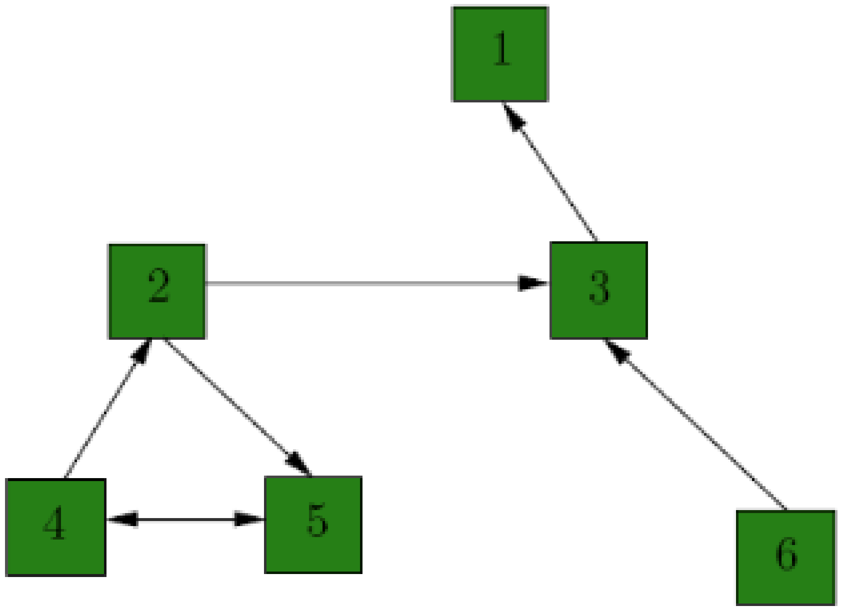
Mathematical Modeling

Inna Williams

Problem 1. Compute the PageRank vector of the following graph, considering the damping constant q to be successively q = 0, q = 0.15, q = 0.5, and respectively q = 1. Interpret your results in terms of the relationship between the number of incoming links that each node has and its rank (depending on q).




```
> with(LinearAlgebra) :
> A := Matrix([[0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 1, 0], [1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 0]])
```

$$A := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (1)$$

```
> deg := Vector(6, 0) : # create the outdegree vector for the transition matrix A
  for i from 1 to 6 do
  for j from 1 to 6 do
    deg[i] := deg[i] + A[i, j]
  end do
  end do; convert(deg, vector)
```

$$\begin{bmatrix} 0 & 2 & 1 & 2 & 1 & 1 \end{bmatrix} \quad (2)$$

```
> T := Matrix(6, 6, 0) :  #create the weighted transition matrix T
  for i from 1 to 6 do
  for j from 1 to 6 do
  if deg[i] > 0 then
```

$$T[j, i] := \frac{A[i, j]}{deg[i]}$$

```
  else
```

$$T[j, i] := \frac{1}{6}$$

end if
end do
end do; *print*(*T*)

$$\begin{bmatrix} \frac{1}{6} & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & \frac{1}{2} & 0 & 0 & 0 & 1 \\ \frac{1}{6} & 0 & 0 & 0 & 1 & 0 \\ \frac{1}{6} & \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \textbf{(3)}$$

> ***q* := 0.0 #set the damping factor q**

$$q := 0. \qquad \textbf{(4)}$$

> *G* := *Matrix*(6, 6, 0) : *# create the Google matrix as a weighted average between T and a random jump*
for *i* **from** 1 **to** 6 **do**
for *j* **from** 1 **to** 6 **do**

$$G[i, j] := (1 - q) \cdot T[i, j] + \frac{q}{6}$$

end do
end do; *print*(*G*)

$$\begin{bmatrix} 0.1666666667 & 0. & 1. & 0. & 0. & 0. \\ 0.1666666667 & 0. & 0. & 0.5000000000 & 0. & 0. \\ 0.1666666667 & 0.5000000000 & 0. & 0. & 0. & 1. \\ 0.1666666667 & 0. & 0. & 0. & 1. & 0. \\ 0.1666666667 & 0.5000000000 & 0. & 0.5000000000 & 0. & 0. \\ 0.1666666667 & 0. & 0. & 0. & 0. & 0. \end{bmatrix} \qquad \textbf{(5)}$$

> $(v, e) := Eigenvectors(G)$

$$\begin{bmatrix} 1.00000000003243 + 0.\,I \\ 0.608866271135962 + 0.\,I \\ -0.329002388803389 + 0.440943881584140\,I \\ -0.329002388803389 - 0.440943881584140\,I \\ -0.392097413430808 + 0.268988771610400\,I \\ -0.392097413430808 - 0.268988771610400\,I \end{bmatrix},$$

$$\begin{bmatrix} -0.35921 + 0.\,I & -0.65384 + 0.\,I & -0.77574 + 0.\,I & -0.77574 + -0.\,I & -0.62646 + 0.\,I & -0.62646 + -0.\,I \\ -0.35921 + 0.\,I & 0.22382 + 0.\,I & 0.026154 + 0.18746I & 0.026154 - 0.18746\,I & -0.33716 + 0.072025\,I & -0.33716 - 0.072025I \\ -0.29934 + 0.\,I & -0.28913 + 0.\,I & 0.38451 - 0.34205\,I & 0.38451 + 0.34205\,I & 0.35004 - 0.16851\,I & 0.35004 + 0.16851\,I \\ -0.59868 + 0.\,I & 0.49050 + 0.\,I & 0.076048 - 0.10028\,I & 0.076048 + 0.10028\,I & 0.43448 - 0.23787\,I & 0.43448 + 0.23787\,I \\ -0.53881 + 0.\,I & 0.40762 + 0.\,I & 0.14849 + 0.066527\,I & 0.14849 - 0.06652\,I & -0.0019628 + 0.21013\,I & -0.00196 - 0.2101\,I \\ -0.059868 + 0.\,I & -0.17897 + 0.\,I & 0.14053 + 0.18835\,I & 0.14053 - 0.18835\,I & 0.18107 + 0.12421\,I & 0.18107 - 0.12421\,I \end{bmatrix}$$

>

> edom := Column(e, 1) #select the dominant eigenvector (corresponding to eigenvalue 1) from the list (the first entry in this case)

$$edom := \begin{bmatrix} -0.359210604076077 + 0.\, I \\ -0.359210604050836 + 0.\, I \\ -0.299342170063074 + 0.\, I \\ -0.598684340075665 + 0.\, I \\ -0.538815906070428 + 0.\, I \\ -0.0598684340227115 + 0.\, I \end{bmatrix} \qquad (6)$$

> **grank := $\frac{edom}{Norm(edom, 1)}$: convert(simplify(grank), vector)   #normalize the dominant eigenvector**

$$\begin{bmatrix} -0.1621621622 & -0.1621621622 & -0.1351351352 & -0.2702702703 & -0.2432432433 & -0.02702702704 \end{bmatrix} \qquad (7)$$

Check that the normalized dominant eigenvector is indeed the steady-state distribution of matrix G.

> MatrixPower(G, 50)

$$\begin{bmatrix} 0.1621621 & 0.162162 & 0.1621621 & 0.1621621 & 0.1621621 & 0.1621621 \\ 0.1621621 & 0.162162 & 0.1621621 & 0.1621621 & 0.1621621 & 0.162162 \\ 0.1351351 & 0.135135 & 0.1351351 & 0.1351351 & 0.1351351 & 0.1351351 \\ 0.2702702 & 0.270270 & 0.2702702 & 0.270270 & 0.27027 & 0.2702702 \\ 0.2432432 & 0.243243 & 0.2432432 & 0.243243 & 0.2432432 & 0.2432432 \\ 0.0270270 & 0.027027 & 0.0270272 & 0.027027 & 0.0270270 & 0.02702702 \end{bmatrix}$$

>

> **q := 0.15 #set the damping factor q**

$$q := 0.15 \qquad (8)$$

```
> G := Matrix(6, 6, 0) :  # create the Google matrix as a weighted average between T and a random jump
  for i from 1 to 6 do
  for j from 1 to 6 do
  G[i, j] := (1 - q)·T[i, j] + q/6
  end do
  end do; print(G)
```

$$\begin{bmatrix} 0.1666666667 & 0.02500000000 & 0.8750000000 & 0.02500000000 & 0.02500000000 & 0.02500000000 \\ 0.1666666667 & 0.02500000000 & 0.02500000000 & 0.4500000000 & 0.02500000000 & 0.02500000000 \\ 0.1666666667 & 0.4500000000 & 0.02500000000 & 0.02500000000 & 0.02500000000 & 0.8750000000 \\ 0.1666666667 & 0.02500000000 & 0.02500000000 & 0.02500000000 & 0.8750000000 & 0.02500000000 \\ 0.1666666667 & 0.4500000000 & 0.02500000000 & 0.4500000000 & 0.02500000000 & 0.02500000000 \\ 0.1666666667 & 0.02500000000 & 0.02500000000 & 0.02500000000 & 0.02500000000 & 0.02500000000 \end{bmatrix} \qquad (9)$$

> (v, e) := Eigenvectors(G)

$$\begin{bmatrix} 1.00000000003744 + 0.\, I \\ 0.517536330460277 + 0.\, I \\ -0.279652030482412 + 0.374802299348323\, I \\ -0.279652030482412 - 0.374802299348323\, I \\ -0.333282801416445 + 0.228640455868547\, I \\ -0.333282801416445 - 0.228640455868547\, I \end{bmatrix}$$

$$\begin{bmatrix} 0.43209 + 0.\, I & -0.65384 + 0.\, I & -0.77574 + 0.\, I & -0.77574 + -0.\, I & -0.62646 + 0.\, I & -0.62646 + -0.\, I \\ 0.34925 + 0.\, I & 0.22382 + 0.\, I & 0.02615 + 0.18746 I & 0.026154 - 0.18746\, I & -0.33716 + 0.07202\, I & -0.33716 - 0.07202 I \\ 0.36844 + 0.\, I & -0.28913 + 0.\, I & 0.38451 - 0.34205\, I & 0.38451 + 0.34205\, I & 0.35004 - 0.16851\, I & 0.350047 + 0.16851\, I \\ 0.54196 + 0.\, I & 0.49050 + 0.\, I & 0.07604 - 0.10028\, I & 0.07604 + 0.10028\, I & 0.43448 - 0.23787\, I & 0.43448 + 0.23787\, I \\ 0.49769 + 0.\, I & 0.40762 + 0.\, I & 0.14849 + 0.06652\, I & 0.14849 - 0.06652\, I & -0.00196 + 0.21013\, I & -0.00196 - 0.21013\, I \\ 0.11892 + 0.\, I & -0.17897 + 0.\, I & 0.14053 + 0.18835 I & 0.14053 - 0.1883\, I & 0.18107 + 0.12421\, I & 0.18107 - 0.12421\, I \end{bmatrix}$$

> *edom* := *Column*(*e*, 1) *#select the dominant eigenvector (corresponding to eigenvalue 1) from the list (the first entry in this case)*

$$edom := \begin{bmatrix} 0.432099460058892 + 0.\,I \\ 0.349256645409087 + 0.\,I \\ 0.368442400552055 + 0.\,I \\ 0.541960531332565 + 0.\,I \\ 0.497690719702392 + 0.\,I \\ 0.118923419601370 + 0.\,I \end{bmatrix} \tag{10}$$

> $grank := \dfrac{edom}{Norm(edom, 1)}$ **: *convert*(*simplify*(*grank*), *vector*) *#normalize the dominant eigenvector***

$$\begin{bmatrix} \mathbf{0.1871878708} & \mathbf{0.1512999063} & \mathbf{0.1596112813} & \mathbf{0.2347802932} & \mathbf{0.2156023665} & \mathbf{0.05151828170} \end{bmatrix} \tag{11}$$

Check that the normalized dominant eigenvector is indeed the steady-state distribution of matrix G.

> *MatrixPower*(*G*, 50)

$$\begin{bmatrix} 0.187187 & 0.1871878 & 0.187187 & 0.1871878 & 0.1871878 & 0.1871878 \\ 0.151299 & 0.151299 & 0.151299 & 0.1512999 & 0.1512999 & 0.1512999 \\ 0.159611 & 0.1596112 & 0.1596112 & 0.1596112 & 0.1596112 & 0.1596112 \\ 0.2347802 & 0.234780 & 0.2347802 & 0.234780 & 0.2347802 & 0.2347802 \\ 0.2156023 & 0.2156023 & 0.215602 & 0.215602 & 0.2156023 & 0.2156023 \\ 0.05151828 & 0.05151828 & 0.0515182 & 0.0515182 & 0.0515182 & 0.0515182 \end{bmatrix}$$

>

> ***q* := 0.5 *#set the damping factor q***

$$q := 0.5 \tag{12}$$

> *G* := *Matrix*(6, 6, 0) : *# create the Google matrix as a weighted average between T and a random jump*
for *i* **from** 1 **to** 6 **do**
for *j* **from** 1 **to** 6 **do**
$G[i, j] := (1 - q)\cdot T[i, j] + \dfrac{q}{6}$
end do
end do; *print*(*G*)

$$\begin{bmatrix} 0.1666666667 & 0.08333333333 & 0.5833333333 & 0.08333333333 & 0.08333333333 & 0.08333333333 \\ 0.1666666667 & 0.08333333333 & 0.08333333333 & 0.3333333333 & 0.08333333333 & 0.08333333333 \\ 0.1666666667 & 0.3333333333 & 0.08333333333 & 0.08333333333 & 0.08333333333 & 0.5833333333 \\ 0.1666666667 & 0.08333333333 & 0.08333333333 & 0.08333333333 & 0.5833333333 & 0.08333333333 \\ 0.1666666667 & 0.3333333333 & 0.08333333333 & 0.3333333333 & 0.08333333333 & 0.08333333333 \\ 0.1666666667 & 0.08333333333 & 0.08333333333 & 0.08333333333 & 0.08333333333 & 0.08333333333 \end{bmatrix} \tag{13}$$

> $(v, e) :=$ *Eigenvectors*(*G*)

$$\begin{bmatrix} 0.999999999987912 + 0.\,I \\ 0.304433135538688 + 0.\,I \\ -0.164501194387996 + 0.220471940787191\,I \\ -0.164501194387996 - 0.220471940787191\,I \\ -0.196048706700304 + 0.134494385787019\,I \\ -0.196048706700304 - 0.134494385787019\,I \end{bmatrix}$$

$$\begin{bmatrix} 0.46158+0.\,I & -0.65384+0.\,I & -0.77574+0.\,I & -0.77574+-0.\,I & -0.62646+0.\,I & -0.62646+-0.\,I \\ 0.35336+0.\,I & 0.22382+0.\,I & 0.026154+0.18746\,I & 0.026154-0.18746\,I & -0.33716+0.072025\,I & -0.33716-0.07202\,I \\ 0.44612+0.\,I & -0.28913+0.\,I & 0.38451-0.34205\,I & 0.38451+0.34205\,I & 0.35004-0.16851\,I & 0.35004+0.16851\,I \\ 0.45938+0.\,I & 0.49050+0.\,I & 0.076048-0.10028\,I & 0.076048+0.10028\,I & 0.43448-0.23787\,I & 0.43448+0.23787\,I \\ 0.44171+0.\,I & 0.40762+0.\,I & 0.14849+0.066527\,I & 0.14849-0.066527\,I & -0.0019628+0.21013\,I & -0.0019628-0.2101I \\ 0.23852+0.\,I & -0.17897+0.\,I & 0.14053+0.18835\,I & 0.14053-0.18835\,I & 0.18107+0.12421\,I & 0.18107-0.12421\,I \end{bmatrix}$$

> *edom* := *Column*(*e*, 1) *#select the dominant eigenvector (corresponding to eigenvalue 1) from the list (the first entry in this case)*

$$edom := \begin{bmatrix} 0.461589143475168+0.\,I \\ 0.353369679212063+0.\,I \\ 0.446129220009458+0.\,I \\ 0.459380582973317+0.\,I \\ 0.441712099005545+0.\,I \\ 0.238524533481126+0.\,I \end{bmatrix} \tag{14}$$

> $\textbf{grank} := \dfrac{\textbf{edom}}{\textbf{Norm}(\textbf{edom}, 1)}$ **: convert(simplify(grank), vector) #normalize the dominant eigenvector**

$$\begin{bmatrix} \mathbf{0.1922723091} & \mathbf{0.1471941122} & \mathbf{0.1858325667} & \mathbf{0.1913523459} & \mathbf{0.1839926403} & \mathbf{0.09935602577} \end{bmatrix} \tag{15}$$

Check that the normalized dominant eigenvector is indeed the steady-state distribution of matrix G.

> *MatrixPower*(*G*, 50)

$$\begin{bmatrix} 0.1922723 & 0.1922723 & 0.1922723 & 0.1922723 & 0.1922723 & 0.1922723 \\ 0.147194 & 0.1471941 & 0.14719411 & 0.1471941 & 0.1471941 & 0.1471941 \\ 0.1858325 & 0.1858325 & 0.1858325 & 0.1858325 & 0.1858325 & 0.1858325 \\ 0.1913523 & 0.1913523 & 0.1913523 & 0.1913523 & 0.1913523 & 0.1913523 \\ 0.1839926 & 0.1839926 & 0.1839926 & 0.1839926 & 0.1839926 & 0.1839926 \\ 0.0993560 & 0.099356 & 0.0993560 & 0.09935602 & 0.09935602 & 0.0993560 \end{bmatrix}$$

>

> *q* := 1.0 *#set the damping factor q*

$$q := 1.0 \tag{16}$$

> *G* := *Matrix*(6, 6, 0) : *# create the Google matrix as a weighted average between T and a random jump*
for *i* **from** 1 **to** 6 **do**
for *j* **from** 1 **to** 6 **do**

$$G[i, j] := (1-q)\cdot T[i, j] + \frac{q}{6}$$

end do
end do; *print*(*G*)

$$\begin{bmatrix} 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \\ 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \\ 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \\ 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \\ 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \\ 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 & 0.1666666667 \end{bmatrix} \tag{17}$$

> $(v, e) := Eigenvectors(G)$

$$\begin{bmatrix} 1.00000000020000 + 0.\,I \\ 6.16297582203915\ 10^{-33} + 0.\,I \\ 9.06493303673678\ 10^{-17} + 0.\,I \\ 0.\,I \\ 0.\,I \\ 0.\,I \end{bmatrix}$$

$$\begin{bmatrix} 0.408248 + 0.\,I & 4.5442\ 10^{-17} + 0.\,I & -0.816496 + 0.\,I & -3.7728\ 10^{-17} + 0.\,I & -3.7728\ 10^{-17} + 0.\,I & -3.7728\ 10^{-17} + 0.\,I \\ 0.40824 + 0.\,I & 0.89442 + 0.\,I & 0.563299 + 0.\,I & -0.199007 + 0.\,I & -0.19900 + 0.\,I & -0.19900 + 0.\,I \\ 0.40824 + 0.\,I & -0.22360 + 0.\,I & 0.063299 + 0.\,I & -0.43771 + 0.\,I & -0.43771 + 0.\,I & -0.43771 + 0.\,I \\ 0.40824 + 0.\,I & -0.22360 + 0.\,I & 0.0632993 + 0.\,I & 0.86219 + 0.\,I & -0.11273 + 0.\,I & -0.11273 + 0.\,I \\ 0.40824 + 0.\,I & -0.22360 + 0.\,I & 0.063299 + 0.\,I & -0.11273 + 0.\,I & 0.86219 + 0.\,I & -0.11273 + 0.\,I \\ 0.40824 + 0.\,I & -0.22360 + 0.\,I & 0.063299 + 0.\,I & -0.11273 + 0.\,I & -0.11273 + 0.\,I & 0.86219 + 0.\,I \end{bmatrix}$$

> *edom* := *Column*(*e*, 1) *#select the dominant eigenvector (corresponding to eigenvalue 1) from the list (the first entry in this case)*

$$edom := \begin{bmatrix} 0.408248290463863 + 0.\,I \\ 0.408248290463863 + 0.\,I \\ 0.408248290463863 + 0.\,I \\ 0.408248290463863 + 0.\,I \\ 0.408248290463863 + 0.\,I \\ 0.408248290463863 + 0.\,I \end{bmatrix} \quad \textbf{(18)}$$

> $grank := \dfrac{edom}{Norm(edom, 1)}$ **: *convert*(*simplify*(*grank*), *vector*) *#normalize the dominant eigenvector***

$$\begin{bmatrix} \mathbf{0.1666666666} & \mathbf{0.1666666666} & \mathbf{0.1666666666} & \mathbf{0.1666666666} & \mathbf{0.1666666666} & \mathbf{0.1666666666} \end{bmatrix} \quad \textbf{(19)}$$

Check that the normalized dominant eigenvector is indeed the steady-state distribution of matrix G.

> *MatrixPower*(*G*, 50)

$$\begin{bmatrix} 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 \\ 0.1666666 & 0.16666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 \\ 0.1666666 & 0.16666666 & 0.1666666 & 0.1666666 & 0.16666666 & 0.1666666 \\ 0.1666666 & 0.1666666 & 0.166666 & 0.1666666 & 0.1666666 & 0.1666666 \\ 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 \\ 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 & 0.1666666 \end{bmatrix}$$

q = 0.00	**0.1621621622**	**0.1621621622**	**0.1351351352**	**0.2702702703**	**0.2432432433**	**0.02702702704**
q = 0.15	**0.1871878708**	**0.1512999063**	**0.1596112813**	**0.2347802932**	**0.2156023665**	**0.0515182817**
q = 0.50	**0.1922723091**	**0.1471941122**	**0.1858325667**	**0.1913523459**	**0.1839926403**	**0.09935602577**
q = 1.00	**0.1666666666**	**0.1666666666**	**0.1666666666**	**0.1666666666**	**0.1666666666**	**0.1666666666**

This graph contain cycle 2-4-5. The total path length on the nodes 4 and 5 will be increasing faster then path length on other nodes and the rank will be higher on the nodes 4 and 5 then on other nodes

If q=0 then Gij = Tij
The random surfer will visit the pages using the simple random walk where G= T

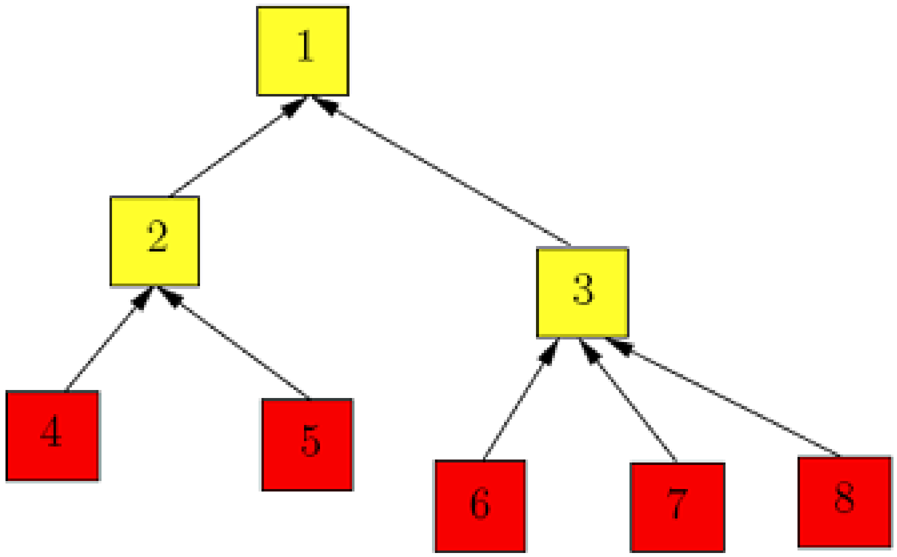
If q = 0,15 and q=0.5 the rank increasses on the Nodes that have no cycle and decreases on the ones that have a cycle

If q=1 then Gij = 1/n = 1/6 = 0.1666

The random surfer will visit the pages using equiprobable random walk where G= 0.166

\>

Problem 2. Compute the PageRank vector of the directed tree depicted below, considering that the damping constant q = 0.15. Interpret your results in terms of the relationship between the number of incoming links that each node has and its rank.



\>

\> $A := Matrix([[0, 0, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0]])$

$$A := \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \textbf{(20)}$$

\> $deg := Vector(8, 0)$: # *create the outdegree vector for the transition matrix A*

for i **from** 1 **to** 8 **do**
for j **from** 1 **to** 8 **do**
$deg[i] := deg[i] + A[i, j]$
end do
end do; $convert(deg, vector)$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \textbf{(21)}$$

\> $T := Matrix(8, 8, 0)$: #*create the weighted transition matrix T*
for i **from** 1 **to** 8 **do**
for j **from** 1 **to** 8 **do**
if $deg[i] > 0$ **then**
$T[j, i] := \frac{A[i, j]}{deg[i]}$
else
$T[j, i] := \frac{1}{8}$
end if
end do

```
end do; print(T)
```

$$\begin{bmatrix} \frac{1}{8} & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

>

> $q := 0.15$ *#set the damping factor q*

$$q := 0.15 \tag{23}$$

> $G := Matrix(8, 8, 0)$: *# create the Google matrix as a weighted average between T and a random jump*

for i **from** 1 **to** 8 **do**
for j **from** 1 **to** 8 **do**

$G[i, j] := (1 - q) \cdot T[i, j] + \frac{q}{8}$

end do
end do; *print*(*G*)

$$\begin{bmatrix} 0.1250000000 & 0.8687500000 & 0.8687500000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.8687500000 & 0.8687500000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.8687500000 & 0.8687500000 & 0.8687500000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \\ 0.1250000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 & 0.01875000000 \end{bmatrix}$$

(24)

> $(v, e) := Eigenvectors(G)$

$$\begin{bmatrix} 1.00000000000000 + 0.\,\mathrm{I} \\ -0.371875000000000 + 0.559706605620302\,\mathrm{I} \\ -0.371875000000000 - 0.559706605620302\,\mathrm{I} \\ -5.53996564307207\ 10^{-18} + 1.07885118435901\ 10^{-8}\,\mathrm{I} \\ -5.53996564307207\ 10^{-18} - 1.07885118435901\ 10^{-8}\,\mathrm{I} \\ -5.57632861953159\ 10^{-17} + 0.\,\mathrm{I} \\ 0.\,\mathrm{I} \\ 0.\,\mathrm{I} \end{bmatrix}$$

$$\begin{bmatrix} 0.78 & 0.81 & 0.81 & 1.06\ 10^{-15}+1.22\ 10^{-23}\ I & -1.06\ 10^{-15}-1.22\ 10^{-23}\ I & 3.29\ 10^{-16} & 6.43\ 10^{-17} & 6.43\ 10^{-17} \\ 0.33 & -0.19+0.19\ I & -0.19-0.19\ I & -0.70-1.65\ 10^{-23}\ I & -0.70+1.65\ 10^{-23}\ I & 0.28 & 1.01\ 10^{-16} & 9.22\ 10^{-17} \\ 0.44 & -0.25+0.34I & -0.25-0.34I & 0.70 & 0.70 & 0.28 & 9.02\ 10^{-17} & 5.89\ 10^{-17} \\ 0.12 & -0.07-0.10\ I & -0.07+0.10\ I & 2.53\ 10^{-16}-1.81\ 10^{-9}\ I & 2.53\ 10^{-16}+1.81\ 10^{-9}\ I & 0.64 & 1.37\ 10^{-16} & 1.47\ 10^{-16} \\ 0.12 & -0.07-0.10I & -0.07+0.10\ I & 4.01\ 10^{-16}-7.15\ 10^{-9}\ I & 4.01\ 10^{-16}+7.15\ 10^{-9}\ I & 0.64 & 2.23\ 10^{-16} & 2.34\ 10^{-16} \\ 0.1 & -0.07-0.10\ I & -0.07+0.10\ I & 1.72\ 10^{-16}+2.99\ 10^{-9}\ I & 1.72\ 10^{-16}-2.99\ 10^{-9}\ I & 4.62\ 10^{-16} & -0.57 & 0.57 \\ 0.12 & -0.07-0.10\ I & -0.07+0.10\ I & 1.72\ 10^{-16}+2.99\ 10^{-9}\ I & 1.72\ 10^{-16}-2.99\ 10^{-9}\ I & 4.62\ 10^{-16} & 0.78 & 0.21 \\ 0.12 & -0.07-0.10\ I & -0.07+0.10\ I & 1.72\ 10^{-16}+2.99\ 10^{-9}\ I & 1.72\ 10^{-16}-2.99\ 10^{-9}\ I & 4.62\ 10^{-16} & -0.2 & 0.78 \end{bmatrix}$$

> *edom* := *Column*(*e*, 1) #*select the dominant eigenvector (corresponding to eigenvalue 1) from the list (the first entry in this case)*

$$edom := \begin{bmatrix} -0.784538967406643+0.\ I \\ -0.335565182098683+0.\ I \\ -0.441206072759379+0.\ I \\ -0.124283400777290+0.\ I \\ -0.124283400777290+0.\ I \\ -0.124283400777290+0.\ I \\ -0.124283400777290+0.\ I \\ -0.124283400777290+0.\ I \end{bmatrix} \quad (25)$$

> $grank := \dfrac{edom}{Norm(edom, 1)}$ **: convert(simplify(grank), vector) #normalize the dominant eigenvector**

$$\begin{bmatrix} \mathbf{-0.35943} & \mathbf{-0.15373} & \mathbf{-0.20213} & \mathbf{-0.056939} & \mathbf{-0.056939} & \mathbf{-0.056939} & \mathbf{-0.056939} & \mathbf{-0.056939} \end{bmatrix}$$

Check that the normalized dominant eigenvector is indeed the steady-state distribution of matrix G.

> *MatrixPower*(*G*, 50)

$$\begin{bmatrix} 0.359430 & 0.3594306 & 0.359430 & 0.359430 & 0.359430 & 0.359430 & 0.35943 & 0.35943 \\ 0.1537366 & 0.153736 & 0.153736 & 0.153736 & 0.153736 & 0.153736 & 0.15373 & 0.15373 \\ 0.2021352 & 0.202135 & 0.202135 & 0.202135 & 0.202135 & 0.202135 & 0.20213 & 0.20213 \\ 0.0569395 & 0.0569395 & 0.056939 & 0.056939 & 0.056939 & 0.0569395 & 0.05693 & 0.05693 \\ 0.0569395 & 0.0569395 & 0.0569395 & 0.056939 & 0.056939 & 0.0569395 & 0.05693 & 0.05693 \\ 0.0569395 & 0.0569395 & 0.0569395 & 0.056939 & 0.056939 & 0.0569395 & 0.05693 & 0.05693 \\ 0.0569395 & 0.0569395 & 0.0569395 & 0.056939 & 0.056939 & 0.0569395 & 0.05693 & 0.05693 \\ 0.0569395 & 0.0569395 & 0.0569395 & 0.056939 & 0.056939 & 0.0569395 & 0.05693 & 0.05693 \end{bmatrix}$$

Graph has no cycles and then path length will be 1 and 2
We can see that the highest rank 0.359 corresponding to the node 1 that has 2 incoming links. Also from 2 and 3 and also 2 more links come through node 2 and 3 more though node 3
total comes to node 1 = 3 from left tree and 4 from right, total = 7

to Node / Links path length from Node	1	2	3	4	5	6	7	8	Total links path length to Node -> corresponding rank
1		1	1	1	1	1	1	1	7 - > 0.359

2				**1**	**1**				**2 - > 0.154**
3						**1**	**1**	**1**	**3 - > 0.202**
4									**0 -> 0.057**
5									**0 -> 0.057**
6									**0 -> 0.057**
7									**0 -> 0.057**
8									**0 -> 0.057**

>

We can see from the table the
Node 1 has the most incoming links and the highest rank
Node 3 has 3 incominglinks second highest rank
Node 2 has 2 link third highest rank
The rest of the nodes have 0 incoming links and the lowest the same rank.
The random serfer will visit the
Node 1 with probability 0.359 -> highest probability
Node 3 with probability 0.202 < 0.359 of Node 1
Node 2 with probability 0.154 < 0.202 of Node 2
Nodes 5,6,7,8 with probability 0.057 < 0.154 of Node 2
We can see that the rank of the node is proportional to its incoming links(or to its in-degree)

>