```
> restart; Restart;
                                   Restart                                  (1)
> n := 6; this := ` `; EdgeCost := Matrix(n, n) :
                                   n := 6
                                   this :=                                  (2)
> brand := rand(2..12) : big := 1000· n^3;
                                   big := 216000                            (3)
>
> MyDist[0] := Matrix(n, n) : FirstStop[0] := Matrix(n, n) : LastStop[0] := Matrix(n, n) :
     PathTraffic := Matrix(n) : EdgeTraffic := Matrix(n) :
> k := 0;
  for i from 1 to n do
   for j from 1 to n do
  MyDist[k][i,j] := brand( ) :#+3·( abs(i-j) - 1)^2 :
  EdgeCost[i,j] := MyDist[k][i,j];
    PathTraffic[i,j] := n + brand( ) :;
    EdgeTraffic[i,j] := 0;
   FirstStop[k][i,j] := j;
   LastStop[k][i,j] := i ;
   if MyDist[k][i,j] > 10 then MyDist[k][i,j] := big; FirstStop[k][i,j] := -1; LastStop[k][i,j]
       := -1; EdgeCost[i,j] := na; end if;
  # MyDist[k][i,j] := (i-j)^2· (brand( ))^2 :
   end do:
   MyDist[k][i, i] := 0 : FirstStop[k][i, i] := 0; LastStop[k][i, i] := 0; EdgeTraffic[i, i] := 0;
     PathTraffic[i, i] := 0; EdgeCost[i, i] := ` `;
   end do:
                                   k := 0                                   (4)
>
> print(k, EdgeCost, PathTraffic);
```

$$0, \begin{bmatrix} & 7 & na & 7 & na & 9 \\ na & & 5 & na & 10 & 3 \\ 9 & 10 & & 8 & 4 & 6 \\ 9 & 4 & 2 & & na & na \\ 3 & 5 & 10 & 10 & & na \\ na & 5 & 8 & 10 & na & \end{bmatrix}, \begin{bmatrix} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{bmatrix} \quad (5)$$

```
>
>
  for k from 1 to n do
    MyDist[k] := Matrix(n, n) : FirstStop[k] := Matrix(n, n); LastStop[k] := Matrix(n, n);
  # print(k - 1, MyDist[k - 1], FirstStop[k - 1], LastStop[k - 1]);
     for i from 1 to n do
      for j from 1 to n do
```

```
            optionDirect := MyDist[k - 1][i, j];
            optionViaK := MyDist[k - 1][i, k] + MyDist[k - 1][k, j] :
            if optionDirect ≤ optionViaK
              then MyDist[k][i, j] := optionDirect; FirstStop[k][i, j] := FirstStop[k - 1][i, j];
           LastStop[k][i, j] := LastStop[k - 1][i, j];
   #lprint(`Dircect: then `, optionDirect, optionViaK, k, i, j);
              else MyDist[k][i, j] := optionViaK; FirstStop[k][i, j] := FirstStop[k - 1][i, k];
          LastStop[k][i, j] := LastStop[k - 1][k, j];
    #lprint(`Via K: else`, optionDirect, optionViaK, k, i, j);
            end if;
       end do: #j
       end do: # i
   # print(k, MyDist[k], FirstStop[k], LastStop[k]);
    end do: # k
> k := 0 : print(k, MyDist[k], FirstStop[k], LastStop[k]);
  k := n : print(k, MyDist[k], FirstStop[k], LastStop[k]);
```

$$0, \begin{bmatrix} 0 & 7 & 216000 & 7 & 216000 & 9 \\ 216000 & 0 & 5 & 216000 & 10 & 3 \\ 9 & 10 & 0 & 8 & 4 & 6 \\ 9 & 4 & 2 & 0 & 216000 & 216000 \\ 3 & 5 & 10 & 10 & 0 & 216000 \\ 216000 & 5 & 8 & 10 & 216000 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 & -1 & 4 & -1 & 6 \\ -1 & 0 & 3 & -1 & 5 & 6 \\ 1 & 2 & 0 & 4 & 5 & 6 \\ 1 & 2 & 3 & 0 & -1 & -1 \\ 1 & 2 & 3 & 4 & 0 & -1 \\ -1 & 2 & 3 & 4 & -1 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0 & 1 & -1 & 1 & -1 & 1 \\ -1 & 0 & 2 & -1 & 2 & 2 \\ 3 & 3 & 0 & 3 & 3 & 3 \\ 4 & 4 & 4 & 0 & -1 & -1 \\ 5 & 5 & 5 & 5 & 0 & -1 \\ -1 & 6 & 6 & 6 & -1 & 0 \end{bmatrix}$$

$$6, \begin{bmatrix} 0 & 7 & 9 & 7 & 13 & 9 \\ 12 & 0 & 5 & 13 & 9 & 3 \\ 7 & 9 & 0 & 8 & 4 & 6 \\ 9 & 4 & 2 & 0 & 6 & 7 \\ 3 & 5 & 10 & 10 & 0 & 8 \\ 15 & 5 & 8 & 10 & 12 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 & 4 & 4 & 4 & 6 \\ 3 & 0 & 3 & 3 & 3 & 6 \\ 5 & 5 & 0 & 4 & 5 & 6 \\ 1 & 2 & 3 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 0 & 2 \\ 3 & 2 & 3 & 4 & 3 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 4 & 1 & 3 & 1 \\ 5 & 0 & 2 & 3 & 3 & 2 \\ 5 & 5 & 0 & 3 & 3 & 3 \\ 4 & 4 & 4 & 0 & 3 & 2 \\ 5 & 5 & 5 & 5 & 0 & 2 \\ 5 & 6 & 6 & 6 & 3 & 0 \end{bmatrix} \quad \textbf{(6)}$$

```
>
> printlevel := 1; evalm(FirstStop[n]);
```

$$printlevel := 1$$

$$\begin{bmatrix} 0 & 2 & 4 & 4 & 4 & 6 \\ 3 & 0 & 3 & 3 & 3 & 6 \\ 5 & 5 & 0 & 4 & 5 & 6 \\ 1 & 2 & 3 & 0 & 3 & 2 \\ 1 & 2 & 3 & 4 & 0 & 2 \\ 3 & 2 & 3 & 4 & 3 & 0 \end{bmatrix} \tag{7}$$

```
> Hops := Matrix(n) : ActualPath := Matrix(n) : for iii from 1 to n do for jjj from 1 to n
      do ActualPath[iii, jjj] := iii  end: end:
  for ii from 1 to n do
  for jj from 1 to n do
   i := ii : j := jj :
       while FirstStop[n][i, j] ≠ 0 do
             ifrom := i :
             i := FirstStop[n][i, j];
            Hops[ii, jj] := Hops[ii, jj] + 1;
            EdgeTraffic[ifrom, i] := EdgeTraffic[ifrom, i] + PathTraffic[ii, jj] :
            ActualPath[ii, jj] := ActualPath[ii, jj] , i;
       end do;
  #     print(`For`, ii, jj, `the actual path is`, ActualPath[ii, jj]);
   end do;
   end do;
> print( EdgeCost, MyDist[n], ActualPath, Hops,  PathTraffic, EdgeTraffic);
```

$$\begin{bmatrix} & 7 & na & 7 & na & 9 \\ na & & 5 & na & 10 & 3 \\ 9 & 10 & & 8 & 4 & 6 \\ 9 & 4 & 2 & & na & na \\ 3 & 5 & 10 & 10 & & na \\ na & 5 & 8 & 10 & na & \end{bmatrix}, \begin{bmatrix} 0 & 7 & 9 & 7 & 13 & 9 \\ 12 & 0 & 5 & 13 & 9 & 3 \\ 7 & 9 & 0 & 8 & 4 & 6 \\ 9 & 4 & 2 & 0 & 6 & 7 \\ 3 & 5 & 10 & 10 & 0 & 8 \\ 15 & 5 & 8 & 10 & 12 & 0 \end{bmatrix}, \tag{8}$$

$$\begin{bmatrix} 1 & (1,2) & (1,4,3) & (1,4) & (1,4,3,5) & (1,6) \\ (2,3,5,1) & 2 & (2,3) & (2,3,4) & (2,3,5) & (2,6) \\ (3,5,1) & (3,5,2) & 3 & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & 4 & (4,3,5) & (4,2,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & 5 & (5,2,6) \\ (6,3,5,1) & (6,2) & (6,3) & (6,4) & (6,3,5) & 6 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 2 & 1 & 3 & 1 \\ 3 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 2 & 0 \end{bmatrix},$$

$$\left[\begin{array}{rrrrrr} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{array}\right], \left[\begin{array}{rrrrrr} 0 & 9 & 0 & 31 & 0 & 12 \\ 0 & 0 & 60 & 0 & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & 0 & 0 \\ 68 & 43 & 12 & 14 & 0 & 0 \\ 0 & 16 & 42 & 8 & 0 & 0 \end{array}\right]$$

```
> FlowPerEdge[0] := Matrix(n, n) : FirstNewStop[0] := Matrix(n, n) : LastNewStop[0]
     := Matrix(n, n) : whattype(FlowPerEdge[0]);
```

$$Matrix \tag{9}$$

```
> for i from 1 to n do for j from 1 to n do
  if MyDist[0][i, j] = big then FlowPerEdge[0][i, j] := big; EdgeTraffic[i, j] := na;
     else FlowPerEdge[0][i, j] := max(0, EdgeTraffic[i, j] ) end if;
  end do;
  FlowPerEdge[0][i, i] := 0
  end do:
  whattype(FlowPerEdge[0]);

  print(FlowPerEdge[0], Hops, ActualPath, PathTraffic, EdgeTraffic);
```

$$Matrix$$

$$\left[\begin{array}{rrrrrr} 0 & 9 & 216000 & 31 & 216000 & 12 \\ 216000 & 0 & 60 & 216000 & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & 216000 & 216000 \\ 68 & 43 & 12 & 14 & 0 & 216000 \\ 216000 & 16 & 42 & 8 & 216000 & 0 \end{array}\right], \left[\begin{array}{rrrrrr} 0 & 1 & 2 & 1 & 3 & 1 \\ 3 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 2 & 0 \end{array}\right], \tag{10}$$

$$\left[\begin{array}{cccccc} 1 & (1,2) & (1,4,3) & (1,4) & (1,4,3,5) & (1,6) \\ (2,3,5,1) & 2 & (2,3) & (2,3,4) & (2,3,5) & (2,6) \\ (3,5,1) & (3,5,2) & 3 & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & 4 & (4,3,5) & (4,2,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & 5 & (5,2,6) \\ (6,3,5,1) & (6,2) & (6,3) & (6,4) & (6,3,5) & 6 \end{array}\right],$$

$$\left[\begin{array}{cccccc} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 9 & na & 31 & na & 12 \\ na & 0 & 60 & na & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & na & na \\ 68 & 43 & 12 & 14 & 0 & na \\ na & 16 & 42 & 8 & na & 0 \end{array}\right]$$

>

>

> $FirstNewStop[0] := Matrix(n, n) : LastNewStop[0] := Matrix(n, n) :$

> $k := 0;$

for i **from** 1 **to** n **do**
for j **from** 1 **to** n **do**
$FirstNewStop[k][i, j] := j;$
$LastNewStop[k][i, j] := i;$
end do:
$FirstNewStop[k][i, i] := 0; LastNewStop[k][i, i] := 0;$
end do:

$$k := 0 \tag{11}$$

> $print(FlowPerEdge[0], Hops, ActualPath, PathTraffic, EdgeTraffic);$

$$\left[\begin{array}{cccccc} 0 & 9 & 216000 & 31 & 216000 & 12 \\ 216000 & 0 & 60 & 216000 & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & 216000 & 216000 \\ 68 & 43 & 12 & 14 & 0 & 216000 \\ 216000 & 16 & 42 & 8 & 216000 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 1 & 2 & 1 & 3 & 1 \\ 3 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 2 & 0 \end{array}\right], \tag{12}$$

$$\left[\begin{array}{cccccc} 1 & (1, 2) & (1, 4, 3) & (1, 4) & (1, 4, 3, 5) & (1, 6) \\ (2, 3, 5, 1) & 2 & (2, 3) & (2, 3, 4) & (2, 3, 5) & (2, 6) \\ (3, 5, 1) & (3, 5, 2) & 3 & (3, 4) & (3, 5) & (3, 6) \\ (4, 1) & (4, 2) & (4, 3) & 4 & (4, 3, 5) & (4, 2, 6) \\ (5, 1) & (5, 2) & (5, 3) & (5, 4) & 5 & (5, 2, 6) \\ (6, 3, 5, 1) & (6, 2) & (6, 3) & (6, 4) & (6, 3, 5) & 6 \end{array}\right],$$

$$\left[\begin{array}{rrrrrr} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 9 & na & 31 & na & 12 \\ na & 0 & 60 & na & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & na & na \\ 68 & 43 & 12 & 14 & 0 & na \\ na & 16 & 42 & 8 & na & 0 \end{array}\right]$$

>

>

>

```
> for k from 1 to n do
    FlowPerEdge[k] := Matrix(n, n) : FirstNewStop[k] := Matrix(n, n); LastNewStop[k]
      := Matrix(n, n);
  # print(k − 1, FlowPerEdge[k − 1], FirstNewStop[k − 1], LastNewStop[k − 1]);
    for i from 1 to n do
     for j from 1 to n do
         optionDirect := FlowPerEdge[k − 1][i, j];
  #        optionViaK:=max(FlowPerEdge[k − 1][i, k], FlowPerEdge[k − 1][k, j]) :
          optionViaK := FlowPerEdge[k − 1][i, k] + FlowPerEdge[k − 1][k, j] :
         if optionDirect ≤ optionViaK
             then FlowPerEdge[k][i, j] := optionDirect; FirstNewStop[k][i, j] := FirstNewStop[k
       − 1][i, j]; LastNewStop[k][i, j] := LastNewStop[k − 1][i, j];
  #lprint(`Dircect: then `, optionDirect, optionViaK, k, i, j);
             else FlowPerEdge[k][i, j] := optionViaK; FirstNewStop[k][i, j] := FirstNewStop[k
       − 1][i, k]; LastNewStop[k][i, j] := LastNewStop[k − 1][k, j];
   #lprint(`Via K: else`, optionDirect, optionViaK, k, i, j);
         end if;
     end do: #j
     end do: # i
  # print(k, FlowPerEdge[k], FirstNewStop[k], LastNewStop[k]);
   end do: # k
```

```
> k := 0; print(k, FlowPerEdge[k], FirstNewStop[k], LastNewStop[k]);
```

$$k := 0$$

$$0, \left[\begin{array}{rrrrrr} 0 & 9 & 216000 & 31 & 216000 & 12 \\ 216000 & 0 & 60 & 216000 & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & 216000 & 216000 \\ 68 & 43 & 12 & 14 & 0 & 216000 \\ 216000 & 16 & 42 & 8 & 216000 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 3 & 4 & 5 & 6 \\ 1 & 2 & 0 & 4 & 5 & 6 \\ 1 & 2 & 3 & 0 & 5 & 6 \\ 1 & 2 & 3 & 4 & 0 & 6 \\ 1 & 2 & 3 & 4 & 5 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 1 & 1 & 1 & 1 & 1 \\ 2 & 0 & 2 & 2 & 2 & 2 \\ 3 & 3 & 0 & 3 & 3 & 3 \\ 4 & 4 & 4 & 0 & 4 & 4 \\ 5 & 5 & 5 & 5 & 0 & 5 \\ 6 & 6 & 6 & 6 & 6 & 0 \end{array}\right] \quad \textbf{(13)}$$

```
> k := n; print(k, FlowPerEdge[k], FirstNewStop[k], LastNewStop[k]);
```

$$k := 6$$

$$6, \begin{bmatrix} 0 & 9 & 21 & 20 & 9 & 12 \\ 12 & 0 & 12 & 14 & 0 & 22 \\ 0 & 0 & 0 & 14 & 0 & 10 \\ 17 & 26 & 38 & 0 & 26 & 29 \\ 12 & 12 & 12 & 14 & 0 & 22 \\ 25 & 16 & 28 & 8 & 16 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 & 2 & 6 & 2 & 6 \\ 5 & 0 & 5 & 5 & 5 & 5 \\ 1 & 2 & 0 & 2 & 2 & 6 \\ 1 & 2 & 2 & 0 & 2 & 1 \\ 3 & 3 & 3 & 4 & 0 & 3 \\ 4 & 2 & 2 & 4 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 5 & 6 & 2 & 1 \\ 3 & 0 & 5 & 5 & 2 & 3 \\ 3 & 3 & 0 & 5 & 2 & 3 \\ 4 & 4 & 5 & 0 & 2 & 1 \\ 3 & 3 & 5 & 5 & 0 & 3 \\ 4 & 6 & 5 & 6 & 2 & 0 \end{bmatrix} \tag{14}$$

```
> Hops2 := Matrix(n) : ActualPath2 := Matrix(n) : for iii from 1 to n do for jjj from 1 to n
     do ActualPath2[iii, jjj] := iii  end: end: MinSneakyPath := Matrix(n) : MaxSneakyPath
      := Matrix(n) : AvgSneakyPath := Matrix(n) :
  for ii from 1 to n do
  for jj from 1 to n do
   i := ii : j := jj :
   MinSneakyPath[ii, jj] := big :
   MaxSneakyPath[ii, jj] := 0 :
   SumSneakyPath[ii, jj] := 0 :
      while FirstStop[n][i, j] ≠ 0 do
            ifrom := i :
            i := FirstNewStop[n][i, j];
           Hops2[ii, jj] := Hops2[ii, jj] + 1;
           MinSneakyPath[ii, jj] := min(MinSneakyPath[ii, jj], FlowPerEdge[k][ifrom, i]) :

            MaxSneakyPath[ii, jj] := max(MinSneakyPath[ii, jj], FlowPerEdge[k][ifrom, i]) :
            AvgSneakyPath[ii, jj] := AvgSneakyPath[ii, jj] + FlowPerEdge[k][ifrom, i] :
   #          TotalFlow[ifrom, i] := TotalFlow[ifrom, i] + PathTraffic[ii, jj] :
           ActualPath2[ii, jj] := ActualPath2[ii, jj] , i;
       end do;
  if Hops2[ii, jj] > 0  then  AvgSneakyPath[ii, jj] := AvgSneakyPath[ii, jj] / Hops2[ii, jj] : end if;

   #    print(`For`, ii, jj, `the actual path is`, ActualPath2[ii, jj]);
   end do;
   end do;
  for iii from 1 to n do MinSneakyPath[iii, iii] := 0  end:
> print(evalm(Hops), evalm(Hops2),  evalm(Hops − Hops2));
   print( ActualPath, ActualPath2);
   print(MinSneakyPath, MaxSneakyPath, map(evalf, AvgSneakyPath));
   print( EdgeCost, PathTraffic, EdgeTraffic, TotalFlow);
```

$$\begin{bmatrix} 0 & 1 & 2 & 1 & 3 & 1 \\ 3 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 3 & 2 & 2 & 1 \\ 3 & 0 & 2 & 2 & 1 & 3 \\ 1 & 1 & 0 & 3 & 2 & 1 \\ 1 & 1 & 3 & 0 & 2 & 2 \\ 2 & 2 & 1 & 1 & 0 & 2 \\ 2 & 1 & 3 & 1 & 2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 & -2 \\ 1 & 1 & 0 & -2 & -1 & 0 \\ 0 & 0 & -2 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -2 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & (1,2) & (1,4,3) & (1,4) & (1,4,3,5) & (1,6) \\ (2,3,5,1) & 2 & (2,3) & (2,3,4) & (2,3,5) & (2,6) \\ (3,5,1) & (3,5,2) & 3 & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & 4 & (4,3,5) & (4,2,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & 5 & (5,2,6) \\ (6,3,5,1) & (6,2) & (6,3) & (6,4) & (6,3,5) & 6 \end{bmatrix},$$

$$\begin{bmatrix} 1 & (1,2) & (1,2,5,3) & (1,6,4) & (1,2,5) & (1,6) \\ (2,5,3,1) & 2 & (2,5,3) & (2,5,4) & (2,5) & (2,5,3,6) \\ (3,1) & (3,2) & 3 & (3,2,5,4) & (3,2,5) & (3,6) \\ (4,1) & (4,2) & (4,2,5,3) & 4 & (4,2,5) & (4,1,6) \\ (5,3,1) & (5,3,2) & (5,3) & (5,4) & 5 & (5,3,6) \\ (6,4,1) & (6,2) & (6,2,5,3) & (6,4) & (6,2,5) & 6 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 9 & 0 & 8 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \\ 17 & 26 & 0 & 0 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 8 & 16 & 0 & 8 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 9 & 12 & 8 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 0 & 0 & 0 & 14 & 0 & 10 \\ 17 & 26 & 12 & 0 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 17 & 16 & 12 & 8 & 0 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0. & 9. & 7. & 10. & 4.500 & 12. \\ 4. & 0. & 6. & 7. & 0. & 7.333 \\ 0. & 0. & 0. & 4.667 & 0. & 10. \\ 17. & 26. & 12.67 & 0. & 13. & 14.50 \\ 6. & 6. & 12. & 14. & 0. & 11. \\ 12.50 & 16. & 9.333 & 8. & 8. & 0. \end{bmatrix}$$

(15)

$$\left[\begin{array}{cccccc} & 7 & na & 7 & na & 9 \\ na & & 5 & na & 10 & 3 \\ 9 & 10 & & 8 & 4 & 6 \\ 9 & 4 & 2 & & na & na \\ 3 & 5 & 10 & 10 & & na \\ na & 5 & 8 & 10 & na & \end{array}\right], \left[\begin{array}{cccccc} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{array}\right], \left[\begin{array}{cccccc} 0 & 9 & na & 31 & na & 12 \\ na & 0 & 60 & na & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & na & na \\ 68 & 43 & 12 & 14 & 0 & na \\ na & 16 & 42 & 8 & na & 0 \end{array}\right], \qquad (15)$$

$TotalFlow$

> $Digits := 4; map(evalf, AvgSneakyPath);$

$$Digits := 4$$

$$\left[\begin{array}{cccccc} 0. & 9. & 7. & 10. & 4.500 & 12. \\ 4. & 0. & 6. & 7. & 0. & 7.333 \\ 0. & 0. & 0. & 4.667 & 0. & 10. \\ 17. & 26. & 12.67 & 0. & 13. & 14.50 \\ 6. & 6. & 12. & 14. & 0. & 11. \\ 12.50 & 16. & 9.333 & 8. & 8. & 0. \end{array}\right] \qquad (16)$$

>

>

>

>

```
> print(`This is an example for the Sneaky Path, November 2016`);
```

This is an example for the Sneaky Path, November 2016 **(17)**

```
> print(` The size n is`, n);
```

The size n is, 6 **(18)**

```
> print(`The Edge Matrix is E`, EdgeCost,
    ` where "na" indicates that there exists no link between these points`);
```

$$\textit{The Edge Matrix is E,} \begin{bmatrix} & 7 & na & 7 & na & 9 \\ na & & 5 & na & 10 & 3 \\ 9 & 10 & & 8 & 4 & 6 \\ 9 & 4 & 2 & & na & na \\ 3 & 5 & 10 & 10 & & na \\ na & 5 & 8 & 10 & na & \end{bmatrix}, \tag{19}$$

where "na" indicates that there exists no link between these points

```
> print(` The  all-pairs-shortest-paths for this given matrix is `, MyDist[n]);
```

$$\textit{The all-pairs-shortest-paths for this given matrix is ,} \begin{bmatrix} 0 & 7 & 9 & 7 & 13 & 9 \\ 12 & 0 & 5 & 13 & 9 & 3 \\ 7 & 9 & 0 & 8 & 4 & 6 \\ 9 & 4 & 2 & 0 & 6 & 7 \\ 3 & 5 & 10 & 10 & 0 & 8 \\ 15 & 5 & 8 & 10 & 12 & 0 \end{bmatrix} \tag{20}$$

```
> print(` The hop-count to obtain these shortest paths is`, Hops);
```

$$\textit{The hop-count to obtain these shortest paths is,} \begin{bmatrix} 0 & 1 & 2 & 1 & 3 & 1 \\ 3 & 0 & 1 & 2 & 2 & 1 \\ 2 & 2 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 2 & 2 \\ 1 & 1 & 1 & 1 & 0 & 2 \\ 3 & 1 & 1 & 1 & 2 & 0 \end{bmatrix} \tag{21}$$

```
> print(`The actual all-pairs-shortest-paths is`, ActualPath);
```

The actual all-pairs-shortest-paths is, **(22)**

$$\begin{bmatrix} 1 & (1,2) & (1,4,3) & (1,4) & (1,4,3,5) & (1,6) \\ (2,3,5,1) & 2 & (2,3) & (2,3,4) & (2,3,5) & (2,6) \\ (3,5,1) & (3,5,2) & 3 & (3,4) & (3,5) & (3,6) \\ (4,1) & (4,2) & (4,3) & 4 & (4,3,5) & (4,2,6) \\ (5,1) & (5,2) & (5,3) & (5,4) & 5 & (5,2,6) \\ (6,3,5,1) & (6,2) & (6,3) & (6,4) & (6,3,5) & 6 \end{bmatrix}$$

> *print*(\`*The total demand flow between two nodes is the matrix F, and given by (this is input to the program)*\`, *PathTraffic*);

The total demand flow between two nodes is the matrix F, and given by (this is input to the program), **(23)**

$$\begin{bmatrix} 0 & 9 & 11 & 12 & 8 & 12 \\ 18 & 0 & 15 & 10 & 17 & 18 \\ 17 & 18 & 0 & 14 & 10 & 10 \\ 17 & 8 & 10 & 0 & 17 & 18 \\ 15 & 9 & 12 & 14 & 0 & 16 \\ 18 & 16 & 15 & 8 & 9 & 0 \end{bmatrix}$$

> *print*(\`*which combines to create traffic load on each edge, wwhich is geiven matrix L  as* \`, *EdgeTraffic*);

which combines to create traffic load on each edge, wwhich is geiven matrix L as , **(24)**

$$\begin{bmatrix} 0 & 9 & na & 31 & na & 12 \\ na & 0 & 60 & na & 0 & 52 \\ 0 & 0 & 0 & 24 & 132 & 10 \\ 17 & 26 & 46 & 0 & na & na \\ 68 & 43 & 12 & 14 & 0 & na \\ na & 16 & 42 & 8 & na & 0 \end{bmatrix}$$

> *print*(\`*Note that there are zeros on the diagonal (as expected), but also zeros away from the diagonal, representing links that everybody considers too long*\`);

Note that there are zeros on the diagonal (as expected), but also zeros away from the diagonal, representing links that everybody considers too long **(25)**

> *print*(\` The  all-pairs-fewest-cars for this given matrix is \`, *FlowPerEdge*[n], \`*(be careful, there are two or more different and  acceptable definitions for fewest cars)*\`);

The all-pairs-fewest-cars for this given matrix is , $\begin{bmatrix} 0 & 9 & 21 & 20 & 9 & 12 \\ 12 & 0 & 12 & 14 & 0 & 22 \\ 0 & 0 & 0 & 14 & 0 & 10 \\ 17 & 26 & 38 & 0 & 26 & 29 \\ 12 & 12 & 12 & 14 & 0 & 22 \\ 25 & 16 & 28 & 8 & 16 & 0 \end{bmatrix}$, **(26)**

(be careful, there are two or more different and acceptable definitions for fewest cars)

> *print*(\` *The hop-count to obtain these fewest cars- paths is*\`, *Hops2*);

(27)

The hop-count to obtain these fewest cars- paths is, $\begin{bmatrix} 0 & 1 & 3 & 2 & 2 & 1 \\ 3 & 0 & 2 & 2 & 1 & 3 \\ 1 & 1 & 0 & 3 & 2 & 1 \\ 1 & 1 & 3 & 0 & 2 & 2 \\ 2 & 2 & 1 & 1 & 0 & 2 \\ 2 & 1 & 3 & 1 & 2 & 0 \end{bmatrix}$ **(27)**

> *print*(\`*The actual all-pairs-fewest-cars-paths, or the all-pairs-sneaky-paths are*\`, *ActualPath2*);

The actual all-pairs-fewest-cars-paths, or the all-pairs-sneaky-paths are, **(28)**

$$\begin{bmatrix} 1 & (1,2) & (1,2,5,3) & (1,6,4) & (1,2,5) & (1,6) \\ (2,5,3,1) & 2 & (2,5,3) & (2,5,4) & (2,5) & (2,5,3,6) \\ (3,1) & (3,2) & 3 & (3,2,5,4) & (3,2,5) & (3,6) \\ (4,1) & (4,2) & (4,2,5,3) & 4 & (4,2,5) & (4,1,6) \\ (5,3,1) & (5,3,2) & (5,3) & (5,4) & 5 & (5,3,6) \\ (6,4,1) & (6,2) & (6,2,5,3) & (6,4) & (6,2,5) & 6 \end{bmatrix}$$

>

> *print*(\`*The Minimum, the maximum, and the average cars per edge on each path is given by* \`, *MinSneakyPath*, *MaxSneakyPath*, *map*(*evalf*, *AvgSneakyPath*));

The Minimum, the maximum, and the average cars per edge on each path is given by , **(29)**

$$\begin{bmatrix} 0 & 9 & 0 & 8 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \\ 17 & 26 & 0 & 0 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 8 & 16 & 0 & 8 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 9 & 12 & 8 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 0 & 0 & 0 & 14 & 0 & 10 \\ 17 & 26 & 12 & 0 & 0 & 12 \\ 0 & 0 & 12 & 14 & 0 & 10 \\ 17 & 16 & 12 & 8 & 0 & 0 \end{bmatrix},$$

$$\begin{bmatrix} 0. & 9. & 7. & 10. & 4.500 & 12. \\ 4. & 0. & 6. & 7. & 0. & 7.333 \\ 0. & 0. & 0. & 4.667 & 0. & 10. \\ 17. & 26. & 12.67 & 0. & 13. & 14.50 \\ 6. & 6. & 12. & 14. & 0. & 11. \\ 12.50 & 16. & 9.333 & 8. & 8. & 0. \end{bmatrix}$$

>