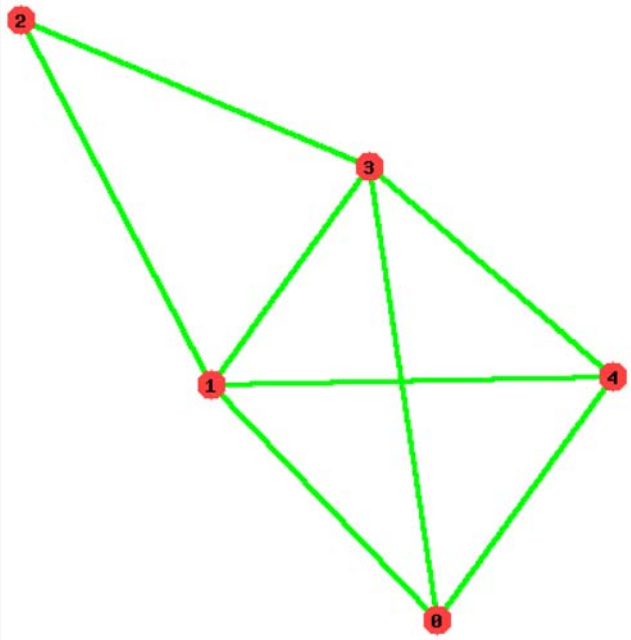
1.4 Anzeige der Adjazenzmatrix und Erzeugen mit einer Adjazenzmatix

```
Aufstellen einer Matrix
```

```
--> A:matrix( [0,1,0,1,1],[1,0,1,1,1],[0,1,0,1,0] ,[1,1,1,0,0] ,[1,1,0,1,0]    );
```
(%o186)
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

```
--> amy:from_adjacency_matrix (A)$
    draw_graph(amy,show_id=true, edge_color=green, edge_width=3);
```

(%t194)



(%o194) *done*

```
--> print_graph(amy);adjazenz_matrix(amy);
```

Graph on 5 vertices with 8 edges.
Adjacencies:
4 : 3 1 0
3 : 4 2 1 0
2 : 3 1
1 : 4 3 2 0
0 : 4 3 1

(%o394) *done*

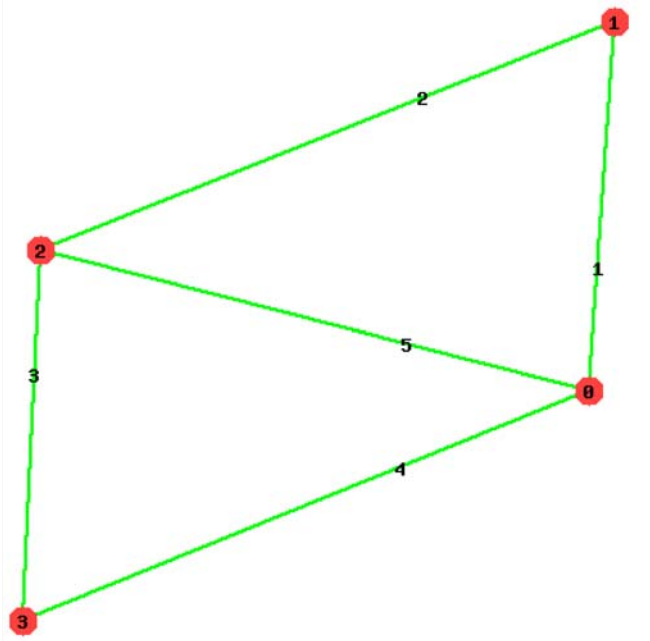
(%o395)
$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

2 Erzeugung eigener Graphen mit Gewichtung

2.1 Kleiner gewichteter Graph

```
--> g : create_graph([0,1,2,3],[[[0,1], 1], [[1,2], 2], [[2,3], 3],
                                [[3,0], 4],[[0,2], 5]])$
    draw_graph(g,show_id=true, edge_color=green,
               edge_width=2,show_weight=true);
    print_graph(g)$
```

(%t416)



(%o416) *done*

Graph on 4 vertices with 5 edges.
Adjacencies:
3 : 0 2
2 : 0 3 1
1 : 2 0
0 : 2 3 1

```
--> A:adjazenz_matrix(g); AA:adjacency_matrix(g);
```

(%o407)
$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

(%o408)
$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

```
Die reparierte Adjazenzmatix ist richtig,
die vom Originalbefehl ausgegebene Adjazenzmatrix ist rückwärts sortiert.
Sie würde beim Konstruieren nicht auf den ursprünglichen Graphen führen.
```

2.2 Spannbaum und kürzeste Wege im Lüneburg-Graphen, S.

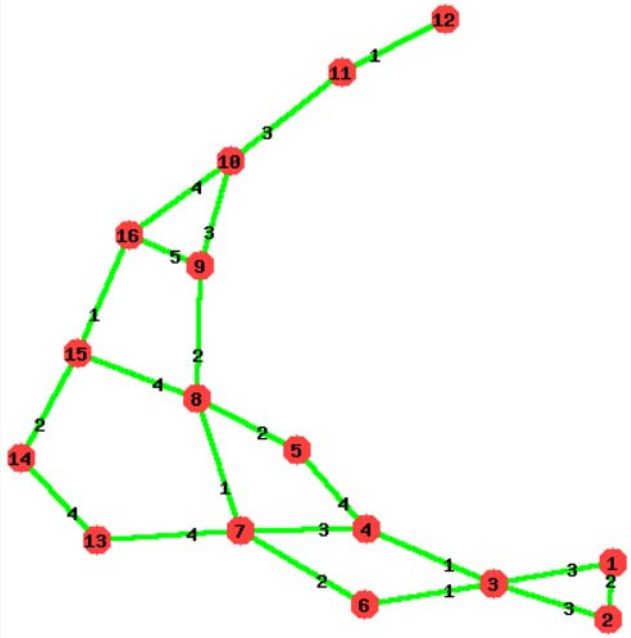
```
(%i36) eckenglg:[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]$
```

```
(%i37) kantenglg:[[[1,2],2],[[1,3],3],[[2,3],3],[[3,4],1],[[4,5],4],[[3,6],1],[[4,7],3],[[5,8],2],[[6,7],2],[
       [[8,9],2],[[9,10],3],[[10,11],3],[[11,12],1],[[7,13],4],[[8,15],4],[[9,16],5],[[10,16],4],[[13,14],4
```

```
(%i38) glg:create_graph(eckenglg,kantenglg)$
```

```
(%i39) draw_graph(glg,show_id=true,show_weight=true,edge_color=green,edge_width=3);
```

(%t39)




```
(%o39) done
```

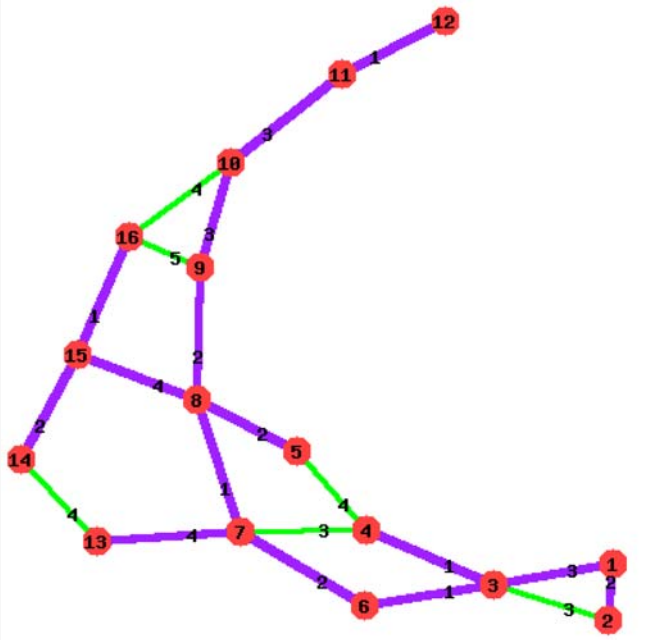
```
(%i40) t : minimum_spanning_tree(glg);print_graph(t);
(%o40) Structure [GRAPH]
```

Graph on 16 vertices with 15 edges.
Adjacencies:
1 : 3 2
2 : 1
3 : 1 6 4
4 : 3
5 : 8
6 : 7 3
7 : 13 6 8
8 : 15 9 5 7
9 : 10 8
10 : 11 9
11 : 10 12
12 : 11
13 : 7
14 : 15
15 : 8 14 16
16 : 15

```
(%o41) done
```

```
(%i42) draw_graph(glg,show_id=true,show_weight=true,edge_color=green,edge_width=3
                  ,show_edges=edges(t),show_edge_color=purple,show_edge_width=5
       );
```

(%t42)



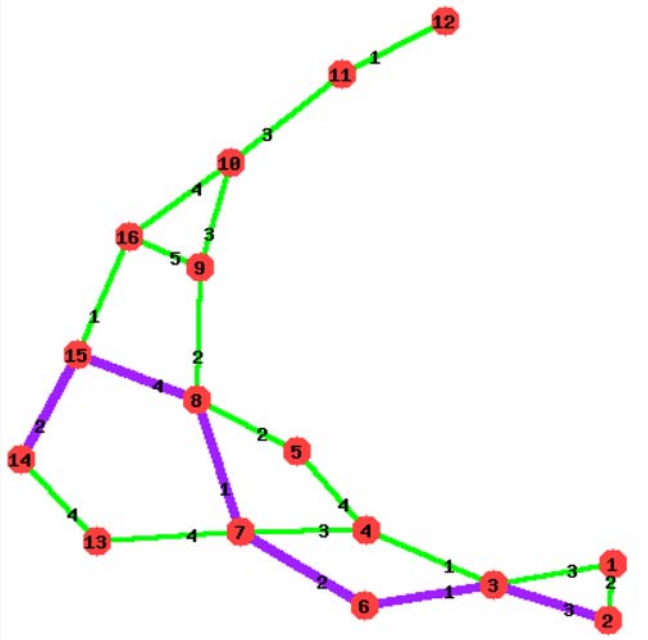
(%o42) *done*

```
(%i43) for i:1 thru 16  do print(shortest_weighted_path(14, i, glg)),endfor;
[13,[14,15,8,7,6,3,1]]
[13,[14,15,8,7,6,3,2]]
[10,[14,15,8,7,6,3]]
[10,[14,15,8,7,4]]
[8,[14,15,8,5]]
[9,[14,15,8,7,6]]
[7,[14,15,8,7]]
[6,[14,15,8]]
[8,[14,15,16,9]]
[7,[14,15,16,10]]
[10,[14,15,16,10,11]]
[11,[14,15,16,10,11,12]]
[4,[14,13]]
[0,[14]]
[2,[14,15]]
[3,[14,15,16]]
(%o43) done
```

```
(%i44) weglg:[[2,3],[3,6],[6,7],[7,8],[8,15],[15,14]]$
```

(%i45)
```
draw_graph(glg,show_id=true,show_weight=true,edge_color=green,edge_width=3
    ,show_edges=weglg,show_edge_color=purple,show_edge_width=5
    );
```

(%t45)



(%o45) *done*

2.3 Der Spannbaum und Dijkstra-Graph aus meinem Buch, S.

(%i46)
```
ecken:[[1,"A"],[2,"B"],[3,"C"],[4,"D"],[5,"E"],[6,"F"],[7,"G"],[8,"H"],[9,"I"]];
```

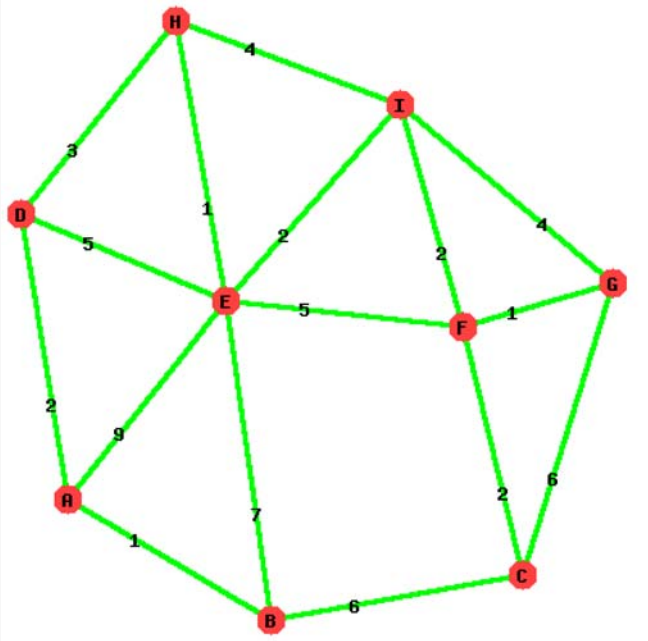
(%o46) $[[1,A],[2,B],[3,C],[4,D],[5,E],[6,F],[7,G],[8,H],[9,I]]$

(%i47)
```
kanten:[[[1,2],1],[[1,5],9],[[1,4],2],[[2,3],6],[[2,5],7],[[3,6],2],[[3,7],6],
    [[4,5],5],[[4,8],3],[[5,6],5],[[5,9],2],[[5,8],1],[[6,7],1],[[6,9],2],[[7,9],4],[[8,9],4]]$
```

(%i48)
```
gb:create_graph(ecken,kanten)$
```

(%i49)
```
draw_graph(gb,show_label=true,show_weight=true,edge_color=green,edge_width=3);
```

(%t49)



(%o49) *done*

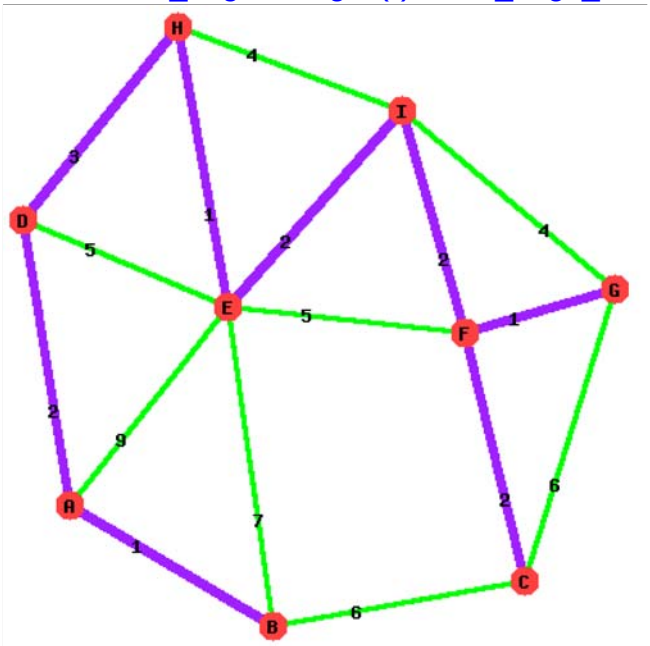
Leider bekomme ich die Schrift nicht größer hin.

Spannbaum

```
(%i50) t : minimum_spanning_tree(gb);print_graph(t);
(%o50) Structure [GRAPH]
Graph on 9 vertices with 8 edges.
Adjacencies:
1 : 4 2
2 : 1
3 : 6
4 : 8 1
5 : 9 8
6 : 9 3 7
7 : 6
8 : 4 5
9 : 6 5
(%o51) done
```

```
(%i52) draw_graph(gb,show_label=true,show_weight=true,edge_color=green,edge_width=3
       ,show_edges=edges(t),show_edge_color=purple,show_edge_width=5 );
```

(%t52)



(%o52) done

So ist es auch auf Seite 65 gefunden.

Kürzeste Wege direkt von Maxima:

```
--> for i:1 thru 9  do print(shortest_weighted_path(1, i, gb)),endfor;
[0,[1]]
[1,[1,2]]
[7,[1,2,3]]
[2,[1,4]]
[6,[1,4,8,5]]
[9,[1,2,3,6]]
[10,[1,2,3,6,7]]
[5,[1,4,8]]
[8,[1,4,8,5,9]]
(%o494) done
```

```
--> weg:shortest_weighted_path(1, 9, gb);
(%o495) [8,[1,4,8,5,9]]
```

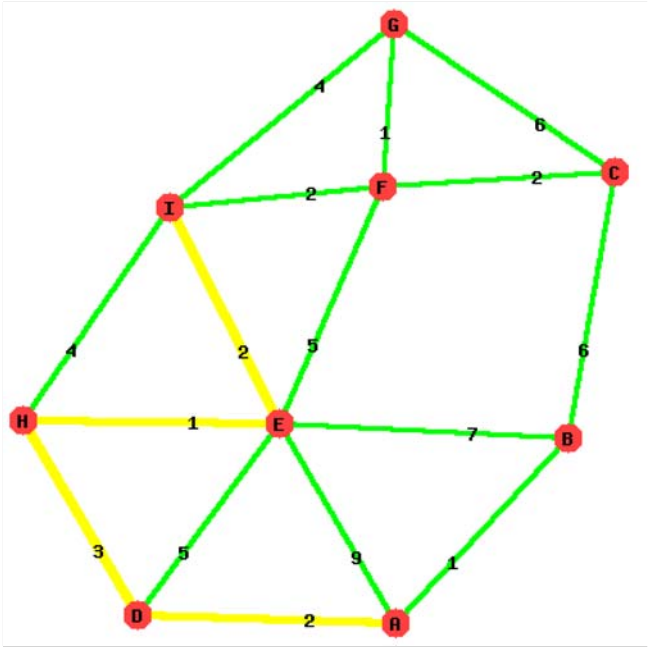
Von Hand habe ich daraus Kantenzüge gemacht.

```
--> dijweg:[[1,4], [4,8],[8,5],[5,9]];dijweg2:[[1,2],[2,3],[3,6],[6,7]];
(%o508) [[1,4],[4,8],[8,5],[5,9]]
(%o509) [[1,2],[2,3],[3,6],[6,7]]
```

Man kann einen Teilgraphen anders Färben aber leider nicht zwei Teilgraphen.

```
--> draw_graph(gb,show_label=true,show_weight=true,edge_color=green,edge_width=3,
      show_edges=dijweg,show_edge_color=yellow,show_edge_width=5
     /* ,show_edges=dijweg2,show_edge_color=purple,show_edge_width=5*/
      );
```

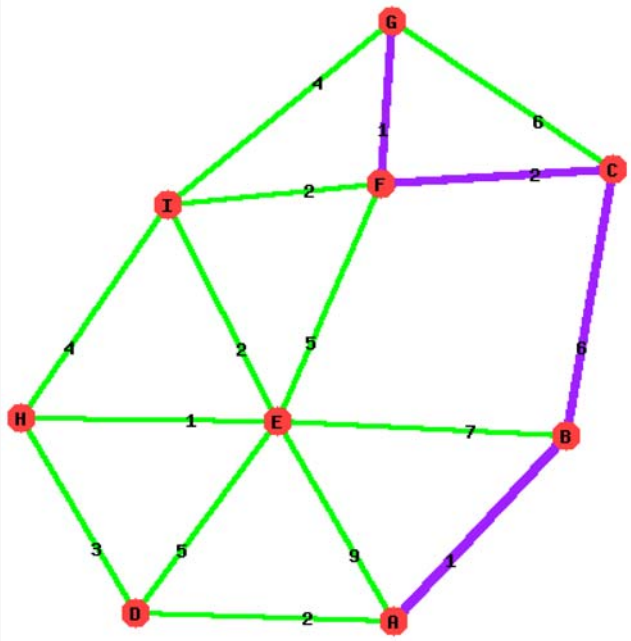
(%t511)



(%o511) done

```
--> draw_graph(gb,show_label=true,show_weight=true,edge_color=green,edge_width=3
      /*, show_edges=dijweg,show_edge_color=yellow,show_edge_width=5  */
      ,show_edges=dijweg2,show_edge_color=purple,show_edge_width=5
      );
```

(%t512)



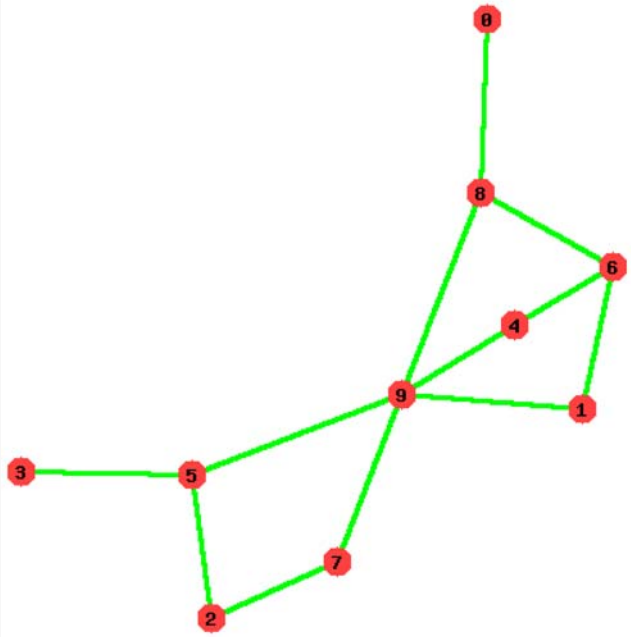
(%o512) *done*

2.4 Weitere Graphen und Spannbäume

(%i59)
```
gsp:random_graph(10,0.3)$
```

(%i61)
```
draw_graph(gsp,show_id=true,edge_color=green,edge_width=3);
```

(%t61)



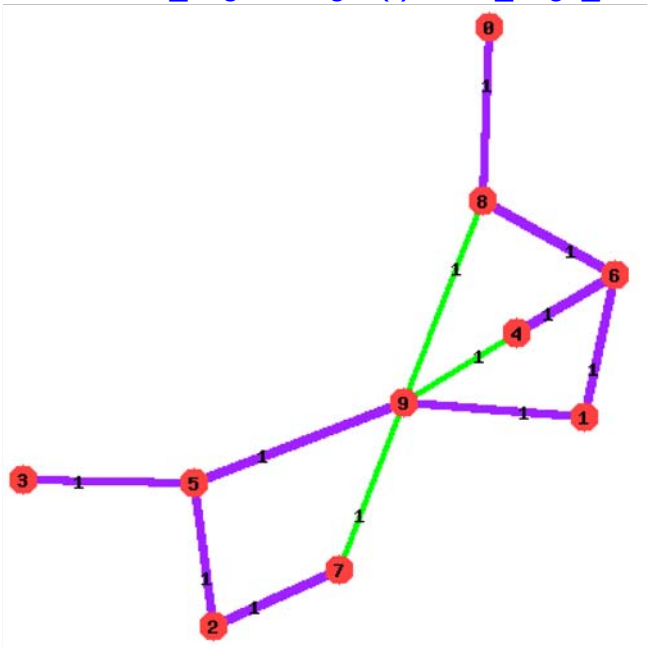
(%o61) *done*

Spannbaum

```
(%i64) t : minimum_spanning_tree(gsp);print_graph(t);
(%o64) Structure [GRAPH]
Graph on 10 vertices with 9 edges.
Adjacencies:
 0 : 8
 1 : 9 6
 2 : 7 5
 3 : 5
 4 : 6
 5 : 9 3 2
 6 : 8 4 1
 7 : 2
 8 : 6 0
 9 : 5 1
(%o65) done
```

```
(%i66) draw_graph(gsp,show_id=true,show_weight=true,edge_color=green,edge_width=3
       ,show_edges=edges(t),show_edge_color=purple,show_edge_width=5 );
```

(%t66)




```
(%o66) done
```

```
Wenn ich mir das ansehe, glaube ich eher, dass "Tiefensuche" programmiert is
```

3 Bipartite und vollständige Graphen

3.1 Erzeugung

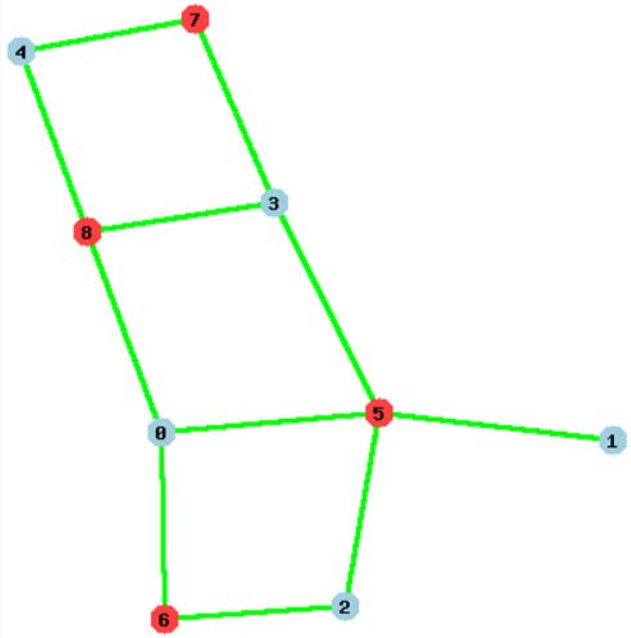
```
(%i93) big:random_bipartite_graph (5, 4, 0.8);
(%o93) Structure [GRAPH]
```

```
(%i94) [A,B]:bipartition(big);
(%o94) [[2,1,0,3,4],[8,6,5,7]]
```

```
(%i95) print_graph(big);
```
Graph on 9 vertices with 11 edges.
Adjacencies:
8 : 4 3 0
7 : 4 3
6 : 2 0
5 : 3 2 1 0
4 : 8 7
3 : 8 7 5
2 : 6 5
1 : 5
0 : 8 6 5

(%o95) *done*

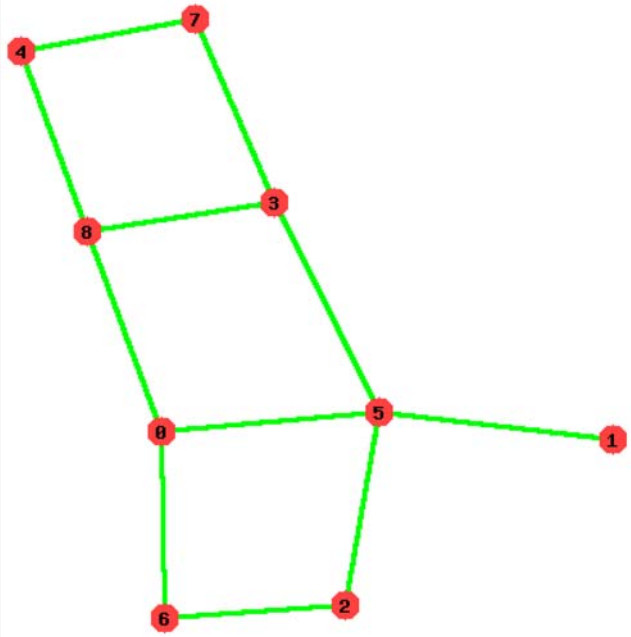
```
(%i96) draw_graph(big,show_id=true,edge_color=green,edge_width=3,show_vertices=A);
```
(%t96)



(%o96) *done*

(%i97) draw_graph(big,show_id=true,edge_color=green,edge_width=3);

(%t97)



(%o97) *done*

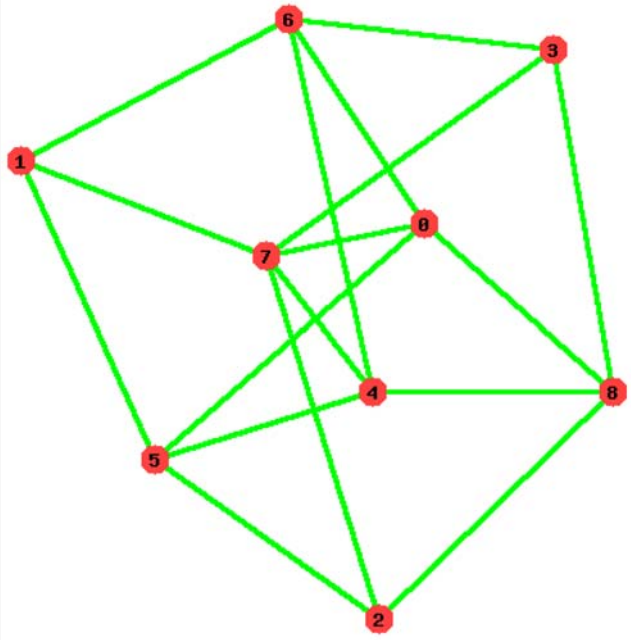
Noch einmal

(%i98) big:random_bipartite_graph (5, 4, 0.8);

(%o98) *Structure [GRAPH]*

(%i102) draw_graph(big,show_id=true,edge_color=green,edge_width=3);

(%t102)



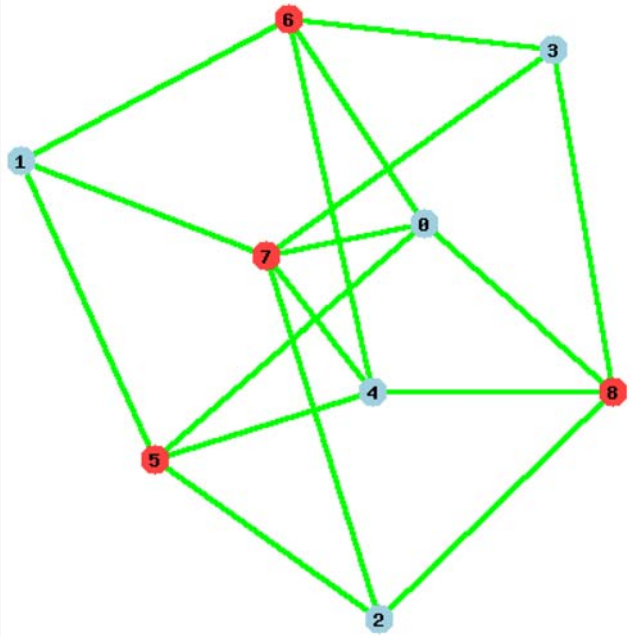
(%o102) *done*

(%i99) [A,B]:bipartition(big);

(%o99) [[3,2,1,0,4],[8,7,6,5]]

```
(%i103) print_graph(big);
```
Graph on 9 vertices with 17 edges.
Adjacencies:
8 : 4 3 2 0
7 : 4 3 2 1 0
6 : 4 3 1 0
5 : 4 2 1 0
4 : 8 7 6 5
3 : 8 7 6
2 : 8 7 5
1 : 7 6 5
0 : 8 7 6 5
(%o103) *done*

```
(%i104) draw_graph(big,show_id=true,edge_color=green,edge_width=3,show_vertices=A);
```
(%t104)



(%o104) *done*

Wenn ich mir das so ansehe, ist [0,3,4] zu [8,7,6] ein vollständiger bipartiter Teilgraph. Damit ist der Graph nicht planar.

```
(%i105) is_planar(big);
```
(%o105) false

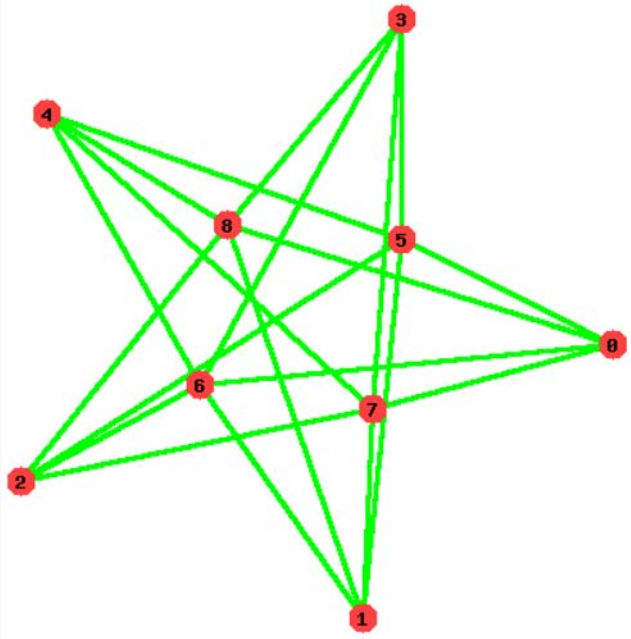
War ja klar.

3.2 Vollständige Graphen

```
(%i106) bipvo:complete_bipartite_graph (5, 4);
```
(%o106) *Structure [GRAPH]*

(%i108) draw_graph(bipvo,show_id=true,edge_color=green,edge_width=3);

(%t108)



(%o108) *done*

(%i109) [A,B]:bipartition(bipvo);

(%o109) [[3,2,1,0,4],[8,7,6,5]]

(%i110) print_graph(bipvo);

Graph on 9 vertices with 20 edges.

Adjacencies:

8 : 4 3 2 1 0

7 : 4 3 2 1 0

6 : 4 3 2 1 0

5 : 4 3 2 1 0

4 : 8 7 6 5

3 : 8 7 6 5

2 : 8 7 6 5

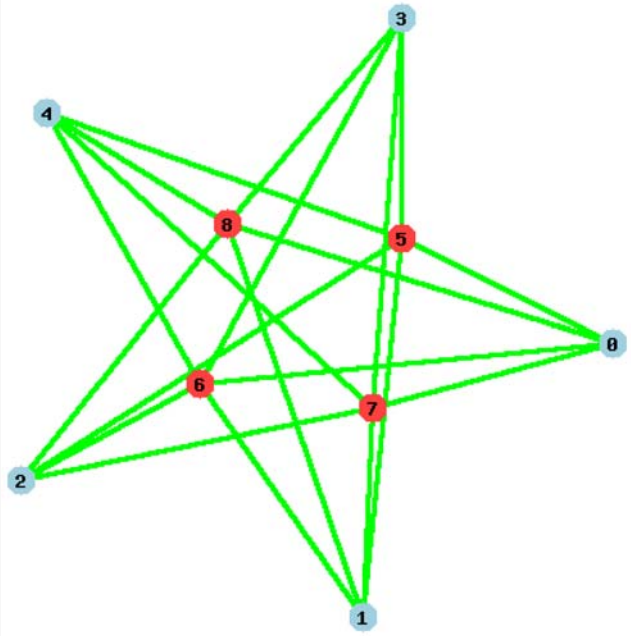
1 : 8 7 6 5

0 : 8 7 6 5

(%o110) *done*

(%i111)
```
draw_graph(bipvo,show_id=true,edge_color=green,edge_width=3,show_vertices=A);
```

(%t111)

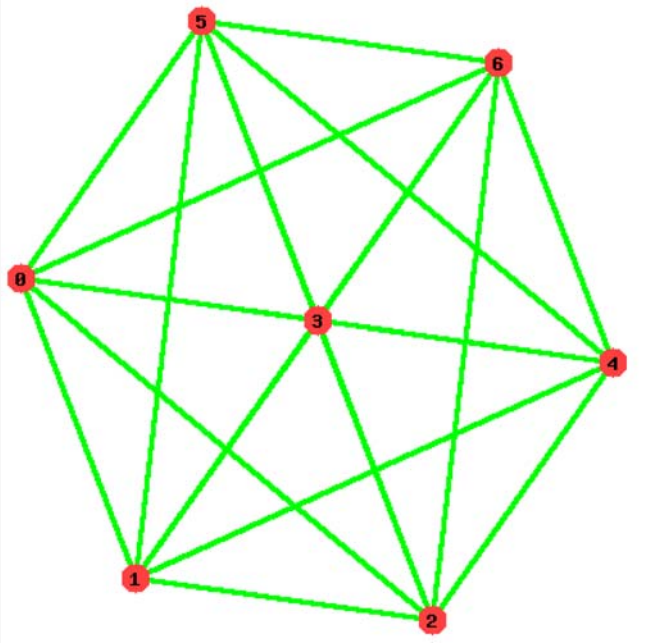


(%o111) *done*

(%i116)
```
vo:complete_graph (7);
draw_graph(vo,show_id=true,edge_color=green,edge_width=3);
```

(%o116) *Structure [GRAPH]*

(%t117)



(%o117) *done*

4 Graphen mit besonderen Namen

Noch nicht ausgeführt. In der Hilfe steht etliches.