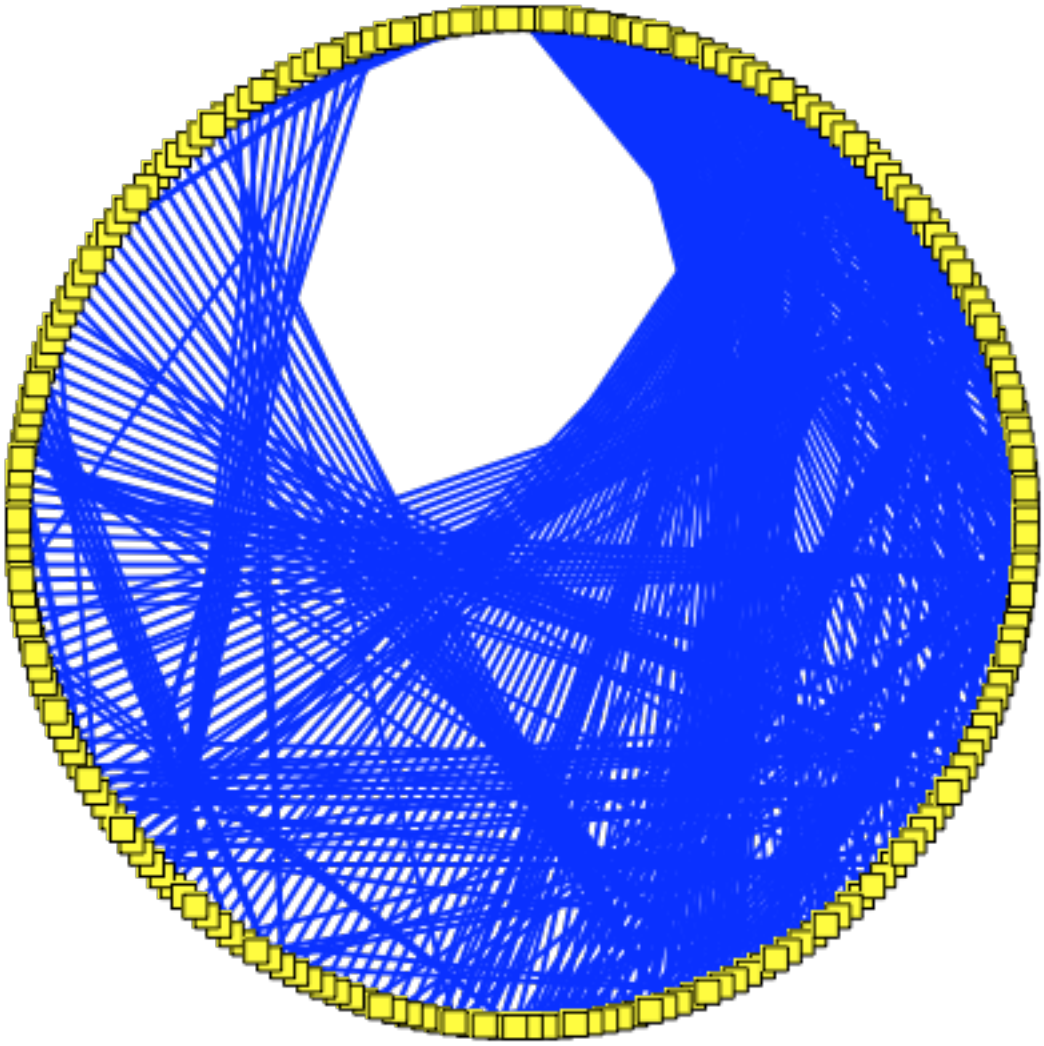
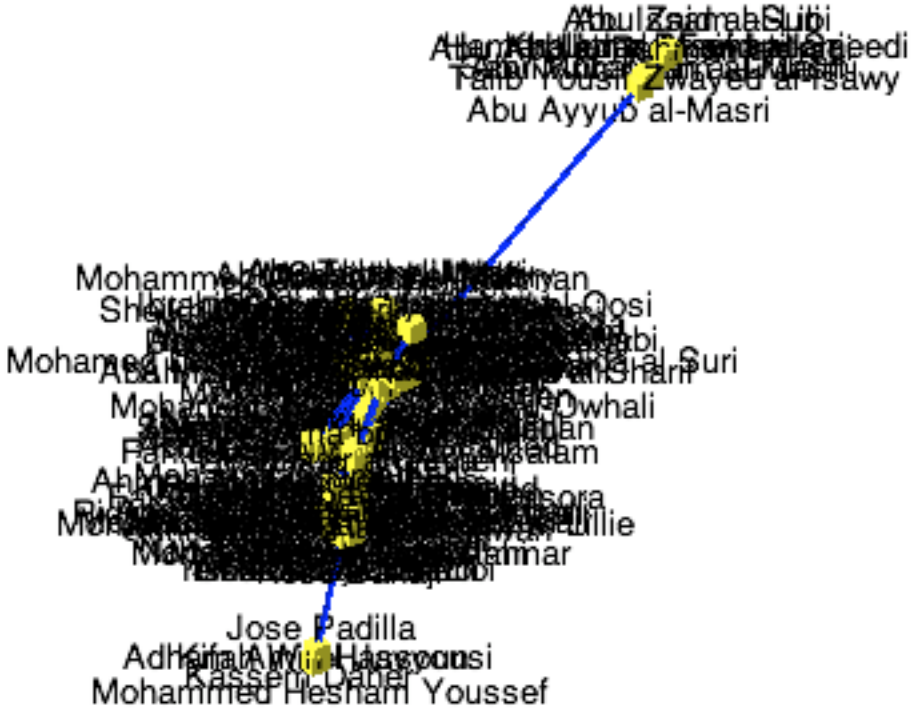
```
Data := ImportMatrix("/Users/sidafa/Documents/Adelaide/Thesis/Matrix.csv", source = csv):
convert(Data, 'matrix'):
Names := ImportMatrix("/Users/sidafa/Documents/Adelaide/Thesis/Names.csv", source = csv):
V := convert(Names,'list'):
with(GraphTheory):
with(networks):
with(Statistics):
G := Graph(undirected, V, Data):
n := nops(Vertices(G)):
m := nops(Edges(G)):
DrawGraph(G)
```



```
DrawGraph(G, style = spring, dimension = 3)
```



DrawGraph (*MinimalSpanningTree* (*G*), *showlabels* = *true*)

$Deg := sort(DegreeSequence(G)):$

$AvgDeg := \mathbf{proc}(n);$

$\quad \mathbf{local}\ k, x;$

$\quad x := 0:$

$\quad \mathbf{for}\ k\ \mathbf{from}\ 1\ \mathbf{to}\ 209\ \mathbf{do}$

$\qquad x := \frac{x + n[k]}{2}\ \mathbf{end\ do};$

$evalf(x);$

$\mathbf{end\ proc}:$

$AvgDeg(Deg)$

$$49.05126345 \qquad (1)$$

$MaxDeg := Deg[209]$

$$59 \qquad (2)$$

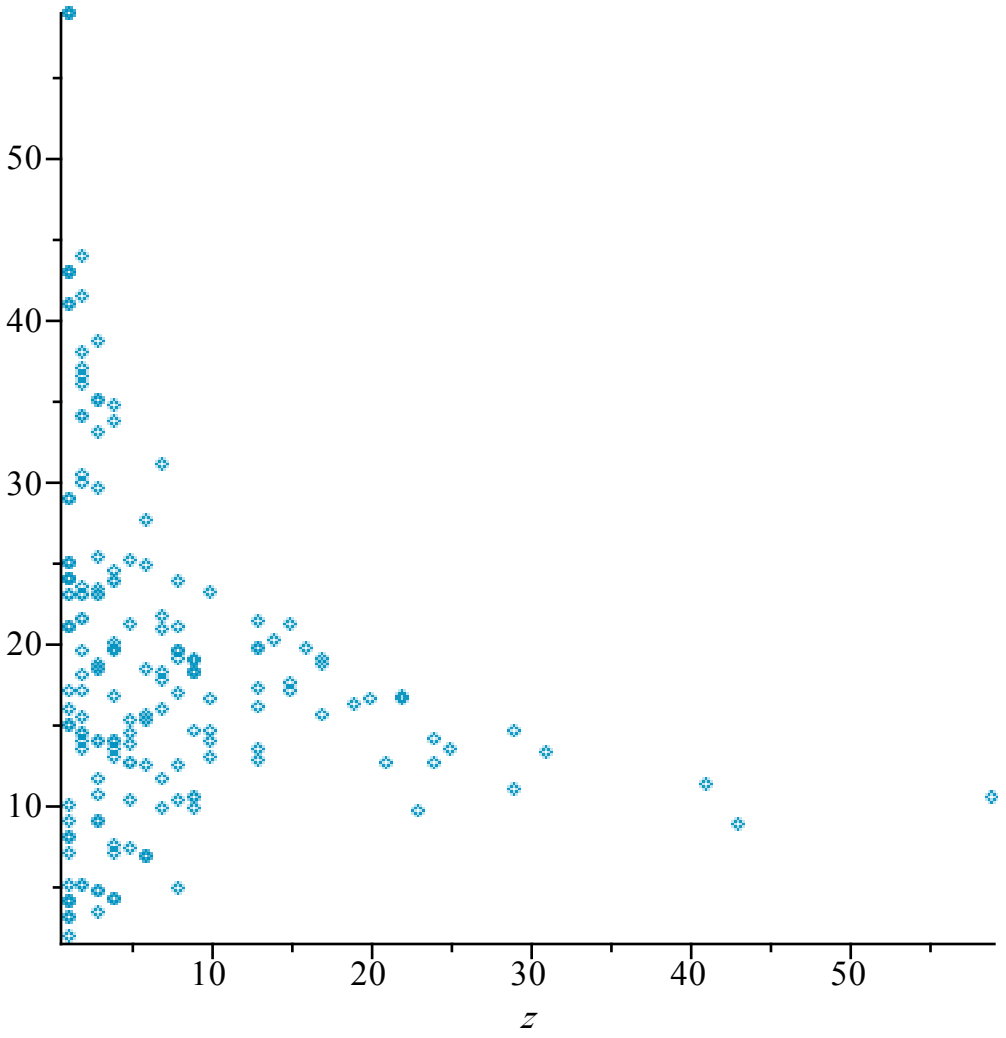
$$LinkDensity := evalf\left(\frac{2 \cdot m}{n \cdot (n-1)}\right)$$

$$0.03045638572 \qquad (3)$$

$Q := Vector(n):$

$T := Vector(n):$

```
for k from 1 to n do
 counter := 0;
     x := Neighbors(G, V[k]);
     y := nops(x);
 T[k] := y;
        for j from 1 to y do
         counter := counter + Degree(G, x[j]);
 end do ;
Q[k] := evalf(counter/y);
 end do:
ScatterPlot(T, Q)
```




```
LinearFit([1, t], T, Q, t)
```

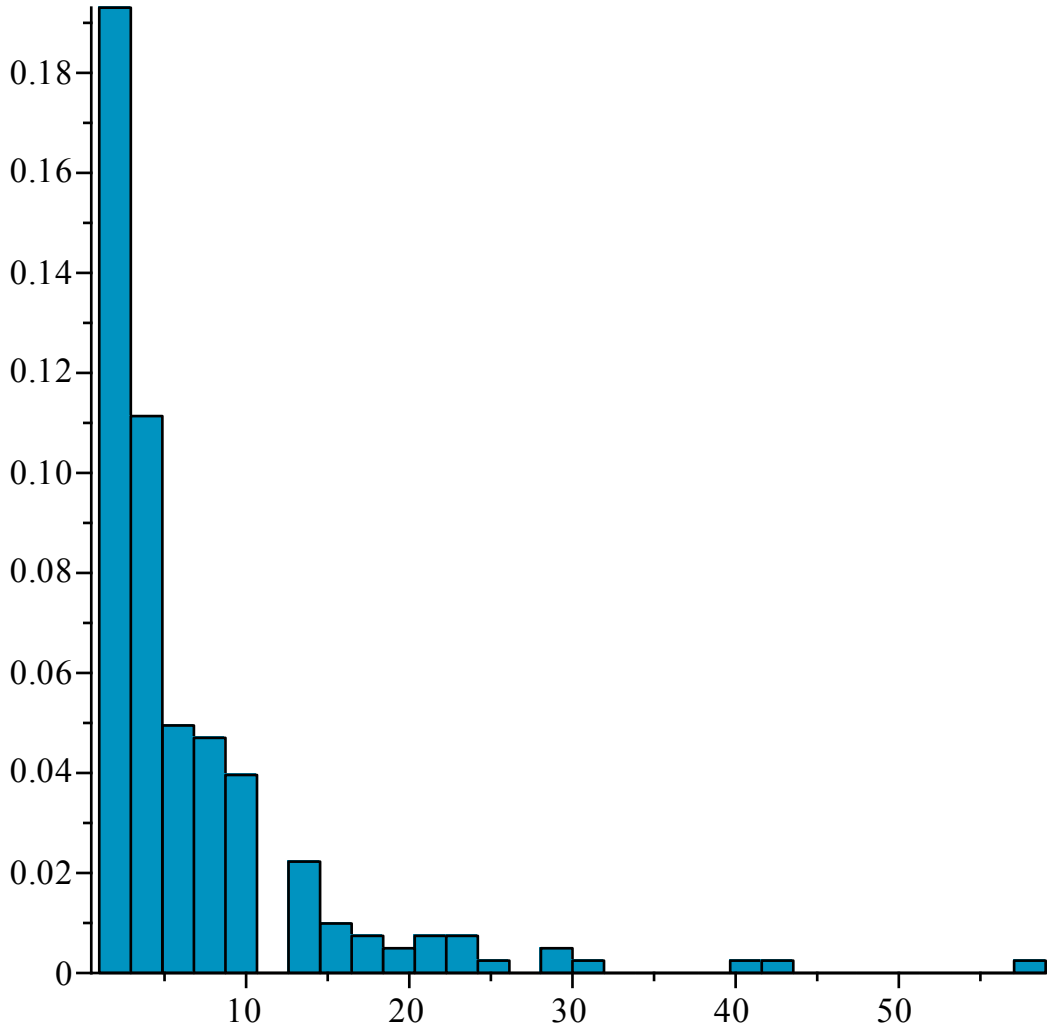
$$21.2980942847207970 - 0.325001260806376369\,t \qquad \textbf{(4)}$$

```
AveragePath := proc(G, V);
 local j, k, sum, counter;
 counter := 0;
 sum := 0;
 for j from 1 to 209 do
```

```
  for k from 1 to 209 do
    if j ≠ k then
       sum := sum + Distance(G, V[j], V[k]);
      counter := counter + 1;
  end if
  end do
end do;
print(sum, counter, evalf(sum/counter));
end proc:
AveragePath(G, V)
```

139790, 43472, 3.215633051 **(5)**

```
Histogram(Deg)
```

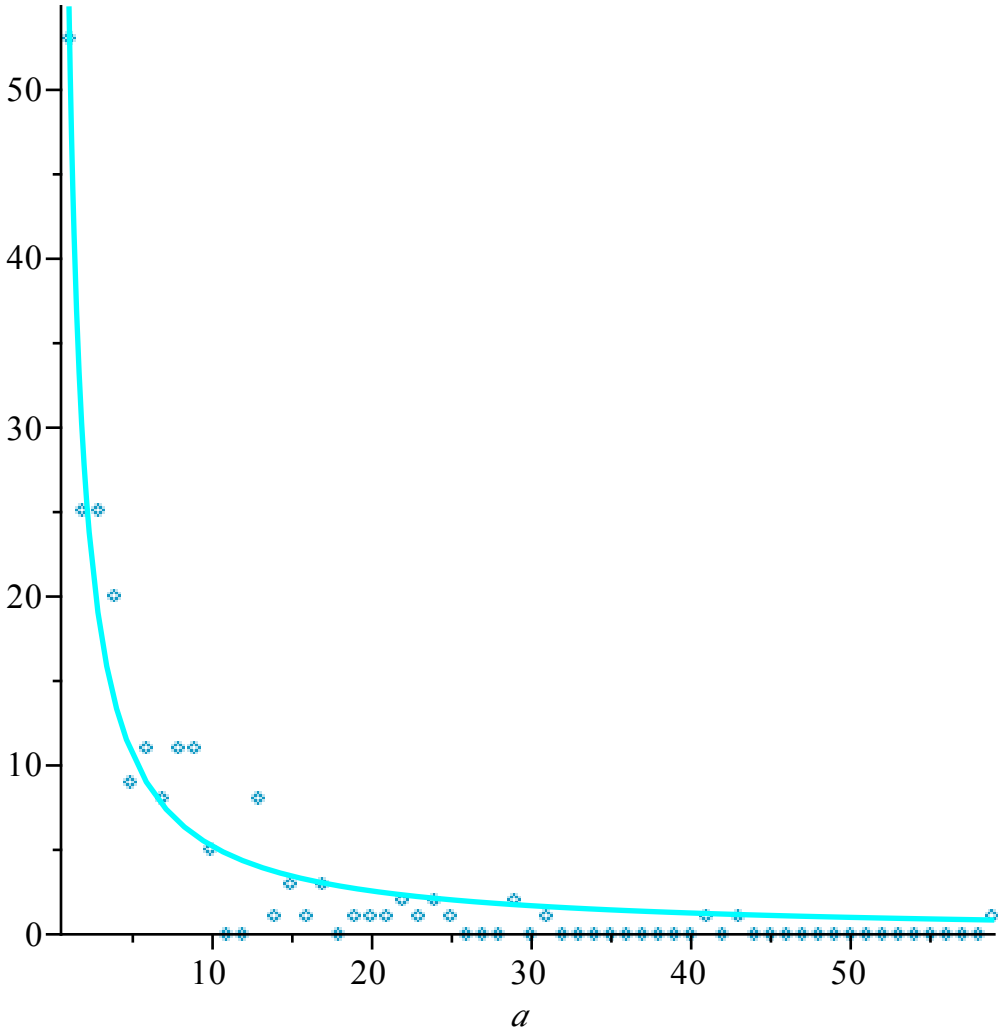



```
X := Vector([seq(1..MaxDeg)]) : Y := Vector(1..MaxDeg, 0) :
for deg from 1 to MaxDeg do
    freq[deg] := 0 :
    for vertex from 1 to 209 do
       if Degree(G, V[vertex]) = deg then freq[deg] := freq[deg] + 1 end if
```

```
end do
end do:
Y := Vector( [ seq( freq[ i ], i = 1 ..MaxDeg ) ] ) :
NonlinearFit( A·a^b, X, Y, a )
```

$$\frac{54.9468818815118212}{a^{1.02308154647380301}} \tag{6}$$

```
ScatterPlot( X, Y, fit = [ A·a^b, a ], thickness = 2 )
```




```
count := 0 :
lnX := Vector( MaxDeg );
lnY := Vector( MaxDeg );
 for deg from 1 to MaxDeg do
 if Y[ deg ] > 0 then
 count := count + 1;
 lnX[ count ] := ln( X[ deg ] ) :
 lnY[ count ] := ln( Y[ deg ] )
 end if
 end do:
 lnXX := Vector( count );
```

```
lnYY := Vector(count);
for i from 1 to count do
lnXX[i] := lnX[i];
lnYY[i] := lnY[i];
end do:
```

$$\begin{bmatrix} 1 .. 59\ Vector_{column} \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix}$$

$$\begin{bmatrix} 1 .. 59\ Vector_{column} \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix}$$

$$\begin{bmatrix} 1 .. 27\ Vector_{column} \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix}$$

$$\begin{bmatrix} 1 .. 27\ Vector_{column} \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \quad \textbf{(7)}$$

```
for i from 1 to count do
print(evalf(lnYY[i]))
end do
```

3.970291914

3.218875824

3.218875824

2.995732274

2.197224578

2.397895273

2.079441542

2.397895273

2.397895273

1.609437912

2.079441542

0.

1.098612289

0.

1.098612289

0.

0.

0.

0.6931471806

0.

0.6931471806

0.

0.6931471806

0.

0. **(8)**

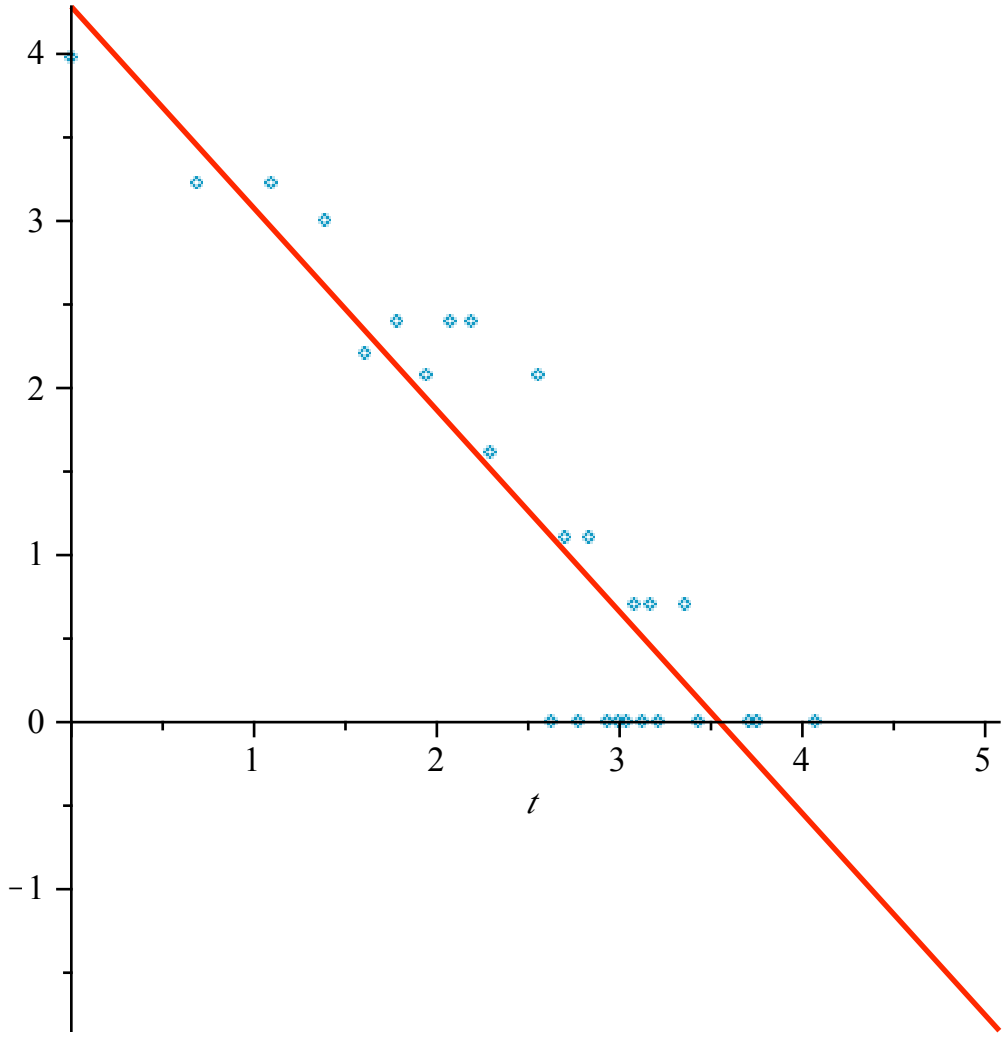
$Plot1 := ScatterPlot(lnXX, lnYY)$

$PLOT(\ldots)$ **(9)**

$Plot2 := plot(LinearFit([1, t], lnXX, lnYY, t), t = 0..(lnXX[count] + 1))$

$PLOT(\ldots)$ **(10)**

$with(plots) : display([Plot1, Plot2], thickness = 2)$



Correlation (*lnXX*, *lnYY*)

$$-0.9142475390 \tag{11}$$

LinearFit ([1, *t*], *lnXX*, *lnYY*, *t*)

$$4.28282405433401792 - 1.20722831711125123\ t \tag{12}$$

```
clusteringcoeff := proc( G, V );
    local k, j, Cv, nv, numn, maxedges, m, y, counter;
 Cv := 0;
 counter := 0;
for k from 1 to n by 1 do nv := Neighbors( G, V [ k ] );
 y := 0;
numn := nops( nv );
if numn ≠ 1 then
maxedges := binomial( numn, 2 );
    for j from 1 to numn − 1 do
        for m from j + 1 to numn do
            if HasEdge( G, { nv [ j ], nv [ m ] } ) then y := y + 1;
 end if ;
 end do
 end do ;
```

$$Cv := \frac{y}{maxedges} + Cv;$$

$counter := counter + 1;$

end if

end do;

$$\text{evalf}\left(\frac{\text{Cv}}{counter}\right);$$

end proc:

$clusteringcoeff(G, V)$

$$0.7186465269 \quad \textbf{(13)}$$

```
with(LinearAlgebra):
EVmatrix := Eigenvalues(evalf(Data)):
EV := convert(EVmatrix, 'list'):
EVsorted := sort(EV):
EVsorted[209]
```

$$15.0730503143760703 + 0.\,I \quad \textbf{(14)}$$