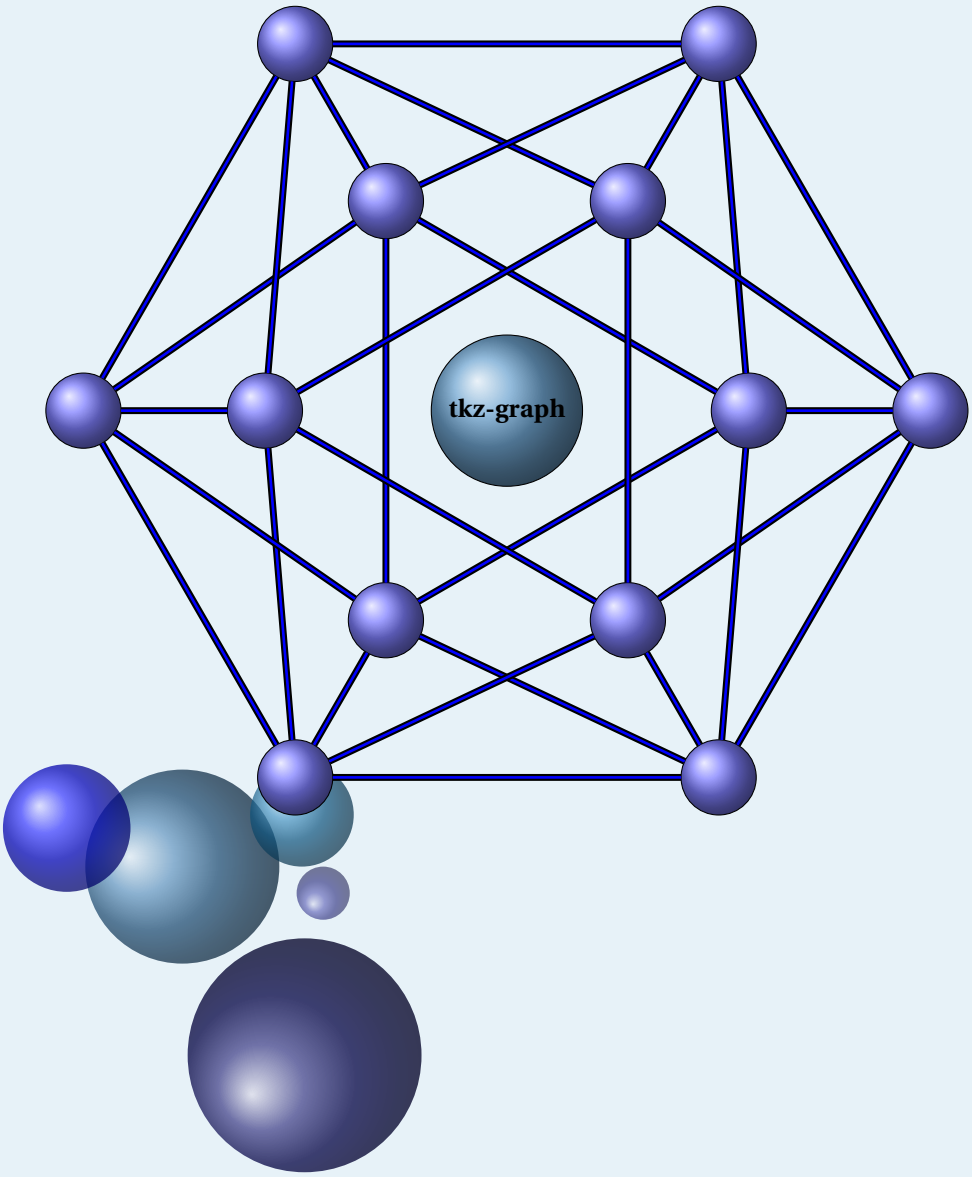
tkz-graph 2.30c

AlterMundus



Alain Matthes

November 2, 2026 Documentation V.2.30c

http://altermundus.fr

tkz-graph

AlterMundus

Alain Matthes

tkz-graph is a package to draw graphs with Ti*k*Z as simply as possible: vertices, edges, loops, labels and styles are created with a few LaTeX macros. It belongs to a series of packages whose common aim is the creation of drawings useful for teaching mathematics. It comes with the package **tkz-berge**, which draws the classic graphs of graph theory.

☞ I would like to thank **Till Tantau** for creating the wonderful tool Ti*k*Z.

☞ You will find many examples on my site: altermundus.fr

Please send your comments and bug reports to Alain Matthes.
This file can be redistributed and/or modified under the terms of the LaTeX Project Public License, version 1.3c or later (https://www.latex-project.org/lppl.txt).

Contents

1 Getting started 6
1.1 A first graph . . . 6
1.2 A classic example . . . 6
1.3 Loading the package . . . 7
1.3.1 Short names and prefixed names . . . 7
1.3.2 Upgrading from version 2.00 . . . 8

2 Vertices 9
2.1 \Vertex . . . 9
2.1.1 Cartesian coordinates . . . 9
2.1.2 Polar coordinates . . . 9
2.1.3 Option Node: a vertex on an existing node . . . 10
2.1.4 Option position: placement with Ti*k*Z . . . 10
2.1.5 Option empty: an invisible vertex . . . 10
2.2 Relative placement: the shortcuts . . . 11
2.2.1 The shortcuts with the default unit . . . 11
2.2.2 A new unit with \SetGraphUnit . . . 11
2.2.3 The units of Ti*k*Z: x=2cm,y=1cm . . . 12
2.2.4 A graph built from a coordinate . . . 12
2.2.5 A local unit . . . 12

3 Several vertices at once 13
3.1 A line from a given point . . . 13
3.2 A line from an existing node . . . 13
3.3 Lines with a direction . . . 14
3.4 Triangles . . . 14
3.5 A square . . . 14
3.6 A circle . . . 15
3.6.1 Options rotation and clockwise . . . 15
3.6.2 A circle and outside labels . . . 15
3.6.3 A circle and a shortcut . . . 16

4 Labels of the vertices 17
4.1 Options of \Vertex for the labels . . . 17
4.1.1 Option L . . . 17
4.1.2 Option Math . . . 17
4.1.3 Option NoLabel . . . 17
4.1.4 Options LabelOut, Lpos and Ldist . . . 17
4.2 Macros for all the vertices . . . 17

5 Edges 19
5.1 \Edge . . . 19
5.1.1 Straight edges and arcs . . . 19
5.1.2 The option style . . . 20
5.1.3 Arrows . . . 20
5.1.4 Double edges . . . 20
5.1.5 Labels . . . 20
5.2 \Edges . . . 21
5.3 A loop: \Loop . . . 21
5.4 Default values for the edges: \SetUpEdge . . . 23
5.5 The styles EdgeStyle and LabelStyle . . . 23

6 Styles 24
6.1 \GraphInit . . . 24
6.1.1 The predefined styles . . . 24
6.2 A new style: \tkzDefGraphStyle . . . 24
6.3 Settings: \tkzSetUpGraph . . . 26
6.4 Refining a style . . . 28
6.5 Colors of the predefined styles . . . 30
6.6 Other macros . . . 31
6.7 Redefining VertexStyle with \tikzset . . . 31

7 Coloring and highlighting 32
7.1 Coloring vertices: \AddVertexColor . . . 32
7.2 Highlighting a path: \tkzHighlightPath . . . 32
7.2.1 A Hamiltonian cycle of the cube: the Gray code . . . 33

8 Graphs and matrices 34
8.1 The adjacency matrix of a graph: \tkzAdjacencyMatrix . . . 34
8.2 A graph from a matrix: \tkzGraphFromMatrix . . . 34
8.2.1 Placed vertices . . . 35

9 Probabilistic graphs 36
9.1 Any number of states: \grProbMatrix . . . 36
9.2 Two states: \grProb . . . 37
9.3 Three states: \grProbThree . . . 38

10 Examples 39
10.1 Coloring with the Welsh–Powell algorithm . . . 39
10.2 Dijkstra's algorithm . . . 40
10.3 Classic graphs with tkz-berge . . . 41

11 Computations with LuaLaTeX: the option lua 42
11.1 An optional part of the package . . . 42
11.1.1 The recorded graph . . . 42
11.2 Order, size and degrees . . . 42
11.3 Shortest path: \tkzGraphDijkstra . . . 44
11.3.1 A directed graph . . . 46
11.3.2 A probabilistic graph . . . 46
11.4 Coloring: \tkzGraphColoring . . . 47
11.5 Connectivity and Eulerian paths . . . 49
11.5.1 The envelope . . . 49
11.5.2 The bridges of Königsberg . . . 50
11.5.3 A cycle in a directed graph . . . 50
11.6 Powers of the adjacency matrix . . . 51
11.7 Texts of the tables . . . 51

12 Exercises from the French baccalaureate 52
12.1 North America, June 2003 . . . 52
12.2 French West Indies and Guiana, June 2003 . . . 54
12.3 Asia, June 2003 . . . 56
12.4 France, June 2003 . . . 58
12.5 Foreign centers, June 2003 . . . 59
12.6 North America, June 2004 . . . 61
12.7 Foreign centers, May 2004 . . . 63
12.8 France, June 2004 . . . 64

12.9 Réunion, June 2004 65
12.10 South America, November 2006 67
12.11 Lebanon, June 2006 69

Index 71

1 Getting started

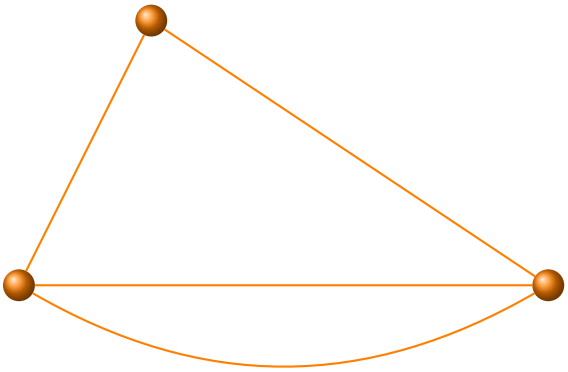
TikZ is a pleasant tool for drawing graphs, so pleasant that one may wonder whether a package is needed at all. **tkz-graph** contains no graph theory: it only provides tools to *draw* graphs. Four reasons justify it:

1. some users do not want to learn Ti*k*Z; a few LaTeX macros with a simple syntax are enough to draw the graphs used in teaching;
2. the styles of the vertices and of the edges are managed for you: a graph can be drawn without a single coordinate, and a new look is obtained by changing a style;
3. the package **tkz-berge** relies on **tkz-graph** to draw the classic graphs of graph theory (Petersen, Heawood, hypercubes, trees…);
4. it is a gentle introduction to Ti*k*Z: every macro accepts Ti*k*Z options.

1.1 A first graph

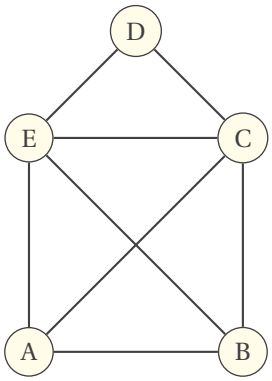
Before reading the description of the macros, check that the package is installed with this complete document.

```
\documentclass[border=3mm]{standalone}
\usepackage{tkz-graph}
\begin{document}
\begin{tikzpicture}[scale=1.75]
  \GraphInit[vstyle=Art]
  \Vertex{A}
  \Vertex[x=4,y=0]{B}
  \Vertex[x=1,y=2]{C}
  \Edge[style={bend left}](B)(A)
  \Edges(A,B,C,A)
\end{tikzpicture}
\end{document}
```



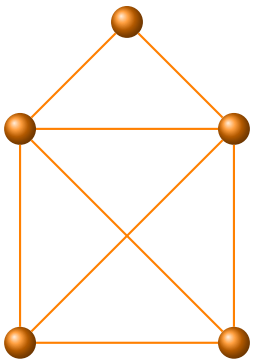
1.2 A classic example

The style **Normal** is the default one. **\Vertices** places a list of vertices on a circle, **\NOEA** places a vertex to the north-east of another one and **\Edges** draws a chain of edges.




```
\begin{tikzpicture}
  \GraphInit[vstyle=Normal]
  \SetGraphUnit{2}
  \Vertices[rotation=-135]{circle}{A,B,C,E}
  \NOEA[unit=1.414](E){D}
  \Edges(A,B,E,D,C,E,A,C,B)
\end{tikzpicture}
```

The same graph with another style:



```
\begin{tikzpicture}
  \GraphInit[vstyle=Art]
  \SetGraphUnit{2}
  \Vertices[rotation=-135]{circle}{A,B,C,E}
  \NOEA[unit=1.414](E){D}
  \Edges(A,B,E,D,C,E,A,C,B)
\end{tikzpicture}
```

1.3 Loading the package

```
\usepackage{tkz-graph}                % or
\usepackage[noshortnames]{tkz-graph}
```

The package loads Ti*kZ* (with its library **arrows**) and **ifthen**. It works with **pdflatex**, **xelatex** and **lualatex**.

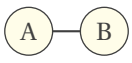
1.3.1 Short names and prefixed names

The drawing macros exist under two names:

short name	prefixed name
\Vertex, \Vertices	\tkzVertex, \tkzVertices
\Edge, \Edges, \Loop	\tkzEdge, \tkzEdges, \tkzLoop
\EA, \WE, \NO, \SO	\tkzEA, \tkzWE, \tkzNO, \tkzSO
\NOEA, \NOWE, \SOEA, \SOWE	\tkzNOEA, \tkzNOWE, \tkzSOEA, \tkzSOWE

Short names like \EA or \NO are often used by other packages or by the user (\EA is a classic abbreviation of \expandafter). For this reason:

- the prefixed names are always defined;
- the short names are installed at the beginning of each **tikzpicture** environment. Outside the pictures, a definition of \EA or \NO made by the user or by another package keeps its meaning;
- if a short name is already defined when **tkz-graph** is loaded, a warning gives the prefixed name to use outside the pictures;
- with the option **noshortnames**, the short names are never defined.



ok

```
\let\EA\expandafter % user definition
\begin{tikzpicture}
  \Vertex{A} \EA(A){B} % EA of tkz-graph
  \Edge(A)(B)
\end{tikzpicture}

\EA\def\csname test\endcsname{ok}
\test % \EA is \expandafter again
```

The macros of settings (\GraphInit, \tkzSetUpGraph, \SetUpEdge…) have long names and are always defined.

1.3.2 Upgrading from version 2.00

Documents written for the version 2.00 give the same result, with two exceptions:

- **xkeyval** is no longer loaded: the keys use **pgfkeys**. A document that uses **xkeyval** without loading it must load it.
- A second **\GraphInit[vstyle=…]** in the same group was ignored by the version 2.00 (and its value was written in the text); it now works.

tkz-berge must be updated at the same time (version 2.2 or later): the old **tkz-berge** needs **xkeyval**. The list of all the changes is in the file **CHANGELOG.md**.

2 Vertices

\Vertex is the main macro: it places a vertex. A vertex can be placed with cartesian or polar coordinates, on an existing node, or relatively to another vertex.

2.1 \Vertex

\Vertex[⟨local options⟩]{⟨name⟩}

A vertex has a name, a position, a label and a style.

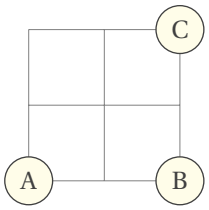
The argument name is mandatory: it is the name of the TikZ node, used later to draw the edges. Do not confuse it with the label, the text displayed in the vertex: the name can be M1 and the label M_1.

Options	Default	Definition
x	{}	abscissa
y	{}	ordinate
a	{}	angle (polar coordinates)
d	{}	distance (polar coordinates)
Node	false	the vertex is placed on an existing node
position	{}	TikZ relative placement (right=of A…)
unit	\GraphUnit	distance for the shortcuts
dir	\EA	direction used by \Vertices
style	{}	TikZ options added to the node
empty	false	the vertex is an invisible node
NoLabel	false	no label
LabelOut	false	the label is outside the node
L	{}	the label
Math	false	the label is written in math mode
Ldist	0cm	distance between the node and an outside label
Lpos	0	angle of an outside label

If L *is empty, the label is the name of the vertex. Without* x, y, a, d *and* position*, the vertex is placed at the origin.*

2.1.1 Cartesian coordinates

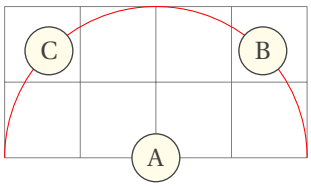
\Vertex[x=⟨number⟩,y=⟨number⟩]{⟨name⟩}.




```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (2,2);
  \Vertex{A} % x=0, y=0
  \Vertex[x=2,y=0]{B}
  \Vertex[x=2,y=2]{C}
\end{tikzpicture}
```

2.1.2 Polar coordinates

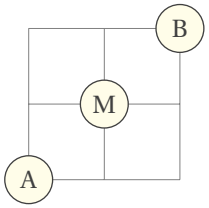
\Vertex[a=⟨angle⟩,d=⟨distance⟩]{⟨name⟩}.




```
\begin{tikzpicture}
  \draw[help lines] (-2,0) grid (2,2);
  \draw[red] (2,0) arc (0:180:2cm);
  \Vertex{A}
  \Vertex[a=45,d=2cm]{B}
  \Vertex[a=135,d=2cm]{C}
\end{tikzpicture}
```

2.1.3 Option Node: a vertex on an existing node

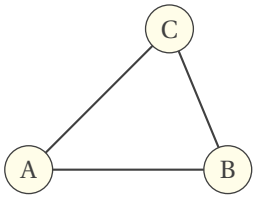
The vertex is placed on a node or on a **coordinate** already defined.




```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (2,2);
  \Vertex{A} \Vertex[x=2,y=2]{B}
  \coordinate (M) at ($(A)!.5!(B)$);
  \Vertex[Node]{M}
\end{tikzpicture}
```

2.1.4 Option position: placement with TikZ

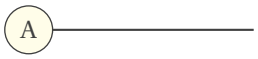
With the library **positioning**:




```
\begin{tikzpicture}[node distance=2cm]
  \Vertex{A}
  \Vertex[position={right=of A}]{B}
  \Vertex[position={above right=of A}]{C}
  \Edges(A,B,C,A)
\end{tikzpicture}
```

2.1.5 Option empty: an invisible vertex

The node exists (edges can reach it) but nothing is drawn.




```
\begin{tikzpicture}
  \Vertex{A}
  \Vertex[x=3,y=0,empty]{B}
  \Edge(A)(B)
\end{tikzpicture}
```

2.2 Relative placement: the shortcuts

A distance between two vertices is needed: the unit of the graph.

`\SetGraphUnit{⟨number⟩}`

Sets the distance between the centers of two vertices placed with a shortcut. The number is expressed in the units of TikZ (1 cm by default). The default value is 1. *The definition is local to a group or a* `scope` *environment.*

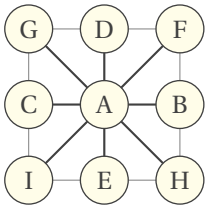
`\EA[⟨local options⟩](⟨vertex A⟩){⟨vertex B⟩}`

The shortcuts create `vertex B` *relatively to* `vertex A`. *The distance is* `unit` *horizontally and vertically:* `unit` *is given by* `\SetGraphUnit` *(or by* `\GraphInit`, *or* `\tkzSetUpGraph`*) or by the option* `unit`. *The options are those of* `\Vertex`.

Shortcut	Definition
`\EA`	east
`\WE`	west
`\NO`	north
`\SO`	south
`\NOEA`	north-east (north then east)
`\NOWE`	north-west
`\SOEA`	south-east
`\SOWE`	south-west

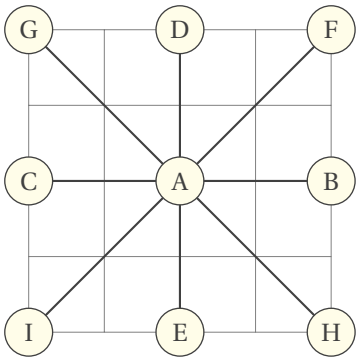
With `\NOEA`, *the distance between the two vertices is* $\sqrt{2}\times$ `unit`.

2.2.1 The shortcuts with the default unit



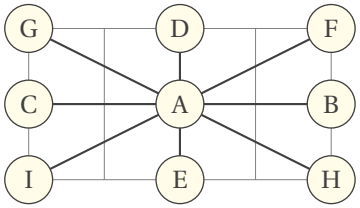

```
\begin{tikzpicture}
  \draw[help lines] (-1,-1) grid (1,1);
  \Vertex{A}
  \EA(A){B}   \WE(A){C}   \NO(A){D}   \SO(A){E}
  \NOEA(A){F} \NOWE(A){G} \SOEA(A){H} \SOWE(A){I}
  \foreach \v in {B,C,D,E,F,G,H,I}{\Edge(A)(\v)}
\end{tikzpicture}
```

2.2.2 A new unit with \SetGraphUnit



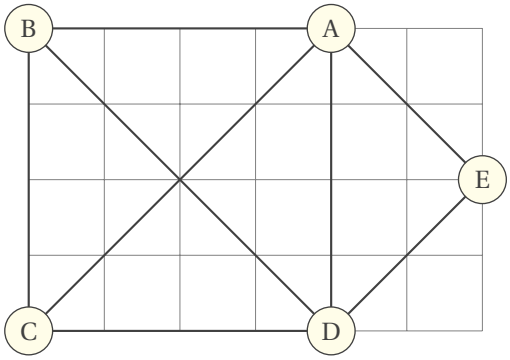

```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (2,2);
  \SetGraphUnit{2}
  \Vertex{A}
  \EA(A){B}   \WE(A){C}   \NO(A){D}   \SO(A){E}
  \NOEA(A){F} \NOWE(A){G} \SOEA(A){H} \SOWE(A){I}
  \foreach \v in {B,C,D,E,F,G,H,I}{\Edge(A)(\v)}
\end{tikzpicture}
```

2.2.3 The units of TikZ: x=2cm,y=1cm




```
\begin{tikzpicture}[x=2cm,y=1cm]
  \draw[help lines] (-1,-1) grid (1,1);
  \Vertex{A}
  \EA(A){B}   \WE(A){C}   \NO(A){D}   \SO(A){E}
  \NOEA(A){F} \NOWE(A){G} \SOEA(A){H} \SOWE(A){I}
  \foreach \v in {B,C,D,E,F,G,H,I}{\Edge(A)(\v)}
\end{tikzpicture}
```

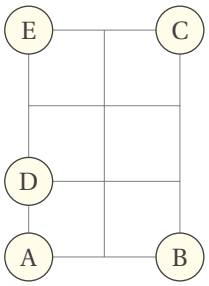
2.2.4 A graph built from a coordinate




```
\begin{tikzpicture}
  \draw[help lines] (-2,-2) grid (4,2);
  \SetGraphUnit{2}
  \coordinate (O) at (0,0);
  \NOEA(O){A} \NOWE(O){B} \SOEA(O){D}
  \SOWE(O){C} \NOEA(D){E}
  \Edges(B,C,D,A,E,D,B,A,C)
\end{tikzpicture}
```

2.2.5 A local unit

The option **unit**, a **scope** environment or a TEX group limit the change of unit.




```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (2,3);
  \SetGraphUnit{2}
  \Vertex{A}  \EA(A){B}
  \NO[unit=3](B){C}
  {\SetGraphUnit{1} \NO(A){D}}
  \NO(D){E}
\end{tikzpicture}
```

3 Several vertices at once

`\Vertices` places a list of vertices on a line or on a simple geometric figure: triangle, square or circle.

`\Vertices[⟨local options⟩]{⟨form⟩}{⟨list of vertices⟩}`

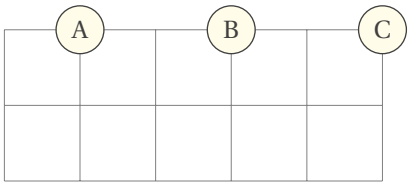
First argument	Definition
line	vertices on a line; the option **dir** gives the direction
tr1	first form of triangle
tr2	second form of triangle
tr3	third form of triangle
tr4	fourth form of triangle
square	four vertices on a square
circle	vertices on a circle (regular polygon)

The second argument is a list of names of vertices.

Options	Default	Definition
dir	\EA	direction of a line
unit	\GraphUnit	side, or radius of the circle
rotation	0	**circle**: angle of the first vertex
clockwise	false	**circle**: the vertices turn clockwise

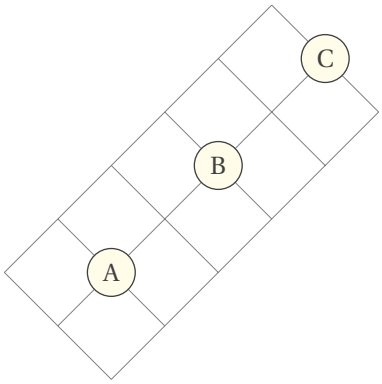
The other options are those of **\Vertex***; they apply to every vertex. For a line, the possible directions are* **\EA***,* **\WE***,* **\NO***,* **\SO***,* **\NOEA***,* **\NOWE***,* **\SOEA** *and* **\SOWE***.*

3.1 A line from a given point



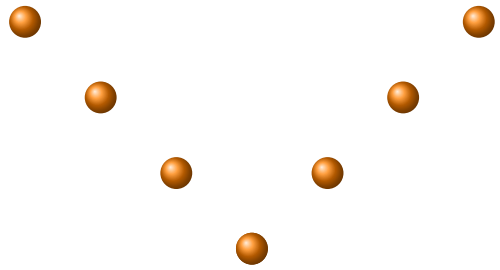

```
\begin{tikzpicture}
  \SetGraphUnit{2}
  \draw[help lines] (0,0) grid (5,2);
  \Vertices[x=1,y=2]{line}{A,B,C}
\end{tikzpicture}
```

3.2 A line from an existing node




```
\begin{tikzpicture}[rotate=45]
  \SetGraphUnit{2}
  \draw[help lines] (0,0) grid (5,2);
  \coordinate (A) at (1,1);
  \Vertices[Node]{line}{A,B,C}
\end{tikzpicture}
```

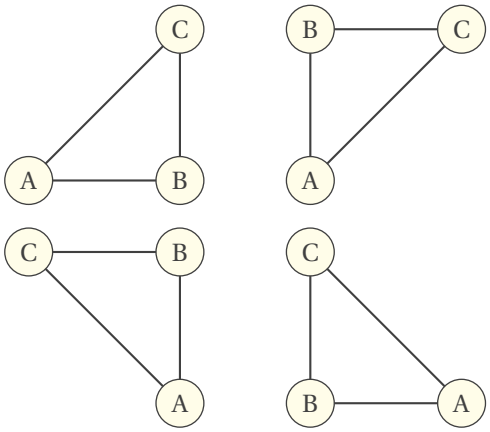
3.3 Lines with a direction



```
\begin{tikzpicture}
  \GraphInit[vstyle=Art]
  \Vertices[dir=\NOEA]{line}{A,B,C,D}
  \Vertices[dir=\NOWE]{line}{A,E,F,G}
\end{tikzpicture}
```

3.4 Triangles

The triangles are right isosceles triangles. Here are the forms **tr1**, **tr2**, **tr3** and **tr4**.

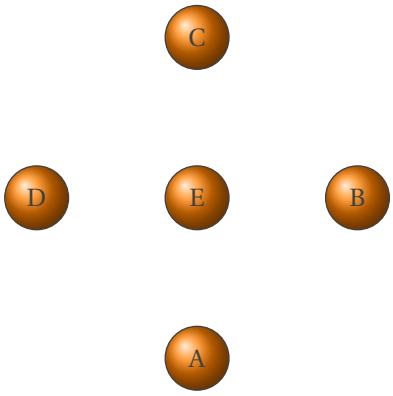



```
\begin{tikzpicture}\SetGraphUnit{2}
  \Vertices{tr1}{A,B,C}\Edges(A,B,C,A)
\end{tikzpicture}\hspace*{1cm}
\begin{tikzpicture}\SetGraphUnit{2}
  \Vertices{tr2}{A,B,C}\Edges(A,B,C,A)
\end{tikzpicture}
```

```
\begin{tikzpicture}\SetGraphUnit{2}
  \Vertices{tr3}{A,B,C}\Edges(A,B,C,A)
\end{tikzpicture}\hspace*{1cm}
\begin{tikzpicture}\SetGraphUnit{2}
  \Vertices{tr4}{A,B,C}\Edges(A,B,C,A)
\end{tikzpicture}
```

3.5 A square

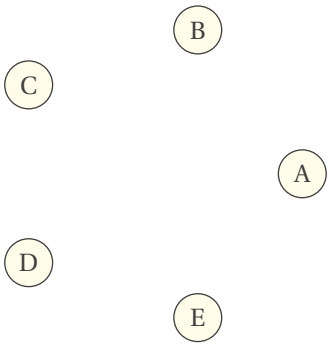
The center of the square is obtained with an intersection (see the **pgfmanual**).




```
\begin{tikzpicture}
  \SetGraphUnit{3}
  \GraphInit[vstyle=Shade]
  \Vertices{square}{A,B,C,D}
  \coordinate (E) at (intersection of A--C and B--D);
  \Vertex[Node]{E}
\end{tikzpicture}
```

3.6 A circle

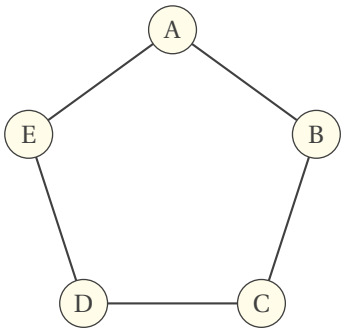
By default the first vertex is at the angle 0 and the vertices turn counterclockwise.




```
\begin{tikzpicture}
  \SetGraphUnit{2}
  \Vertices{circle}{A,B,C,D,E}
\end{tikzpicture}
```

3.6.1 Options rotation and clockwise

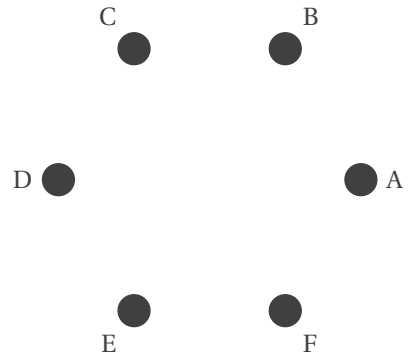
rotation gives the angle of the first vertex; with **clockwise**, the vertices turn clockwise.



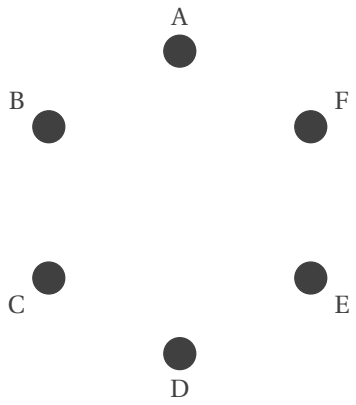

```
\begin{tikzpicture}
  \SetGraphUnit{2}
  \Vertices[rotation=90,clockwise]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A)
\end{tikzpicture}
```

3.6.2 A circle and outside labels

With an outside label, **Lpos** is added to the angle of each vertex: the labels follow the rotation of the figure.

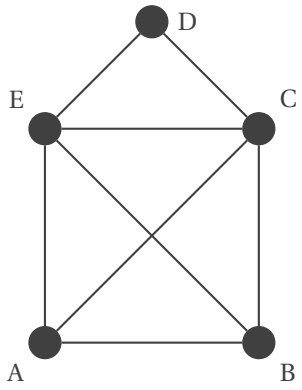



```
\begin{tikzpicture}
  \SetGraphUnit{2}
  \GraphInit[vstyle=Classic]
  \Vertices{circle}{A,B,C,D,E,F}
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \GraphInit[vstyle=Classic]
  \Vertices[unit=2,rotation=90]{circle}{A,B,C,D,E,F}
\end{tikzpicture}
```

3.6.3 A circle and a shortcut




```
\begin{tikzpicture}[scale=.5]
  \SetGraphUnit{4}
  \GraphInit[vstyle=Classic]
  \Vertices[rotation=45]{circle}{C,E,A,B}
  \NOEA[unit=2.828](E){D}
  \Edges(A,B,E,D,C,E,A,C,B)
\end{tikzpicture}
```

4 Labels of the vertices

If no label is given, the label of a vertex is its name. The behavior of the labels can be changed locally with the options of `\Vertex`, or for all the vertices with the macros of this section.

4.1 Options of \Vertex for the labels

4.1.1 Option L




```
\begin{tikzpicture}
  \Vertex[L=$\alpha$]{a}
  \EA[unit=4](a){b}
\end{tikzpicture}
```

4.1.2 Option Math

The label is written in math mode; there is no need to give `L` in math mode.




```
\begin{tikzpicture}
  \Vertex[Math]{A_1}
  \Vertex[Math,L=\alpha,x=4,y=0]{a}
\end{tikzpicture}
```

4.1.3 Option NoLabel

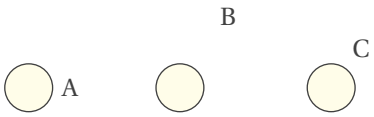
The label is not displayed. To remove the labels of all the vertices, use `\SetVertexNoLabel`.



```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex[NoLabel]{A}
  \EA[NoLabel](A){B}
\end{tikzpicture}
```

4.1.4 Options LabelOut, Lpos and Ldist

The first option places the label outside the node, the second one gives its direction (an angle) and the last one the distance between the label and the node.




```
\begin{tikzpicture}
  \Vertex[LabelOut]{A}
  \Vertex[LabelOut,Lpos=60,Ldist=.5cm,x=2,y=0]{B}
  \Vertex[LabelOut,Lpos=60,x=4,y=0]{C}
\end{tikzpicture}
```

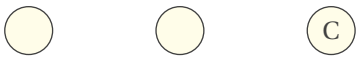
4.2 Macros for all the vertices

\SetVertexNoLabel

The labels of the next vertices are removed (option **NoLabel***).*

\SetVertexLabel

The labels are displayed again.




```
\begin{tikzpicture}
  \SetVertexNoLabel
  \SetGraphUnit{2}
  \Vertex{A}     \EA(A){B}
  \SetVertexLabel  \EA(B){C}
\end{tikzpicture}
```

\SetVertexMath

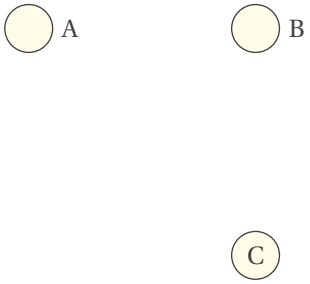
The labels are written in math mode (option **Math***). The macro also applies to the graphs of* **tkz-berge***.* **\SetVertexNoMath** *cancels it.*




```
\begin{tikzpicture}
  \SetVertexMath
  \Vertex{A_1}  \EA[unit=3](A_1){A_2}
\end{tikzpicture}
```

\SetVertexLabelOut

The labels are placed outside the vertices (option **LabelOut***);* **Lpos** *and* **Ldist** *give their position.* **\SetVertexLabelIn** *writes the labels inside the vertices again.*




```
\begin{tikzpicture}
  \SetGraphUnit{3}
  \SetVertexLabelOut
  \Vertex{A}  \EA(A){B}
  \SetVertexLabelIn  \SO(B){C}
\end{tikzpicture}
```

These macros act until the end of the current group (a **scope** environment, a **tikzpicture** or braces).

5 Edges

5.1 \Edge

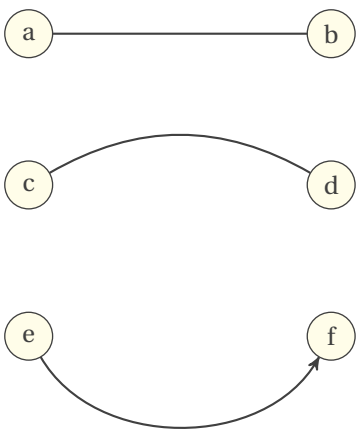
\Edge[⟨local options⟩](⟨vertex A⟩)(⟨vertex B⟩)

Options	Default	Definition
color	\EdgeColor	color of the edge
lw	\EdgeLineWidth	line width of the edge
double	{}	color of the border of a double edge
dd	\EdgeDoubleDistance	line width of the border
label	{}	label of the edge
labeltext	\LabelTextColor	color of the text of the label
labelcolor	\LabelFillColor	color of the background of the label
labelstyle	{}	TikZ options of the label
style	pos=.5	TikZ options of the edge
local	false	the style **EdgeStyle** is not used
->		arrow from A to B
<-		arrow from B to A
<->		arrows at both ends
-		no arrow
arrow	{}	any TikZ arrow specification

\Edge draws an edge between two vertices. The edge uses the style **EdgeStyle** *(defined by* **\GraphInit***) and its label the style* **LabelStyle***. The keys* **->**, **<-** *and* **<->** *use the tip given by* **\tkzSetUpGraph[arrow tip=…]** *(***stealth'** *by default).*

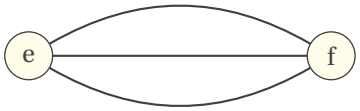
5.1.1 Straight edges and arcs

An edge is a segment or an arc; it can have a label. The style **EdgeStyle** can be redefined for all the following edges.




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{a}
  \EA(a){b}
  \SO[unit=2](a){c}
  \EA(c){d}
  {\SetGraphUnit{2}
  \SO(c){e}}
  \EA(e){f}
  \Edge(a)(b)
  \tikzset{EdgeStyle/.style = {bend left}}
  \Edge(c)(d)
  \tikzset{EdgeStyle/.style = {bend right=60}}
  \Edge[->](e)(f)
\end{tikzpicture}
```

5.1.2 The option style




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{e}
  \EA(e){f}
  \Edge(f)(e)
  \Edge[style={bend left}](f)(e)
  \Edge[style={bend right}](f)(e)
\end{tikzpicture}
```

5.1.3 Arrows




```
\begin{tikzpicture}
  \SetGraphUnit{2}
  \Vertices{line}{A,B,C,D}
  \Edge[->](A)(B)
  \Edge[<->,color=red](B)(C)
  \Edge[<-,style={bend left}](C)(D)
  \Edge[arrow={-to},style={bend right}](C)(D)
\end{tikzpicture}
```

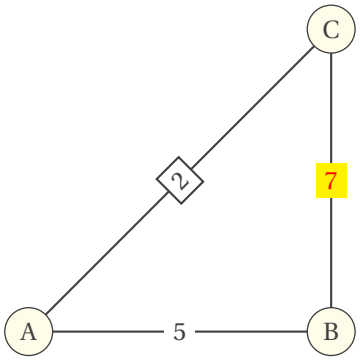
5.1.4 Double edges

`double` gives the color of the border, `lw` the width of the inside and `dd` the width of the border.




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{A} \EA(A){B}
  \Edge[double=black,color=orange,lw=2pt](A)(B)
\end{tikzpicture}
```

5.1.5 Labels

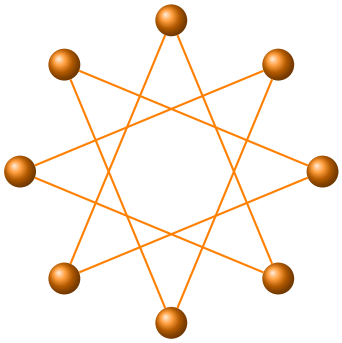



```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{A} \EA(A){B} \NO(B){C}
  \Edge[label=$5$](A)(B)
  \Edge[label=$7$,labelcolor=yellow,labeltext=red](B)(C)
  \Edge[label=$2$,labelstyle={draw,sloped}](C)(A)
\end{tikzpicture}
```

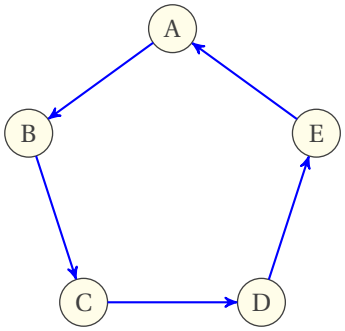
5.2 \Edges

\Edges[⟨local options⟩](⟨vertex A,vertex B,…⟩)

\Edges draws a chain of edges A–B, B–C.... The options are those of \Edge; they apply to every edge.



```
\begin{tikzpicture}
  \GraphInit[vstyle=Art]
  \Vertices[unit=2]{circle}{a0,a1,a2,a3,a4,a5,a6,a7}
  \Edges(a0,a3,a6,a1,a4,a7,a2,a5,a0)
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \Vertices[unit=2,rotation=90]{circle}{A,B,C,D,E}
  \Edges[->,color=blue](A,B,C,D,E,A)
\end{tikzpicture}
```

5.3 A loop: \Loop

\Loop[⟨local options⟩](⟨vertex⟩)

Options	Default	Definition
dir	WE	direction: WE, EA, NO, SO, NOWE, NOEA, SOWE, SOEA
dist	4cm	size of the loop
color	black	color of the loop
label	{}	label
labelstyle	{}	TikZ options of the label
style	pre,thick	TikZ options of the loop

The direction is not case sensitive. \grLoop is another name of this macro, always defined.



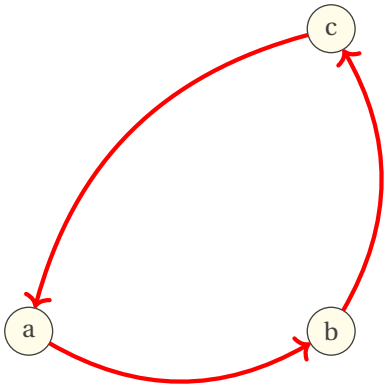

```
\begin{tikzpicture}
  \useasboundingbox (-2,-2) rectangle (8,2);
  \SetVertexSimple
  \SetGraphUnit{5}
  \Vertex{A}
  \EA(A){B}
  \Edge[->](A)(B)
  \Loop[dist=3cm,dir=EA,style={thick,->}](B)
  \Loop[dist=5cm,dir=WE,style={thick,->},label=$0.5$,labelstyle=left](A)
\end{tikzpicture}
```

5.4 Default values for the edges: \SetUpEdge

\SetUpEdge[⟨local options⟩]

Options	Default	Definition
color	\EdgeColor	color of the edges
lw	\EdgeLineWidth	line width
label	{}	label
labeltext	\LabelTextColor	color of the text of the labels
labelcolor	\LabelFillColor	background of the labels
labelstyle	{}	TikZ options of the labels
style	{}	TikZ options of the edges

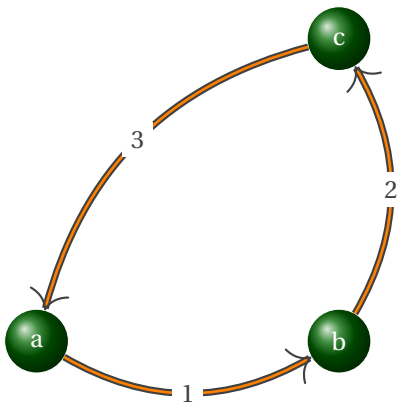
The given values become the default values of the next edges, until the end of the group. **\SetUpEdge** *also empties the styles* **EdgeStyle** *and* **LabelStyle**. **\DisableEdgeStyle** *only empties* **EdgeStyle**.




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{a}
  \EA(a){b}
  \NO(b){c}
  \SetUpEdge[style={->,bend right,ultra thick},
             color=red]
  \Edge(a)(b)
  \Edge(b)(c)
  \Edge(c)(a)
\end{tikzpicture}
```

5.5 The styles EdgeStyle and LabelStyle

These two TikZ styles can be redefined with **\tikzset**.




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \tikzset{VertexStyle/.style = {draw,
     shape = circle, shading = ball,
     ball color = green!40!black,
     minimum size = 24pt, color = white}}
  \tikzset{EdgeStyle/.style = {->,bend right,
     thick, double = orange,
     double distance = 1pt}}
  \Vertex{a}
  \EA(a){b}
  \NO(b){c}
  \tikzset{LabelStyle/.style = {fill=white}}
  \Edge[label=$1$](a)(b)
  \Edge[label=$2$](b)(c)
  \Edge[label=$3$](c)(a)
\end{tikzpicture}
```

6 Styles

The look of a graph depends on three Ti*k*Z styles: `VertexStyle`, `EdgeStyle` and `LabelStyle` (the labels of the edges). There are several ways to change them; the last definition wins.

1. `\GraphInit` chooses a predefined style (`vstyle`); `\tkzDefGraphStyle` defines a new one.
2. `\tkzSetUpGraph` changes the values used by the styles: colors, sizes, line widths…
3. The macros `\SetVertexSimple`, `\SetVertexNormal`, `\SetVertexShade`, `\SetVertexArt`, `\SetUpVertex` and `\SetUpEdge` refine a style.
4. The styles can be redefined with `\tikzset`: `VertexStyle/.style={…}` defines a new style, `VertexStyle/.append style={…}` completes the current one.

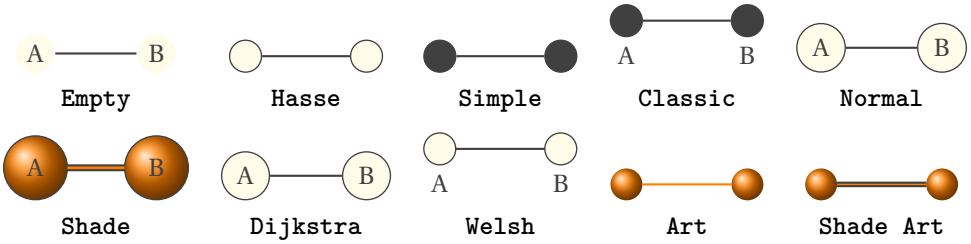
6.1 \GraphInit

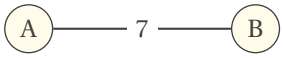
`\GraphInit[⟨local options⟩]`

Options	Default	Definition
vstyle	Normal	predefined style
unit	\GraphUnit	unit of the graph

The predefined styles are **Empty**, **Hasse**, **Simple**, **Classic**, **Normal**, **Shade**, **Dijkstra**, **Welsh**, **Art** *and* **Shade Art**. *The name of the style is not case sensitive. A style also sets the behavior of the labels (inside, outside or no label).*

6.1.1 The predefined styles



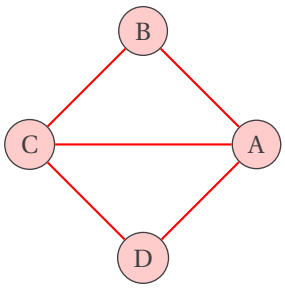



```
\begin{tikzpicture}
  \GraphInit[vstyle=Dijkstra]
  \SetGraphUnit{3}
  \Vertex{A}\EA(A){B}
  \Edge[label=$7$](A)(B)
\end{tikzpicture}
```

6.2 A new style: \tkzDefGraphStyle

`\tkzDefGraphStyle{⟨name⟩}{⟨code⟩}`

Defines (or redefines) the style **name** *for* `\GraphInit[vstyle=name]`. *The code is executed each time the style is chosen; it can use the macros of this section or* `\tikzset`. *The name is not case sensitive.*




```
\tkzDefGraphStyle{Red}{%
  \GraphInit[vstyle=Normal]
  \SetGraphColor{red!20}{red}}
\begin{tikzpicture}
  \GraphInit[vstyle=red]
  \Vertices[unit=1.5]{circle}{A,B,C,D}
  \Edges(A,B,C,D,A,C)
\end{tikzpicture}
```

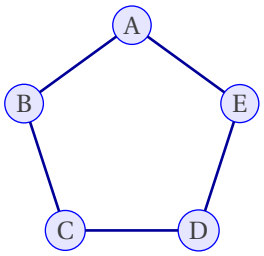
6.3 Settings: \tkzSetUpGraph

The predefined styles use macros like \VertexShape or \EdgeLineWidth. \tkzSetUpGraph changes them with keys.

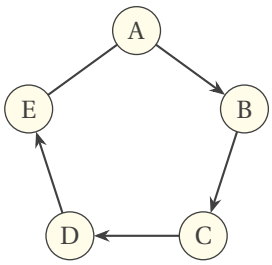
\tkzSetUpGraph[⟨options⟩]

Options	Default	Macro changed
vertex shape	circle	\VertexShape
vertex line width	0.5pt	\VertexLineWidth
vertex line color	black	\VertexLineColor
vertex text color	black	\VertexTextColor
vertex fill	black	\VertexFillColor
light fill	white	\VertexLightFillColor
dark fill	black	\VertexDarkFillColor
ball color	orange	\VertexBallColor
inner sep	2pt	\VertexInnerSep
outer sep	0pt	\VertexOuterSep
small size	12pt	\VertexSmallMinSize
medium size	18pt	\VertexInterMinSize
big size	24pt	\VertexBigMinSize
edge color	black	\EdgeColor
edge width	0.8pt	\EdgeLineWidth
edge fill	orange	\EdgeFillColor
art color	orange	\EdgeArtColor
double distance	1pt	\EdgeDoubleDistance
label text	black	\LabelTextColor
label fill	white	\LabelFillColor
unit	1	\GraphUnit
directed		an arrow on every edge
arrow tip	stealth'	tip of the arrows

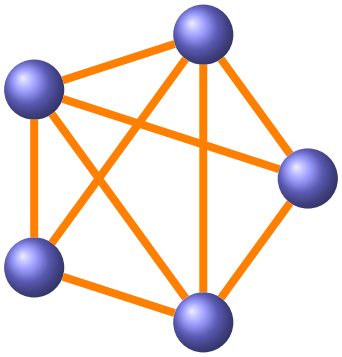
The changes are local: inside a **tikzpicture** *or a group they stop at its end; in the preamble they apply to the whole document.* **directed=false** *cancels* **directed**, *and the key* **-** *of* **\Edge** *removes the arrow of one edge. The tips* **Stealth**, **Latex**… *need the library* **arrows.meta**, *which is not loaded by* **tkz-graph**.




```
\begin{tikzpicture}
  \tkzSetUpGraph[medium size=14pt,
     vertex line color=blue,
     light fill=blue!10,
     edge color=blue!60!black,
     edge width=1pt]
  \Vertices[unit=1.5,rotation=90]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A)
\end{tikzpicture}
```




```
% \usetikzlibrary{arrows.meta}
\begin{tikzpicture}
  \tkzSetUpGraph[directed,
    arrow tip={Stealth[length=6pt]}]
  \Vertices[unit=1.5,rotation=90,
            clockwise]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E)
  \Edge[-](E)(A)
\end{tikzpicture}
```



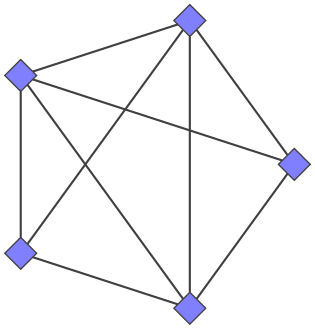
```
\begin{tikzpicture}
  \GraphInit[vstyle=Art]
  \tkzSetUpGraph[inner sep=8pt,
     edge width=3pt,ball color=blue!50]
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A,C,E,B,D)
\end{tikzpicture}
```

6.4 Refining a style

`\SetVertexSimple[⟨local options⟩]`

Options	Default	Definition
Shape	\VertexShape	shape of the node
MinSize	\VertexSmallMinSize	minimum size
LineWidth	\VertexLineWidth	line width
LineColor	\VertexLineColor	color of the border
FillColor	\VertexFillColor	color of the inside
InnerSep	\VertexInnerSep	inner sep
OuterSep	\VertexOuterSep	outer sep

Small filled vertices without label.

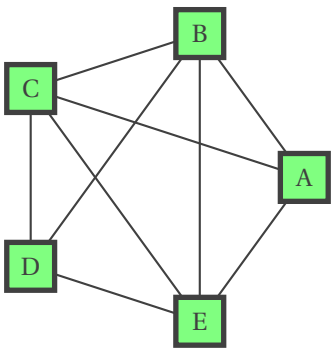


```
\begin{tikzpicture}
  \SetVertexSimple[Shape=diamond,
                   FillColor=blue!50]
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A,C,E,B,D)
\end{tikzpicture}
```

`\SetVertexNormal[⟨local options⟩]`

Options	Default	Definition
Shape	\VertexShape	shape of the node
MinSize	\VertexInterMinSize	minimum size
LineWidth	\VertexLineWidth	line width
LineColor	\VertexLineColor	color of the border
FillColor	\VertexLightFillColor	color of the inside
TextColor	\VertexTextColor	color of the label
InnerSep	\VertexInnerSep	inner sep
OuterSep	\VertexOuterSep	outer sep

The labels are inside the vertices.



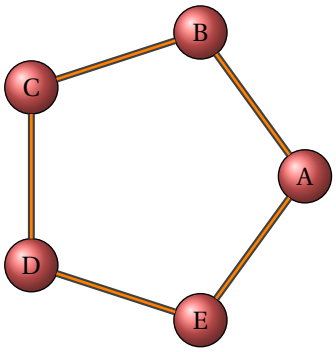

```
\begin{tikzpicture}
  \SetVertexNormal[Shape=rectangle,
                   LineWidth=2pt,
                   FillColor=green!50]
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A,C,E,B,D)
\end{tikzpicture}
```

`\SetVertexShade[⟨local options⟩]`

Options **Shape**, **MinSize** (**\VertexBigMinSize**), **LineWidth**, **LineColor**, **BallColor** (**\VertexBallColor**), **InnerSep**, **OuterSep**. *Shaded vertices with their label; double edges.*

`\SetVertexArt[⟨local options⟩]`

Same options as **\SetVertexShade**, *with* **\VertexSmallMinSize** *for* **MinSize**. *Small shaded vertices without label.*

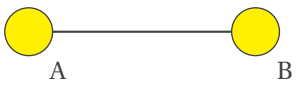



```
\begin{tikzpicture}
  \SetVertexShade[BallColor=red!60,
                  MinSize=20pt]
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A)
\end{tikzpicture}
```

`\SetUpVertex[⟨local options⟩]`

Options	Default	Definition
Lpos	0	angle of an outside label
Ldist	0cm	distance of an outside label
Style	{}	TikZ options added to the vertices
NoLabel	false	no label
LabelOut	false	outside labels
Math	false	labels in math mode
Unit	\GraphUnit	unit of the shortcuts
MinSize	\VertexInterMinSize	minimum size
LineWidth	\VertexLineWidth	line width
LineColor	\VertexLineColor	color of the border
FillColor	\VertexLightFillColor	color of the inside
TextColor	\VertexTextColor	color of the label
InnerSep	\VertexInnerSep	inner sep
OuterSep	\VertexOuterSep	outer sep

Changes the current style of the vertices and the default values of **\Vertex**.




```
\begin{tikzpicture}
  \SetGraphUnit{3}
  \SetUpVertex[Lpos=-60,LabelOut,
               FillColor=yellow]
  \Vertex{A}\EA(A){B}\Edge(A)(B)
\end{tikzpicture}
```

6.5 Colors of the predefined styles

`\SetGraphShadeColor{⟨ball color⟩}{⟨edge color⟩}{⟨double color⟩}`

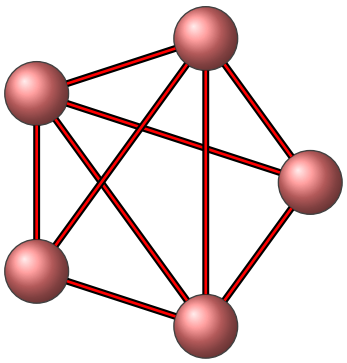
Colors of the style **Shade**.

`\SetGraphArtColor{⟨ball color⟩}{⟨edge color⟩}`

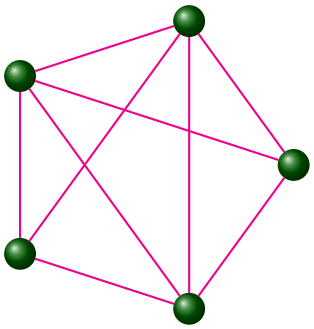
Colors of the style **Art**.

`\SetGraphColor{⟨fill color⟩}{⟨edge color⟩}`

Colors of the style **Normal**.



```
\begin{tikzpicture}
  \GraphInit[vstyle=Shade]
  \SetVertexNoLabel
  \SetGraphShadeColor{red!50}{black}{red}
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A,C,E,B,D)
\end{tikzpicture}
```

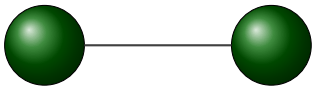


```
\begin{tikzpicture}
  \GraphInit[vstyle=Art]
  \SetGraphArtColor{green!40!black}{magenta}
  \Vertices[unit=2]{circle}{A,B,C,D,E}
  \Edges(A,B,C,D,E,A,C,E,B,D)
\end{tikzpicture}
```

6.6 Other macros

- `\SetVertexLabeledSmall`, `\SetVertexNotLabeledSmall`: small vertices with or without label and double edges;
- `\SetVertexForPresentation`{⟨ball⟩}{⟨edge⟩}{⟨double⟩}: style **Shade** without label, with these colors;
- `\tkzSetUpColors[background=…,text=…]`: the default colors of the background and of the text, used by the styles (this macro also changes the color of the page).

6.7 Redefining VertexStyle with \tikzset



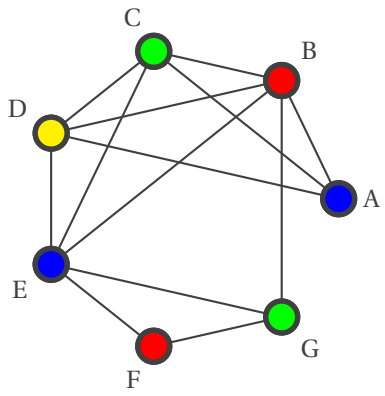
```
\begin{tikzpicture}
  \tikzset{VertexStyle/.style = {
      shape = circle, shading = ball,
      ball color = green!40!black,
      minimum size = 30pt, draw}}
  \SetVertexNoLabel
  \Vertex{A}\EA[unit=3](A){B}
  \Edge(A)(B)
\end{tikzpicture}
```

7 Coloring and highlighting

7.1 Coloring vertices: \AddVertexColor

\AddVertexColor{⟨color⟩}{⟨list of vertices⟩}

The vertices of the list, already drawn, are drawn again with this color as fill color (and without label). The style **Welsh** *is designed for this use.*




```
\begin{tikzpicture}
  \GraphInit[vstyle=Welsh]
  \tkzSetUpGraph[vertex line width=2pt]
  \Vertices[unit=2]{circle}{A,B,C,D,E,F,G}
  \SetVertexNoLabel
  \AddVertexColor{red}{B,F}
  \AddVertexColor{blue}{E,A}
  \AddVertexColor{green}{C,G}
  \AddVertexColor{yellow}{D}
  \Edges(G,E,F,G,B,D,E,C,D,A,C,B,A)
  \Edges(B,E)
\end{tikzpicture}
```

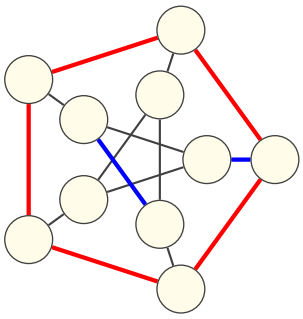
A complete example (the Welsh–Powell algorithm) is given in the section 10.1.

7.2 Highlighting a path: \tkzHighlightPath

\tkzHighlightPath[⟨local options⟩]{⟨list⟩}

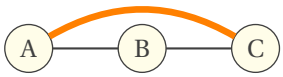
Options	Default	Definition
color	red	color of the path
lw	1.6pt	line width
style	{}	TikZ options (to follow a bent edge…)
cycle	false	the last vertex is joined to the first one

The list is a path **A,B,C,D** *(the edges A–B, B–C, C–D are drawn again) or a list of edges* **A/B,C/D**. *The vertices must exist; the edges are drawn over the graph, so the macro is used at the end of the picture. With* **tkz-berge**, *it highlights the edges of a graph drawn by a single macro.*



```
\begin{tikzpicture}[scale=.6]
  \GraphInit[vstyle=Normal]
  \SetVertexNoLabel
  \grPetersen[RA=3,RB=1.5]
  \tkzHighlightPath[cycle]{a0,a1,a2,a3,a4}
  \tkzHighlightPath[color=blue]{a0/b0,b2/b4}
\end{tikzpicture}
```

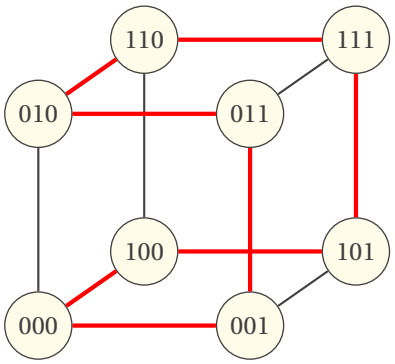
A bent edge needs the same **style**:




```
\begin{tikzpicture}
  \Vertices[unit=1.5]{line}{A,B,C}
  \Edge[style={bend left=30}](A)(C)
  \Edges(A,B,C)
  \tkzHighlightPath[color=orange,lw=2.5pt,
     style={bend left=30}]{A,C}
\end{tikzpicture}
```

7.2.1 A Hamiltonian cycle of the cube: the Gray code

With **tkz-berge**, \grHypercube names the vertices of the cube by their binary word; a Hamiltonian cycle gives the Gray code.




```
\begin{tikzpicture}[scale=.8]
  \GraphInit[vstyle=Normal]
  \tkzSetUpGraph[medium size=20pt]
  \grHypercube[RA=3.5,prefix={}]{3}
  \tkzHighlightPath[cycle]%
    {000,001,011,010,110,111,101,100}
\end{tikzpicture}
```

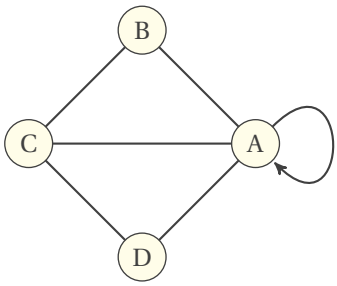
8 Graphs and matrices

8.1 The adjacency matrix of a graph: \tkzAdjacencyMatrix

`\tkzAdjacencyMatrix[⟨local options⟩]`

Options	Default	Definition
directed	false	an edge A--B counts only for the entry (A,B)
order	{}	order of the vertices: order={A,B,C}

Writes the adjacency matrix of the graph drawn in the last **tikzpicture**. *Each picture records its vertices (in order of creation) and its edges. The entry (i, j) is the number of edges between the i-th and the j-th vertex; a loop counts* 1. *The macro can be used in text mode or in math mode.*



$$M = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

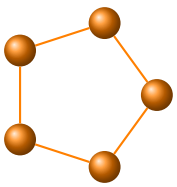
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

```
\begin{tikzpicture}
  \Vertices[unit=1.5]{circle}{A,B,C,D}
  \Edges(A,B,C,D,A)
  \Edge(A)(C)
  \Loop[dir=EA,dist=1.5cm](A)
\end{tikzpicture}

$M = \tkzAdjacencyMatrix$

\tkzAdjacencyMatrix[order={D,C,B,A}]
```

With **tkz-berge**:



$$C_5 : \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{pmatrix}$$

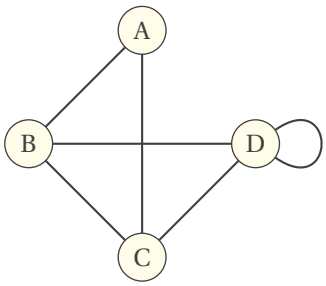
```
\begin{tikzpicture}[scale=.5]
  \GraphInit[vstyle=Art]
  \grCycle[RA=2]{5}
\end{tikzpicture}
\quad $C_5\colon \tkzAdjacencyMatrix$
```

8.2 A graph from a matrix: \tkzGraphFromMatrix

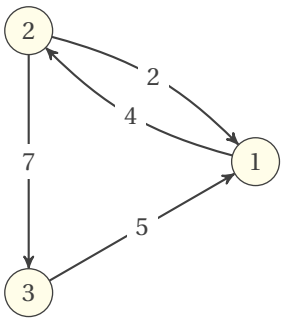
`\tkzGraphFromMatrix[⟨local options⟩]{⟨list of vertices⟩}{⟨matrix⟩}`

Options	Default	Definition
placed	false	the vertices already exist
unit	\GraphUnit	radius of the circle
rotation	0	angle of the first vertex
clockwise	false	the vertices turn clockwise
directed	false	arrows; i→j and j→i are bent
weighted	false	the entries are written on the edges
style	{}	TikZ options added to each edge
bend	15	angle of the bent edges
loop distance	1cm	size of the loops

The rows of the matrix are separated by `;` *and the entries by* `,` *(use a decimal point). A non-zero entry* (i, j) *gives an edge between the* i*-th and the* j*-th vertex, a non-zero entry on the diagonal a loop, directed away from the center of the graph. Without* `placed`*, the vertices are placed on a circle, like* `\Vertices{circle}`*; the options of the vertices (*`\SetVertexMath`*,* `\GraphInit`*…) apply. Without* `directed`*, only the upper triangle of the matrix is read.*



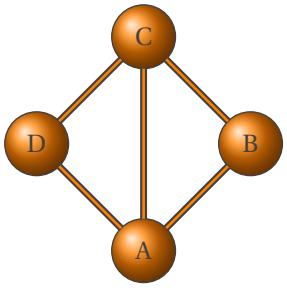

```
\begin{tikzpicture}
  \tkzGraphFromMatrix[unit=1.5,rotation=90]%
    {A,B,C,D}{0,1,1,0;
              1,0,1,1;
              1,1,0,1;
              0,1,1,1}
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \tkzGraphFromMatrix[directed,weighted,
                      unit=2]%
    {1,2,3}{0,4,0; 2,0,7; 5,0,0}
\end{tikzpicture}
```

8.2.1 Placed vertices

With `placed`, the vertices are drawn first, anywhere.




```
\begin{tikzpicture}
  \GraphInit[vstyle=Shade]
  \Vertices[unit=2]{square}{A,B,C,D}
  \tkzGraphFromMatrix[placed]%
    {A,B,C,D}{0,1,1,1;
              1,0,1,0;
              1,1,0,1;
              1,0,1,0}
\end{tikzpicture}
```

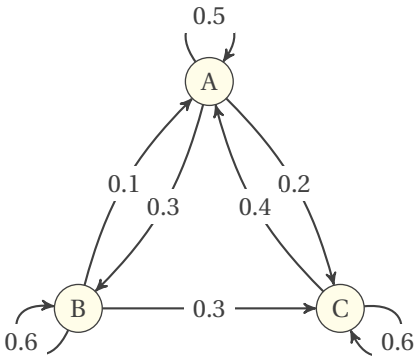
9 Probabilistic graphs

A probabilistic graph (the graph of a Markov chain) is a weighted directed graph: the weight of the edge $i \to j$ is the probability of going from the state i to the state j.

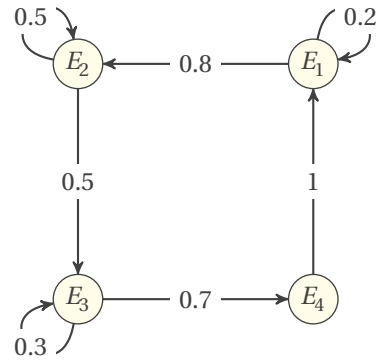
9.1 Any number of states: \grProbMatrix

`\grProbMatrix[⟨local options⟩]{⟨list of states⟩}{⟨transition matrix⟩}`

The graph of the transition matrix: **`\tkzGraphFromMatrix`** *with the options* **`directed`** *and* **`weighted`**. *The other options are those of* **`\tkzGraphFromMatrix`**. *The pairs $i \to j$, $j \to i$ are bent and the loops are directed away from the center.*




```
\begin{tikzpicture}
  \grProbMatrix[unit=2,rotation=90]{A,B,C}%
    {0.5,0.3,0.2;
     0.1,0.6,0.3;
     0.4,0,0.6}
\end{tikzpicture}
```




```
\begin{tikzpicture}
  \SetVertexMath
  \grProbMatrix[unit=2.2,rotation=45]%
    {E_1,E_2,E_3,E_4}%
    {0.2,0.8,0,0;
     0,0.5,0.5,0;
     0,0,0.3,0.7;
     1,0,0,0}
\end{tikzpicture}
```

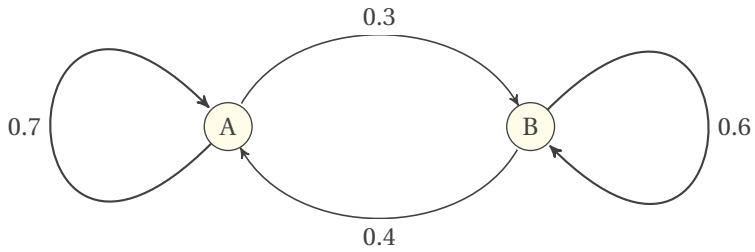
9.2 Two states: \grProb

\grProb[⟨local options⟩]{⟨left⟩}{⟨right⟩}{⟨N⟩}{⟨S⟩}{⟨W⟩}{⟨E⟩}

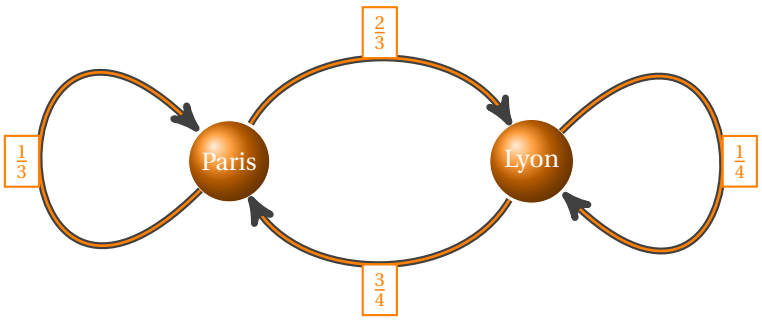
Arguments	Definition
left	name of the vertex on the left
right	name of the vertex on the right
N	label of the top edge (left to right)
S	label of the bottom edge (right to left)
W	label of the loop on the left
E	label of the loop on the right

Options	Default	Definition
unit	4	distance between the vertices
LposA	180	angle of an outside label of the left vertex
LposB	0	angle of an outside label of the right vertex
Ldist	0cm	distance between a node and its label
LoopDist	4cm	size of the loops

The labels of the edges are written in math mode. The vertices are named **grA** *and* **grB**.




```
\begin{tikzpicture}
  \useasboundingbox (-2.5,-2) rectangle (7.5,2);
  \grProb{A}{B}{0.3}{0.4}{0.7}{0.6}
\end{tikzpicture}
```

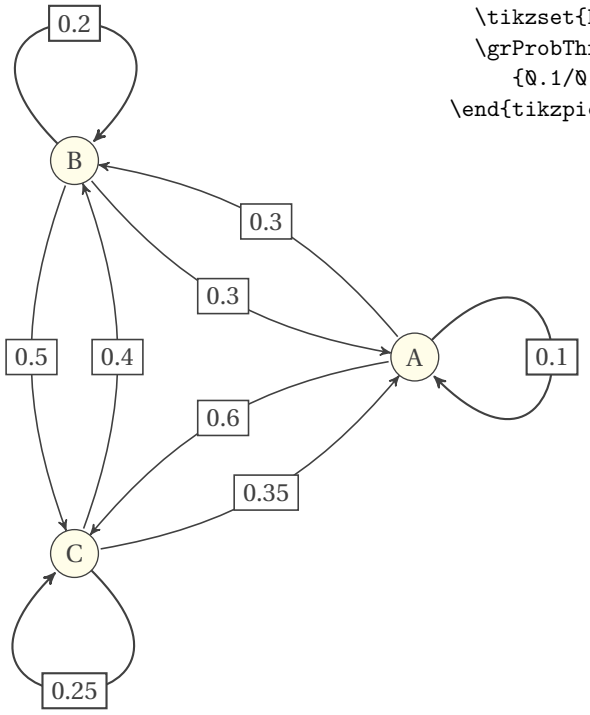


```
\begin{tikzpicture}
  \useasboundingbox (-2.5,-2.5) rectangle (7.5,2.5);
  \tikzset{VertexStyle/.style = {shape = circle,
      shading = ball, ball color = orange,
      minimum size = 20pt, draw, color = white}}
  \tikzset{LabelStyle/.style = {draw,color=orange,fill=white}}
  \tikzset{EdgeStyle/.style = {->, thick,
      double = orange, double distance = 1pt}}
  \grProb[Ldist=0.1cm,LposA=0,LposB=180]{Paris}{Lyon}%
     {\frac{2}{3}}{\frac{3}{4}}{\frac{1}{3}}{\frac{1}{4}}
\end{tikzpicture}
```

9.3 Three states: \grProbThree

\grProbThree[⟨local options⟩]{⟨A⟩}{⟨B⟩}{⟨C⟩}{⟨aa/ab/ac⟩}{⟨bb/ba/bc⟩}{⟨cc/ca/cb⟩}

The three states are placed on a circle and named ***grA***, ***grB***, ***grC***; *the arguments 4 to 6 give the labels of the loop and of the two edges leaving each state. Same options as* **\grProb**. **\grProbMatrix** *is now simpler for this case.*



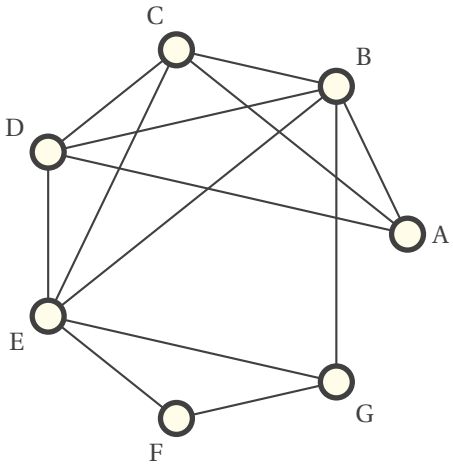

```
\begin{tikzpicture}[scale=.75]
  \tikzset{LabelStyle/.style = {draw,fill=white}}
  \grProbThree[unit=4]{A}{B}{C}%
     {0.1/0.3/0.6}{0.2/0.3/0.5}{0.25/0.35/0.4}
\end{tikzpicture}
```

10 Examples

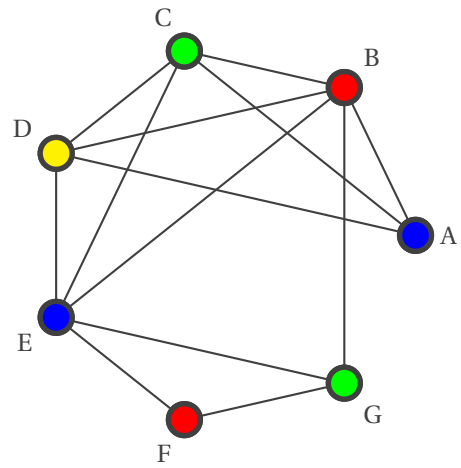
More examples, with a street map, Dijkstra's algorithm and colorings, are given in the next section (exercises of the French baccalaureate).

10.1 Coloring with the Welsh--Powell algorithm

An airline offers direct flights between the cities A, B, C, D, E, F and G. In the graph below, the vertices are the cities and the edges are the flights. Each airport receives a color, and two airports linked by a direct flight must have different colors.
The vertices are sorted by decreasing degree; the first color goes to the first vertex and to every following vertex not adjacent to a vertex of this color, and so on.



Vertex	*Degree*	*Color*
B	5	red
E	5	blue
C	4	green
D	4	yellow
A	3	blue
G	3	green
F	2	red

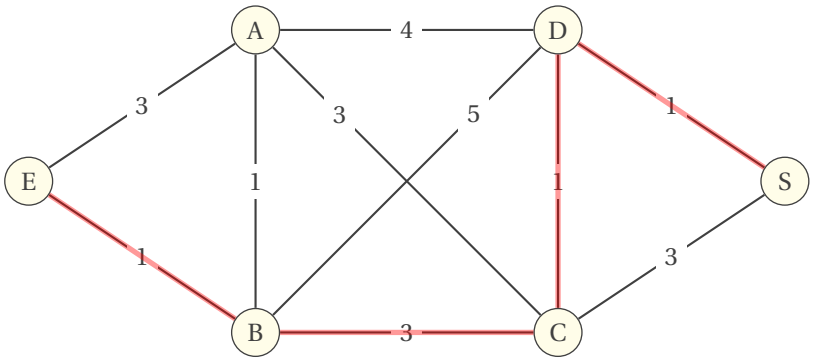



```
\begin{tikzpicture}
  \GraphInit[vstyle=Welsh]
  \tkzSetUpGraph[vertex line width=2pt]
  \Vertices[unit=2.5]{circle}{A,B,C,D,E,F,G}
  \SetVertexNoLabel
  \AddVertexColor{red}{B,F}
  \AddVertexColor{blue}{E,A}
  \AddVertexColor{green}{C,G}
  \AddVertexColor{yellow}{D}
  \Edges(G,E,F,G,B,D,E,C,D,A,C,B,A)
  \Edges(B,E)
\end{tikzpicture}
```

The graph contains the complete subgraph $\{B,C,D,E\}$ of order 4: four colors are necessary and the chromatic number is 4.

10.2 Dijkstra's algorithm

The shortest path from the vertex E to the vertex S.



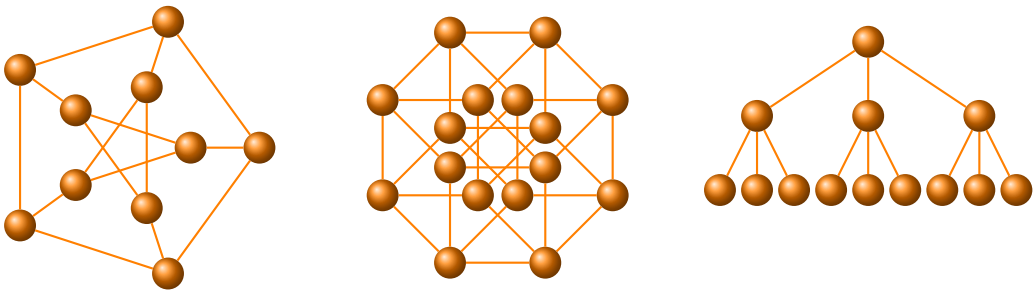

```
\begin{tikzpicture}
  \GraphInit[vstyle=Dijkstra]
  \Vertex{E}
  \Vertex[x=3,y=2]{A}  \Vertex[x=3,y=-2]{B}
  \Vertex[x=7,y=-2]{C} \Vertex[x=7,y=2]{D}
  \Vertex[x=10,y=0]{S}
  \Edge[label=$3$](E)(A)
  \Edge[label=$1$](E)(B)
  \Edge[label=$1$](A)(B)
  \Edge[label=$3$](B)(C)
  \Edge[label=$3$,style={pos=.25}](A)(C)
  \Edge[label=$5$,style={pos=.75}](B)(D)
  \Edge[label=$4$](A)(D)
  \Edge[label=$1$](S)(D)
  \Edge[label=$3$](C)(S)
  \Edge[label=$1$](C)(D)
  \tkzHighlightPath[color=red,lw=2pt,style={opacity=.4}]{E,B,C,D,S}
\end{tikzpicture}
```

E	A	B	C	D	S	Choice
0	∞	∞	∞	∞	∞	E
	$3(E)$	$1(E)$	∞	∞	∞	B
	$2(B)$		$4(B)$	$6(B)$	∞	A
			$4(B)$	$6(B)$	∞	C
				$5(C)$	$7(C)$	D
					$6(D)$	S

The shortest path is $EBCDS$, of length 6.

10.3 Classic graphs with tkz-berge

tkz-berge draws the classic graphs of graph theory with **tkz-graph**; all the macros and styles of this documentation apply.



```
\begin{tikzpicture}[scale=.7]
  \GraphInit[vstyle=Art]
  \grPetersen[RA=2.5,RB=1.2]
  \begin{scope}[xshift=7cm]
    \grHypercube[RA=1.8]{4}
  \end{scope}
  \begin{scope}[xshift=14cm,yshift=2cm]
    \grTree[RA=.7,RB=1.4]{3}{2}
  \end{scope}
\end{tikzpicture}
```

11 Computations with LuaLaTeX: the option lua

11.1 An optional part of the package

With the option **lua**, **tkz-graph** computes on the graph drawn in a **tikzpicture**: order, size, degrees, shortest path (Dijkstra's algorithm), coloring (Welsh–Powell algorithm), connectivity, Eulerian paths and powers of the adjacency matrix. The computations are written in Lua (file **tkz-graph.lua**), so this option needs **lualatex**.

```
\usepackage[lua]{tkz-graph}  % or \usepackage[lua]{tkz-berge}
```

Without the option, **tkz-graph** and **tkz-berge** keep working with **pdflatex** and **xelatex**. With **pdflatex**, the option gives the error "The options lua and 3d need LuaLaTeX" and is ignored.

11.1.1 The recorded graph

Each **tikzpicture** starts a new graph. The vertices and the edges created with **\Vertex**, **\Vertices**, **\Edge**, **\Edges**, **\Loop**, **\tkzGraphFromMatrix** and the macros of **tkz-berge** are recorded. The commands of this section can be used in the picture, after the edges, or after the picture, until the next **tikzpicture**.

- The *weight* of an edge is its label when the label is a number: **3**, **3**, **0.5**, **$0{,}5$** or **$\frac{1}{2}$**. An edge without numeric label has the weight 1. With a decimal comma in the labels, the results are written with a decimal comma.
- An edge is *oriented* when it has an arrow: the keys **->**, **<-**, **<->**, **arrow**, or an arrow in its option **style**. The other edges can be used in both directions, unless the option **directed** is given.
- A loop counts 2 in the degree of its vertex.

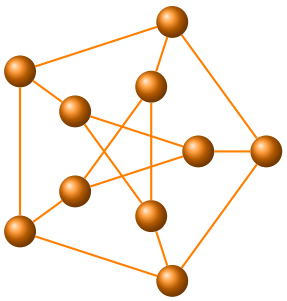
11.2 Order, size and degrees

\tkzGraphDegree{⟨vertex⟩}

Commands	Result
\tkzGraphOrder	number of vertices
\tkzGraphSize	number of edges (loops included)
\tkzGraphDegree{⟨vertex⟩}	degree of the vertex
\tkzGraphInDegree{⟨vertex⟩}	edges that arrive at the vertex
\tkzGraphOutDegree{⟨vertex⟩}	edges that leave the vertex
\tkzGraphDegreeSequence[⟨sep⟩]	degrees in decreasing order
\tkzGraphDegreeTable[⟨options⟩]	table of the degrees

Options of the tables	Default	Definition
math	false	names of the vertices in math mode
sort	true	vertices sorted by decreasing degree

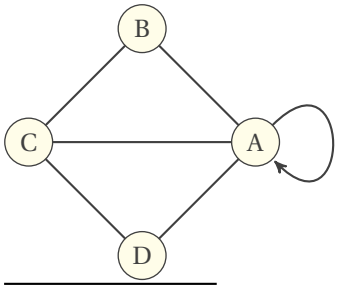
The first five commands are expandable. For the in- and out-degree, an edge without arrow counts in both directions. The separator of the sequence is **{, }** *by default.*



```
\begin{tikzpicture}[scale=.6]
  \GraphInit[vstyle=Art]
  \grPetersen[RA=3,RB=1.5]
\end{tikzpicture}
\par
The Petersen graph has \tkzGraphOrder\
  vertices and \tkzGraphSize\ edges.
  Degrees: \tkzGraphDegreeSequence.
```

The Petersen graph has 10 vertices and 15 edges.
Degrees: 3, 3, 3, 3, 3, 3, 3, 3, 3, 3.

The sum of the degrees is twice the number of edges.



Vertex	Degree
A	5
C	3
B	2
D	2

$\deg A = 5$, size = 6.

```
\begin{tikzpicture}
  \Vertices[unit=1.5]{circle}{A,B,C,D}
  \Edges(A,B,C,D,A)
  \Edge(A)(C)
  \Loop[dir=EA,dist=1.5cm](A)
\end{tikzpicture}
\par
\tkzGraphDegreeTable
\par\medskip
$\deg A = \tkzGraphDegree{A}$,
  size $= \tkzGraphSize$.
```

11.3 Shortest path: \tkzGraphDijkstra

`\tkzGraphDijkstra[⟨local options⟩](⟨start⟩)(⟨end⟩)`

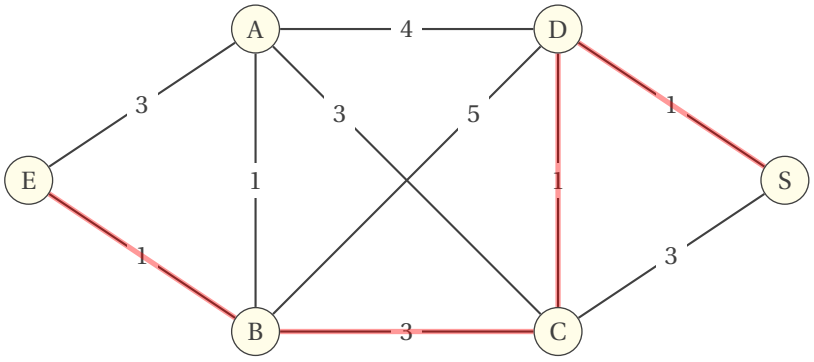
Options	Default	Definition
color	red	color of the path
lw	1.6pt	line width of the path
style	{}	TikZ options of the path
draw	true	the path is drawn over the edges
directed	false	an edge without arrow goes from its first vertex to the second

Dijkstra's algorithm. In the picture, the shortest path is drawn again over its edges; each edge keeps its own option **style** *(for example a bend). The weights must be positive.*

Results	
\tkzGraphPath	the vertices of the path: E,B,C,D,S
\tkzGraphPathLength	the length of the path (∞ without path)
\tkzGraphDijkstraTable[⟨options⟩]	the table of the steps

The options of the table are those of **\tkzGraphDegreeTable**. *In each row, a vertex is marked by* **\tkzGraphDoneMark** *when its distance is final; the last column gives the vertex chosen at this step.*

This is the example of the section "Dijkstra's algorithm", without the table written by hand.



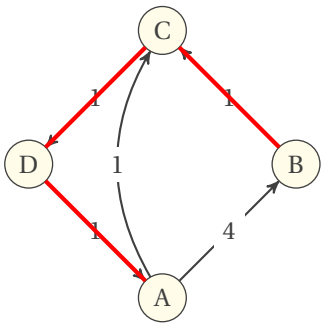
The shortest path is E,B,C,D,S, of length 6.

E	*A*	*B*	*C*	*D*	*S*	Choice
0	∞	∞	∞	∞	∞	*E*
■	3 (*E*)	1 (*E*)	∞	∞	∞	*B*
■	2 (*B*)	■	4 (*B*)	6 (*B*)	∞	*A*
■	■	■	4 (*B*)	6 (*B*)	∞	*C*
■	■	■	■	5 (*C*)	7 (*C*)	*D*
■	■	■	■	■	6 (*D*)	*S*

```
\begin{tikzpicture}
  \GraphInit[vstyle=Dijkstra]
  \Vertex{E}
  \Vertex[x=3,y=2]{A}  \Vertex[x=3,y=-2]{B}
  \Vertex[x=7,y=-2]{C} \Vertex[x=7,y=2]{D}
  \Vertex[x=10,y=0]{S}
  \Edge[label=$3$](E)(A)
  \Edge[label=$1$](E)(B)
  \Edge[label=$1$](A)(B)
  \Edge[label=$3$](B)(C)
  \Edge[label=$3$,style={pos=.25}](A)(C)
  \Edge[label=$5$,style={pos=.75}](B)(D)
  \Edge[label=$4$](A)(D)
  \Edge[label=$1$](S)(D)
  \Edge[label=$3$](C)(S)
  \Edge[label=$1$](C)(D)
  \tkzGraphDijkstra[style={opacity=.4},lw=2pt](E)(S)
\end{tikzpicture}
\par\medskip
The shortest path is \tkzGraphPath, of length $\tkzGraphPathLength$.
\par\medskip
\tkzGraphDijkstraTable[math]
```

11.3.1 A directed graph

All the edges have an arrow: from B, the shortest way to A goes around the square. With the option **directed**, the edges without arrow would go from their first vertex to the second.

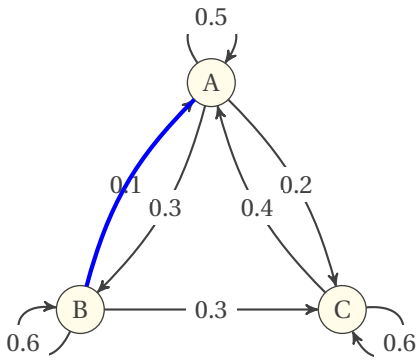


From B to A: B,C,D,A, length 3.

```
\begin{tikzpicture}
  \SetGraphUnit{2.5}
  \Vertices{square}{A,B,C,D}
  \Edge[->,label=$4$](A)(B)
  \Edge[->,label=$1$](B)(C)
  \Edge[->,label=$1$](C)(D)
  \Edge[->,label=$1$](D)(A)
  \Edge[->,label=$1$,style={bend left}](A)(C)
  \tkzGraphDijkstra(B)(A)
\end{tikzpicture}
\par
From B to A: \tkzGraphPath,
  length $\tkzGraphPathLength$.
```

11.3.2 A probabilistic graph

The weights are read in the labels written by **\grProbMatrix**; the path follows the bent edges.



B,A, 0.1.

```
\begin{tikzpicture}
  \grProbMatrix[unit=2,rotation=90]{A,B,C}%
    {0.5,0.3,0.2;
     0.1,0.6,0.3;
     0.4,0,0.6}
  \tkzGraphDijkstra[color=blue](B)(A)
\end{tikzpicture}
\par
\tkzGraphPath, $\tkzGraphPathLength$.
```

11.4 Coloring: \tkzGraphColoring

\tkzGraphColoring[⟨local options⟩]

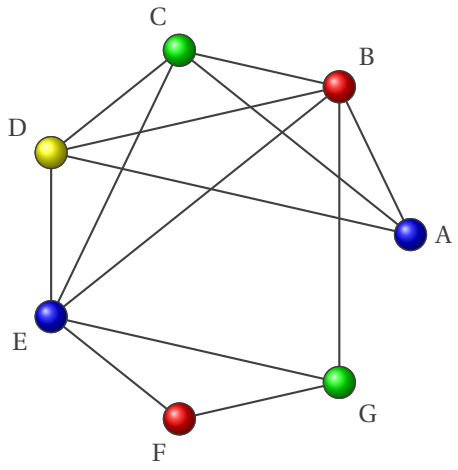
Options	Default	Definition
colors	red,blue,…	list of the colors
draw	true	the vertices are drawn again with their color

Welsh–Powell algorithm: the vertices are sorted by decreasing degree (order of creation for equal degrees); the first color goes to the first vertex and to each following vertex not adjacent to a vertex of this color, and so on. The number of colors is an upper bound of the chromatic number, not always the chromatic number. The default list is ***red, blue, green, yellow, orange, cyan, magenta, brown, lime, pink, teal, violet****.*

Results	
\tkzGraphColorCount	number of colors used
\tkzGraphVertexColor{⟨vertex⟩}	color of the vertex
\tkzGraphColoringTable[⟨options⟩]	table: vertex, degree, color

A vertex is drawn again with ***fill*** *and* ***ball color*** *set to its color, so the shaded styles are also colored. A label inside the vertex is written again (its name, or the name in math mode); a label outside the vertex is kept.*

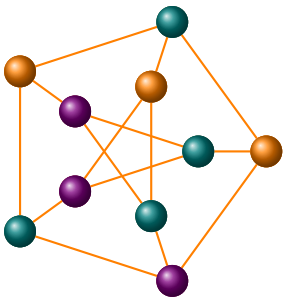
This is the example of the section "Coloring with the Welsh–Powell algorithm": the colors and the table are computed.




```
\begin{tikzpicture}
  \GraphInit[vstyle=Welsh]
  \Vertices[unit=2.5]{circle}{A,B,C,D,E,F,G}
  \Edges(G,E,F,G,B,D,E,C,D,A,C,B,A)
  \Edges(B,E)
  \tkzGraphColoring
\end{tikzpicture}
\par
\tkzGraphColoringTable[math]
\par\medskip
\tkzGraphColorCount\ colors.
```

Vertex	Degree	Color
B	5	red
E	5	blue
C	4	green
D	4	yellow
A	3	blue
G	3	green
F	2	red

4 colors.



3 colors; the vertex a0 is orange.

```
\begin{tikzpicture}[scale=.6]
  \GraphInit[vstyle=Art]
  \grPetersen[RA=3,RB=1.5]
  \tkzGraphColoring[colors={orange,
    teal,violet}]
\end{tikzpicture}
\par
\tkzGraphColorCount\ colors; the vertex
  a0 is \tkzGraphVertexColor{a0}.
```

11.5 Connectivity and Eulerian paths

`\tkzGraphEuler[⟨local options⟩]`

Options	Default	Definition
start	{}	first vertex
numbers	false	the rank of each edge is written on it
pos	.5	position of the numbers on the edges
directed	false	an edge without arrow goes from its first vertex to the second

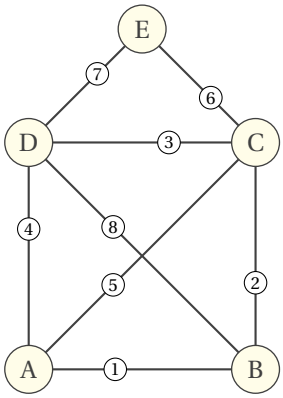
Finds an Eulerian path (each edge used once) or an Eulerian cycle (a path that ends at its first vertex), with Hierholzer's algorithm; the edges are taken in their order of creation. The path is in `\tkzGraphEulerPath`*. For a path,* `start` *must be one of the two odd vertices (by default, the first one). The graph is directed when all its edges have an arrow, or with the option* `directed`*. The numbers use the TikZ style* `EulerNumberStyle`*.*

Commands	Result
`\tkzGraphComponents`	number of connected components
`\tkzGraphIfConnected`{⟨true⟩}{⟨false⟩}	test
`\tkzGraphOddVertices`[⟨sep⟩]	vertices of odd degree
`\tkzGraphEulerType`	`cycle`, `path` or `none`
`\tkzGraphIfEulerian`{⟨cycle⟩}{⟨path⟩}{⟨none⟩}	test

These commands are expandable. The orientation of the edges is ignored for the components. For an Eulerian path, the isolated vertices are ignored and the edges must be in one component. Euler's theorem: a connected graph has an Eulerian cycle when all its degrees are even, an Eulerian path when exactly two degrees are odd.

11.5.1 The envelope

The envelope can be drawn without lifting the pencil: two vertices have an odd degree, the path goes from one to the other.

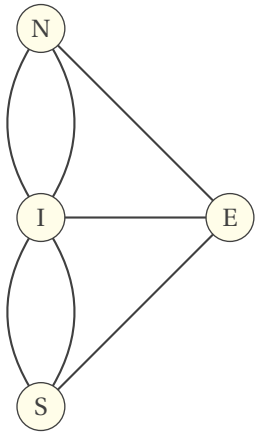


Type: path; odd vertices: A and B.

Path: A,B,C,D,A,C,E,D,B.

```
\begin{tikzpicture}
  \Vertex{A} \Vertex[x=3,y=0]{B}
  \Vertex[x=3,y=3]{C}
  \Vertex[x=0,y=3]{D}
  \Vertex[x=1.5,y=4.5]{E}
  \Edges(A,B,C,D,A,C,E,D,B)
  \tkzGraphEuler[numbers,pos=.35]
\end{tikzpicture}
\par
Type: \tkzGraphEulerType; odd vertices:
  \tkzGraphOddVertices[ and ].
\par\medskip
Path: \tkzGraphEulerPath.
```

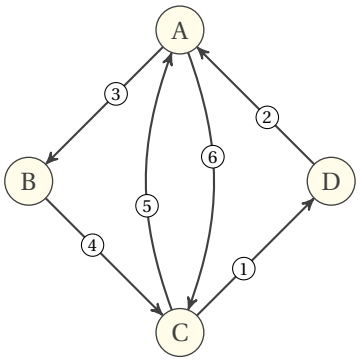
11.5.2 The bridges of Königsberg



The degrees are 5, 3, 3, 3: there is no Eulerian path.

```
\begin{tikzpicture}
  \SetGraphUnit{2.5}
  \Vertex{N} \SO(N){I} \SO(I){S}
  \EA(I){E}
  \Edge[style={bend left}](N)(I)
  \Edge[style={bend right}](N)(I)
  \Edge[style={bend left}](I)(S)
  \Edge[style={bend right}](I)(S)
  \Edges(N,E,S) \Edge(I)(E)
\end{tikzpicture}
\par
The degrees are \tkzGraphDegreeSequence:
  \tkzGraphIfEulerian{there is an Eulerian
  cycle}{there is an Eulerian path}{there
  is no Eulerian path}.
```

11.5.3 A cycle in a directed graph



cycle: C,D,A,B,C,A,C.

```
\begin{tikzpicture}
  \Vertices[unit=2,rotation=90]{circle}{A,B,C,D}
  \Edges[->](A,B,C,D,A)
  \Edge[->,style={bend left=20}](A)(C)
  \Edge[->,style={bend left=20}](C)(A)
  \tkzGraphEuler[numbers,start=C,pos=.4]
\end{tikzpicture}
\par
\tkzGraphEulerType: \tkzGraphEulerPath.
```

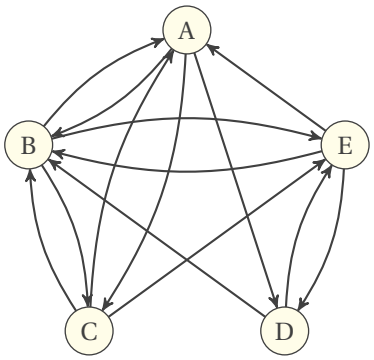
11.6 Powers of the adjacency matrix

`\tkzGraphMatrixPower[⟨local options⟩]{⟨k⟩}`

Options	Default	Definition
directed	false	an edge A--B counts only for the entry (A,B)
order	{}	order of the vertices: order={A,B,C}

M^k, where M is the matrix given by `\tkzAdjacencyMatrix` *(same options). The entry (i,j) of M^k is the number of walks of length k from the i-th vertex to the j-th.* `\tkzGraphWalks[⟨directed⟩]{⟨k⟩}(⟨A⟩)(⟨B⟩)` *gives one entry; it is expandable.*

The directed graph G_2 of the exercise "North America, June 2004" (see the section 12.6): the matrix M^3 of the statement is computed.



$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$$M^3 = \begin{pmatrix} 6 & 6 & 4 & 5 & 3 \\ 5 & 6 & 5 & 3 & 6 \\ 5 & 7 & 4 & 3 & 6 \\ 3 & 5 & 3 & 3 & 3 \\ 6 & 6 & 3 & 3 & 5 \end{pmatrix}$$

Walks of length 3 from A to B: 6.

```
\begin{tikzpicture}
  \tkzGraphFromMatrix[directed,unit=2.2,
                      rotation=90]%
    {A,B,C,D,E}%
    {0,1,1,1,0; 1,0,1,0,1; 1,1,0,0,1;
     0,1,0,0,1; 1,1,0,1,0}
\end{tikzpicture}
\par
$M = \tkzAdjacencyMatrix[directed]$
\par\medskip
$M^3 = \tkzGraphMatrixPower[directed]{3}$
\par\medskip
Walks of length 3 from A to B:
  \tkzGraphWalks[directed]{3}(A)(B).
```

11.7 Texts of the tables

The texts of the tables are given by macros; they can be redefined with `\renewcommand`. With `booktabs`, the tables use its rules.

Macros	Default
`\tkzGraphVertexText`	Vertex
`\tkzGraphDegreeText`	Degree
`\tkzGraphColorText`	Color
`\tkzGraphChoiceText`	Choice
`\tkzGraphDoneMark`	■
`\tkzGraphColorSwatch{⟨color⟩}`	red

12 Exercises from the French baccalaureate

Graph theory was taught in the last year of the French *lycée* (option *mathématiques* of the *série ES*). Here are exercises given at the *baccalauréat* between 2003 and 2006, translated into English. All the graphs are drawn with **tkz-graph**; the code of each graph follows the exercise.

12.1 North America, June 2003

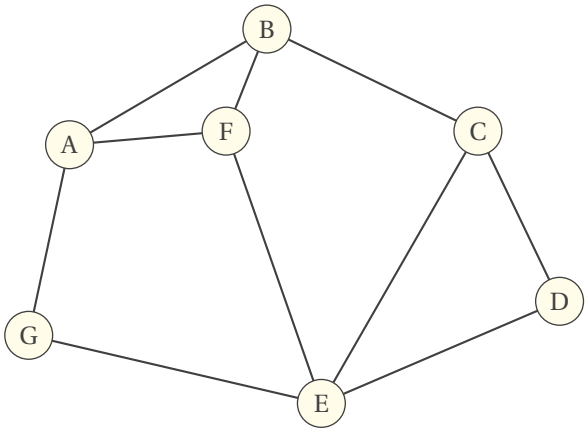
Let G be the graph below, with the vertices A, B, C, D, E, F and G.

1. What is its order? What is the degree of each vertex?
2. Copy and complete the table of the distances between two vertices of G:

Distance	A	B	C	D	E	F	G
A	X						
B	X	X					
C	X	X	X				
D	X	X	X	X			
E	X	X	X	X	X		
F	X	X	X	X	X	X	
G	X	X	X	X	X	X	X

   Deduce the diameter of this graph.

3. a) Give a complete subgraph of order 3 of G. What can you deduce for the chromatic number of G?
   b) Propose a coloring of G and deduce its chromatic number.
4. Give the matrix M of G (rows and columns in alphabetical order).
5. Using the matrix M^2 below, find the number of chains of length 2 starting from A and not returning to A.



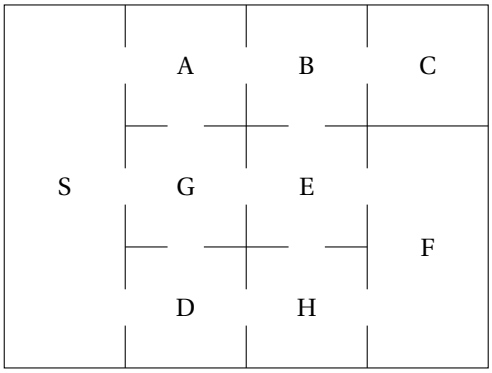
$$M^2 = \begin{pmatrix} 3 & 1 & 1 & 0 & 2 & 1 & 0 \\ 1 & 3 & 0 & 1 & 2 & 1 & 1 \\ 1 & 0 & 3 & 1 & 1 & 2 & 1 \\ 0 & 1 & 1 & 2 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 & 4 & 0 & 0 \\ 1 & 1 & 2 & 1 & 0 & 3 & 2 \\ 0 & 1 & 1 & 1 & 0 & 2 & 2 \end{pmatrix}$$

The answer to question 4 is given by **\tkzAdjacencyMatrix**, used just after the picture: $M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

```
\begin{tikzpicture}
  \Vertex[x=1.3,y=3.8]{A}  \Vertex[x=4.2,y=5.5]{B}
  \Vertex[x=7.3,y=4]{C}    \Vertex[x=8.5,y=1.5]{D}
  \Vertex[x=5,y=0]{E}      \Vertex[x=3.6,y=4]{F}
  \Vertex[x=0.7,y=1]{G}
  \Edges(A,B,C,D,E,G,A,F,E,C) \Edge(B)(F)
\end{tikzpicture}
$M = \tkzAdjacencyMatrix$
```

12.2 French West Indies and Guiana, June 2003

1. A museum has 9 rooms, named A, B, C, D, E, F, G, H and S. Here is the plan of the museum:



A visitor in the room S can go directly to the rooms A, B or G. From the room C, he can go directly to the room B but not to the room F. We study the walk of a visitor in the museum, without caring about how he enters or leaves it. The situation is modeled by a graph: the vertices are the rooms and the edges are the doors.

a) Draw a graph modeling this situation.

b) Is it possible to visit the museum using each door once and only once? Justify with a theorem of the course.

c) The curator wants each room to have a carpet different from the carpets of the neighboring rooms (rooms joined by a door). What is the minimum number of kinds of carpet? Justify.

2. Let M be the 9×9 matrix of this graph, the rooms being in the order S, A, B, C, D, E, F, G, H. The graph is not directed: what is the consequence for the matrix?

3. Here is the matrix

$$M^4 = \begin{pmatrix} 18 & 12 & 11 & 2 & 20 & 12 & 6 & 12 & 12 \\ 12 & 20 & 3 & 6 & 11 & 20 & 5 & 18 & 5 \\ 11 & 3 & 16 & 0 & 19 & 3 & 8 & 4 & 12 \\ 2 & 6 & 0 & 3 & 1 & 7 & 1 & 4 & 1 \\ 20 & 11 & 19 & 1 & 31 & 9 & 11 & 12 & 19 \\ 12 & 20 & 3 & 7 & 9 & 28 & 9 & 20 & 9 \\ 6 & 5 & 8 & 1 & 11 & 9 & 9 & 8 & 9 \\ 12 & 18 & 4 & 4 & 12 & 20 & 8 & 20 & 6 \\ 12 & 5 & 12 & 1 & 19 & 9 & 9 & 6 & 17 \end{pmatrix}$$

a) How many paths of 4 steps start from D and come back to D?

b) How many paths of 4 steps start from S and end in C? Give them.

c) Is it always possible to join two rooms in 4 steps? Justify.

The plan is drawn with Ti*k*Z only:

```
\begin{tikzpicture}
  \draw (0,0) rectangle (8,6);
  \foreach \x in {2,4,6}{%
    \draw (\x,0)--(\x,0.7)   (\x,1.3)--(\x,2.7)
          (\x,3.3)--(\x,4.7) (\x,5.3)--(\x,6);}
  \draw (2,2)--(2.7,2) (3.3,2)--(4.7,2) (5.3,2)--(6,2);
  \draw (2,4)--(2.7,4) (3.3,4)--(4.7,4) (5.3,4)--(8,4);
  \foreach \r/\p in {S/{1,3},G/{3,3},D/{3,1},A/{3,5},H/{5,1},
                     E/{5,3},B/{5,5},F/{7,2},C/{7,5}}
    \node at (\p) {\r};
\end{tikzpicture}
```

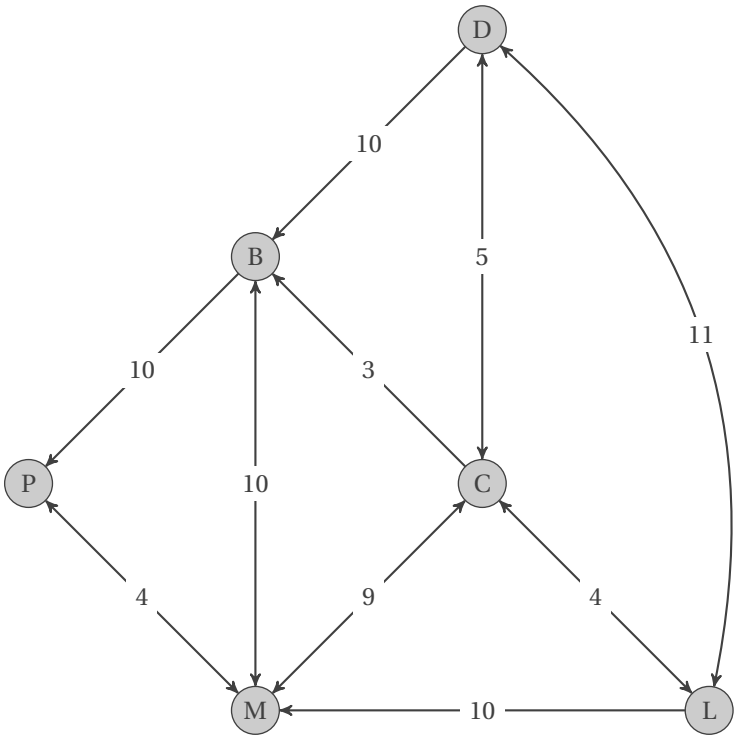
12.3 Asia, June 2003

	B	C	L	M	P
B		X		X	X
C	X		X	X	
L		X		X	
M	X	X	X		X
P	X			X	

In the town of GRAPH, we study the main streets joining the places open to the public: the town hall (M), the shopping center (C), the library (B), the swimming pool (P) and the high school (L). The table gives the streets between these places.

1. Draw a graph representing this situation.
2. Show that there is a route using each street of this plan once and only once. Justify and propose such a route.

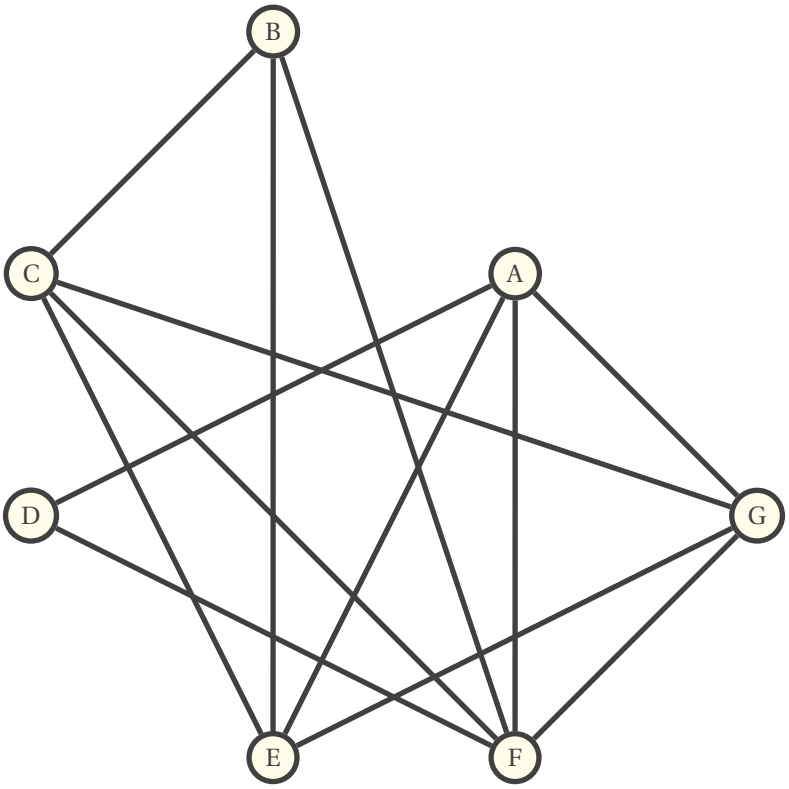
   Is there such a route starting and ending at the same place?
3. Dimitri lives in this town. The graph below gives the **new** plan of the district, with the directions of traffic in the streets and the travel times between the places (D is the home of Dimitri).



```
\begin{tikzpicture}
  \SetGraphUnit{3}
  \tkzSetUpGraph[light fill=gray!40,label fill=white]
  \Vertex{P}
  \NOEA(P){B} \SOEA(P){M} \NOEA(B){D}
  \SOEA(B){C} \SOEA(C){L}
  \Edge[<->,label=$4$](P)(M)
  \Edge[<->,label=$9$](C)(M)
  \Edge[<->,label=$4$](C)(L)
  \Edge[<->,label=$5$](C)(D)
  \Edge[<->,label=$10$](B)(M)
  \Edge[<->,label=$11$,style={bend right}](L)(D)
  \Edge[->,label=$3$](C)(B)
  \Edge[->,label=$10$](D)(B)
  \Edge[->,label=$10$](L)(M)
  \Edge[->,label=$10$](B)(P)
\end{tikzpicture}
```

12.4 France, June 2003

A charity concert is organized in a large hall. Seven international artists are invited: Luther Allunison (A), John Biaise (B), Phil Colline (C), Bob Ditlâne (D), Jimi Endisque (E), Robert Fripe (F) and Rory Garaguerre (G). Some musicians refuse to play with some others, so the organizer must plan several parts in the show. The edges of the graph Γ below show which musicians refuse to play together.

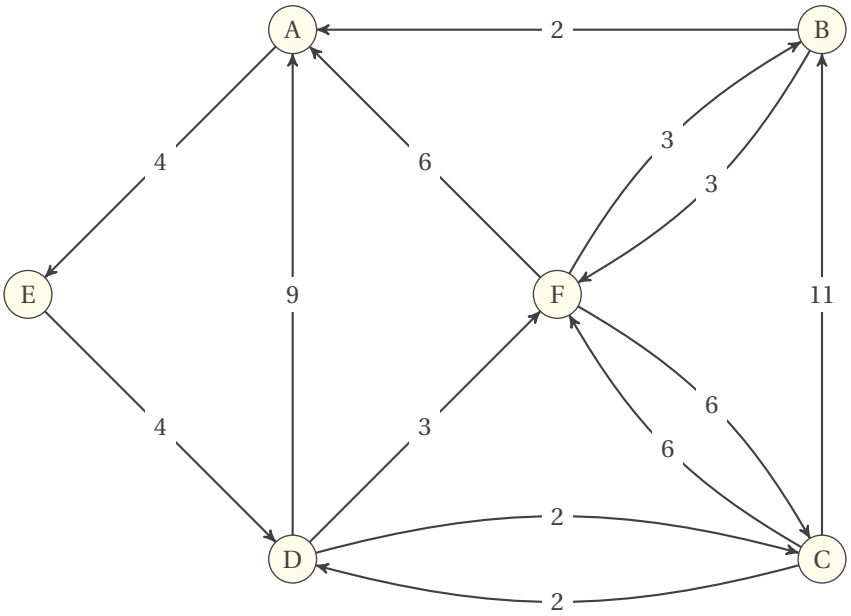


1. Give the matrix of the graph Γ (vertices in alphabetical order).
2. What is the subgraph of Γ with the vertices A, E, F and G? What can you deduce for the chromatic number $\chi(\Gamma)$?
3. Which vertex of Γ has the highest degree? Deduce bounds for $\chi(\Gamma)$.
4. Sort the vertices by decreasing degree and color the graph Γ.
5. How many parts must the organizer plan? Propose a distribution of the musicians in these parts.

```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \GraphInit[vstyle=Normal]
  \tkzSetUpGraph[vertex line width=2pt,edge width=2pt]
  \Vertex{D}
  \SOEA(D){E} \EA(E){F}
  \NOEA(F){G} \NOWE(G){A}
  \NOWE(A){B} \SOWE(B){C}
  \Edges(F,G,A,D,F,B,E,G,C,F,A,E,C,B)
\end{tikzpicture}
```

12.5 Foreign centers, June 2003

In the afternoon, a delivery man loads his truck at the warehouse A, delivers five customers B, C, D, E and F, then comes back to the warehouse. The road network, with the directions of traffic, and the travel times (in minutes) are given by the graph G:



1. Give the matrix M of the graph G, with the rows and the columns in the order A, B, C, D, E, F.
2. Here is the matrix

$$M^6 = \begin{pmatrix} 8 & 6 & 6 & 3 & 4 & 6 \\ 19 & 11 & 12 & 9 & 6 & 16 \\ 36 & 28 & 23 & 22 & 18 & 34 \\ 37 & 24 & 25 & 17 & 15 & 31 \\ 15 & 12 & 9 & 10 & 8 & 15 \\ 28 & 22 & 19 & 15 & 15 & 26 \end{pmatrix}$$

   We study the paths starting from the warehouse A and ending at A.
   a) How many paths of length 6 join A to A?
   b) Give these paths.
   c) Among those going through all the vertices, which one has the shortest travel time?
   d) What can the delivery man deduce from the last result?
3. The delivery man has chosen the fastest route. Unfortunately, the customer C is absent when he arrives, and he decides to end his round with this customer. What is the fastest path from C back to the warehouse A? Justify.

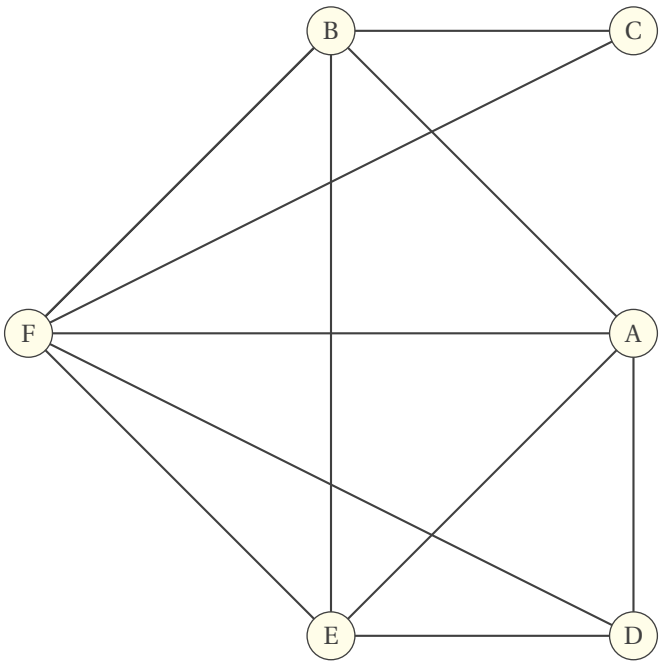
The pairs of opposite edges are drawn with a loop **\foreach**:

```
\begin{tikzpicture}
  \SetGraphUnit{3.5}
  \tkzSetUpGraph[label fill=white]
  \Vertex{F}
  \NOWE(F){A} \NOEA(F){B} \SOEA(F){C}
  \SOWE(F){D} \SOWE(A){E}
  \Edge[->,label=$4$](A)(E)
  \Edge[->,label=$4$](E)(D)
  \Edge[->,label=$9$](D)(A)
  \Edge[->,label=$2$](B)(A)
  \Edge[->,label=$11$](C)(B)
  \Edge[->,label=$3$](D)(F)
  \Edge[->,label=$6$](F)(A)
  \foreach \u/\v/\t in {D/C/2,C/D/2,F/B/3,B/F/3,F/C/6,C/F/6}
    {\Edge[->,label=$\t$,style={bend left=15}](\u)(\v)}
\end{tikzpicture}
```

12.6 North America, June 2004

The parts A and B are independent.

Part A. Let G_1 be the graph below.



1. Justify the following statements.

   A_1: "the graph G_1 has at least one Eulerian chain".

   A_2: "the chain DABCFBEFAE is not an Eulerian chain of G_1".

2. Find a complete subgraph of G_1 of the largest possible order. Deduce a lower bound of the chromatic number γ of this graph.

3. Find an upper bound of this chromatic number (justify).

4. Propose a coloring of G_1 and find its chromatic number.

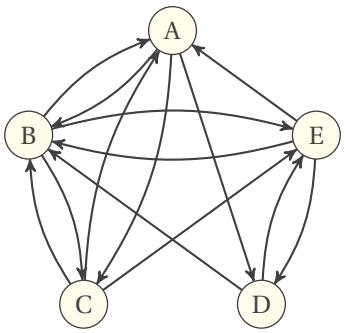
Part B. Let M be the matrix of a directed graph G_2, the vertices A, B, C, D and E being in alphabetical order.

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad M^3 = \begin{pmatrix} 6 & 6 & 4 & 5 & 3 \\ 5 & 6 & 5 & 3 & 6 \\ 5 & 7 & 4 & 3 & 6 \\ 3 & 5 & 3 & 3 & 3 \\ 6 & 6 & 3 & 3 & 5 \end{pmatrix}.$$

1. Draw the graph G_2.

2. Find the number of chains of length 3 joining B to D. Give them all.

The graph G_2 can be drawn directly from its matrix with `\tkzGraphFromMatrix`:




```
\begin{tikzpicture}
  \tkzGraphFromMatrix[directed,unit=2,
                      rotation=90]%
    {A,B,C,D,E}{0,1,1,1,0;
                1,0,1,0,1;
                1,1,0,0,1;
                0,1,0,0,1;
                1,1,0,1,0}
\end{tikzpicture}
```

```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \Vertex{F}
  \NOEA(F){B} \SOEA(F){E}
  \EA(B){C}   \EA(E){D}
  \NO(D){A}
  \Edges(B,F,E,D,A,E,B,A,F,B,C,F,D)
\end{tikzpicture}
```

12.7 Foreign centers, May 2004

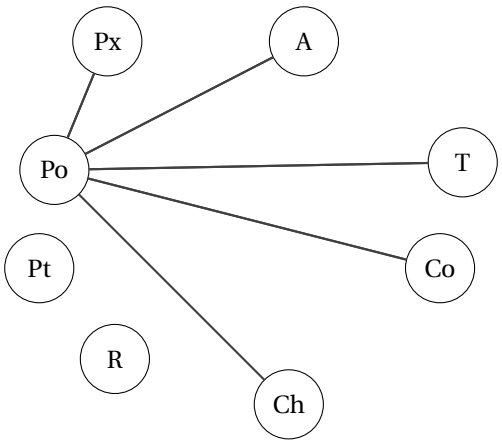
A gardener owns a sunny piece of land with a shadier part. He wants to create plots where he will plant 8 kinds of vegetables: garlic (A), squash (Co), cabbage (Ch), leeks (Px), peas (Po), potatoes (Pt), radishes (R) and tomatoes (T). In an almanac, he finds the incompatibilities of plants given by two tables.

Incompatible exposures	
Partial shade plants	Full sun plants
peas radishes	cabbage tomatoes squash
For example, peas are incompatible with cabbage, tomatoes and squash.	

Incompatible associations in a same plot	
peas	garlic, leeks
potatoes	squash, radishes, tomatoes
cabbage	tomatoes, garlic, leeks, squash
squash	tomatoes
For example, peas are incompatible with garlic and leeks.	

To take these incompatibilities into account, the gardener models the situation by a graph with eight vertices, one per vegetable.

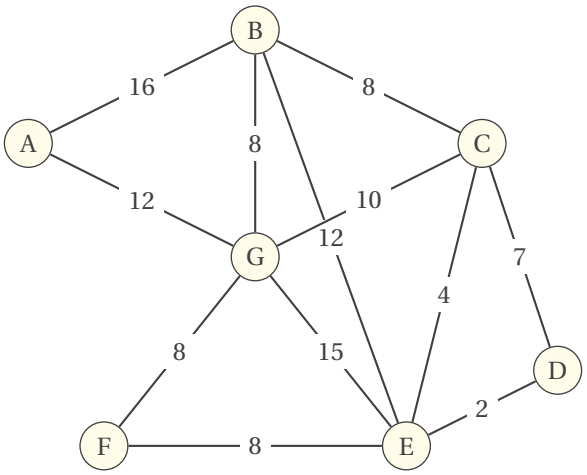
1. Complete the graph below to show the incompatibilities given by the two tables.
2. Compute the sum of the degrees of the vertices and deduce the number of edges.
3. Find a complete subgraph of order 4. What can you deduce for the chromatic number of the graph?
4. Give the chromatic number of the graph and interpret it as the minimum number of plots.
5. Give a distribution of the plants such that each plot contains exactly two kinds of plants, with a minimum number of plots.
6. Give a distribution of the plants such that one plot contains three plants, with a minimum number of plots.




```
\begin{tikzpicture}
  \tikzset{VertexStyle/.style = {shape = circle, fill = white,
                                 minimum size = 26pt, draw}}
  \Vertex[x=1,y=0.8]{R}    \Vertex[x=0.2,y=3.3]{Po}
  \Vertex[x=0,y=2]{Pt}     \Vertex[x=0.9,y=5]{Px}
  \Vertex[x=3.5,y=5]{A}    \Vertex[x=5.6,y=3.4]{T}
  \Vertex[x=5.3,y=2]{Co}   \Vertex[x=3.3,y=0.2]{Ch}
  \Edges(Po,Px,Po,A,Po,T,Po,Co,Po,Ch)
\end{tikzpicture}
```

12.8 France, June 2004

The graph below shows, without scale, the possible routes between the seven buildings A, B, C, D, E, F and G of a large company. A security guard regularly patrols the site; the numbers are his travel times in minutes, in both directions.

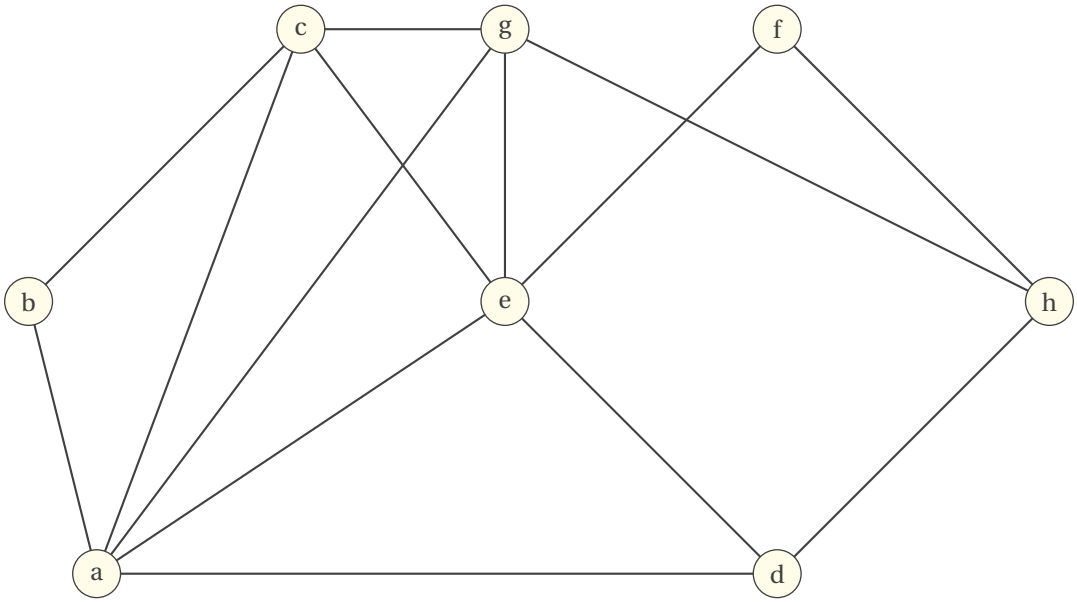


1. Show that the guard can walk once and only once along every path of the site. Give such a route.
2. Can the guard come back to his starting point after walking once and only once along every path? Justify.
3. Every morning, the guard goes from the building A to the building D. With an algorithm, find the path with the shortest travel time and give this time.

```
\begin{tikzpicture}
  \GraphInit[vstyle=Dijkstra]
  \Vertex{G}
  \Vertex[x=-3,y=1.5]{A}  \Vertex[x=0,y=3]{B}
  \Vertex[x=3,y=1.5]{C}   \Vertex[x=4,y=-1.5]{D}
  \Vertex[x=2,y=-2.5]{E}  \Vertex[x=-2,y=-2.5]{F}
  \foreach \u/\v/\t in {A/B/16,A/G/12,B/C/8,B/E/12,B/G/8,C/D/7,
                        C/E/4,C/G/10,D/E/2,E/F/8,E/G/15,F/G/8}
    {\Edge[label=$\t$](\u)(\v)}
\end{tikzpicture}
```

12.9 Réunion, June 2004

Part A. Let G be the graph below and M its matrix, the vertices being in alphabetical order. The matrix M^3 is also given.

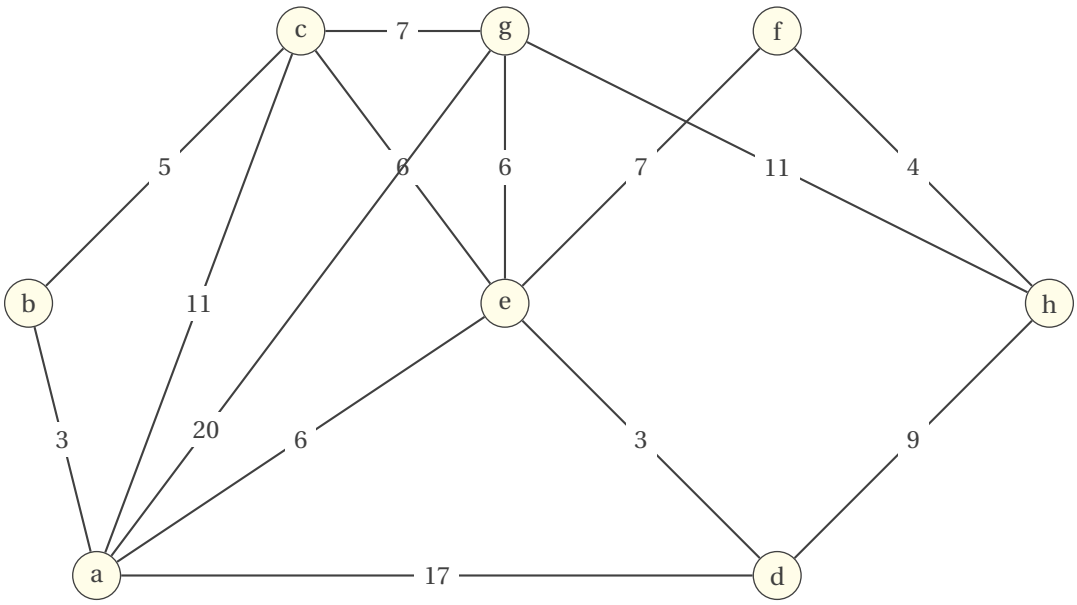


$$M^3 = \begin{pmatrix} 10 & 8 & 11 & 10 & 12 & 5 & 13 & 4 \\ 8 & 2 & 7 & 3 & 5 & 2 & 4 & 3 \\ 11 & 7 & 8 & 6 & 12 & 3 & 10 & 5 \\ 10 & 3 & 6 & 2 & 11 & 1 & 4 & 8 \\ 12 & 5 & 12 & 11 & 8 & 8 & 13 & 3 \\ 5 & 2 & 3 & 1 & 8 & 0 & 2 & 6 \\ 13 & 4 & 10 & 4 & 13 & 2 & 6 & 9 \\ 4 & 3 & 5 & 8 & 3 & 6 & 9 & 0 \end{pmatrix}$$

Say whether the following statements are true or false, and justify.

1. The order of the graph is equal to the largest degree of the vertices.
2. The graph G contains a complete subgraph of order 3.
3. The vertices of G can be colored with three colors, two adjacent vertices having different colors.
4. It is possible to go through this graph using each edge once and only once.
5. For each vertex, there is at least one path of length 3 joining it to each of the seven other vertices.
6. There are 72 paths of length 3 joining the vertex e to the eight vertices of the graph.

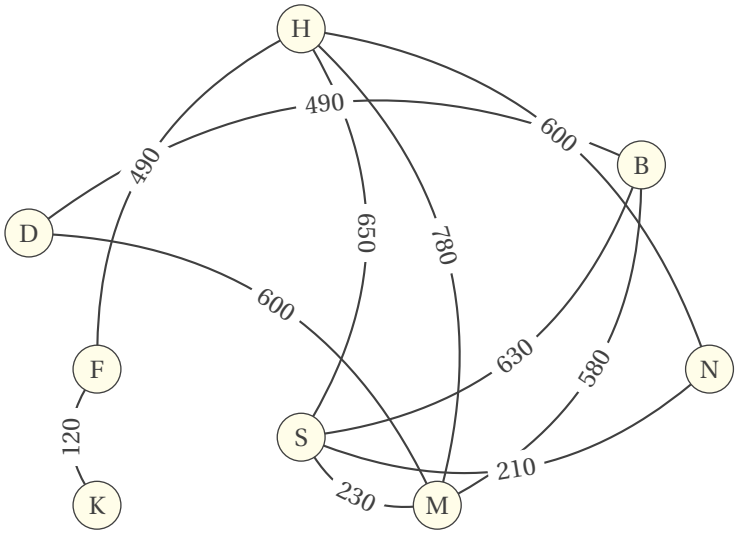
Part B. The graph below is a network of bus lines. The vertices are the stops; the weights are the travel times in minutes between two stops (changes included). With an algorithm, find the minimum time to go from the stop a to the stop h, and give the route.




```
\begin{tikzpicture}
  \SetGraphUnit{4}
  \GraphInit[vstyle=Dijkstra]
  \Vertex{e}
  \NOEA(e){f} \SOEA(e){d} \SOEA(f){h}
  \Vertex[x=0,y=4]{g}
  \Vertex[x=-3,y=4]{c}
  \Vertex[x=-6,y=-4]{a}
  \SOWE(c){b}
  \foreach \u/\v/\t in {a/b/3,a/c/11,a/e/6,a/d/17,c/b/5,c/e/6,
                        c/g/7,f/e/7,d/e/3,d/h/9,g/e/6,f/h/4}
    {\Edge[label=$\t$](\u)(\v)}
  \Edge[style={pos=.25},label=$20$](a)(g)
  \Edge[label=$11$](g)(h)
\end{tikzpicture}
```

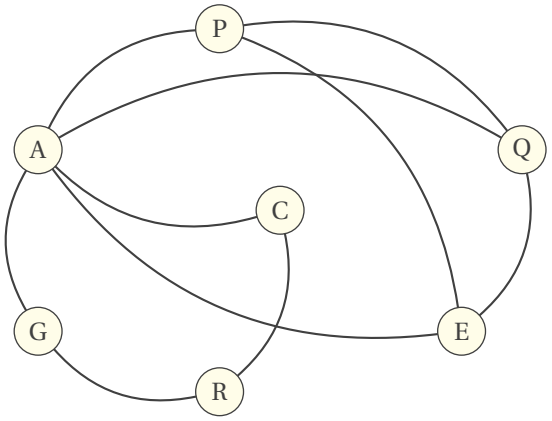
12.10 South America, November 2006

1. For the 2006 football World Cup in Germany, a travel agency organizes coach trips through the cities where a national team plays. The roads are represented by the graph below; the numbers are the distances in kilometers. B, D, F, H, K, M, N and S are Berlin, Dortmund, Frankfurt, Hamburg, Kaiserslautern, Munich, Nuremberg and Stuttgart.



   Explaining the method, find the shortest route from Kaiserslautern to Berlin with the coaches of this agency.

2. For safety reasons, the supporters of some national teams cannot stay in the same hotel. Here is the graph of incompatibility between the supporters of several teams: for example, a supporter of the team A cannot stay with a supporter of the team P.



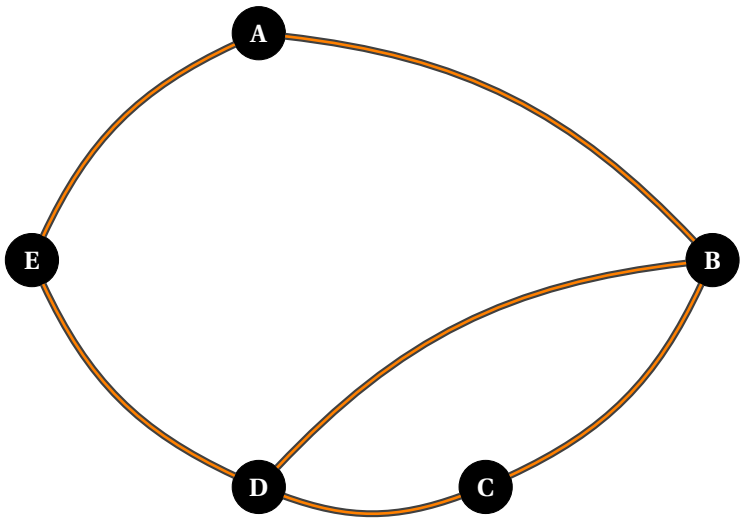
   a) Find the chromatic number of this graph and justify.

   b) Propose a distribution of the supporters in a minimum number of hotels.

```
\begin{tikzpicture}
  \Vertex[x=0,y=0]{K}  \Vertex[x=0,y=2]{F}
  \Vertex[x=-1,y=4]{D} \Vertex[x=3,y=7]{H}
  \Vertex[x=8,y=5]{B}  \Vertex[x=9,y=2]{N}
  \Vertex[x=5,y=0]{M}  \Vertex[x=3,y=1]{S}
  \tikzset{LabelStyle/.style = {fill=white,sloped}}
  \tikzset{EdgeStyle/.style  = {bend left}}
  \Edge[label=$120$](K)(F)
  \Edge[label=$650$](H)(S)
  \Edge[label=$780$](H)(M)
  \Edge[label=$490$](D)(B)
  \Edge[label=$600$](D)(M)
  \Edge[label=$580$](B)(M)
  \Edge[label=$600$](H)(N)
  \Edge[label=$490$](F)(H)
  \tikzset{EdgeStyle/.style  = {bend right}}
  \Edge[label=$630$](S)(B)
  \Edge[label=$210$](S)(N)
  \Edge[label=$230$](S)(M)
\end{tikzpicture}
```

12.11 Lebanon, June 2006

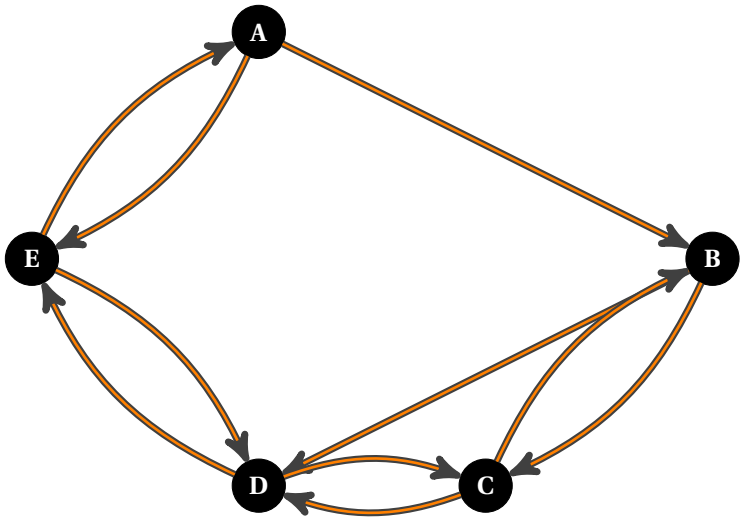
1. In a park, five benches are joined by alleys. The benches are the vertices A, B, C, D, E and the alleys the edges of the graph G:



   a) The benches are painted so that two benches joined by an alley always have different colors. Give bounds for the minimum number of colors and justify. Find this number.

   b) Is it possible to walk along all the alleys of the park without using the same alley twice?

2. An exhibition takes place in the park. As there are too many people, a traffic plan is set up: some alleys become one-way, the others stay two-way. For example, between the benches B and C one can go from B to C and from C to B, but between A and B only from A to B. The directed graph G' below models this new situation.



   a) Give the matrix M of the graph G' (vertices in alphabetical order).

   b) Here is

$$M^5 = \begin{pmatrix} 1 & 6 & 9 & 6 & 10 \\ 4 & 5 & 7 & 11 & 5 \\ 4 & 6 & 6 & 11 & 5 \\ 1 & 5 & 10 & 6 & 10 \\ 6 & 5 & 5 & 14 & 2 \end{pmatrix}$$

   How many paths of length 5 go from the vertex D to the vertex B? Give them all.

c) Show that there is only one cycle of length 5 through the vertex A. What is this cycle? Is it the same for the vertex B?

The graph G' is drawn from its matrix: the vertices are placed first, then `\tkzGraphFromMatrix[placed,directed]` draws the arcs; the pairs of opposite arcs are bent.

```
\begin{tikzpicture}
  \SetGraphUnit{3}
  \tikzset{VertexStyle/.style = {shape = circle, fill = black,
     minimum size = 20pt, text = white, draw}}
  \tikzset{EdgeStyle/.style = {double = orange,
     double distance = 1pt, thick}}
  \Vertex[L=\textbf{E}]{E}
  \NOEA[L=\textbf{A}](E){A} \SOEA[L=\textbf{D}](E){D}
  \EA[L=\textbf{C}](D){C}   \NOEA[L=\textbf{B}](C){B}
  \tkzGraphFromMatrix[placed,directed,bend=20]%
    {A,B,C,D,E}{0,1,0,0,1;
                0,0,1,1,0;
                0,1,0,1,0;
                0,0,1,0,1;
                1,0,0,1,0}
\end{tikzpicture}
```

Index

\AddVertexColor, 32

\DisableEdgeStyle, 23

\EA, 7, 11, 13
\Edge, 7, 19, 21, 26, 42
\Edge: options
    ->, 19
    -, 19
    <->, 19
    <-, 19
    arrow, 19
    color, 19
    dd, 19
    double, 19
    labelcolor, 19
    labelstyle, 19
    labeltext, 19
    label, 19
    local, 19
    lw, 19
    style, 19
\EdgeLineWidth, 26
\Edges, 6, 7, 21, 42
\expandafter, 7

\foreach, 60

\GraphInit, 7, 8, 11, 19, 24, 35
\GraphInit: options
    unit, 24
    vstyle, 24
\grHypercube, 33
\grLoop, 21
\grProb, 37, 38
\grProb: arguments
    E, 37
    N, 37
    S, 37
    W, 37
    left, 37
    right, 37
\grProb: options
    Ldist, 37
    LoopDist, 37
    LposA, 37
    LposB, 37
    unit, 37
\grProbMatrix, 36, 38, 46
\grProbThree, 38

\Loop, 7, 21, 42
\Loop: options
    color, 21
    dir, 21
    dist, 21
    labelstyle, 21

label, 21
style, 21

\NO, 7, 11, 13
\NOEA, 6, 7, 11, 13
\NOWE, 7, 11, 13

\renewcommand, 51

\SetGraphArtColor, 30
\SetGraphColor, 30
\SetGraphShadeColor, 30
\SetGraphUnit, 11
\SetUpEdge, 7, 23, 24
\SetUpEdge: options
color, 23
labelcolor, 23
labelstyle, 23
labeltext, 23
label, 23
lw, 23
style, 23
\SetUpVertex, 24, 30
\SetUpVertex: options
FillColor, 30
InnerSep, 30
LabelOut, 30
Ldist, 30
LineColor, 30
LineWidth, 30
Lpos, 30
Math, 30
MinSize, 30
NoLabel, 30
OuterSep, 30
Style, 30
TextColor, 30
Unit, 30
\SetVertexArt, 24, 29
\SetVertexForPresentation, 31
\SetVertexLabel, 17
\SetVertexLabeledSmall, 31
\SetVertexLabelIn, 18
\SetVertexLabelOut, 18
\SetVertexMath, 18, 35
\SetVertexNoLabel, 17
\SetVertexNoMath, 18
\SetVertexNormal, 24, 28
\SetVertexNormal: options
FillColor, 28
InnerSep, 28
LineColor, 28
LineWidth, 28
MinSize, 28
OuterSep, 28
Shape, 28
TextColor, 28
\SetVertexNotLabeledSmall, 31

\SetVertexShade, 24, 29
\SetVertexSimple, 24, 28
\SetVertexSimple: options
    FillColor, 28
    InnerSep, 28
    LineColor, 28
    LineWidth, 28
    MinSize, 28
    OuterSep, 28
    Shape, 28
\SO, 7, 11, 13
\SOEA, 7, 11, 13
\SOWE, 7, 11, 13

\tikzset, 23, 24, 31
\tkzAdjacencyMatrix, 34, 51, 52
\tkzAdjacencyMatrix: options
    directed, 34
    order, 34
\tkzDefGraphStyle, 24
\tkzEA, 7
\tkzEdge, 7
\tkzEdges, 7
\tkzGraphChoiceText, 51
\tkzGraphColorCount, 47
\tkzGraphColoring, 47
\tkzGraphColoring: options
    colors, 47
    draw, 47
\tkzGraphColoringTable, 47
\tkzGraphColorSwatch, 51
\tkzGraphColorText, 51
\tkzGraphComponents, 49
\tkzGraphDegree, 42
\tkzGraphDegree: options
    math, 42
    sort, 42
\tkzGraphDegreeSequence, 42
\tkzGraphDegreeTable, 42, 44
\tkzGraphDegreeText, 51
\tkzGraphDijkstra, 44
\tkzGraphDijkstra: options
    color, 44
    directed, 44
    draw, 44
    lw, 44
    style, 44
\tkzGraphDijkstraTable, 44
\tkzGraphDoneMark, 44, 51
\tkzGraphEuler, 49
\tkzGraphEuler: options
    directed, 49
    numbers, 49
    pos, 49
    start, 49
\tkzGraphEulerPath, 49
\tkzGraphEulerType, 49
\tkzGraphFromMatrix, 34, 36, 42, 61, 70

\tkzGraphFromMatrix: options
bend, 34
clockwise, 34
directed, 34
loop distance, 34
placed, 34
rotation, 34
style, 34
unit, 34
weighted, 34
\tkzGraphIfConnected, 49
\tkzGraphIfEulerian, 49
\tkzGraphInDegree, 42
\tkzGraphMatrixPower, 51
\tkzGraphMatrixPower: options
directed, 51
order, 51
\tkzGraphOddVertices, 49
\tkzGraphOrder, 42
\tkzGraphOutDegree, 42
\tkzGraphPath, 44
\tkzGraphPathLength, 44
\tkzGraphSize, 42
\tkzGraphVertexColor, 47
\tkzGraphVertexText, 51
\tkzGraphWalks, 51
\tkzHighlightPath, 32
\tkzHighlightPath: options
color, 32
cycle, 32
lw, 32
style, 32
\tkzLoop, 7
\tkzNO, 7
\tkzNOEA, 7
\tkzNOWE, 7
\tkzSetUpColors, 31
\tkzSetUpGraph, 7, 11, 19, 24, 26
\tkzSetUpGraph: options
arrow tip, 26
art color, 26
ball color, 26
big size, 26
dark fill, 26
directed, 26
double distance, 26
edge color, 26
edge fill, 26
edge width, 26
inner sep, 26
label fill, 26
label text, 26
light fill, 26
medium size, 26
outer sep, 26
small size, 26
unit, 26
vertex fill, 26

vertex line color, 26
vertex line width, 26
vertex shape, 26
vertex text color, 26
\tkzSO, 7
\tkzSOEA, 7
\tkzSOWE, 7
\tkzVertex, 7
\tkzVertices, 7
\tkzWE, 7

\Vertex, 7, 9, 11, 13, 17, 30, 42
\Vertex: arguments
name, 9
\Vertex: options
LabelOut, 9
Ldist, 9
Lpos, 9
L, 9
Math, 9
NoLabel, 9
Node, 9
a, 9
dir, 9
d, 9
empty, 9
position, 9
style, 9
unit, 9
x, 9
y, 9
\VertexBallColor, 29
\VertexBigMinSize, 29
\VertexShape, 26
\VertexSmallMinSize, 29
\Vertices, 6, 7, 9, 13, 35, 42
\Vertices: arguments
circle, 13
line, 13
square, 13
tr1, 13
tr2, 13
tr3, 13
tr4, 13
\Vertices: options
clockwise, 13
dir, 13
rotation, 13
unit, 13

\WE, 7, 11, 13