Documentation of `mptrees.mp`

Olivier PÉAULT[*]

April 27, 2023

Contents

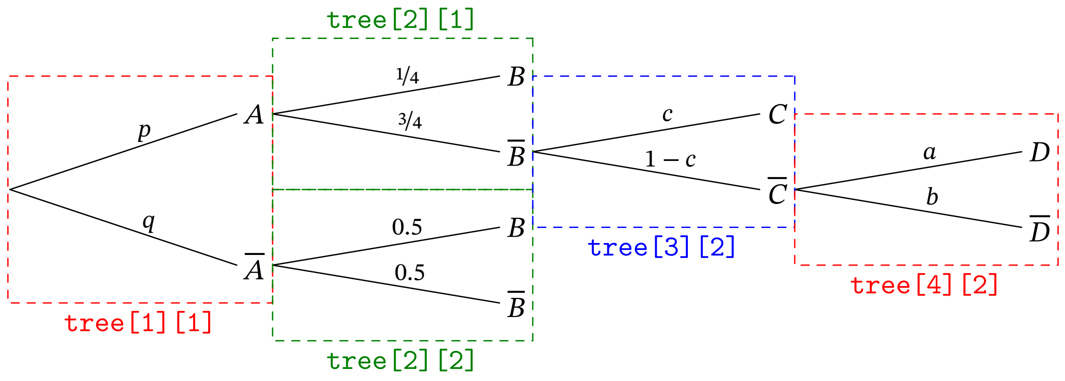
1 Overview

This package is intended to simplify the drawing of probability trees 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.

`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).



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

`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

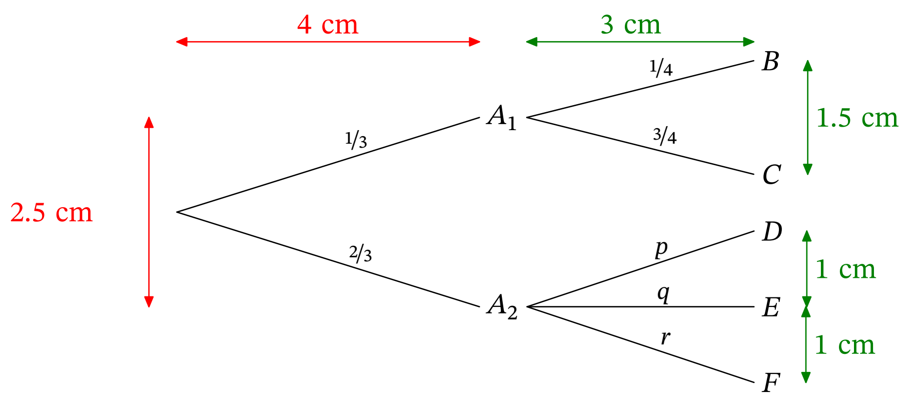
2.1 Different kinds of trees

`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.

Exemple 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;
```

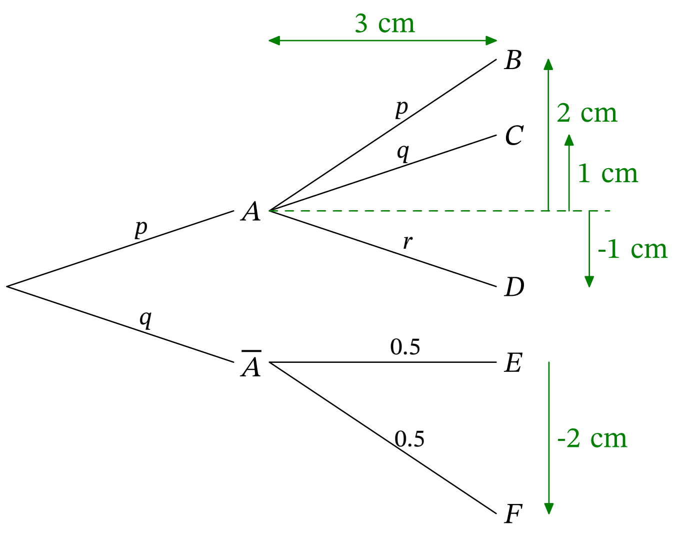


`tree[<i>][<j>](<width>,<vsp1>,<vsp2>,...)(<ev1>,<p1>,<ev2>,<p2>,...)` picture

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.

Exemple 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;
```

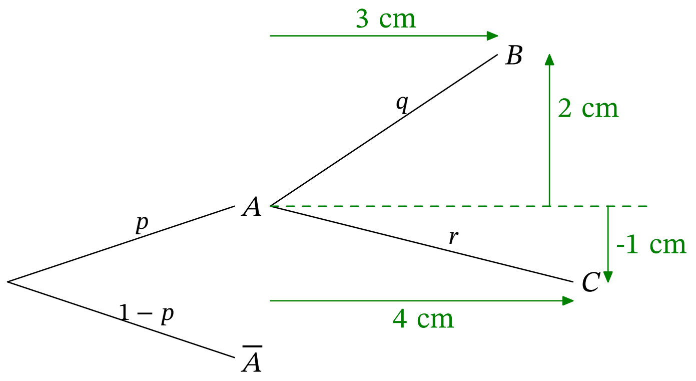


`tree[<i>][<j>](<pair1>,<pair2>,...)(<ev1>,<prob1>,<ev2>,<prob2>,...)` picture

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

Exemple 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;
```



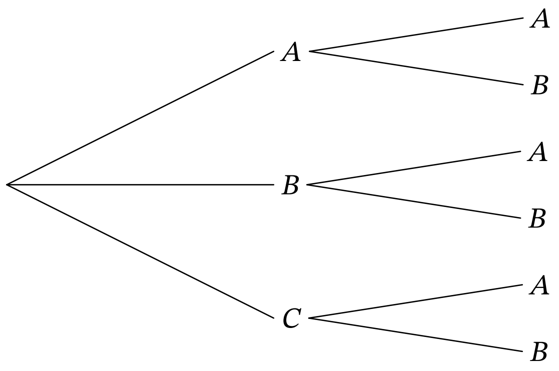
2.2 Simple trees

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

Same as previous except that there are no probabilities.

Exemple 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;
```



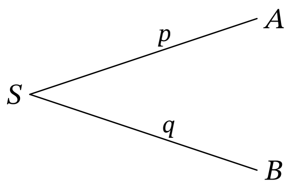
2.3 Start and end labels

`startlabel(<s>)` picture

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

Exemple 5

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

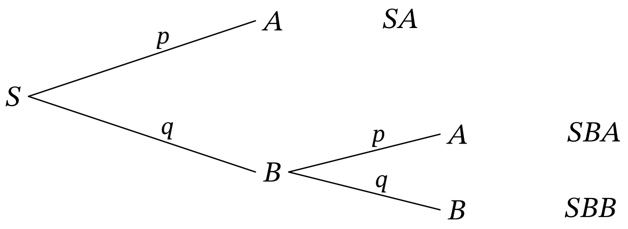


`endlabel[<i>][<j>](<s>)` picture

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`.

Exemple 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;
```



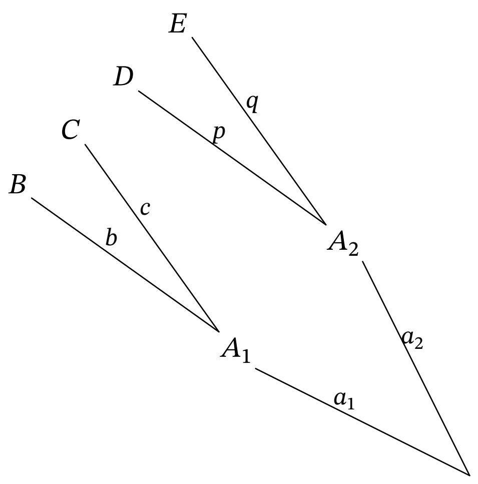
3 Direction

`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.

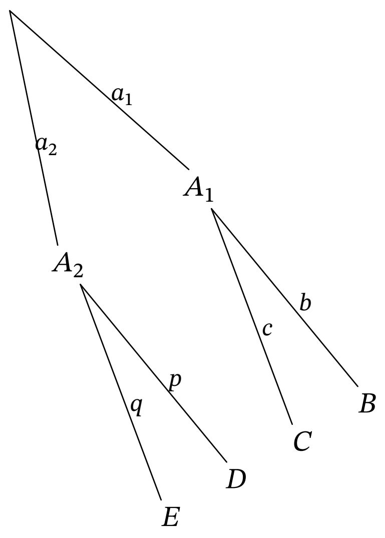
Exemple 7

```
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;
```



Exemple 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;
```

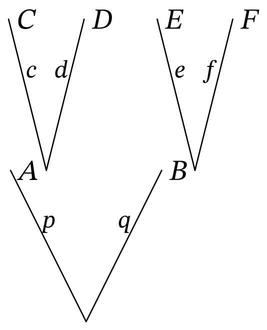


dirlabel numeric, default : 0

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.

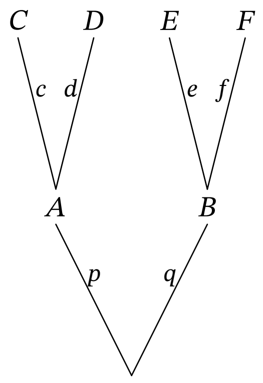
Exemple 9

```
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;
```



Exemple 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;
```



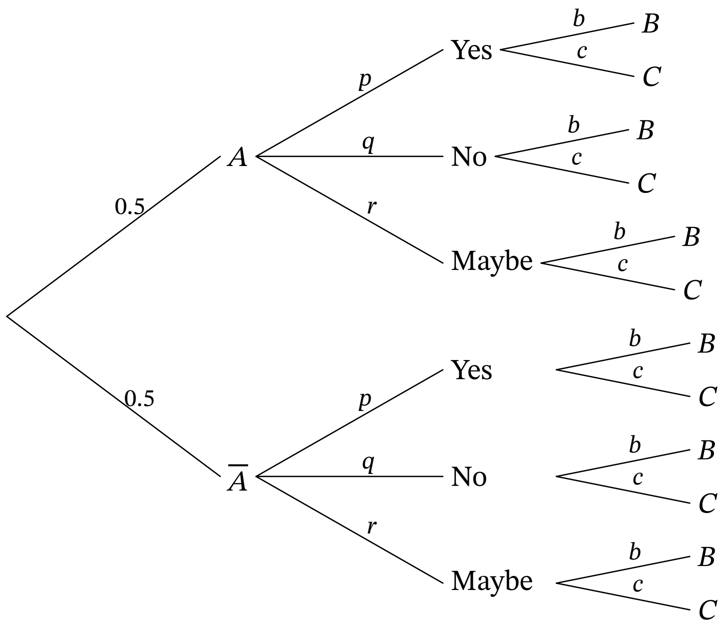
4 Dealing with alignment

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.

Exemple 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;
```

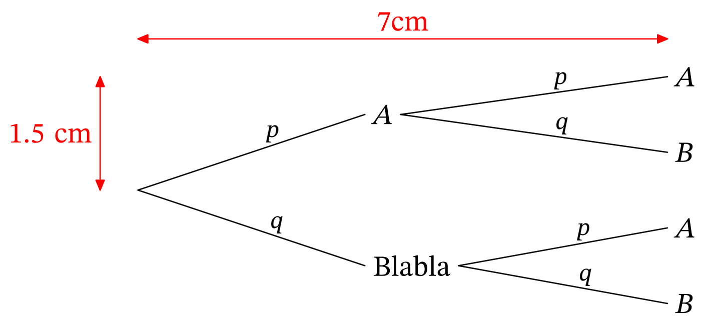


`abscoord` **boolean, default : `false`**

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.

Exemple 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.

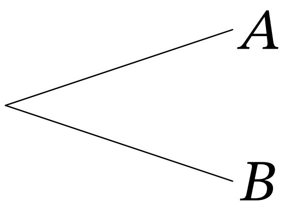
5.1 Event

`scaleev` numeric, default : 1

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

Exemple 13

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

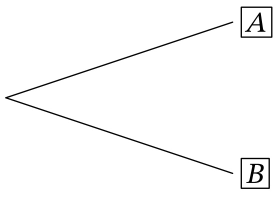


`nodeformat` string, default : ""

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

Exemple 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

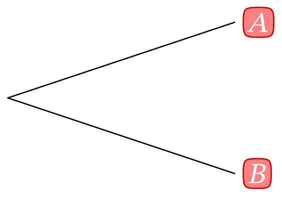
Color of the background of the region delimited by the previous path.

`nodefgcolor` color, default : black

Color of the text.

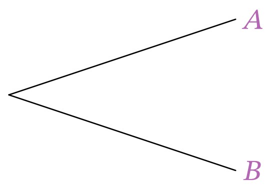
Exemple 15

```
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;
```



Exemple 16

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



5.2 Leaves

begintree;
endtree;

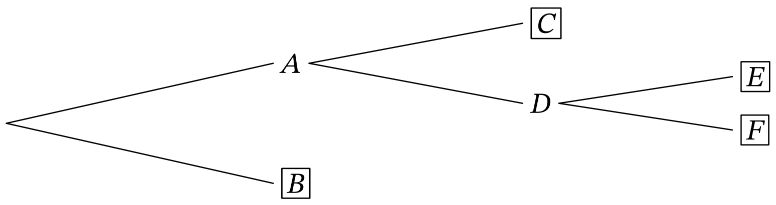
You may want to format the leaves in a different way from the nodes. A tree using the following parameters must be enclosed in a `begintree;`...`endtree;` "environment".

`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"`.

Exemple 17

```
beginfig(17);
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`

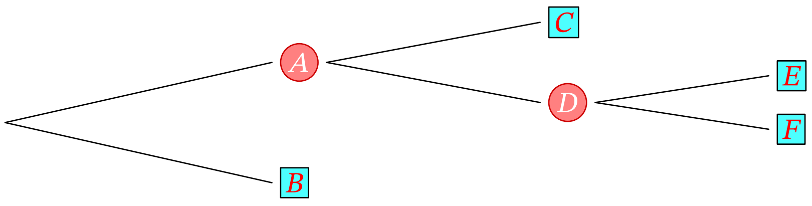
Color of the background of the region delimited by the previous path.

`leavefgcolor` color, default : `black`

Color of the text.

Exemple 18

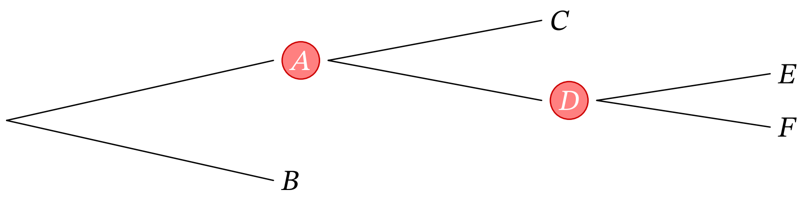
```
beginfig(18);
begintree;
nodeformat:="circle";
nodelinecolor:=(0.8,0,0); nodebgcolor:=(1,0.5,0.5); nodefgcolor:=white;
leaveformat:="bbox";
leavebgcolor:=(0.3,1,1); leavefgcolor:=red;
draw stree[1][1](100,45)("$A$","$B$");
draw stree[2][1](80,30)("$C$","$D$");
draw stree[3][2](65,20)("$E$","$F$");
endtree;
endfig;
```



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

Exemple 19

```
beginfig(19);
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;
```



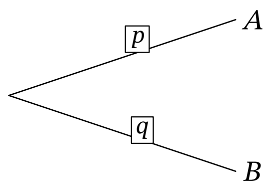
5.3 Probability

`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"`.

Exemple 20

```
beginfig(20);
 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

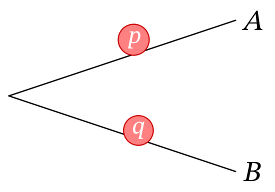
Color of the background of the region delimited by the previous path.

`probfgcolor` color, default : black

Color of the text.

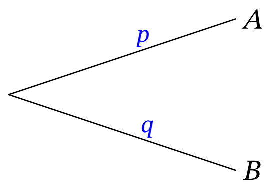
Exemple 21

```
beginfig(21);
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;
```



Exemple 22

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

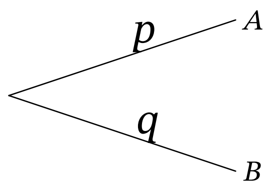


`scaleprob` numeric, default : 0.85

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

Exemple 23

```
beginfig(23);
 scaleprob:=1.5;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

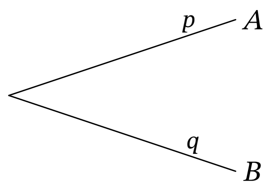


`posprob` numeric, default : 0.6

Numeric controlling the position of the label above the edge.

Exemple 24

```
beginfig(24);
 posprob:=0.8;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

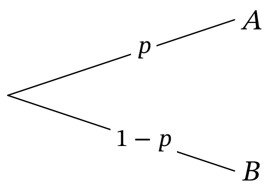


`typeprob` numeric, default : 1

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).

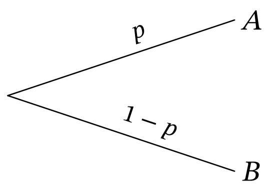
Exemple 25

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



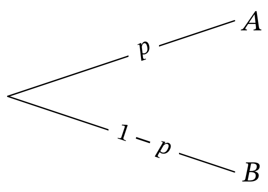
Exemple 26

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



Exemple 27

```
beginfig(27);
 typeprob:=4;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$1-p$");
endfig;
```

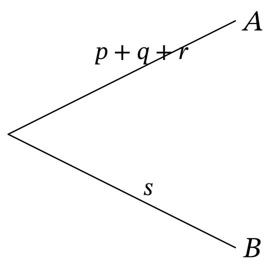


`proboffset` numeric, default : 3bp

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

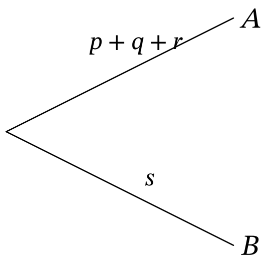
Exemple 28

```
beginfig(28);
 draw tree[1][1](3cm,3cm)("$A$","$p+q+r$","$B$","$s$");
endfig;
```



Exemple 29

```
beginfig(29);
 proboffset:=6bp;
 draw tree[1][1](3cm,3cm)("$A$","$p+q+r$","$B$","$s$");
endfig;
```



5.4 Edge

`linewidth` numeric, default : 0.5bp

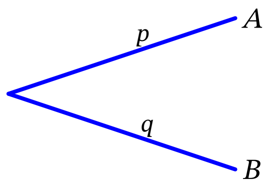
Width of the lines.

`linecolor` color, default : black

Color of the lines.

Exemple 30

```
beginfig(30);
 linewidth:=1.5;
 linecolor:=blue;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```

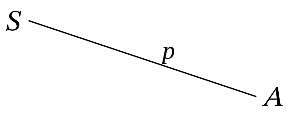


`endedgeshift` numeric, default : 0

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

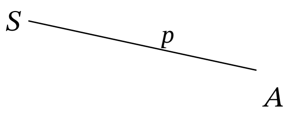
Exemple 31

```
beginfig(31);
 draw startlabel("$S$");
 draw tree[1][1]((3cm,-1cm))("$A$","$p$");
endfig;
```



Exemple 32

```
beginfig(32);
 endedgeshift:=10;
 draw startlabel("$S$");
 draw tree[1][1]((3cm,-1cm))("$A$","$p$");
endfig;
```

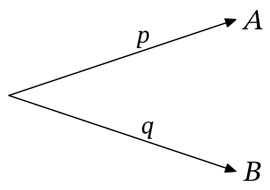


`edgearrow` boolean, default : false

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

Exemple 33

```
beginfig(33);
 edgearrow:=true;
 draw tree[1][1](3cm,2cm)("$A$","$p$","$B$","$q$");
endfig;
```



`branchtype` string, default : "segment"

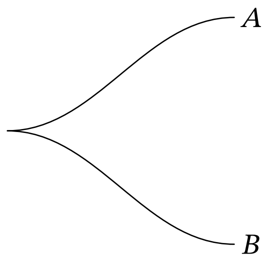
String which indicates the shape of the edge. Possible values are `segment`, `curve`, `broken`.
Note that double quotes have to be replaced by single quotes when this parameter is changed locally inside the `tree` macro.

`tenscurve` numeric, default : 0

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

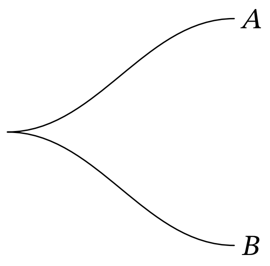
Exemple 34

```
beginfig(34);
 branchtype:="curve";
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



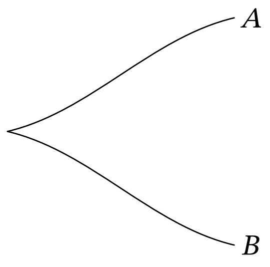
Exemple 35

```
beginfig(35);
 draw stree[1][1](3cm,3cm,"branchtype:='curve'")
                                    ("$A$","$B$");
endfig;
```



Exemple 36

```
beginfig(36);
 branchtype:="curve";
 tenscurve:=0.5;
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```

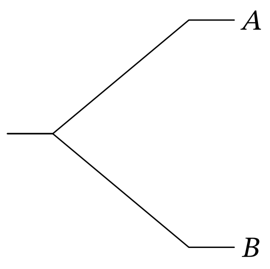


`brokenlineratio` numeric, default : 0.2

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.

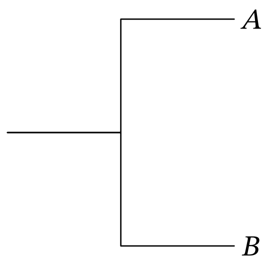
Exemple 37

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



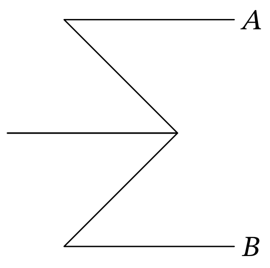
Exemple 38

```
beginfig(38);
 branchtype:="broken";
 posprob:=0.8;
 brokenlineratio:=0.5;
 draw stree[1][1](3cm,3cm)("$A$","$B$");
endfig;
```



Exemple 39

```
beginfig(39);
 branchtype:="broken";
 posprob:=0.8;
 brokenlineratio:=0.75;
 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

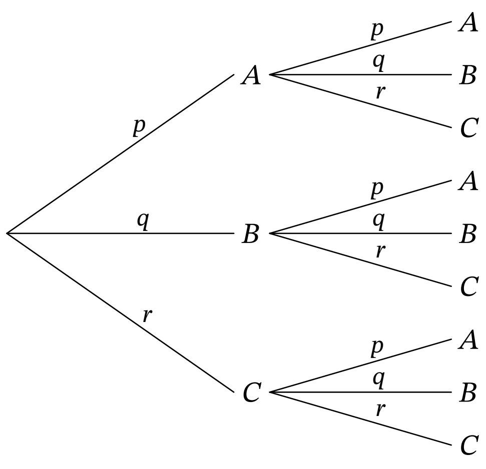
Tree describing the repetition of n identical and independent random experiments. `l` is the horizontal length of the first edges and `h` is the vertical space between two leaves.

`scalebranch` numeric, default : `0.8`

Ratio between edges width of consecutive level.

Exemple 40

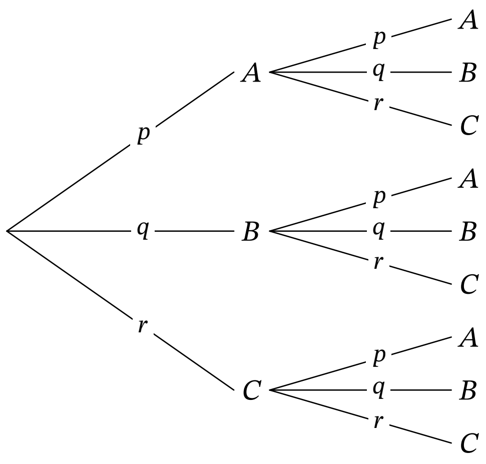
```
beginfig(40);
draw regulartree(2)(3cm,0.7cm)
            ("$A$","$p$","$B$","$q$","$C$","$r$");
endfig;
```



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

Exemple 41

```
beginfig(41);
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 : `"$q$"`

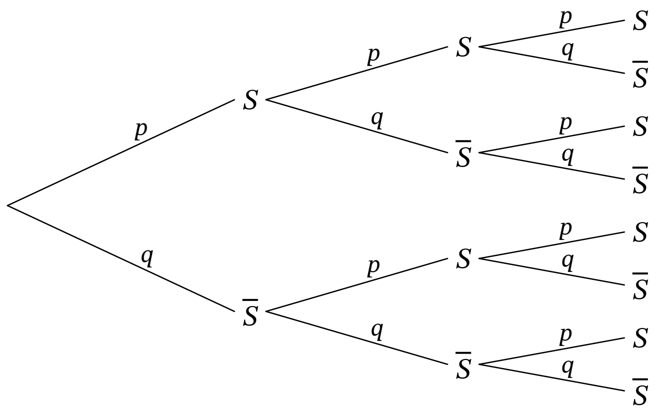
String printed above every edge representing a failure.

`bernoulliscalebranch` numeric, default : `0.8`

Ratio between width of consecutive edges.

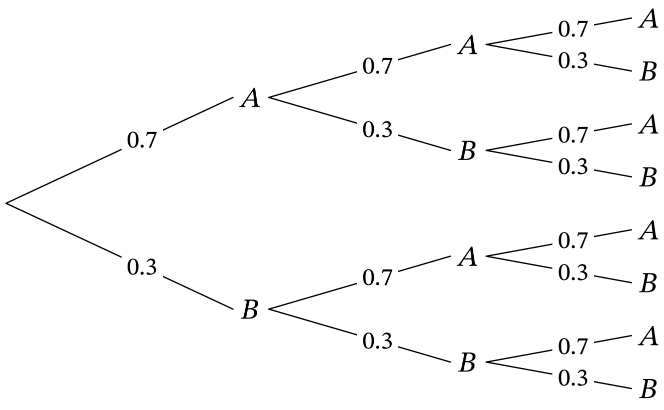
Exemple 42

```
beginfig(42);
draw bernoulliprocess(3)(3cm,0.7cm)();
endfig;
```



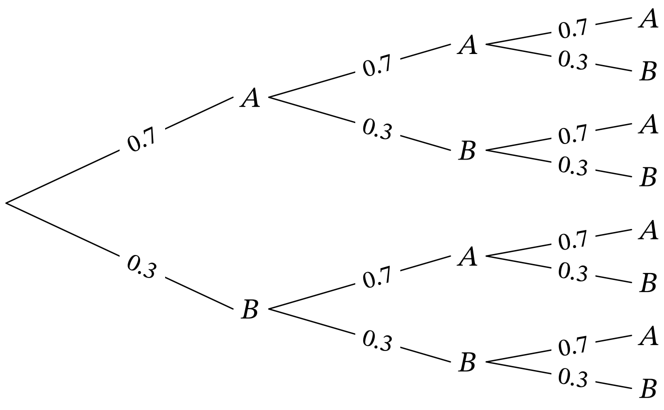
Exemple 43

```
beginfig(43);
draw bernoulliprocess(3)
        (3cm,0.7cm,"typeprob:=2;")
        ("$A$","$0.7$","$B$","$0.3$");
endfig;
```



Exemple 44

```
beginfig(44);
typeprob:=4;
bernoullisuccessevent:="$A$";
bernoullifailureevent:="$B$";
bernoullisuccessprob:="$0.7$";
bernoullifailureprob:="$0.3$";
draw bernoulliprocess(3)(3cm,0.7cm)();
endfig;
```



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

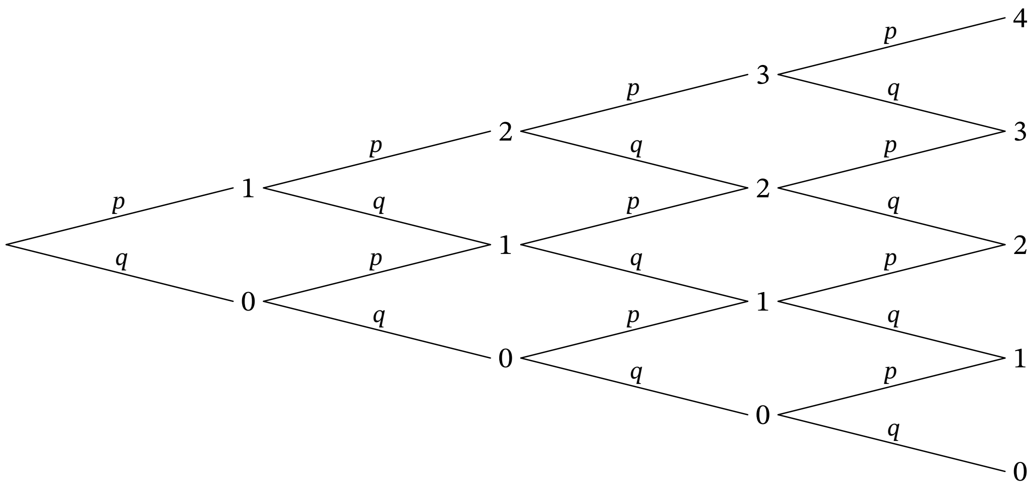
Tree describing the binomial distribution with n trials. `l` is the length of the first edges and `h` is the space between two final nodes. It uses `bernoullisuccesprob` and `bernoullifailureprob` but `bernoulliscalebranch` is set to `1`.

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

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

Exemple 45

```
beginfig(45);
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
```

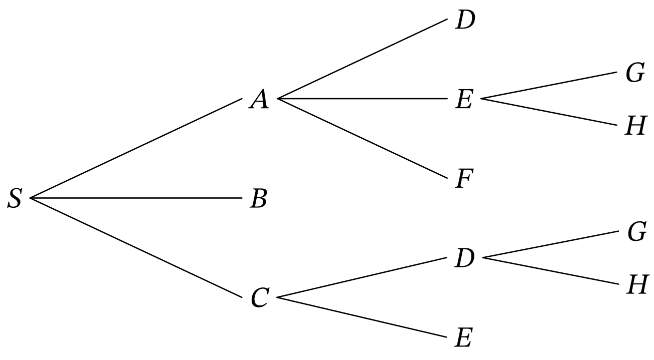
When the first set of parameters is left empty, the dimensions of the tree are calculated. The calculations use the parameters described below.

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

Same as above for “simple” trees.

Exemple 46

```
beginfig(46);
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`

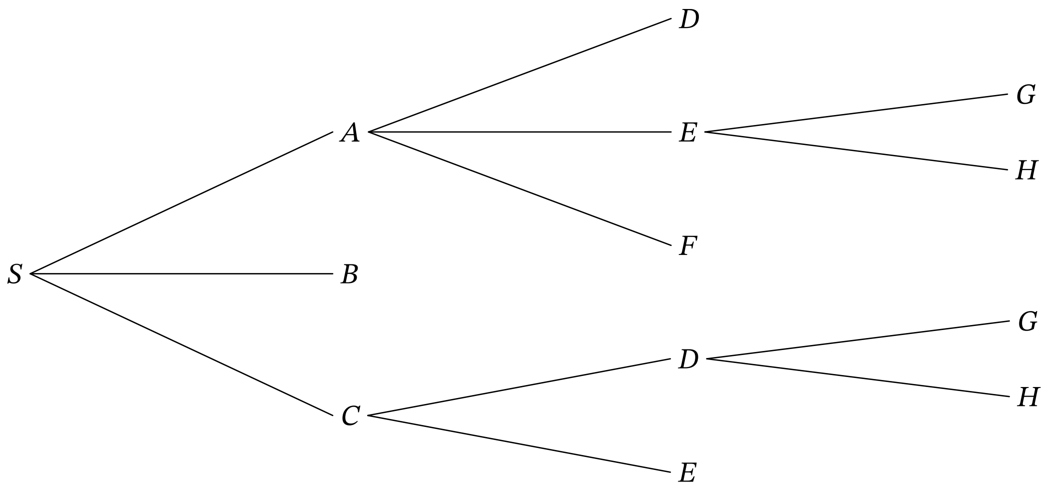
Minimal vertical space between two nodes of the last level trees.

`scalebranch` `numeric, default : 0.8`

Ratio between edges width of consecutive level.

Exemple 47

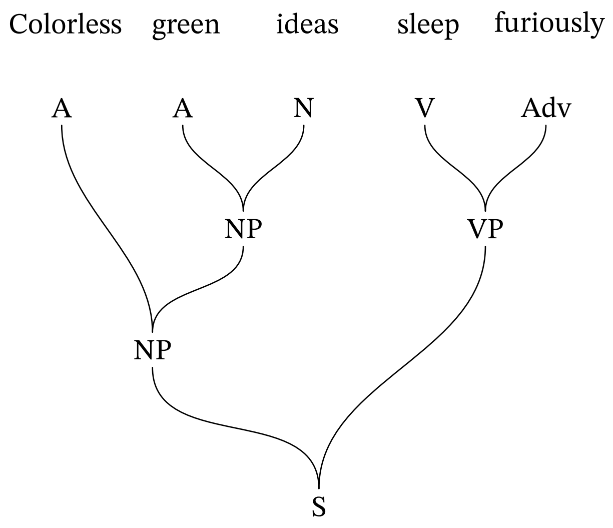
```
beginfig(47);
begintree;
widthbranch:=4cm;
scalebranch:=1;
gapnode:=1cm;
draw startlabel("$S$");
draw stree[1][1]()("$A$","$B$","$C$");
draw stree[2][1]()("$D$","$E$","$F$");
draw stree[2][3]()("$D$","$E$");
draw stree[3][2]()("$G$","$H$");
draw stree[3][4]()("$G$","$H$");
endtree;
endfig;
```



8 Examples

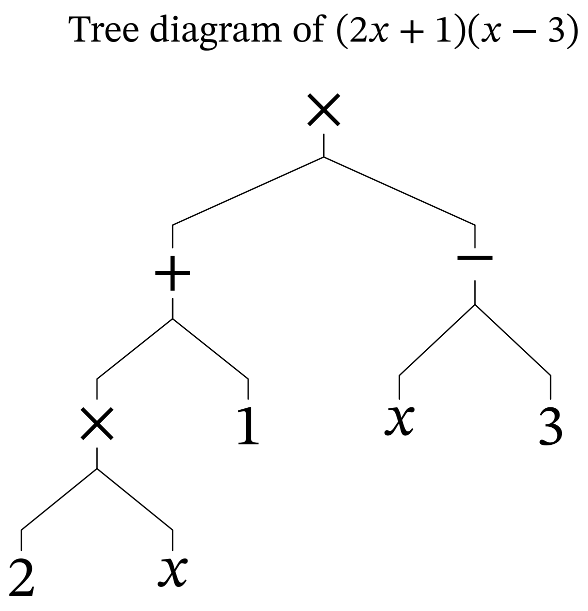
Exemple 48

```
beginfig(48);
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;
```



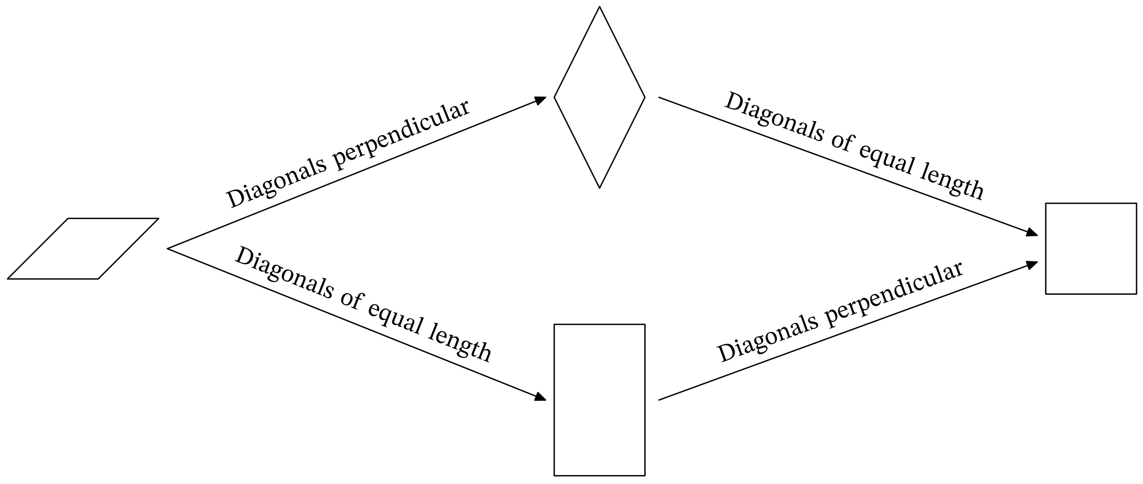
Exemple 49

```
beginfig(49);
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;
```



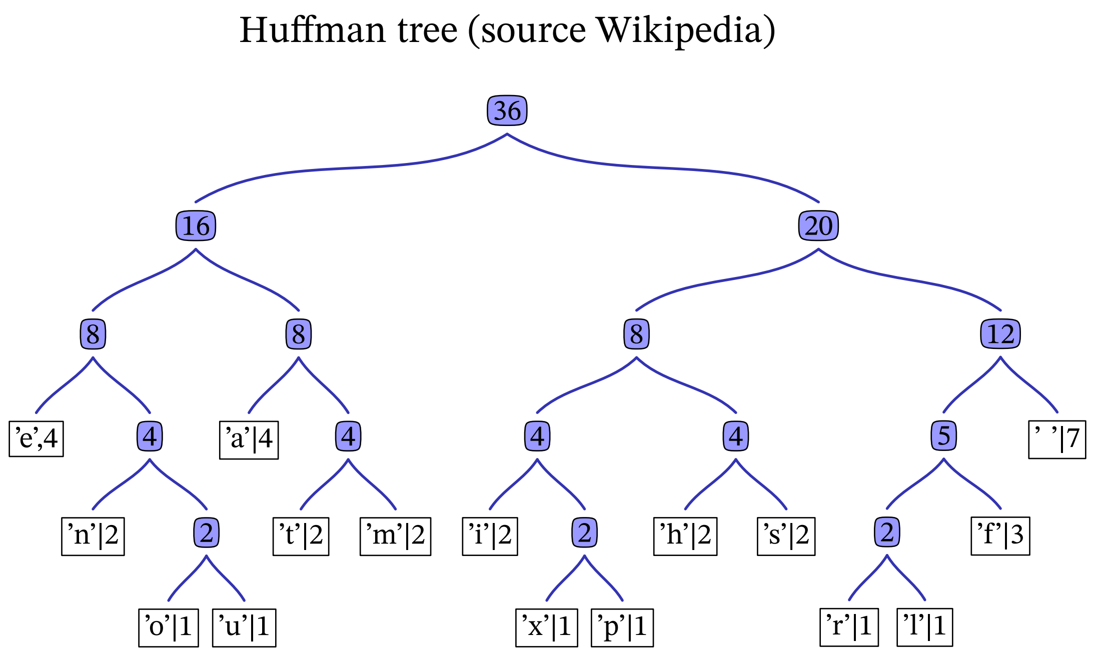
Exemple 50

```
beginfig(50);
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;
```



Exemple 51

```
beginfig(51);
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)
36
16
20
8
8
8
12
'e',4
4
'a'|4
4
4
4
5
' '|7
'n'|2
2
't'|2
'm'|2
'i'|2
2
'h'|2
's'|2
2
'f'|3
'o'|1
'u'|1
'x'|1
'p'|1
'r'|1
'l'|1