Documentation of `mptrees.mp`

Olivier PÉAULT*

April 8, 2024

Contents

Part I
Trees

1 Overview

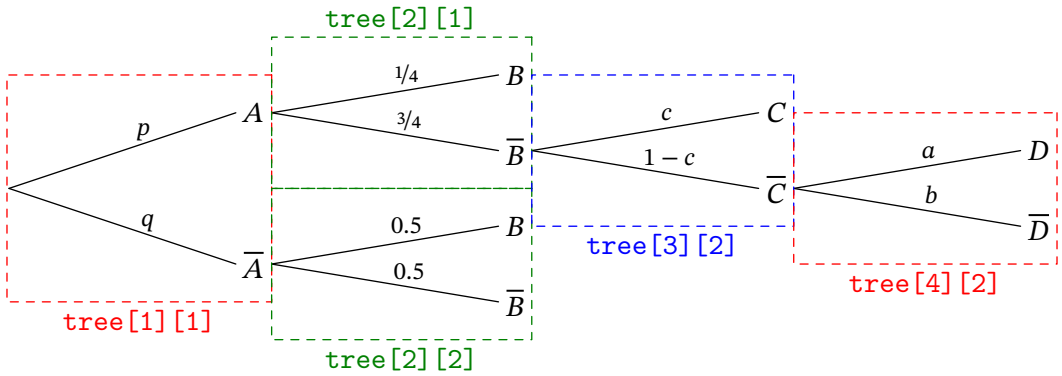
The first aim of this package is to simplify the drawing of probability trees (or other kind of "tree") with METAPOST. It provides one main command and several parameters to control the output.

It can be used in standalone files with two compilations (`latexmp` package is loaded) but also with LuaLaTeX and `luamplib` package.

*E-mail : o.peault@posteo.net

`tree[<i>][<j>](<dim1>,<dim2>,...)(<ev1>,<prob1>,<ev2>,<prob2>,...)` `picture`

Probability tree located in column `i` and row `j` (see figure below). `dim1`, `dim2`,... can be numerics or pairs and control the dimension of the tree. `ev1`, `prob1`... can be strings or pictures and will be printed (using `latexmp` if strings) at the end of the edge (the event) and above the edge (the probability).



`begintree; endtree;`

Note that you can use these commands inside any `beginfig();...endfig;` but sometimes, for some constructions, they need to be enclosed between `begintree` and `endtree` commands. Such commands are indicated with a margin note.

2 Trees

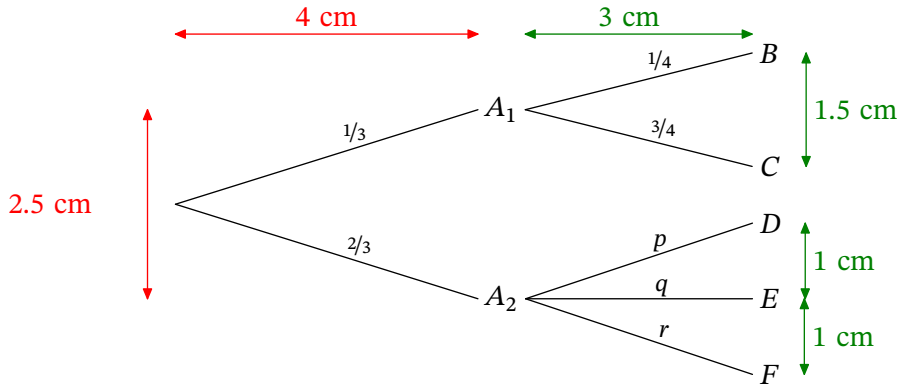
2.1 Different kinds of trees

`tree[<i>][<j>](<width>,<vspace>)(<ev1>,<prob1>,<ev2>,<prob2>,...)` `picture`

Regular tree where `width` is the horizontal width of the tree and `vspace` the vertical space between two consecutive nodes.

Example 1

```
beginfig(1);
draw tree[1][1](4cm,2.5cm)("$A_1$","$\nicefrac{1}{3}$","$A_2$","$\nicefrac{2}{3}$");
draw tree[2][1](3cm,1.5cm)("$B$","$\nicefrac{1}{4}$","$C$","$\nicefrac{3}{4}$");
draw tree[2][2](3cm,1cm)("$D$","$p$","$E$","$q$","$F$","$r$");
endfig;
```

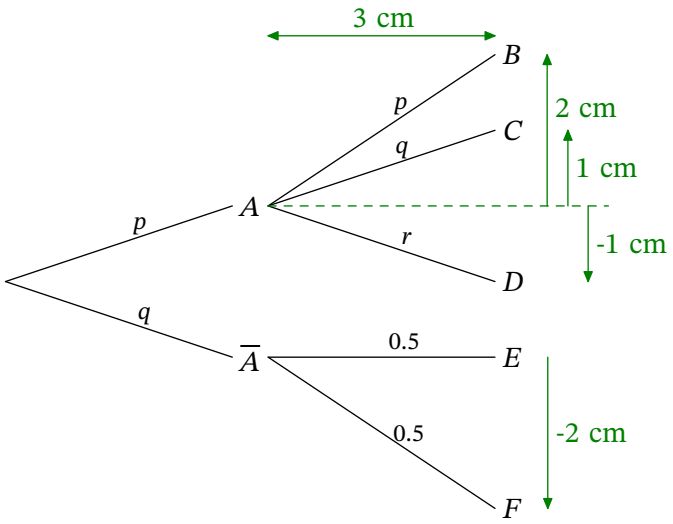


`tree[<i>][<j>](<width>,<vsp1>,<vsp2>,...)(<ev1>,<p1>,<ev2>,<p2>,...)` picture

Tree where `width` is the horizontal width of the tree while each `vsp` indicates the vertical space between the node and the origin of the tree.

Example 2

```
beginfig(2);
draw tree[1][1](3cm,2cm)("$A$","$p$","$\overline{A}$","$q$");
draw tree[2][1](3cm,2cm,1cm,-1cm)("$B$","$p$","$C$","$q$","$D$","$r$");
draw tree[2][2](3cm,0cm,-2cm)("$E$","$0.5$","$F$","$0.5$");
endfig;
```

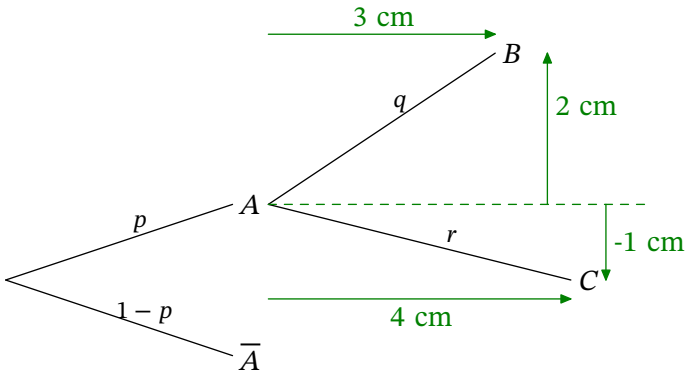


`tree[<i>][<j>](<pair1>,<pair2>,...)(<ev1>,<prob1>,<ev2>,<prob2>,...)` picture

Tree where `pair1`, `pair2`... indicate the coordinates of each node from the origin of the tree.

Example 3

```
beginfig(3);
draw tree[1][1](3cm,2cm)("$A$","$p$","$\overline{A}$","$1-p$");
draw tree[2][1]((3cm,2cm),(4cm,-1cm))("$B$","$q$","$C$","$r$");
endfig;
```



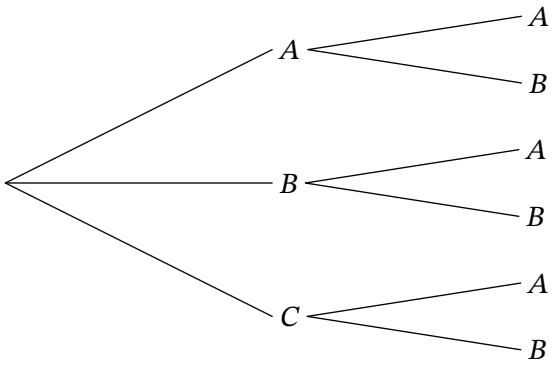
2.2 Simple trees

`stree[<i>][<j>](...)(<ev1>,<ev2>)` picture

Same as previous except that there are no probabilities.

Example 4

```
beginfig(4);
draw stree[1][1](100,50)("$A$","$B$","$C$");
draw stree[2][1](80,25)("$A$","$B$");
draw stree[2][2](80,25)("$A$","$B$");
draw stree[2][3](80,25)("$A$","$B$");
endfig;
```



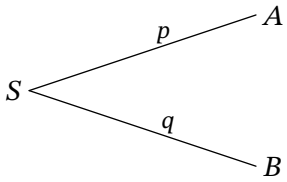
2.3 Start and end labels

`startlabel(<s>)` picture

Prints `s` (can be a string or a picture) at the origin of the tree.

Example 5

```
beginfig(5);
 draw startlabel("$S$");
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

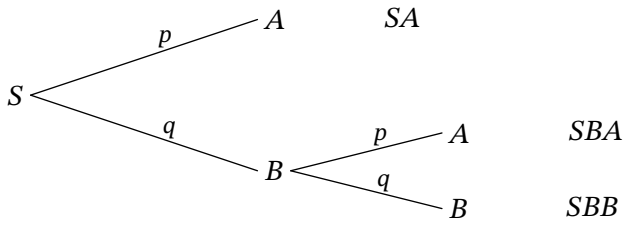


`endlabel[<i>][<j>](<s>)` picture

Prints `s` at the end of a branch. The space between the previous label ans `s` is controlled by the numeric `endlabelspace` which defaults to `1cm`.

Example 6

```
beginfig(6);
 draw startlabel("$S$");
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
 draw tree[2][2](2cm,1cm)("$A$","$p$","$B$","$q$");
 draw endlabel[2][1]("$SA$");
 draw endlabel[3][1]("$SBA$");
 draw endlabel[3][2]("$SBB$");
endfig;
```



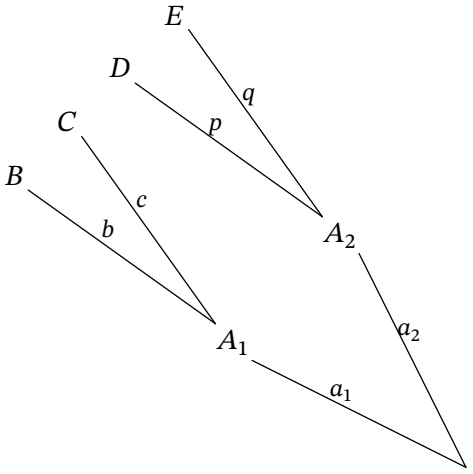
3 Direction

`dirtree` `numeric, default : 0`

All trees are construct horizontally by default. `ditree` indicates the angle in degrees between the horizontal and the main direction of the tree.

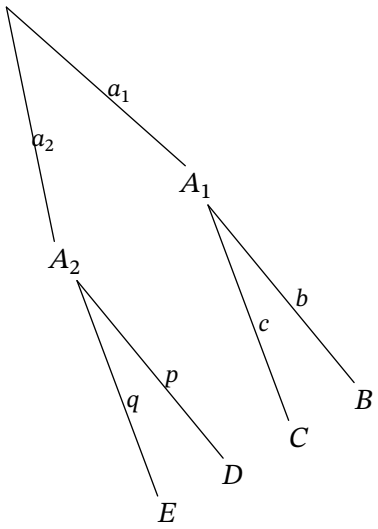
Example 7

```
beginfig(7);
dirtree:=135;
draw tree[1][1](3cm,2cm)("$A_1$","$a_1$","$A_2$","$a_2$");
draw tree[2][1](3cm,1cm)("$B$","$b$","$C$","$c$");
draw tree[2][2](3cm,1cm)("$D$","$p$","$E$","$q$");
endfig;
```



Example 8

```
beginfig(8);
dirtree:=-60;
draw tree[1][1](3cm,2cm)("$A_1$","$a_1$","$A_2$","$a_2$");
draw tree[2][1](3cm,1cm)("$B$","$b$","$C$","$c$");
draw tree[2][2](3cm,1cm)("$D$","$p$","$E$","$q$");
endfig;
```

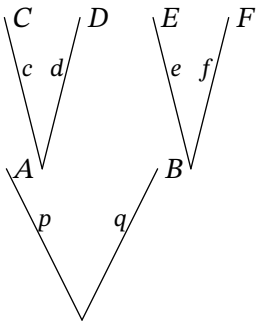


`dirlabel` `numeric, default : 0`

All the trees are viewed as "horizontal" trees, so the space between two subtrees is horizontal too. With `dirtree`, the whole (horizontal) tree is rotated. But if the tree is designed vertically, spacing is wrong. In this case, one can use `dirlabel` to indicate the orientation of the tree.

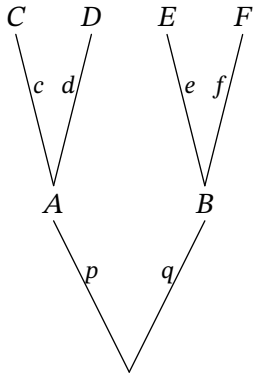
Example 9

```
beginfig(9);
draw tree[1][1]((-1cm,2cm),(1cm,2cm))
                          ("$A$","$p$","$B$","$q$");
draw tree[2][1]((-0.5cm,2cm),(0.5cm,2cm))
                          ("$C$","$c$","$D$","$d$");
draw tree[2][2]((-0.5cm,2cm),(0.5cm,2cm))
                          ("$E$","$e$","$F$","$f$");
endfig;
```



Example 10

```
beginfig(10);
dirlabel:=90;
draw tree[1][1]((-1cm,2cm),(1cm,2cm))
                          ("$A$","$p$","$B$","$q$");
draw tree[2][1]((-0.5cm,2cm),(0.5cm,2cm))
                          ("$C$","$c$","$D$","$d$");
draw tree[2][2]((-0.5cm,2cm),(0.5cm,2cm))
                          ("$E$","$e$","$F$","$f$");
endfig;
```



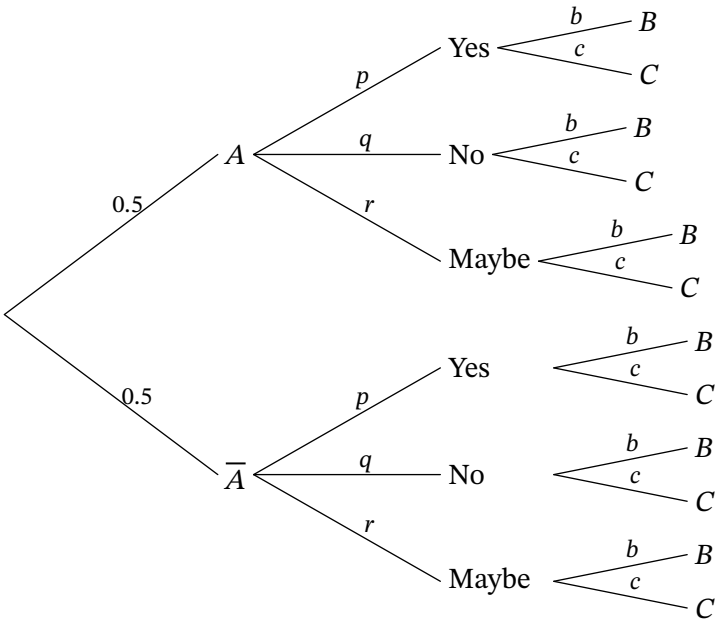
4 Dealing with alignment

`shiftev` — `numeric`, default : `-1`

The origin of each tree is located at the right side of the bounding box of the previous event name. Thus different subtrees may begin at different places. The numeric `shiftev`, if positive, indicates the fixed horizontal space between the end of the edges and the beginning of following subtrees. It can be used inside the first set of parameters of the tree (see example below) or as a global variable.

Example 11

```
beginfig(11);
 draw tree[1][1](80,120)("$A$","$0.5$","$\overline{A}$","$0.5$");
 draw tree[2][1](70,40)("Yes","$p$","No","$q$","Maybe","$r$");
 draw tree[2][2](70,40,"shiftev:=1.5cm")("Yes","$p$","No","$q$","Maybe","$r$");
 draw tree[3][1](50,20)("$B$","$b$","$C$","$c$");
 draw tree[3][2](50,20)("$B$","$b$","$C$","$c$");
 draw tree[3][3](50,20)("$B$","$b$","$C$","$c$");
 draw tree[3][4](50,20)("$B$","$b$","$C$","$c$");
 draw tree[3][5](50,20)("$B$","$b$","$C$","$c$");
 draw tree[3][6](50,20)("$B$","$b$","$C$","$c$");
endfig;
```

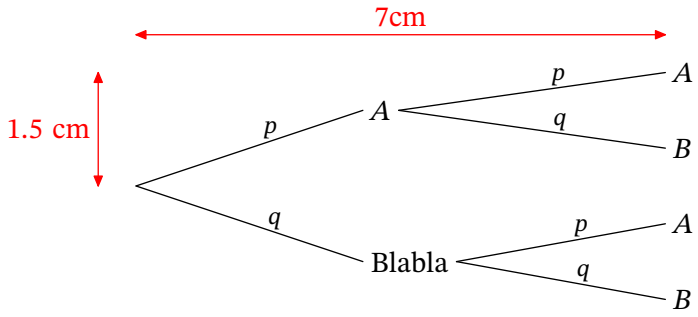


`abscoord` — `boolean`, default : `false`

With the boolean `abscoord` set to `true`, all the coordinates are given from the origin of the *first* tree instead of the origin of the subtree, which makes easier the alignment of all the subtrees.

Example 12

```
beginfig(12);
abscoord:=true;
draw tree[1][1](3cm,2cm)("$A$","$p$","Blabla","$q$");
draw tree[2][1]((7cm,1.5cm),(7cm,0.5cm))("$A$","$p$","$B$","$q$");
draw tree[2][2]((7cm,-0.5cm),(7cm,-1.5cm))("$A$","$p$","$B$","$q$");
endfig;
```



5 Parameters

All following parameters can be changed globally before drawing the tree or changed locally inside the first set of parameters:

```
scaleev:=2;
draw tree[1][1](3cm,2cm)(...);
draw tree[2][1](3cm,2cm)(...);
```

or

```
draw tree[1][1](3cm,2cm,"scaleev:=2")(...);
draw tree[2][1](3cm,2cm)(...);
```

In the fisrt case, `scaleev` is changed globally while in the second case, the change only applies to the first tree.

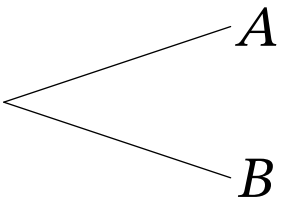
5.1 Event

`scaleev` `numeric, default : 1`

Numeric controlling the scale of the label at the end of the edge (the event).

Example 13

```
beginfig(13);
 scaleev:=2;
 draw stree[1][1](3cm,2cm)("$A$","$B$");
endfig;
```

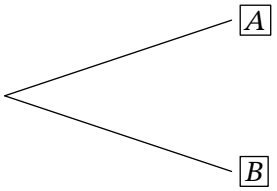


`nodeformat` — `string`, default : `""`

String that indicates how the events are printed (the shape of path around the event). Possible values are (for now) `"bbox"`, `"circle"`, `"superellipse"`.
See below for personalised values.

Example 14

```
beginfig(14);
 nodeformat:="bbox";
 draw stree[1][1](3cm,2cm)("$A$","$B$");
endfig;
```



`nodelinecolor` — `color`, default : `black`

Color of the path around the node

`nodebgcolor` — `color`, default : `white`

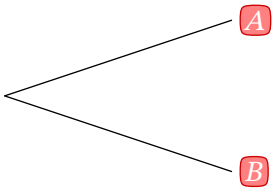
Color of the background of the region delimited by the previous path.

`nodefgcolor` — `color`, default : `black`

Color of the text.

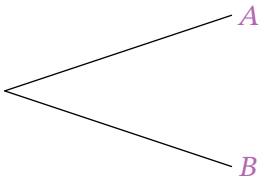
Example 15

```
beginfig(15);
nodeformat:="superellipse";
nodelinecolor:=(0.8,0,0);
nodebgcolor:=(1,0.5,0.5);
nodefgcolor:=white;
draw stree[1][1](3cm,2cm)("$A$","$B$");
endfig;
```



Example 16

```
beginfig(16);
nodefgcolor:=(0.7,0.4,0.7);
draw stree[1][1](3cm,2cm)("$A$","$B$");
endfig;
```

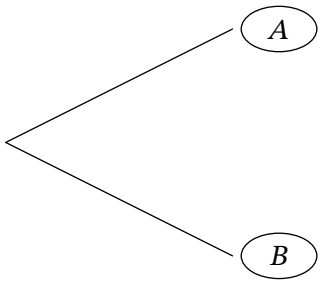


`nodeshape(expr A,B)` — `cycle path`

It is also possible to redefine the path defining the node. Simply rewrite the `nodeshape` macro, which must take a picture as parameter and whose result must be a cycle path.
The parameter passed to the macro is the figure representing the drawing of the event (normally the name of the event). You may use it, or not, in the macro definition.

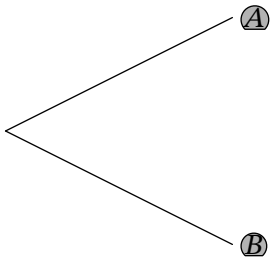
Example 17

```
beginfig(17);
vardef nodeshape(expr p)=
  fullcircle xscaled 1cm yscaled 0.6cm
enddef;

draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



Example 18

```
beginfig(18);
vardef nodeshape(expr p)=
  llcorner p..ulcorner p..urcorner p..lrcorner p--cycle
enddef;

nodebgcolor:=0.7white;
draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



5.2 Leaves

`begintree; endtree;`

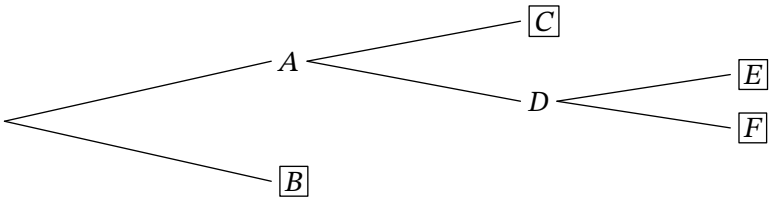
You may want to format the leaves in a different way from the nodes. A tree using the following parameters must be enclosed in a `begintree;`...`endtree;` "environment".

`leaveformat` `string,` default: `""`

String that indicates how the events are printed (the shape of path around the event). Possible values are (for now) `"bbox"`, `"circle"`, `"superellipse"` and `"none"`.

Example 19

```
beginfig(19);
begintree;
leaveformat:="bbox";
draw stree[1][1](100,45)("$A$","$B$");
draw stree[2][1](80,30)("$C$","$D$");
draw stree[3][2](65,20)("$E$","$F$");
endtree;
endfig;
```



`leavelinecolor` `color,` default: `black`

Color of the path around the leave

`leavebgcolor` `color`, default : `white`

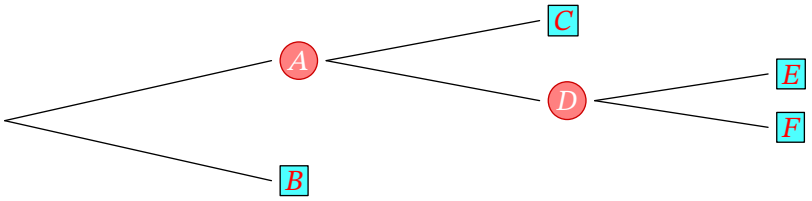
Color of the background of the region delimited by the previous path.

`leavefgcolor` `color`, default : `black`

Color of the text.

Example 20

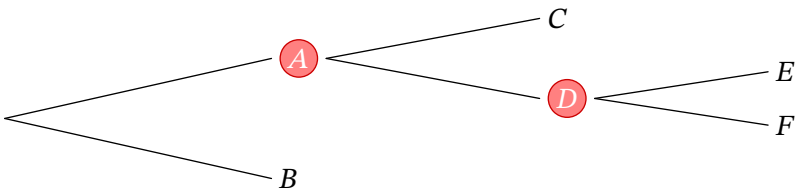
```
beginfig(20);
begintree;
nodeformat:="circle";
nodelinecolor:=(0.8,0,0); nodebgcolor:=(1,0.5,0.5); nodefgcolor:=white;
leaveformat:="bbox";
leavebgcolor:=(0.3,1,1); leavefgcolor:=red;
draw stree[1][1](100,45)("$A$","$B$");
draw stree[2][1](80,30)("$C$","$D$");
draw stree[3][2](65,20)("$E$","$F$");
endtree;
endfig;
```



Note that `nodeformat` applies to both nodes and leaves. To avoid formatting the leaves, use the value `"none"` for `leaveformat`.

Example 21

```
beginfig(21);
begintree;
nodeformat:="circle";
nodelinecolor:=(0.8,0,0); nodebgcolor:=(1,0.5,0.5); nodefgcolor:=white;
leaveformat:="none";
draw stree[1][1](100,45)("$A$","$B$");
draw stree[2][1](80,30)("$C$","$D$");
draw stree[3][2](65,20)("$E$","$F$");
endtree;
endfig;
```



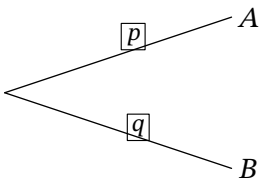
5.3 Probability

`probformat` `string`, default : `""`

String that indicates how the probabilities are printed (the shape of path around the probability). Possible values are (for now) `"bbox"`, `"circle"`, `"superellipse"`.

Example 22

```
beginfig(22);
 probformat:="bbox";
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```



`problinecolor` `color`, default : `black`

Color of the path around the probability

`probbgcolor` `color`, default : `white`

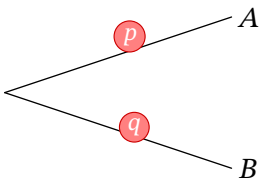
Color of the background of the region delimited by the previous path.

`probfgcolor` `color`, default : `black`

Color of the text.

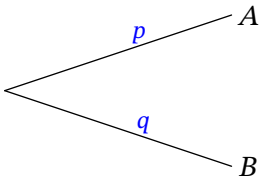
Example 23

```
beginfig(23);
probformat:="circle";
problinecolor:=(0.8,0,0);
probbgcolor:=(1,0.5,0.5);
probfgcolor:=white;
draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```



Example 24

```
beginfig(24);
probfgcolor:=blue;
draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

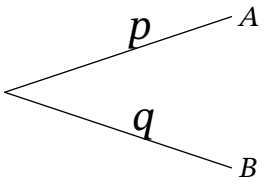


`scaleprob` `numeric`, default : `0.85`

Numeric controlling the scale of the label above the edge (the probability).

Example 25

```
beginfig(25);
 scaleprob:=1.5;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

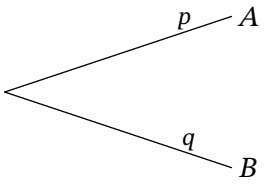


`posprob` `numeric, default : 0.6`

Numeric controlling the position of the label above the edge.

Example 26

```
beginfig(26);
 posprob:=0.8;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

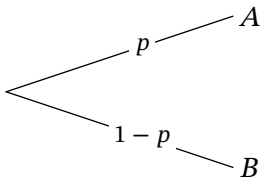


`typeprob` `numeric, default : 1`

Numeric controlling how the label is printed. Values can be 1 (the label is printed above the edge), 2 (the label is printed on the edge), 3 (the label is printed above the edge and rotated) or 4 (the label is printed on the edge and rotated).

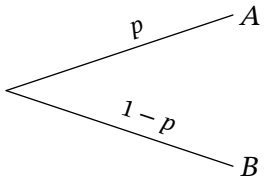
Example 27

```
beginfig(27);
 typeprob:=2;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$1-p$");
endfig;
```



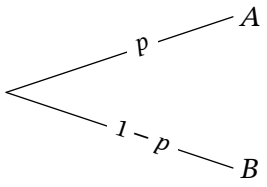
Example 28

```
beginfig(28);
 typeprob:=3;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$1-p$");
endfig;
```



Example 29

```
beginfig(29);
 typeprob:=4;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$1-p$");
endfig;
```

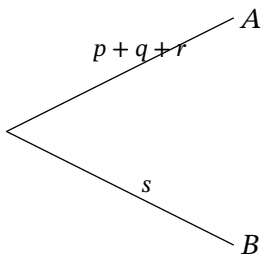


`proboffset` `numeric, default : 3bp`

Numeric controlling the amount by which the label above the edge is offset.

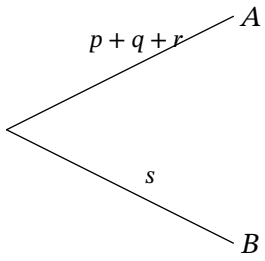
Example 30

```
beginfig(30);
 draw tree[1][1](3cm,3cm)("$A$","$p+q+r$","$B$","$s$");
endfig;
```



Example 31

```
beginfig(31);
 proboffset:=6bp;
 draw tree[1][1](3cm,3cm)("$A$","$p+q+r$","$B$","$s$");
endfig;
```



5.4 Edge

`linewidth` `numeric, default: 0.5bp`

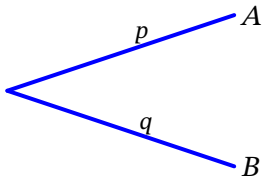
Width of the lines.

`linecolor` `color, default: black`

Color of the lines.

Example 32

```
beginfig(32);
 linewidth:=1.5;
 linecolor:=blue;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

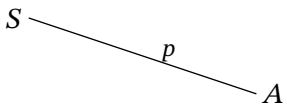


`endedgeshift` `numeric, default: 0`

Vertical space added at the end of the edge. Useful when various edges end at the same point.

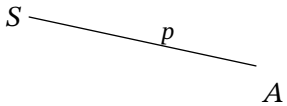
Example 33

```
beginfig(33);
 draw startlabel("$S$");
 draw tree[1][1]((3cm,-1cm))("$A$","$p$");
endfig;
```



Example 34

```
beginfig(34);
 endedgeshift:=10;
 draw startlabel("$S$");
 draw tree[1][1]((3cm,-1cm))("$A$","$p$");
endfig;
```

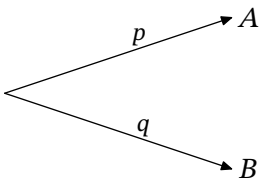


`edgearrow` — `boolean`, default : `false`

When the boolean `edgearrow` is set to true, edges end with an arrow.

Example 35

```
beginfig(35);
 edgearrow:=true;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```



`branchtype` — `string`, default : `"segment"`

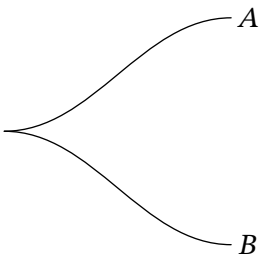
String which indicates the shape of the edge. Possible values are `segment`, `curve`, `broken`.
Note that double quotes have to be replaced by single quotes when this parameter is changed locally inside the `tree` macro.
See below for personalised values.

`tenscurve` — `numeric`, default : `0`

If string `branchtype` is set to `curve`, `tenscurve` indicates the "tension". When sets to `1`, the curve is a segment.

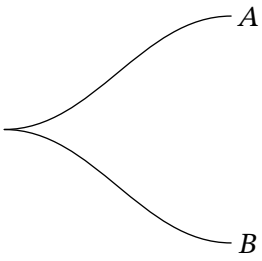
Example 36

```
beginfig(36);
 branchtype:="curve";
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



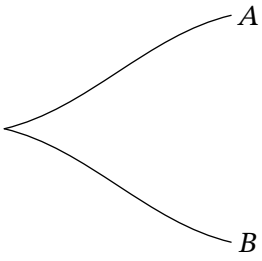
Example 37

```
beginfig(37);
 draw stree[1][1](3cm,3cm,"branchtype:='curve'")
                                  ("$A$","$B$");
endfig;
```



Example 38

```
beginfig(38);
 branchtype:="curve";
 tenscurve:=0.5;
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```

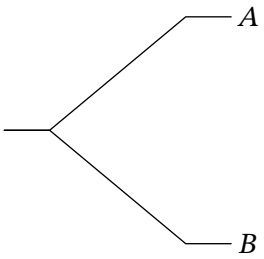


`brokenlineratio` `numeric, default : 0.2`

If string `branchtype` is set to `broken`, `brokenlineratio` indicates the ratio between the length of the first segment of the broken line and the total length of the horizontal space.

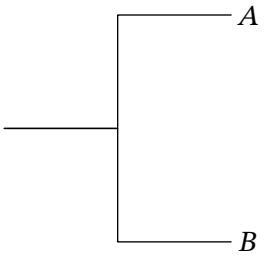
Example 39

```
beginfig(39);
 branchtype:="broken";
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



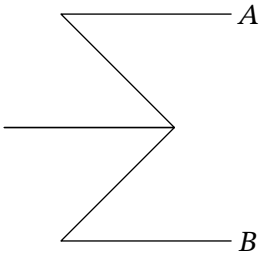
Example 40

```
beginfig(40);
 branchtype:="broken";
 posprob:=0.8;
 brokenlineratio:=0.5;
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



Example 41

```
beginfig(41);
 branchtype:="broken";
 posprob:=0.8;
 brokenlineratio:=0.75;
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```

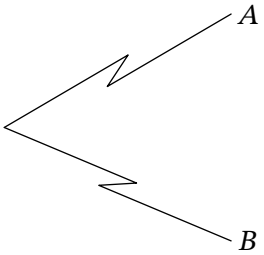


`edgeshape(expr A,B)` `path`

It is possible to completely redefine the path defining the edge. Simply rewrite the `edgeshape` macro, which must take two pairs as parameters and whose result must be a path.
The first parameter represents the start point and the second the end point.

Example 42

```
beginfig(42);
vardef edgeshape(expr S,E)=
 save a;numeric a;a=angle(E-S);
 S--((0.25cm,0) rotated (a+30) shifted 0.5[S,E])
  --((0.25cm,0) rotated (a-150) shifted 0.5[S,E])
  --E
enddef;

draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



6 Regular trees

6.1 Ordinary regular trees

`regulartree(<n>)(<l>,<h>)(<ev1>,<prob1>,<ev2>,<prob2>,...)` picture

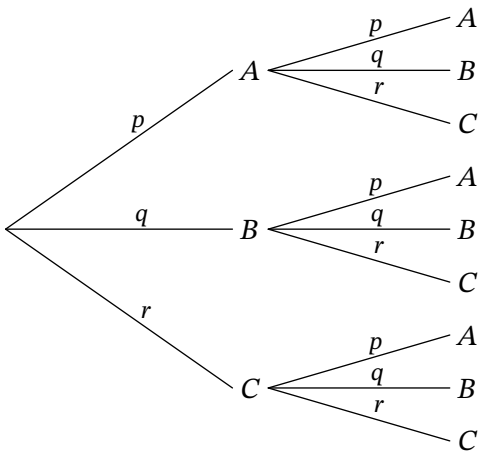
Tree describing the repetition of n identical and independent random experiments. `l` is the horizontal length of the first edges and `h` is the vertical space between two leaves.

`scalebranch` numeric, default : 0.8

Ratio between edges width of consecutive level.

Example 43

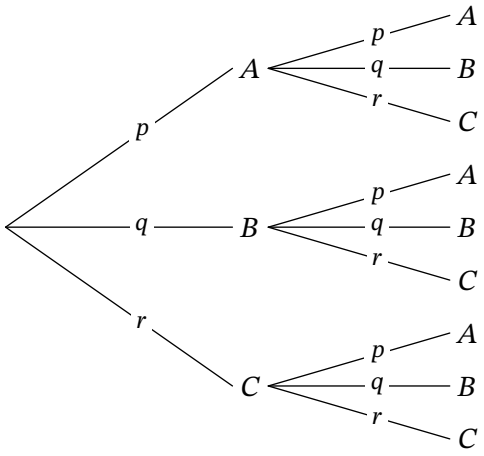
```
beginfig(43);
draw regulartree(2)(3cm,0.7cm)
            ("$A$","$p$","$B$","$q$","$C$","$r$");
endfig;
```



Note that you can change variable values inside the first set of parameters.

Example 44

```
beginfig(44);
draw regulartree(2)(3cm,0.7cm,"typeprob:=2")
            ("$A$","$p$","$B$","$q$","$C$","$r$");
endfig;
```



6.2 Binomial trees

`bernoulliprocess(<n>)(<l>,<h>)(<ev1>,<prob1>,<ev2>,<prob2>)` picture

Tree describing the Bernoulli process with n trials. `l` is the horizontal length of the first edges and `h` is the vertical space between two final nodes. If the last set of parameters is omitted, the values are set according to the following parameters.

`bernoulliprocessL(<n>)(<L>,<H>)(<ev1>,<prob1>,<ev2>,<prob2>)` picture

Same as above where `L` is the whole width of the tree and `H` its height.

Several parameters control the output:

`bernoullisuccessevent` `string`, default: `"$S$"`

String printed at every node representing a success.

`bernoullifailureevent` `string`, default: `"$\overline{S}$"`

String printed at every node representing a failure.

`bernoullisuccessprob` `string`, default: `"$p$"`

String printed above every edge representing a success.

`bernoullifailureprob` `string`, default: `"$q$"`

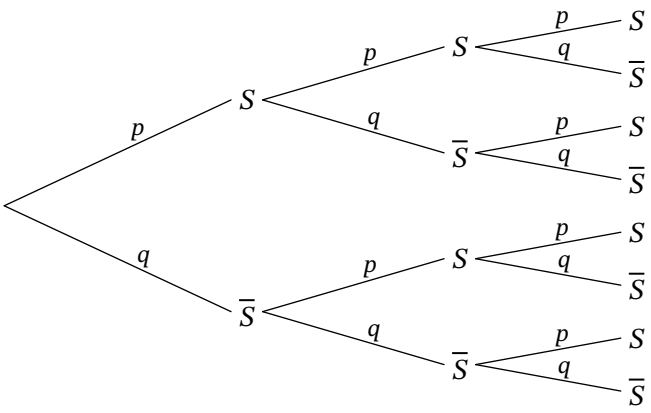
String printed above every edge representing a failure.

`bernoulliscalebranch` `numeric`, default: `0.8`

Ratio between width of consecutive edges.

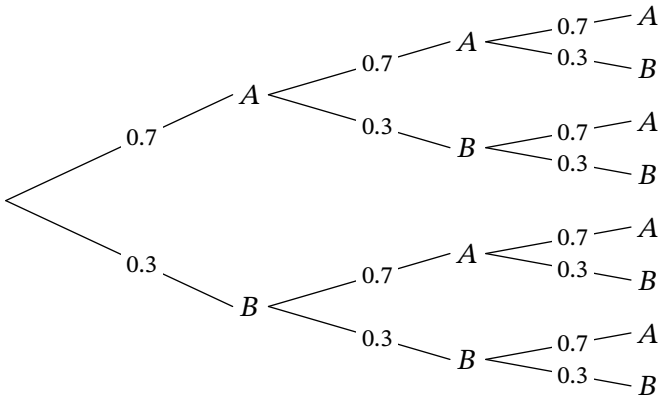
Example 45

```
beginfig(45);
draw bernoulliprocess(3)(3cm,0.7cm)();
endfig;
```



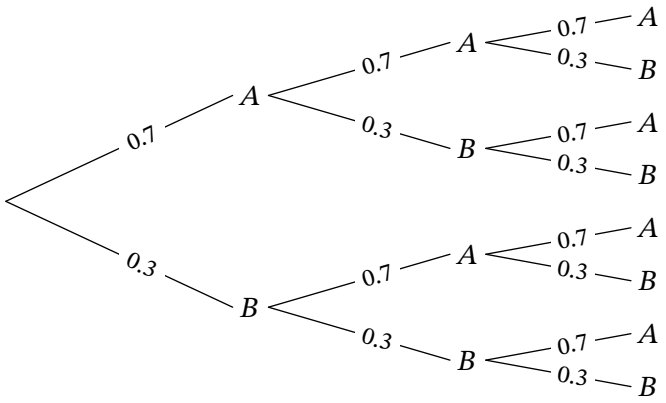
Example 46

```
beginfig(46);
draw bernoulliprocess(3)
        (3cm,0.7cm,"typeprob:=2;")
        ("$A$","$0.7$","$B$","$0.3$");
endfig;
```



Example 47

```
beginfig(47);
typeprob:=4;
bernoullisuccessevent:="$A$";
bernoullifailureevent:="$B$";
bernoullisuccessprob:="$0.7$";
bernoullifailureprob:="$0.3$";
draw bernoulliprocess(3)(3cm,0.7cm)();
endfig;
```



`binomialtree(<n>)(<l>,<h>)` picture

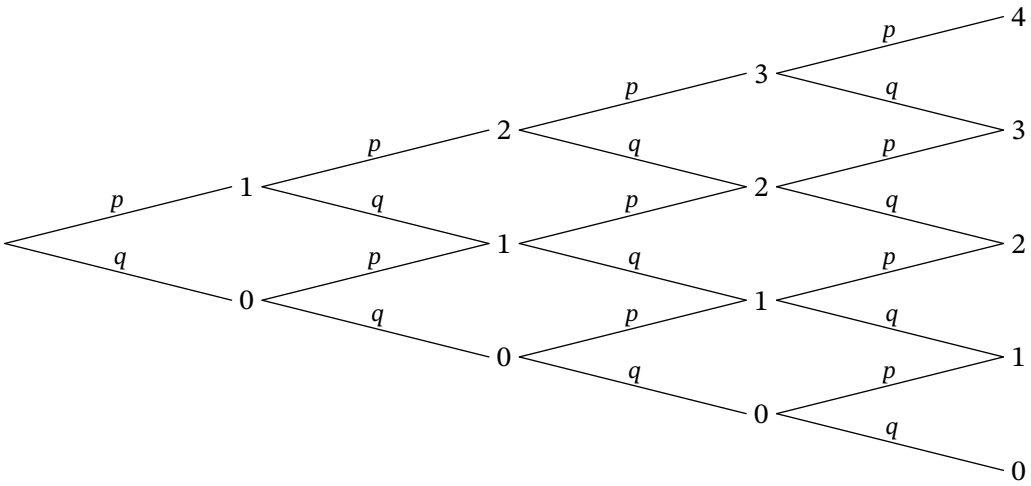
Tree describing the binomial distribution with n trials. `l` is the length of the first edges and `h` is the space between two final nodes. It uses `bernoullisuccesprob` and `bernoullifailureprob` but `bernoulliscalebranch` is set to 1.

`binomialtreeL(<n>)(<L>,<H>)` picture

Same as above where `L` is the whole width of the tree and `H` its height.

Example 48

```
beginfig(48);
draw binomialtree(4)(3cm,1.5cm);
endfig;
```



7 “Calculated” trees

`begintree;` `endtree;`

The following commands are experimental and need to be enclosed in a `begintree;...endtree;` “environment”.

`tree[<i>][<j>]()(<ev1>,<prob1>,<ev2>,<prob2>,...)` picture

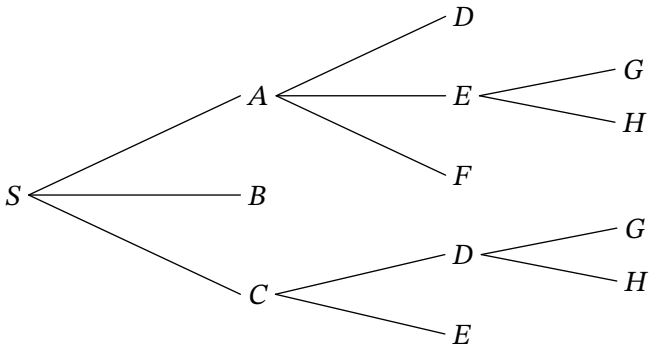
When the first set of parameters is left empty, the dimensions of the tree are calculated. The calculations use the parameters described below.

`stree[<i>][<j>]()(<ev1>,<ev2>,...)` `picture`

Same as above for “simple” trees.

Example 49

```
beginfig(49);
begintree;
draw startlabel("$S$");
draw stree[1][1]()("$A$","$B$","$C$");
draw stree[2][1]()("$D$","$E$","$F$");
draw stree[2][3]()("$D$","$E$");
draw stree[3][2]()("$G$","$H$");
draw stree[3][4]()("$G$","$H$");
endtree;
endfig;
```



`widthbranch` `numeric, default : 3.5cm`

Horizontal width of the first level tree.

`gapnode` `numeric, default : 0.7cm`

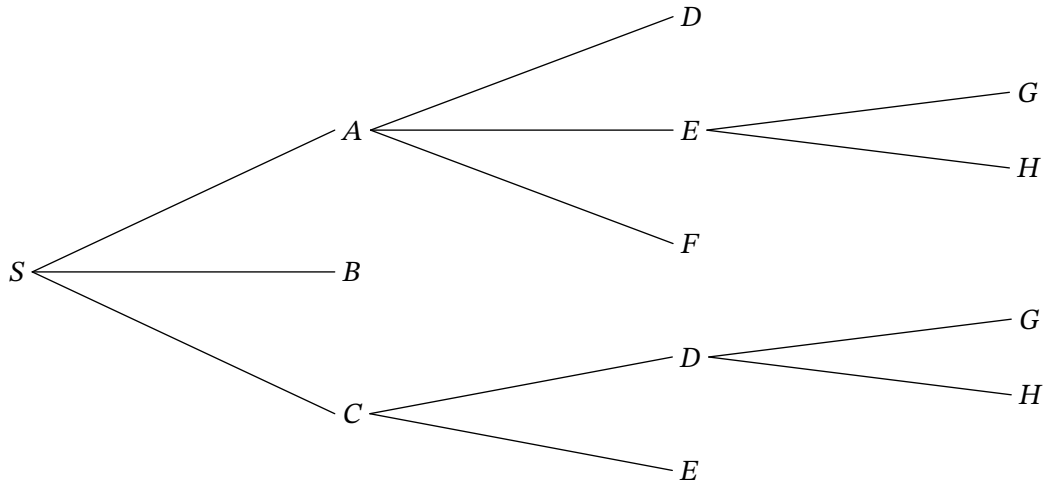
Minimal vertical space between two nodes of the last level trees.

`scalebranch` `numeric, default : 0.8`

Ratio between edges width of consecutive level.

Example 50

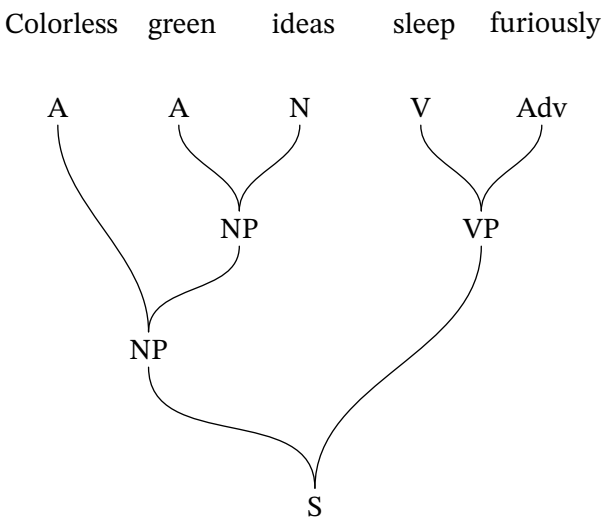
```
beginfig(50);
begintree;
widthbranch:=4cm;
scalebranch:=1;
gapnode:=1cm;
draw startlabel("$S$");
draw stree[1][1]()("$A$","$B$","$C$");
draw stree[2][1]()("$D$","$E$","$F$");
draw stree[2][3]()("$D$","$E$");
draw stree[3][2]()("$G$","$H$");
draw stree[3][4]()("$G$","$H$");
endtree;
endfig;
```



8 Examples

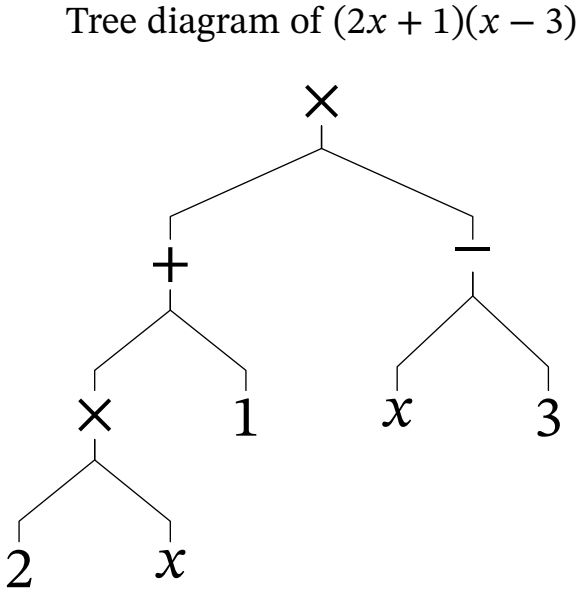
Example 51

```
beginfig(51);
u:=0.4cm;
branchtype:="curve";
dirlabel:=90;
abscoord:=true;
endlabelspace:=0.5cm;
draw startlabel("S");
draw stree[1][1]((-5.5u,4u),(5.5u,8u))("NP","VP");
draw stree[2][1]((-8.5u,12u),(-2.5u,8u))("A","NP");
draw stree[2][2]((3.5u,12u),(7.5u,12u))("V","Adv");
draw stree[3][2]((-4.5u,12u),(-0.5u,12u))("A","N");
draw endlabel[3][1]("Colorless");
draw endlabel[4][1]("green");
draw endlabel[4][2]("ideas");
draw endlabel[3][3]("sleep");
draw endlabel[3][4]("furiously");
endfig;
```



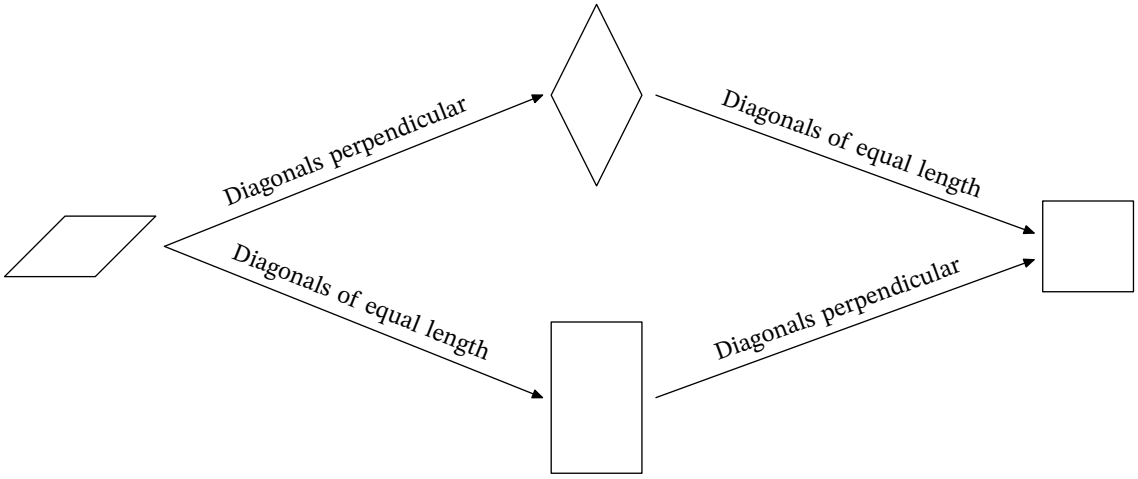
Example 52

```
beginfig(52);
u:=1cm;
branchtype:="broken";
dirlabel:=-90;
abscoord:=true;
scaleev:=2;
label.top(textext("\Large␣Tree␣diagram␣of␣$(2x+1)(x-3)$"),(0,1cm));
draw startlabel("$\times$");
draw stree[1][1]((-2u,-1.5u),(2u,-1.5u))("$+$","$-$");
draw stree[2][1]((-3u,-3.5u),(-1u,-3.5u))("$\times$","$1$");
draw stree[2][2]((1u,-3.5u),(3u,-3.5u))("$x$","$3$");
draw stree[3][1]((-4u,-5.5u),(-2u,-5.5u))("$2$","$x$");
endfig;
```



Example 53

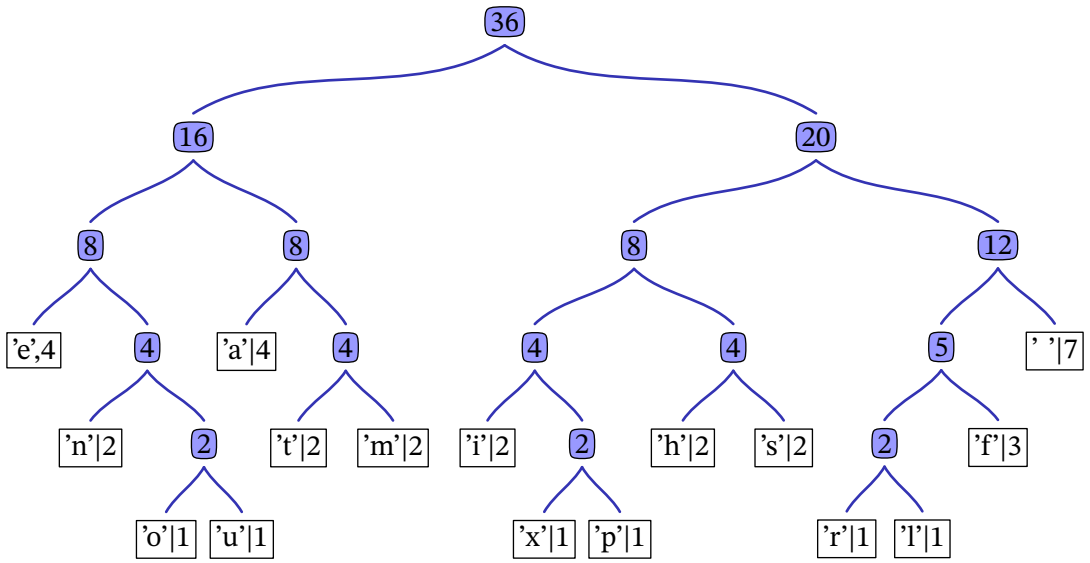
```
beginfig(53);
posprob:=0.5;
typeprob:=3;
shiftev:=1.5cm;
edgearrow:=true;
u:=0.2cm;
vardef paral = ((2,-2)--(6,2)--(0,2)--(-4,-2)--cycle) scaled u enddef;
vardef rhombus = ((3,0)--(0,6)--(-3,0)--(0,-6)--cycle) scaled u enddef;
vardef rectangle = ((3,5)--(-3,5)--(-3,-5)--(3,-5)--cycle) scaled u enddef;
vardef square = ((3,3)--(-3,3)--(-3,-3)--(3,-3)--cycle) scaled u enddef;
draw startlabel(paral);
draw tree[1][1](5cm,4cm)(rhombus,"Diagonals␣perpendicular",%
                        rectangle,"Diagonals␣of␣equal␣length");
endedgeshift:=5;
draw tree[2][1]((5cm,-2cm))("","Diagonals␣of␣equal␣length");
draw tree[2][2]((5cm,2cm))(square,"Diagonals␣perpendicular");
endfig;
```



Example 54

```
beginfig(54);
dirtree:=-90;
branchtype:="curve"; tenscurve:=0.75;
linewidth:=1;      linecolor:=(0.2,0.2,0.7);
widthbranch:=1cm;  scalebranch:=0.9;
gapnode:=1cm;
leaveformat:="bbox";
nodeformat:="superellipse"; nodebgcolor:=(0.6,0.6,1);
begintree;
label.top(textext("\Large␣Huffman␣tree␣(source␣Wikipedia)"),(0,1cm));
draw startlabel("36");
draw stree[1][1]()("20","16");
draw stree[2][1]()("12","8");
draw stree[2][2]()("8","8");
draw stree[3][1]()("'␣'|7","5");
draw stree[3][2]()("4","4");
draw stree[3][3]()("4","'a'|4");
draw stree[3][4]()("4","'e',4");
draw stree[4][2]()("'f'|3","2");
draw stree[4][3]()("'s'|2","'h'|2");
draw stree[4][4]()("2","'i'|2");
draw stree[4][5]()("'m'|2","'t'|2");
draw stree[4][7]()("2","'n'|2");
draw stree[5][2]()("'l'|1","'r'|1");
draw stree[5][5]()("'p'|1","'x'|1");
draw stree[5][9]()("'u'|1","'o'|1");
endtree;
endfig;
```

Huffman tree (source Wikipedia)



Part II
Graphs

9 Overview

This package can also be used to draw graphs. In this case, you need to define the nodes by their coordinates and indicate the edges to be drawn. Various commands allow you to draw graphs quickly, but it is also possible to finely control the drawing of each node and edge.

10 Nodes

10.1 Definition

`defnodes(<pair1>,<pair2>,...)`

The easiest way to define the nodes : just put a list of pairs. This command does not draw anything, it just defines an array of pairs corresponding to the coordinates. By default the array name is `A[]`.

`defnodename` `string,` default : `"A"`

Default node array name.

`nodename` `string,` default : `"array"`

String indicating how the nodes are stored and labels are drawns:

- with `nodename="array"`, nodes are stored in `A[]` and labelled `A1`, `A2`,...
- with `nodename="Alph"`, nodes are stored and labelled `A`, `B`,...
- with `nodename="alph"`, nodes are stored and labelled `a`, `b`,...
- with `nodename="arabic"`, nodes are stored in `A[]`,... but are labelled 1, 2,...

`defnode(<token>,<pair>)`

Nodes can be defined one by one. `<token>` is the node's name and `<pair>` its coordinates.

10.2 Drawing

`drawnodes.<pos>(<nodei>,<nodej>,...)`

Draw all the indicated nodes. These nodes can be designated either by name or by rank. If the list is empty, all the nodes are drawn. `pos` can be empty or `lft`, `rt`, etc. or an angle.

Example 55

```
beginfig(55);
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes(1,3);
endfig;
```

Example 56

```
beginfig(56);
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes(A1,A3);
endfig;
```

Example 57

```
beginfig(57);
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

Example 58

```
beginfig(58);
defnodename:="N";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

Example 59

```
beginfig(59);
nodename:="Alph";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

Example 60

```
beginfig(60);
nodename:="alph";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

Example 61

```
beginfig(61);
nodename:="arabic";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

`printnodename` `boolean, default: true`

When set to false, node names are not printed.

Example 62

```
beginfig(62);
printnodename:=false;
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

`nodeformat` `string,` default : `""`

String that indicates how the nodes are printed (the shape of path around the node). Possible values are (for now) `"bbox"`, `"circle"`, `"superellipse"` and `"square"`.
See below for personalised values.

`nodewidth` `numeric,` default : `0.6cm`

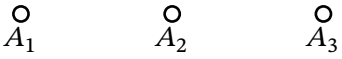
Node width...

Example 63

```
beginfig(63);
nodeformat:="square";
nodewidth:=1cm;
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

Example 64

```
beginfig(64);
nodewidth:=0.2cm;
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes.bot();
endfig;
```



`naturalwidth` `boolean,` default : `false`

When set to true, the width of the node adapts to the content.

Example 65

```
beginfig(65);
defnodename:="AA";
naturalwidth:=true;
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

`nodelinewidth` `numeric,` default : `1`

Line width of the path around the node.

`nodelinecolor` `color, default : black`

Color of the path around the node.

`nodebgcolor` `color, default : white`

Color of the background of the region delimited by the previous path.

`nodefgcolor` `color, default : black`

Color of the text.

Example 66

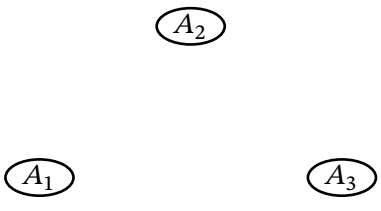
```
beginfig(66);
nodelinecolor:=(0.8,0,0);
nodebgcolor:=(1,0.5,0.5);
nodefgcolor:=white;
nodeformat:="superellipse";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes();
endfig;
```

`nodeshape(expr A,B)` `(cycle) path`

In the same way as for trees, it is also possible to redefine the path defining the node. Simply rewrite the `nodeshape` macro, which must take a picture as parameter and whose result must be a cycle path.
The parameter passed to the macro is the figure representing the drawing of the event (normally the name of the event). You may use it, or not, in the macro definition.
Note that in the path definition, the coordinates $(0, 0)$ correspond to the node coordinates.

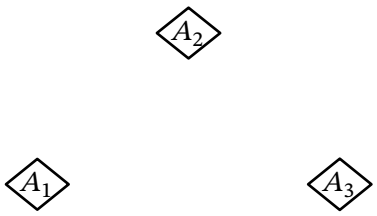
Example 67

```
beginfig(67);
vardef nodeshape(expr p)=
  fullcircle xscaled 1.5nodewidth yscaled 0.8nodewidth
enddef;

defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
endfig;
```



Example 68

```
beginfig(68);
vardef nodeshape(expr p)=
  pair c,h,v;
  h:=lrcorner p - llcorner p;
  v:=ulcorner p - llcorner p;
  c:=center p;
  (c-h -- c+v -- c+h -- c-v -- cycle) shifted -c
enddef;

defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
endfig;
```



`drawnodes(<nodes>) withbgcolor <color>`

While `withcolor` applies to lines and labels, `withbgcolor` applies to the filling of nodes.

Example 69

```
beginfig(69);
nodelinecolor:=(0.8,0,0);
nodebgcolor:=(1,0.5,0.5);
nodefgcolor:=white;
nodeformat:="superellipse";
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnodes() withcolor black withbgcolor (0.6,0.6,1);
endfig;
```

`drawnode.<pos>(<node>,<label>)`

The nodes can be drawn one by one, specifying the text to be displayed if necessary.

Example 70

```
beginfig(70);
defnodes((0cm,0cm),(2cm,0),(4cm,0));
drawnode[60](1);
drawnode(2,"$\sqrt{2}$") withbgcolor 0.7white;
drawnode.bot(A3,"Node");
endfig;
```

11 Edges

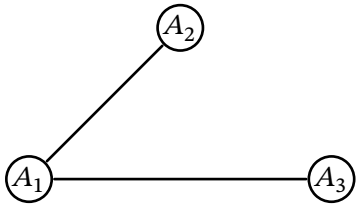
11.1 Undirected edges

`drawedges.<pos>(<(Na,Nb)>,<string1>,<(Nc,Nd)>,<string2>,...)(<angle1>,<angle2>)`

Draw undirected edges between Node Na and node Nb, between node Nc and node Nd,... Na, Nb, Nc, Nd... must be integers. You can't refer to the nodes with their name.
If present, the string is printed at the position indicated in relation to the middle of the edge (by default).
By default, the edge is a segment. If an angle is specified, it is added to the angle at the start of the edge and subtracted from the angle at the end of the edge. If two angles are specified, they are added to the start and end angles respectively.

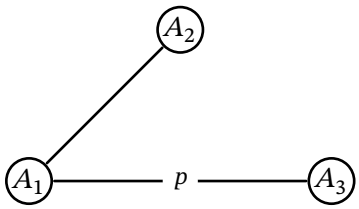
Example 71

```
beginfig(71);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1))();
endfig;
```



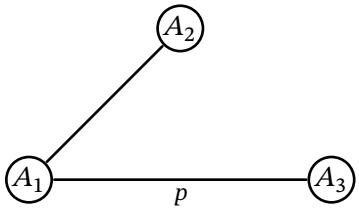
Example 72

```
beginfig(72);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1),"$p$")();
endfig;
```



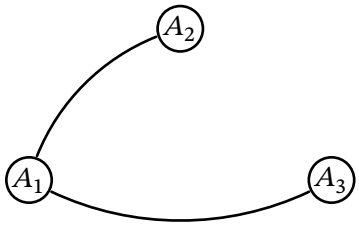
Example 73

```
beginfig(73);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges.bot((1,2),(3,1),"$p$")();
endfig;
```



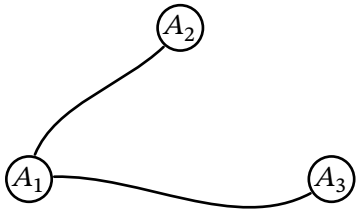
Example 74

```
beginfig(74);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1))(30);
endfig;
```



Example 75

```
beginfig(75);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1))(40,10);
endfig;
```

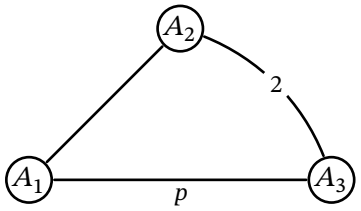


`drawedge.<pos>(<Na>,<Nb>)(<angle1>,<angle2>,<string>)`

The edges can be drawn one by one.In this case, Na, Nb can be the node name. The label or weight must be indicated in the second set of parameters.

Example 76

```
beginfig(76);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedge(1,2)();
drawedge.bot(1,3)("$p$");
drawedge(2,3)(30,"2");
endfig;
```



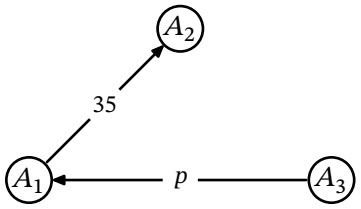
11.2 Directed edges

`drawdiredges.<pos>(<(Na,Nb)>,<string1>,<(Nc,Nd)>,<string2>,...)(<angle1>,<angle2>)`

Draw directed edges from Node Na to node Nb, from node Nc to node Nd,... Na, Nb, Nc, Nd... must be integers. You can't refer to the nodes with their name.
If present, the string is printed at the position indicated in relation to the middle of the edge (by default).
By default, the edge is a segment. If an angle is specified, it is added to the angle at the start of the edge and subtracted from the angle at the end of the edge. If two angles are specified, they are added to the start and end angles respectively.

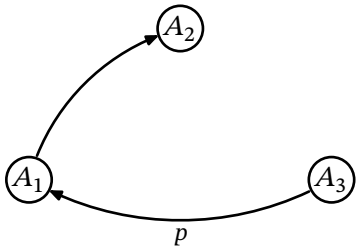
Example 77

```
beginfig(77);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawdiredges((1,2),"35",(3,1),"$p$")();
endfig;
```



Example 78

```
beginfig(78);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawdiredges.bot((1,2),(3,1),"$p$")(30);
endfig;
```

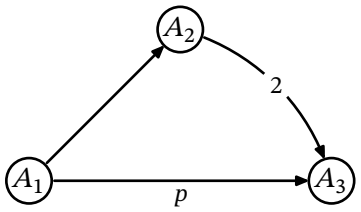


`drawdiredge.<pos>(<Na>,<Nb>)(<angle1>,<angle2>,<string>)`

The directed edges can be drawn one by one. In this case, Na, Nb can be the node name. The label or weight must be indicated in the second set of parameters.

Example 79

```
beginfig(79);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawdiredge(1,2)();
drawdiredge.bot(1,3)("$p$");
drawdiredge(2,3)(30,"2");
endfig;
```



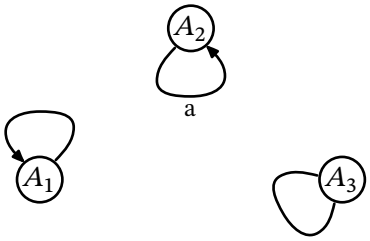
11.3 Loops

`draw(dir)edge.<pos>(<Na>,<Na>)(<angle1>,<angle2>,<string>)`

Previous commands can be used to draw loops. If present, `angle1` indicates the angle of the loop with horizontal direction. Default value is 90.

Example 80

```
beginfig(80);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawdiredge(1,1)();
drawedge(3,3)(-150);
drawdiredge.bot(2,2)(-90,"a");
endfig;
```



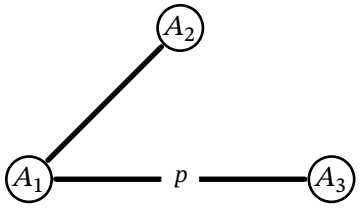
11.4 Parameters

`edgelinewidth` `numeric, default : 1`

Width of the edge.

Example 81

```
beginfig(81);
edgelinewidth:=2;
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1),"$p$")();
endfig;
```

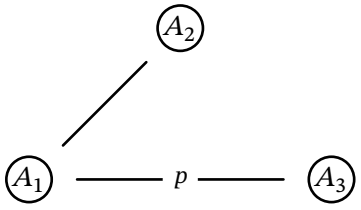


`nodeedgeoffset` `numeric, default : 0`

Space between the node and the edge.

Example 82

```
beginfig(82);
nodeedgeoffset:=0.3cm;
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1),"$p$")();
endfig;
```



`startedgeshift` `numeric, default : 0`

Space added at the beginning of the edge.

`endedgeshift` `numeric, default : 0`

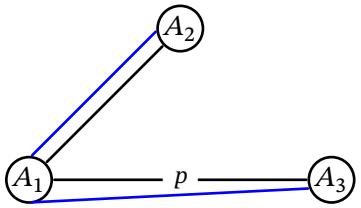
Space added at the end of the edge.

`edgeshift` `numeric, default : 0`

Space added at both start and end of the edge.

Example 83

```
beginfig(83);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(1,3),"$p$")();
startedgeshift:=-0.3cm;endedgeshift:=-0.1cm;
drawedges((1,3))() withcolor blue;
edgeshift:=0.2cm;
drawedges((1,2))() withcolor blue;
endfig;
```



`probformat` `string, default : ""`

String that indicates how the weights are printed (the shape of the path). Possible values are (for now) `"bbox"`, `"circle"`, `"superellipse"` and `"square"`.

`problinecolor` `color, default : black`

Color of the path around the weight.

`probgcolor` `color, default : white`

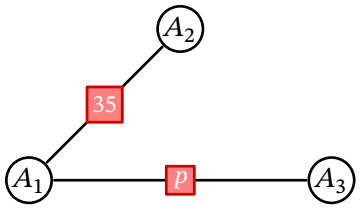
Color of the background of the region delimited by the previous path.

`probfgcolor` `color, default : black`

Color of the text.

Example 84

```
beginfig(84);
problinecolor:=(0.8,0,0);
probbgcolor:=(1,0.5,0.5);
probfgcolor:=white;
probformat:="square";
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),"35",(3,1),"$p$")();
endfig;
```

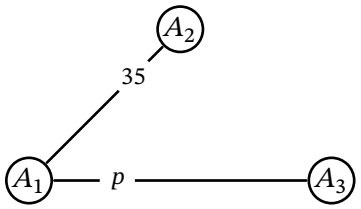


`posprob` `numeric, default : 0.5`

Position of the weight on the edge.

Example 85

```
beginfig(85);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
posprob:=0.75;
drawnodes();
drawedges((1,2),"35",(3,1),"$p$")();
endfig;
```

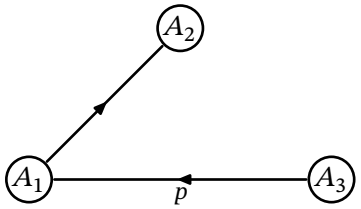


`edgearrowpos` `numeric, default : 1`

Position of arrowhead on the edge.

Example 86

```
beginfig(86);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
edgearrowpos:=0.5;
drawnodes();
drawdiredges.bot((1,2),(3,1),"$p$")();
endfig;
```

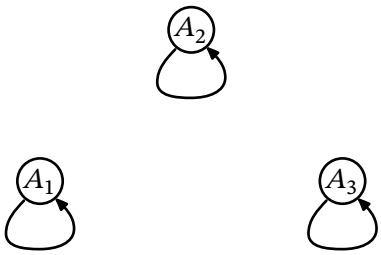


`loopangle` `numeric, default : 90`

Loop direction. It can be specified in the command parameters.

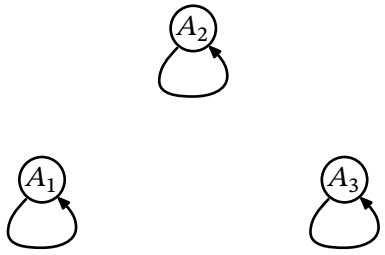
Example 87

```
beginfig(87);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
loopangle:=-90;
drawnodes();
drawdiredges((1,1),(2,2),(3,3))();
endfig;
```



Example 88

```
beginfig(88);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawdiredges((1,1),(2,2),(3,3))(-90);
endfig;
```

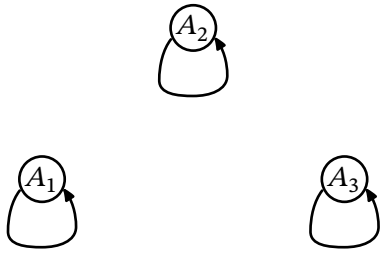


`loopstartangle` `numeric, default : 30`

Angle between the path direction at node center and loop direction.

Example 89

```
beginfig(89);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
loopstartangle:=90;
drawnodes();
drawdiredges((1,1),(2,2),(3,3))(-90);
endfig;
```



`loopsize` `numeric, default : 1.5nodesize`

Loop size.

Example 90

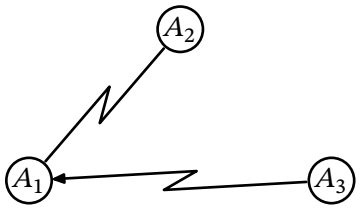
```
beginfig(90);
defnodes((0cm,0cm),(2cm,0cm),(4cm,0));
loopsize:=1.5cm;
drawnodes();
drawdiredges((1,1),(2,2),(3,3))();
endfig;
```

`edgeshape(expr A,B)` `path`

In the same way as for trees, it is also possible to completely redefine the path defining the edge. Simply rewrite the `edgeshape` macro, which must take two pairs as parameters and whose result must be a path.
The first parameter represents the start point and the second the end point.

Example 91

```
beginfig(91);
vardef edgeshape(expr A,B)=
 save a;numeric a;a=angle(B-A);
 A--((0.25cm,0) rotated (a+30) shifted 0.5[A,B])
  --((0.25cm,0) rotated (a-150) shifted 0.5[A,B])
  --B
enddef;

defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedge(1,2)();
drawdiredge(3,1)();
endfig;
```



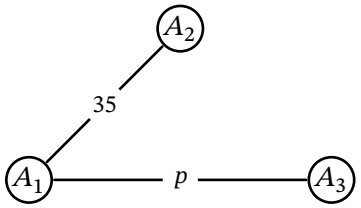
12 Complete graphs

`draw(dir)graph.<pos>(<(Na,Nb)>,<string1>,<(Nc,Nd)>,<string2>,...)(<angle1>,<angle2>)`

These commands are shortcuts for drawing both nodes and edges.

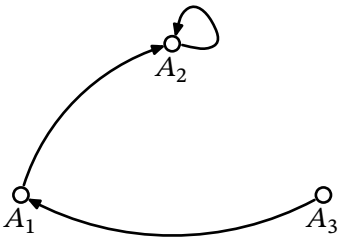
Example 92

```
beginfig(92);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawgraph((1,2),"35",(3,1),"$p$")();
endfig;
```



Example 93

```
beginfig(93);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
nodewidth:=0.2cm;
drawdirgraph.bot((1,2),(3,1),(2,2))(30);
endfig;
```



13 Grids

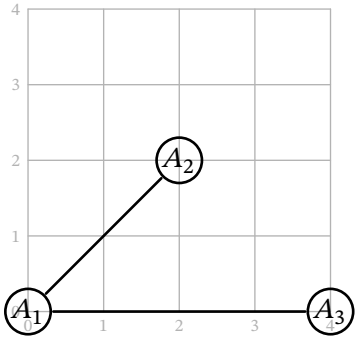
To facilitate node placement, use the following command to draw a grid.

`drawgrid(<xmin>,<ymin>,<xmax>,<ymax>)(<unit>,<step>)`

Command for drawing a grid. Minimum values are optional and equal to 0 by default. The unit and step values are also optional and default to 1 cm and 1 respectively.

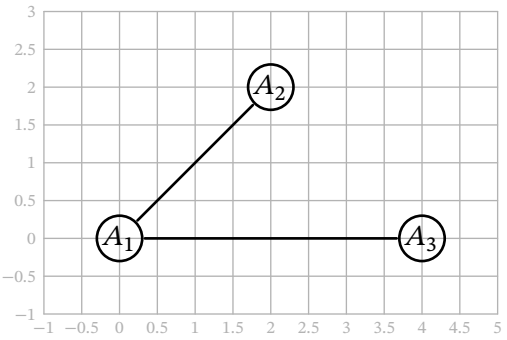
Example 94

```
beginfig(94);
drawgrid(4,4)();
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1))();
endfig;
```



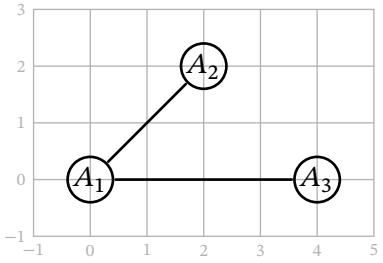
Example 95

```
beginfig(95);
drawgrid(-1,-1,5,3)(1cm,0.5);
defnodes((0cm,0cm),(2cm,2cm),(4cm,0));
drawnodes();
drawedges((1,2),(3,1))();
endfig;
```



Example 96

```
beginfig(96);
u:=0.75cm;
drawgrid(-1,-1,5,3)(u);
defnodes((0,0),(2u,2u),(4u,0));
drawnodes();
drawedges((1,2),(3,1))();
endfig;
```

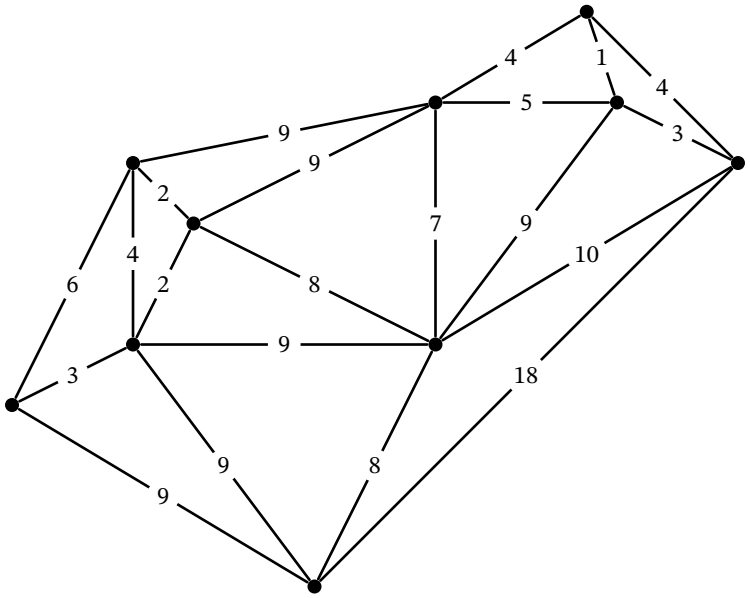


14 Examples

Weighted graph

Example 97

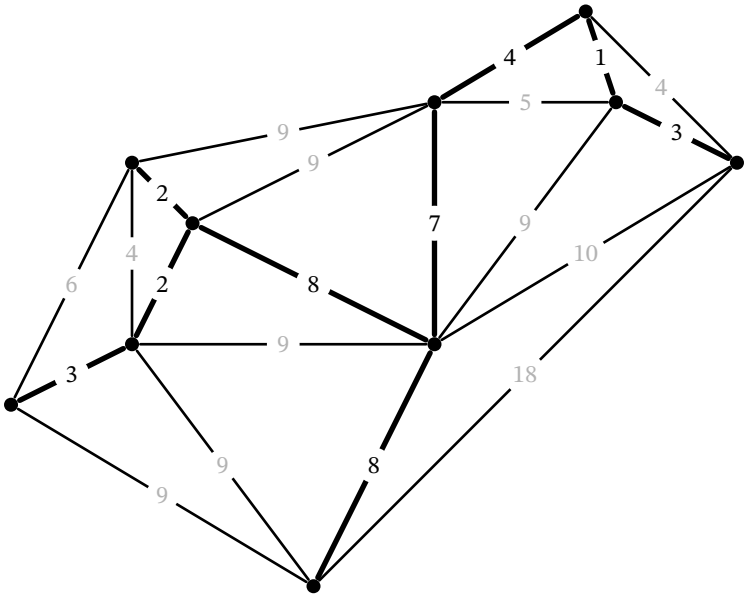
```
beginfig(97);
u:=0.8cm;
nodewidth:=0.15cm;
nodebgcolor:=black;
printnodename:=false;
defnodes((5u,0),(0,3u),(2u,4u),(7u,4u),(3u,6u),(2u,7u),(12u,7u),(7u,8u),
                                                        (10u,8u),(9.5u,9.5u));
drawnodes();
drawedges((1,2),"9",(1,3),"9",(1,4),"8",(1,7),"18")();
drawedges((2,3),"3",(2,6),"6")();
drawedges((3,4),"9",(3,5),"2",(3,6),"4")();
drawedges((4,5),"8",(4,7),"10",(4,8),"7",(4,9),"9")();
drawedges((5,6),"2",(5,8),"9")();
drawedges((6,8),"9")();
drawedges((7,9),"3",(7,10),"4")();
drawedges((8,9),"5",(8,10),"4")();
drawedges((9,10),"1")();
endfig;
```



Minimum spanning tree

Example 98

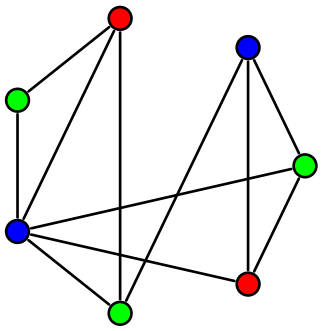
```
beginfig(98);
u:=0.8cm;
nodewidth:=0.15cm;
nodebgcolor:=black;
printnodename:=false;
defnodes((5u,0),(0,3u),(2u,4u),(7u,4u),(3u,6u),(2u,7u),(12u,7u),(7u,8u),
                                                          (10u,8u),(9.5u,9.5u));
drawnodes();
drawoptions(withcolor 0.7white);
drawedges((1,2),"9",(1,3),"9",(1,7),"18")();
drawedges((2,6),"6")();
drawedges((3,4),"9",(3,6),"4")();
drawedges((4,7),"10",(4,9),"9")();
drawedges((5,8),"9")();
drawedges((6,8),"9")();
drawedges((7,10),"4")();
drawedges((8,9),"5")();
drawoptions(withcolor black);
edgelinewidth:=2;
drawedges((1,4),"8",(2,3),"3",(3,5),"2",(4,5),"8",(5,6),"2",(4,8),"7",
                                          (8,10),"4",(9,10),"1",(7,9),"3")();
endfig;
```



Graph colouring

Example 99

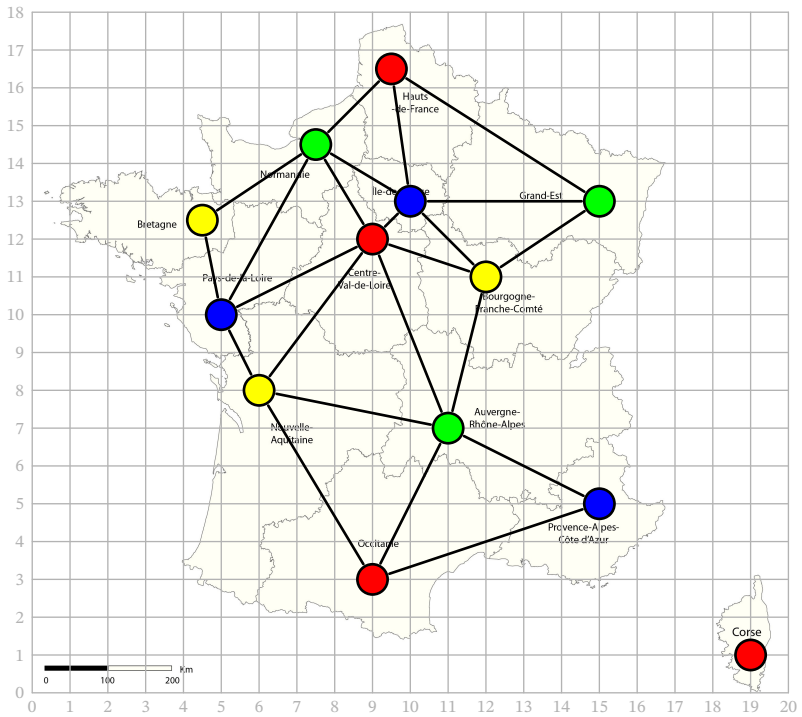
```
beginfig(99);
printnodename:=false;
nodewidth:=0.3cm;
for i=1 upto 7:
  defnodes(((2cm,0) rotated (360*i/7)));
endfor
nodebgcolor:=red;
drawnodes(2,6);
nodebgcolor:=blue;
drawnodes(1,4);
nodebgcolor:=green;
drawnodes(3,5,7);
drawedges((1,5),(1,6),(1,7),(2,3),(2,4),(2,5),
          (3,4),(4,5),(4,6),(4,7),(6,7))();
endfig;
```



With LuaLaTeX and `luamplib` package you can use `\includegraphics` inside a METAPOST code:

Example 100

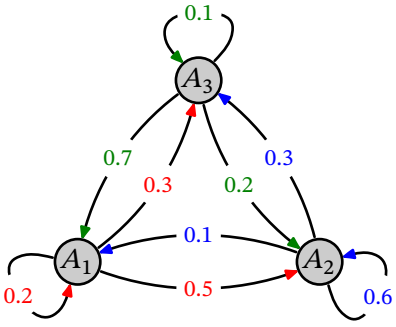
```
beginfig(100);
u:=0.5cm;
nodewidth:=0.4cm;
nodename:="arabic";
printnodename:=false;
draw btex \includegraphics[width=10cm]{france-region.jpg} etex;
% picture from https://capcarto.fr/telechargements/
draw grid(20,18)(u);
defnodes((4.5u,12.5u),(7.5u,14.5u),(5u,10u),(9.5u,16.5u),(10u,13u),
         (15u,13u),(9u,12u),(12u,11u),(6u,8u),(11u,7u),(9u,3u),
         (15u,5u),(19u,1u));
nodebgcolor:=red;
drawnodes(7,4,11,13);
nodebgcolor:=green;
drawnodes(2,10,6);
nodebgcolor:=blue;
drawnodes(5,3,12);
nodebgcolor:=(1,1,0);
drawnodes(1,8,9);
drawedges((1,2),(1,3),(2,3),(2,4),(2,5),(2,7),(3,7),(3,9),(4,5),
         (4,6),(5,6),(5,7),(5,8),(6,8),(7,8),(7,9),(7,10),(8,10),
         (9,10),(9,11),(10,11),(10,12),(11,12))();
endfig;
```



Markov chain

Example 101

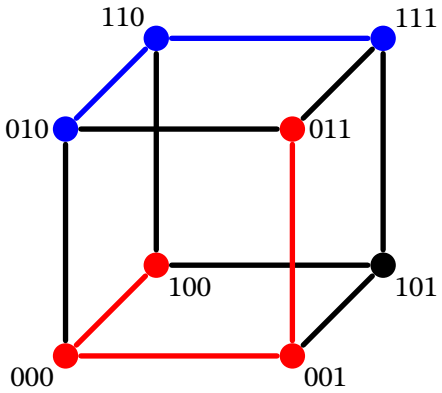
```
beginfig(101);
scaleprob:=0.8;
u:=1.6cm;
nodebgcolor:=0.8white;
defnodes((0,0),(2u,0),(u,1.5u));
drawnodes();
drawoptions(withcolor red);
drawdiredges((1,2),"$0.5$",(1,3),"$0.3$")(-25);
drawdiredge(1,1)("$0.2$",210);
drawoptions(withcolor blue);
drawdiredges((2,1),"$0.1$",(2,3),"$0.3$")(-25);
drawdiredge(2,2)("$0.6$",-30);
drawoptions(withcolor 0.5green);
drawdiredges((3,1),"$0.7$",(3,2),"$0.2$")(-25);
drawdiredge(3,3)("$0.1$");
endfig;
```



Hypercube Q_3

Example 102

```
beginfig(102);
u:=3cm;
nodewidth:=0.3cm;
edgelinewidth:=2;
pair p; p:=(0.4u,0.4u);
defnodes((0,0),(u,0),(0,u),(u,u));
defnodes((0,0)+p,(u,0)+p,(0,u)+p,(u,u)+p);
nodebgcolor:=black;nodelinecolor:=black;
drawnode.lrt(6,"101");
nodebgcolor:=red;nodelinecolor:=red;
drawnode.llft(1,"000");drawnode.lrt(2,"001");
drawnode.rt(4,"011");drawnode.lrt(5,"100");
nodebgcolor:=blue;nodelinecolor:=blue;
drawnode.lft(3,"010");drawnode.ulft(7,"110");
drawnode.urt(8,"111");
drawedges((1,3),(3,4),(2,6),(4,8),(6,8),(5,6),(5,7))();
drawedges((3,7),(7,8))() withcolor blue;
drawedges((1,2),(1,5),(2,4))() withcolor red;
endfig;
```

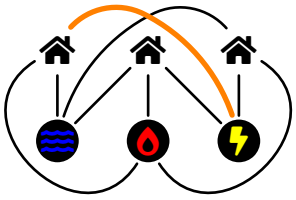


Three utilities problem

With LuaLaTeX, `luamplib` and `fontawesome5` packages.

Example 103

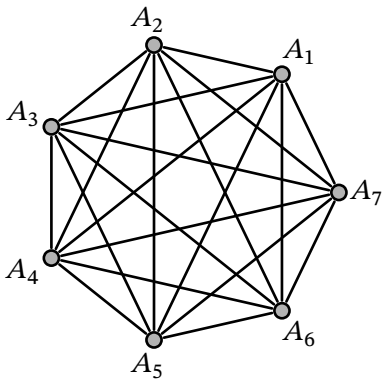
```
beginfig(103);
u:=1.2cm;
defnodes((u,2u),(2u,2u),(3u,2u),(u,u),(2u,u),(3u,u));
nodelinecolor:=white;
drawnode(1,"\large␣\faHome");
drawnode(2,"\large␣\faHome");
drawnode(3,"\large␣\faHome");
nodebgcolor:=black;nodefgcolor:=blue;
drawnode(4,"\faWater");
nodefgcolor:=red;drawnode(5,"\faBurn");
nodefgcolor:=(1,1,0);drawnode(6,"\faBolt");
drawedges((1,4),(2,4),(2,5),(2,6),(3,6))();
drawedges((1,5),(5,3))(-115);
vardef edgeshape(expr A,B)=
save a;numeric a;a=angle(B-A);s:=if A>B: - fi 1;
A..(0.7[A,B]+(0.35(B-A) rotated (s*90))){dir a}..B
enddef;
drawedges((4,3))();
edgelinewidth:=2;drawedges((6,1))() withcolor (1,0.5,0);
endfig;
```



Various

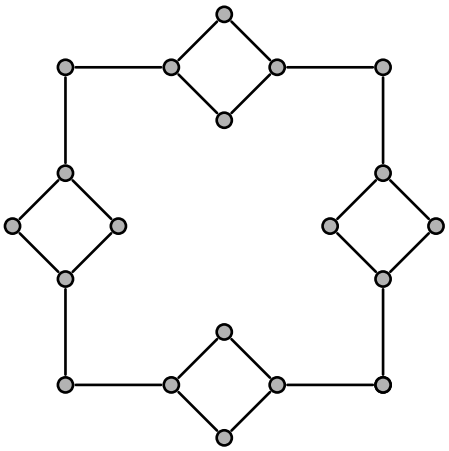
Example 104

```
beginfig(104);
nodebgcolor:=0.7white;
nodewidth:=0.2cm;

for i=1 upto 7:
  defnode(A[i],(2cm,0) rotated (360*i/7));
  drawnode[360*i/7](i);
  for j=1 upto i-1:
    drawedge(i,j)();
  endfor
endfor
endfig;
```



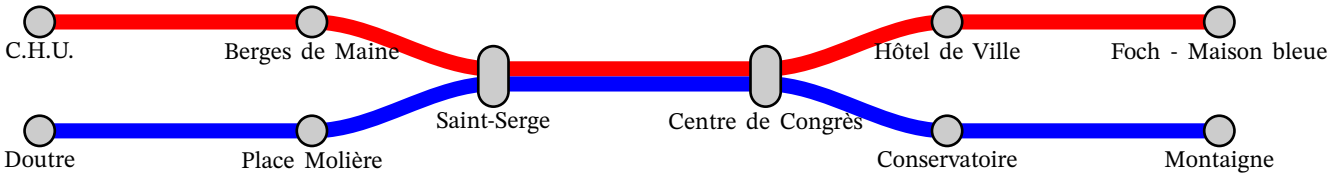
Example 105

```
beginfig(105);
nodebgcolor:=0.7white;
nodewidth:=0.2cm;
printnodename:=false;

u:=0.7cm;
defnodes((3u,-3u));
drawnode(1);
for i=1 upto 4:
 for j=1 upto 4:
  defnodes((0,-2u) rotatedaround((0,-3u),90j)
                   rotated 90i);
 endfor
 defnodes((3u,-3u) rotated (90i));
 drawnodes(5i-3,5i-2,5i-1,5i,5i+1);
 drawedges((5i-4,5i-3),(5i-3,5i-2),(5i-2,5i-1),
           (5i-1,5i),(5i,5i-3),(5i-1,5i+1))();
endfor;
endfig;
```



Example 106

```
beginfig(106);
u:=1.2cm;
nodewidth:=0.4cm;
defnodes((0,0.6u),(3u,0.6u),(5u,0),(8u,0),(10u,0.6u),(13u,0.6u),(0,-0.6u),
                                        (3u,-0.6u),(10u,-0.6u),(13u,-0.6u));
nodeedgeoffset:=-0.2cm;
nodebgcolor:=0.8white;
edgelinewidth:=0.2cm;
vardef edgeshape (expr A,B)=A{dir 0}..{dir 0}B enddef;
drawedges((1,2))() withcolor red;
drawedges((5,6))() withcolor red;
drawedges((7,8))() withcolor blue;
drawedges((9,10))() withcolor blue;
endedgeshift:=0.1cm; drawedges((2,3))() withcolor red;
endedgeshift:=-0.1cm; drawedges((8,3))() withcolor blue;
endedgeshift:=0cm;
startedgeshift:=0.1cm; drawedges((4,5))() withcolor red;
startedgeshift:=-0.1cm;drawedges((4,9))() withcolor blue;
edgeshift:=0.1cm;   drawedges((3,4))() withcolor red;
edgeshift:=-0.1cm;  drawedges((3,4))() withcolor blue;
drawnode.bot(1,"C.H.U.");
drawnode.bot(2,"Berges␣de␣Maine");
drawnode.bot(5,"Hôtel␣de␣Ville");
drawnode.bot(6,"Foch␣-␣Maison␣bleue");
drawnode.bot(7,"Doutre");
drawnode.bot(8,"Place␣Molière");
drawnode.bot(9,"Conservatoire");
drawnode.bot(10,"Montaigne");
vardef nodeshape(expr p)=
  (halfcircle shifted (0,0.5)--
   halfcircle scaled -1 shifted (0,-0.5)--
    cycle) scaled nodewidth
enddef;
drawnode.bot(3,"Saint-Serge");
drawnode.bot(4,"Centre␣de␣Congrès");
endfig;
```



Example 107

```
beginfig(107);
vardef nodeshape(expr p)=
% From "Drawing with Metapost" by Toby Thurston
% https://github.com/thruston/Drawing-with-Metapost/
 for i=1 upto 20:
  (0.4cm if odd(i): - else: + fi 3+ uniformdeviate 3,0) rotated (i*360/20) --
 endfor cycle
enddef;
vardef edgeshape(expr A,B)=
 save an;numeric an;an:=angle(B-A);
 A{dir (an + 60*(uniformdeviate 2-1))}..
 0.5[A,B]{dir (180+an)}..
 {dir (an + 60*(uniformdeviate 2-1))}B
enddef;
nodebgcolor:=(1,0.5,0);
nodelinecolor:=0.6red;
nodefgcolor:=0.6red;
linecolor:=0.5green;
edgelinewidth:=1.5;
defnodes((0cm,0cm),(4cm,2cm),(8cm,1cm),(5cm,-3cm));
drawnodes();
drawedges((1,2),(2,3),(3,4),(1,4))();
endfig;
```