As an alignment environment, **\psmatrix** is similar to AMS-TEX's \matrix. There is no argument for specifying the columns. Instead, you can just use as many columns as you need. The entries are horizontally centered. Rows are ended by \\. **\psmatrix** can be used in or out of math mode.

Our first example wasn't very interesting, because we didn't make use of the nodes. Actually, each entry is a node. The name of the node in row *row* and column *col* is {*row*,*col*}, with no spaces. Let's see some node connections:

$$
\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, "g"] \\
Y \arrow[r, dotted, "h"'] & Z
\end{tikzcd}
$$

```
$
\psmatrix[colsep=1cm]
    & X \\
  Y & Z
\endpsmatrix
\everypsbox{\scriptstyle}%
\psset{nodesep=3pt,arrows=->}
\ncline{1,2}{2,1}
\tlput{f}
\ncline{1,2}{2,2}
\trput{g}
\ncline[linestyle=dotted]{2,1}{2,2}
\tbput{h}
$
```

You can include the node connections inside the **\psmatrix**, in the last entry and right before **\endpsmatrix**. One advantage to doing this is that **shortput=tab** is the default within a **\psmatrix**.

$$
\begin{tikzcd}
U \arrow[dr, "y"'] \arrow[drr, dash, dashed, equal, "x"] & & \\
 & X \times_Z Y \arrow[r, "p"'] \arrow[d, "q"'] & X \arrow[d, "f"] \\
 & Y \arrow[r, "g"'] & Z
\end{tikzcd}
$$

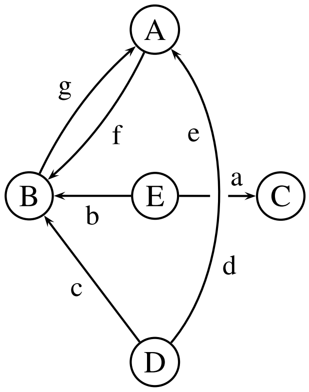
```
$
  \begin{psmatrix}
    U \\
      &  X\times_Z Y  &  X \\
      &        Y         & Z
    \psset{arrows=->,nodesep=3pt}
    \everypsbox{\scriptstyle}
    \ncline{1,1}{2,2}_{y}
    \ncline[doubleline=true,linestyle=dashed]{-}{1,1}{2,3}^{x}
    \ncline{2,2}{3,2}<{q}
    \ncline{2,2}{2,3}_{p}
    \ncline{2,3}{3,3}>{f}
    \ncline{3,2}{3,3}_{g}
  \end{psmatrix}
$
```

You can change the kind of nodes that are made by setting the

mnode=*type* **Default: R**

parameter. Valid types are R, r, C, f, p, circle, oval, dia, tri, dot and none, standing for **\Rnode**, **\rnode**, **\Cnode**, **\fnode**, **\pnode**, **\circlenode**, **\ovalnode**, **\dotnode** and no node, respectively. Note that for circles, you use **mnode=C** and set the radius with the **radius** parameter.

For example:



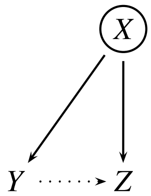

```
\psmatrix[mnode=circle,colsep=1]
    & A \\
  B & E & C \\
    & D &
\endpsmatrix
\psset{shortput=nab,arrows=->,labelsep=3pt}
\small
\ncline{2,2}{2,3}^[npos=.75]{a}
\ncline{2,2}{2,1}^{b}
\ncline{3,2}{2,1}^{c}
\ncarc[arcangle=-40,border=3pt]{3,2}{1,2}
  _[npos=.3]{d}^[npos=.7]{e}
\ncarc[arcangle=12]{1,2}{2,1}^{f}
\ncarc[arcangle=12]{2,1}{1,2}^{g}
```

Note that a node is made only for the non-empty entries. You can also specify a node for the empty entries by setting the

emnode=*type* **Default: none**

parameter.

You can change parameters for a single entry by starting with entry with the parameter changes, enclosed in square brackets. Note that the changes affect the way the node is made, but not contents of the entry (use **\psset** for this purpose). For example:




```
$
\psmatrix[colsep=1cm]
     & [mnode=circle] X \\
   Y & Z
\endpsmatrix
\psset{nodesep=3pt,arrows=->}
\ncline{1,2}{2,1}
\ncline{1,2}{2,2}
\ncline[linestyle=dotted]{2,1}{2,2}
$
```

If you want your entry to begin with a [that is not meant to indicate parameter changes, the precede it by {}.

You can assign your own name to a node by setting the

name=*name* **Default:**

parameter at the beginning of the entry, as described above. You can still refer to the node by {*row*,*col*}, but here are a few reasons for giving your own name to a node:

- The name may be easier to keep track of;
- Unlike the {*row*,*col*} names, the names you give remain valid even when you add extra rows or columns to your matrix.
- The names remain valid even when you start a new **\psmatrix** that reuses the {*row*,*col*} names.

Here a few more things you should know:

- The baselines of the nodes pass through the centers of the nodes. **\psmatrix** achieves this by setting the

  nodealign=*true/false* **Default: false**

  parameter to true. You can also set this parameter outside of **\psmatrix** when you want this kind of alignment.
- You can left or right-justify the nodes by setting the

  mcol=*l/r/c* **Default: c**

  parameter. l, r and c stand for left, right and center, respectively.
- The space between rows and columns is set by the

rowsep=*dim* **Default: 1.5cm**
colsep=*dim* **Default: 1.5cm**

parameters.

- If you want all the nodes to have a fixed with, set

  mnodesize=*dim* **Default: -1pt**

  to a positive value.

- If **\psmatrix** is used in math mode, all the entries are set in math mode, but you can switch a single entry out of math mode by starting and ending the entry with $.

- The radius of the c **mnode** (corresponding to **\cnode**) is set by the

  radius=*dim* **Default: 2pt**

  parameter.

- Like in LaTeX, you can end a row with \\[*dim*] to insert an extra space *dim* between rows.

- The command \psrowhookii is executed, if defined, at the beginning of every entry in row ii (row 2), and the command \pscolhookv is executed at athe beginning of every entry in column v (etc.). You can use these hooks, for example, to change the spacing between two columns, or to use a special **mnode** for all the entries in a particular row.

- An entry can itself be a node. You might do this if you want an entry to have two shapes.

- If you want an entry to stretch across several (*int*) columns, use the

  \psspan{*int*}

  at the end of the entry. This is like Plain TeX's **\multispan**, or LaTeX's **\multicolumn**, but the template for the current column (the first column that is spanned) is still used. If you want wipe out the template as well, use \multispan{*int*} *at the beginning of the entry* instead. If you just want to wipe out the template, use \omit before the entry.

- **\psmatrix** can be nested, but then all node connections and other references to the nodes in the {*row*,*col*} form for the nested matrix *must go inside* the **\psmatrix**. This is how PSTricks decides which matrix you are referring to. It is still neatest to put all the node connections towards the end; just be sure to put them before **\endpsmatrix**. Be careful also not to refer to a node until it actually appears. The whole matrix can itself go inside a node, and node connections can be made as usual. This is not the same as connecting nodes from two different **\psmatrix**'s. To do this, you must give the nodes names and refer to them by these names.

12 Obsolete put commands

This is old documentation, but these commands will continue to be supported.

There is also an obsolete command **\Lput** for putting labels next to node connections. The syntax is

```
\Lput{labelsep}[refpoint]{rotation}(pos){stuff}
```

It is a combination of **\Rput** and **\lput**, equivalent to

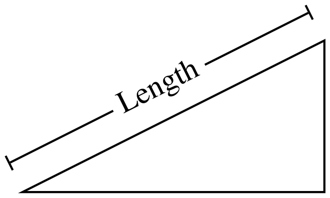
```
\lput(pos){\Rput{labelsep}[refpoint]{rotation}(0,0){stuff}}
```

\Mput is a short version of **\Lput** with no {*rotation*} or (*pos*) argument. **\Lput** and **\Mput** remain part of PSTricks only for backwards compatibility.

Here are the node label commands:

\lput*[*refpoint*]{*rotation*}***(pos){stuff}***

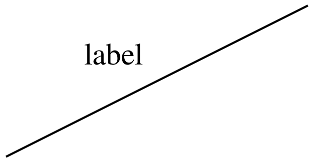
The l stands for "label". Here is an example illustrating the use of the optional star and :*angle* with **\lput**, as well as the use of the **offset** parameter with **\pcline**:




```
\pspolygon(0,0)(4,2)(4,0)
\pcline[offset=12pt]{|-|}(0,0)(4,2)
\lput*{:U}{Length}
```

(Remember that with the put commands, you can omit the coordinate if you include the angle of rotation. You are likely to use this feature with the node label commands.)

With **\lput** and **\rput**, you have a lot of control over the position of the label. E.g.,

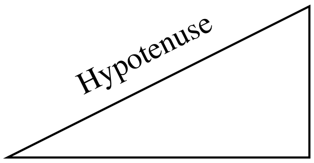



```
\pcline(0,0)(4,2)
\lput{:U}{\rput[r]{N}(0,.4){label}}
```

puts the label upright on the page, with right side located .4 centimeters "above" the position .5 of the node connection (above if the node connection points to the right). However, the **\aput** and **\bput** commands described below handle the most common cases without **\rput**.[8]

\aput*[*labelsep*]{*angle*}***(*pos*){*stuff*}**

stuff is positioned distance **\pslabelsep** *above* the node connection, given the convention that node connections point to the right. \aput is a node-connection variant of **\uput**. For example:




```
\pspolygon(0,0)(4,2)(4,0)
\pcline[linestyle=none](0,0)(4,2)
\aput{:U}{Hypotenuse}
```

\bput*[*labelsep*]{*angle*}***(*pos*){*stuff*}**

This is like **\aput**, but *stuff* is positioned *below* the node connection.

It is fairly common to want to use the default position and rotation with these node connections, but you have to include at least one of these arguments. Therefore, PSTricks contains some variants:

---

[8] There is also an obsolete command **\Lput** for putting labels next to node connections. The syntax is

\Lput{*labelsep*}[*refpoint*]{*rotation*}(*pos*){*stuff*}

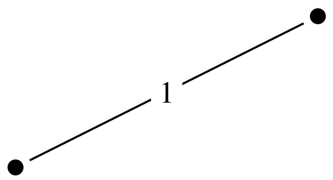
It is a combination of **\Rput** and **\lput**, equivalent to

\lput(*pos*){\Rput{*labelsep*}[*refpoint*]{*rotation*}(0,0){*stuff*}}

\Mput is a short version of **\Lput** with no {*rotation*} or (*pos*) argument. **\Lput** and **\Mput** remain part of PSTricks only for backwards compatibility.

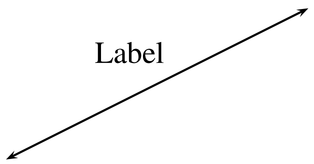
\mput*[*refpoint*]***{stuff}***
\Aput*[*labelsep*]***{stuff}***
\Bput*[*labelsep*]***{stuff}***

of **\lput**, **\aput** and **\bput**, respectively, that have no angle or positioning argument. For example:




```
\cnode*(0,0){3pt}{A}
\cnode*(4,2){3pt}{B}
\ncline[nodesep=3pt]{A}{B}
\mput*{1}
```

Here is another:




```
\pcline{<->}(0,0)(4,2)
\Aput{Label}
```

II Trees

13 Overview

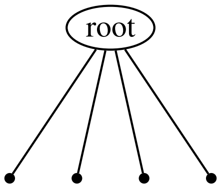
pstree

The node and node connections are perfect tools for making trees. The file pstree.tex / pstree.sty contains a high-level interface for positioning the nodes in a tree.

The main tree macro is

\pstree{*(root)node*}{*(sub)trees and (terminal)nodes*}

This positions the root node above its successors.

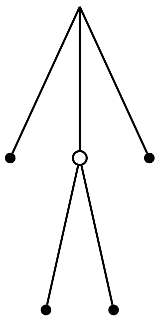



```
\pstree{\Toval{root}}{\TC*  \TC*  \TC*  \TC*}
```

\pstree produces a box that encloses all the nodes, and whose baseline passes through the center of the root node.

For most of the nodes described in Section 6 (e.g., **\ovalnode**), there is a variant for use within a tree (e.g., **\Toval**). Note that there is no distinction between a terminal node and a root node, other than their position in the **\pstree** command.

A tree, when included in the list or successors, becomes a subtree.



```
\pstree{\Tp}{%
   \TC*
   \pstree{\Tc{3pt}}{\TC*  \TC*}
   \TC*}
```

14 Tree Nodes

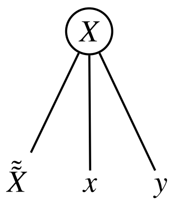
For most nodes described in Section 6, you can add strip node from the end of the name and add T add the beginning to obtain a node for use in trees. The syntax of a tree node is the same as of its corresponding “normal” node, except that:

- there is always an optional argument for setting graphics parameters, even if the original node did not have one,
- there is no argument for specifying the name of the node, and
- there is never a coordinate argument for positioning the node.
- to set the reference point with **\Tr**, set the **ref** parameter.

Here is the list of such tree nodes:

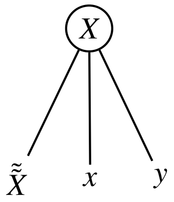
\Tp*[par]
\Tc*[par]{dim}
\TC*[par]
\Tf*[par]
\Tdot*[par]
\Tr*[par]{stuff}
\TR*[par]{stuff}
\Tcircle*[par]{stuff}
\Toval*[par]{stuff}
\Tdia*[par]{stuff}
\Ttri*[par]{stuff}

\Rnode is a good choice when you want the baselines of the text in the nodes to line up horizontally.




```
$
\pstree[nodesepB=3pt]{\Tcircle{X}}{%
   \TR{\tilde{\tilde{X}}}
   \TR{x}
   \TR{y}}
$
```

Compare the preceding example with the next one, which uses **\rnode**:

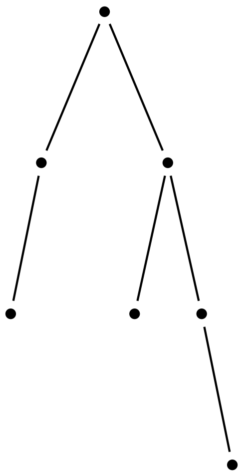



```
$
\pstree[nodesepB=3pt]{\Tcircle{X}}{%
   \Tr{\tilde{\tilde{X}}}
   \Tr{x}
   \Tr{y}}
$
```

There is also a null tree node

\Tn

It is meant to be just a place holder.



```
\pstree[nodesep=3pt]{\TC*}{%
   \pstree{\TC*}{\TC*  \Tn}
   \pstree{\TC*}{%
      \TC*
      \pstree{\TC*}{\Tn\TC*}}}
```

Actually, if I was going to do this a lot I would define some short-cuts:

```
\def\mytree{\pstree{\TC*}}
\def\ltree#1{\mytree{#1\Tn}}
\def\rtree#1{\mytree{\Tn#1}}
\psset{nodesep=3pt}
\mytree{%
   \ltree{\TC*}
   \mytree{%
      \TC*
      \rtree{\TC*}}}
```

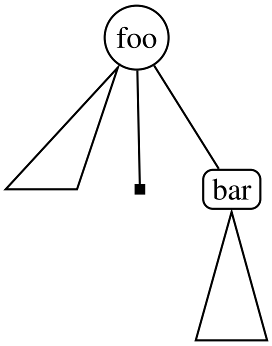
There is also a special tree node that doesn't have a "normal" version and that can't be used as the root node of a whole tree:

\Tfan*[*par*]

This draws a triangle whose base is

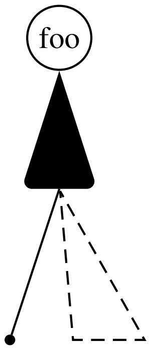
fansize=*dim* **Default: 1cm**

and whose opposite corner is the predecessor node, adjusted by the value of **nodesepA** and **offsetA**. For example:




```
\pstree{\Tcircle{foo}}{%
  \Tfan
  \Tf*[framesize=4pt]
  \pstree{\Tr{\psframebox[framearc=.5]{bar}}}{\Tfan}}
```

Here is another example illustrating that a **\Tfan** can have successors:




```
\pstree{\Tcircle{foo}}{%
  \pstree{\Tfan*[linearc=.1]}{%
    \Tc*{2pt}
    \Tfan[linestyle=dashed]}}
```

15 Trees

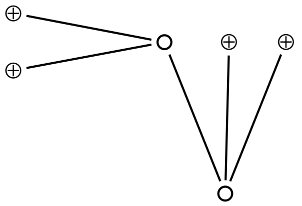
This section describes several graphics parameters for **\pstree**. Any settings of graphics parameters for **\pstree** affects all of its successors, including subtrees. but not the root node.

The

treemode=*R/L/U/D* **Default:**

parameter controls the direction in which the tree grows. R, L, U and D stand for “right”, “left”, “up” and “down”, respectively. When you change the **treemode**, the **treemode** of all nested trees changes as well.

For example, here is a tree that grows up, and then to the left:



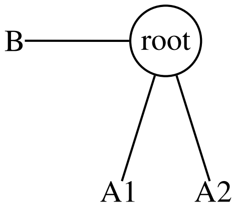
```
\pstree[treemode=U,dotstyle=oplus,dotsize=6pt,
  nodesep=2pt]{\Tc{3pt}}{%
  \pstree[treemode=L]{\Tc{3pt}}{%
    \Tdot
    \Tdot}
  \Tdot
  \Tdot}
```

When the tree goes up or down, the successors are lined up from left to right in the order they appear in **\pstree**'s argument. When the tree goes left or right, the successors are lined up from top to bottom. As an afterthought, you might want to flip the order of the nodes. The

treeflip=*true/false* **Default: false**

let's you do this.

A tree can also be root node. This is useful when the nested tree goes off in a different direction. If *treeB* is the root node of *treeA*, then the root of *treeB* is also the root node *treeA*.




```
\pstree{%
  \pstree[treemode=L]{\Tcircle{root}}{%
    \Tr{B}}%
  }{%
    \Tr{A1}
    \Tr{A2}}
```

A node can also contain a tree, but that is another story.

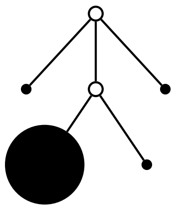
The distance between successors and between levels is given by the

treesep=*dim* **Default: .75cm**
levelsep=dim*** **Default: 2cm**

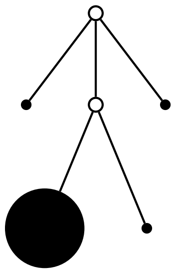
parameters.

The distance between successors takes into account the size of the nodes, but the distance between levels does not, at least by default. If you include the optional * when setting **levelsep**, the level sep is in addition to the size of the nodes. However, PSTricks needs a second run through TeX (without any changes between runs) to get the spacing right, and it writes to the .aux file with LaTeX, and to the file \jobname.pst with other macro packages. (Even then, there is no guarantee it will get the spacing right.)

Here are two exaggerated examples that illustrates the difference between relative and absolute spacing between levels:



```
\pstree[levelsep=1cm,radius=2pt]{\Tc{3pt}}{%
  \TC*
  \pstree{\Tc{3pt}}{%
    \Tc*{15pt}
    \TC*}
  \TC*}
```

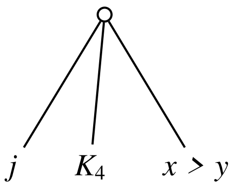


```
\psset{levelsep=*1cm,radius=2pt}
\pstree{\Tc{3pt}}{%
  \TC*
  \pstree{\Tc{3pt}}{%
    \Tc*{15pt}
    \TC*}
  \TC*}
```

If you set the

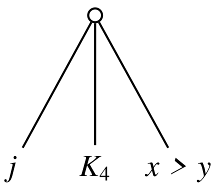
treenodesize=*dim* **Default: -1pt**

to a non-negative value, then PSTricks uses **treenodesize** as a fixed size of the successors (in the direction of their neighbors, i.e., a fixed width for vertical trees and a fixed height/depth for horizontal trees). For example, sometimes it is esthetically pleasing to smooth over small variations in the sizes of the nodes. Compare




```
\pstree[nodesepB=-8pt]{\Tc{3pt}}{%
  \TR{$j$}%
  \TR{$K_4$}%
  \TR{$x>y$}}
```

with




```
\pstree[treenodesize=.4cm,treesep=.3cm,nodesepB=-8pt]{\Tc{3pt}}{%
  \TR{$j$}%
  \TR{$K_4$}%
  \TR{$x>y$}}
```

A subtree's profile varies from level to level. **\pstree** has two modes for fitting subtrees together:

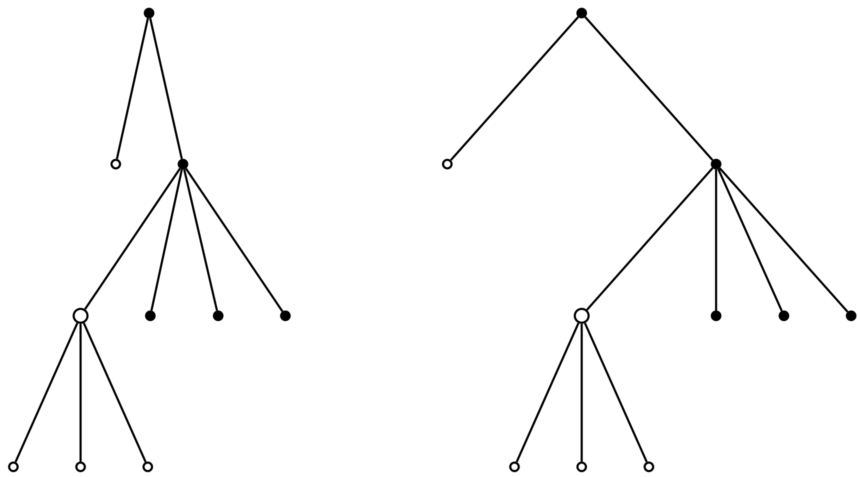
tight With tight fit, the subtrees are fit together so that the minimum distance on any level is **treesep**. This is the default.

loose With loose fit, the distance between the subtrees' bounding boxes is **treesep**. Except when you have exceptionally large intermediate nodes, the effect is that the horizontal distance (or vertical distance, for horizontal trees) between all the terminal nodes is the same.

You select the mode with the

treefit=*tight/loose* **Default: tight**

parameter.

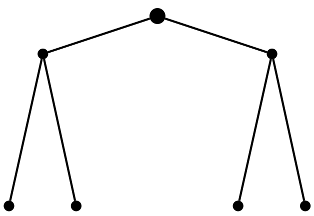


treefit=tight **treefit=loose**

As noted at the beginning of this section, parameter changes made with **\pstree** affect all subtrees. However, there are variants of some of these parameters for making local changes, i.e, changes that affects only the current level:

thistreesep=*dim* **Default:**
thistreenodesize=*dim* **Default:**
thistreefit=*tight/loose* **Default:**
thislevelsep=****dim*** **Default:**

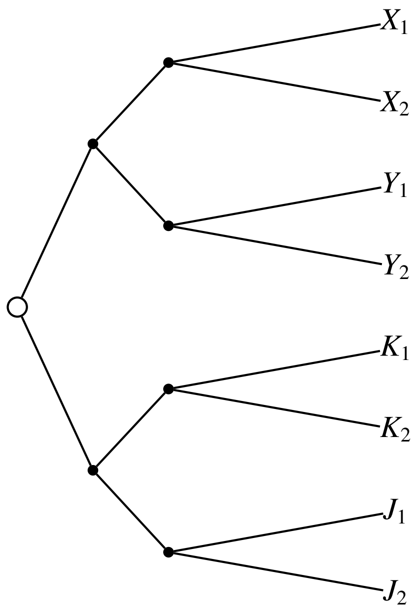
For example:



```
\pstree[thislevelsep=.5cm,thistreesep=2cm,radius=2pt]{\Tc*{3pt}}{%
    \pstree{\TC*}{\TC*  \TC*}
    \pstree{\TC*}{\TC*  \TC*}}
```

There are some things you may want set uniformly across a level in the tree, such as the **levelsep**. At level *n*, the command \pstreehook*roman(n)* (e.g., \pstreehookii) is executed, if it is defined (the root node of the whole tree is level 0, the successor tree objects and the node connections from the root node to these successors is level 1, etc.). In the following example, the **levelsep** is changed for level 2, without having to set the **thislevelsep** parameter for each of the three subtrees that make of level 2:

```
\[
  \def\pstreehookiii{\psset{thislevelsep=3cm}}
  \pstree[treemode=R,levelsep=1cm,radius=2pt]{\Tc{4pt}}{%
    \pstree{\TC*}{%
      \pstree{\TC*}{\Tr{X_1}  \Tr{X_2}}
      \pstree{\TC*}{\Tr{Y_1}  \Tr{Y_2}}}
    \pstree{\TC*}{%
      \pstree{\TC*}{\Tr{K_1}  \Tr{K_2}}
      \pstree{\TC*}{\Tr{J_1}  \Tr{J_2}}}}
\]
```



16 Edges

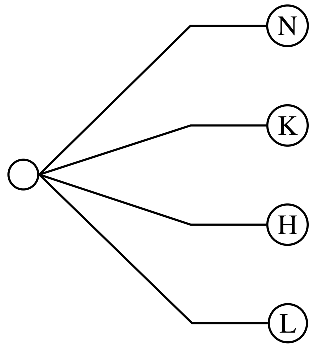
A tree node is really a composite object. In addition to creating a new node, it also draws a node connection between itself and its predecessor, if there is one.

When a tree node has made the new node, the command **\pssucc** is equal to the name of this node, and **\pspred** is equal to the name of its predecessor. Then the tree node executes

```
\psedge{\pspred}{\pssucc}
```

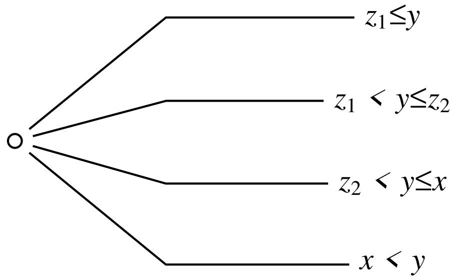
You can define **\psedge** to make whatever node connection you want (see Section **??**). For example, here I use **\ncdiag**, with **armA=0**, to get all the node connections to emanate from the same point in the predecessor:

```
\def\psedge{\ncdiag[armA=0,angleB=180,armB=1cm]}
% Or: \renewcommand{\psedge}{ ... }
\pstree[treemode=R,levelsep=3.5cm,framesep=2pt]{\Tc{6pt}}{%
  \small \Tcircle{N} \Tcircle{K} \Tcircle{H} \Tcircle{L}}
```



Here is another example with **\ncdiagg**. Note the use of negative the **armA** value so that the corners of the edges are vertically aligned, even though the nodes have different sizes:

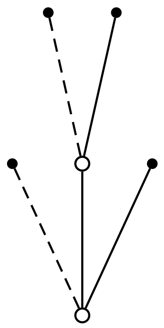
```
$
\def\psedge#1#2{\ncdiagg[angleA=180, armA=-3cm,
  nodesep=4pt]{#2}{#1}}
% Or: \renewcommand{\psedge}[2]{ ... }
\pstree[treemode=R, levelsep=5cm]{\Tc{3pt}}{%
  \Tr{z_1\leq y}
  \Tr{z_1<y\leq z_2}
  \Tr{z_2<y\leq x}
  \Tr{x<y}}
$
```



Another way to define **\psedge** is with the

edge=*command* **Default: \ncline**

parameter. Be sure to enclose the value in braces {} if it contains commas or other parameter delimiters. This gets messy if your command is long, and you can't use arguments like in the preceding example, but for simple changes it is useful. For example, if I want to switch between a few node connections frequently, I might define a command for each node connection, and then use the **edge** parameter.

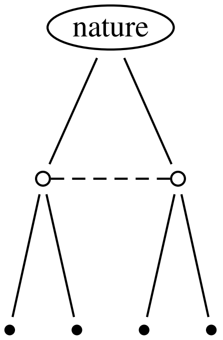


```
\def\dedge{\ncline[linestyle=dashed]}
\pstree[treemode=U,radius=2pt]{\Tc{3pt}}{%
  \TC*[edge=\dedge]
  \pstree{\Tc{3pt}}{\TC*[edge=\dedge]  \TC*}
  \TC*}
```

You can also set **edge=none** to suppress the node connection.

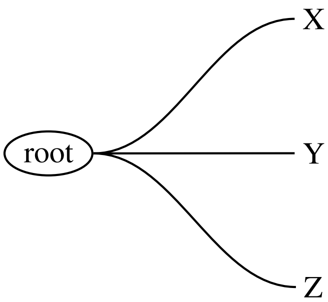
edge is the only parameter which, when set in a tree node's parameter argument, affects the drawing of the node connection (e.g., if you want to change the **nodesep**, your edge has to include the parameter change, or you have to set it before the node).

If you want to draw a node connection between two nodes that are not direct predecessor and successor, you have to give the nodes a name that you can refer to, using the **name** parameter. For example, here I connect two nodes on the same level:

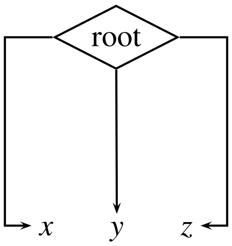



```
\pstree[nodesep=3pt,radius=2pt]{\Toval{nature}}{%
   \pstree{\Tc[name=top]{3pt}}{\TC*  \TC*}
   \pstree{\Tc[name=bot]{3pt}}{\TC*  \TC*}}
 \ncline[linestyle=dashed]{top}{bot}
```

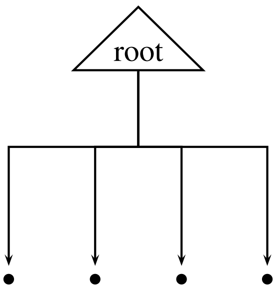
We conclude with the more examples.




```
\def\psedge{\nccurve[angleB=180,  nodesepB=3pt]}
\pstree[treemode=R,  treesep=1.5,  levelsep=3.5]%
   {\Toval{root}}{\Tr{X}  \Tr{Y}  \Tr{Z}}
```




```
\pstree[nodesepB=3pt,  arrows=->,  xbbl=15pt,
   xbbr=15pt,  levelsep=2.5cm]{\Tdia{root}}{%
   $
   \TR[edge={\ncbar[angle=180]}]{x}
   \TR{y}
   \TR[edge=\ncbar]{z}
   $}
```

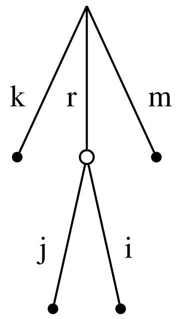



```
\psset{armB=1cm,  levelsep=3cm,  treesep=1cm,
   angleB=-90,  angleA=90,  arrows=<-,  nodesepA=3pt}
\def\psedge#1#2{\ncangle{#2}{#1}}
\pstree[radius=2pt]{\Ttri{root}}{\TC*  \TC*  \TC*  \TC*}
```

17 Edge and node labels

Right after a node, an edge has typically been drawn, and you can attach labels using **\ncput \tlput**, etc.

With **\tlput**, **\trput**, **\taput** and **\tbput**, you can align the labels vertically or horizontally, just like the nodes. This can look nice, at least if the slopes of the node connections are not too different.



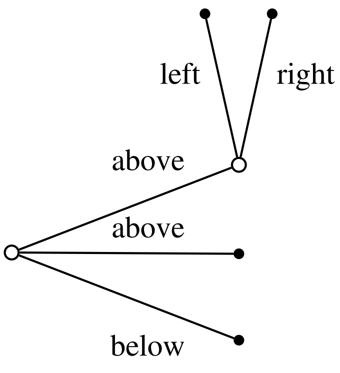

```
\pstree[radius=2pt]{\Tp}{%
  \psset{tpos=.6}
  \TC*  \tlput{k}
  \pstree{\Tc{3pt}  \tlput[labelsep=3pt]{r}}{%
    \TC*  \tlput{j}
    \TC*  \trput{i}}
  \TC*  \trput{m}}
```

Within trees, the **tpos** parameter measures this distance from the predecessor to the successor, whatever the orientation of the true. (Outside of trees it measures the distance from the top to bottom or left to right nodes.)

PSTricks also sets **shortput=tab** within trees. This is a special **shortput** option that should not be used outside of trees. It implements the following abbreviations, which depend of the orientation of the true:

	Short for:	
Char.	*Vert.*	*Horiz.*
^	**\tlput**	**\taput**
_	**\trput**	**\tbput**

(The scheme is reversed if **treeflip=true**.)

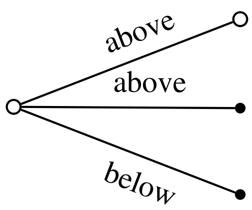



```
\psset{tpos=.6}
\pstree[treemode=R,  thistreesep=1cm,
  thislevelsep=3cm,radius=2pt]{\Tc{3pt}}{%
  \pstree[treemode=U,  xbbr=20pt]{\Tc{3pt}^{above}}{%
    \TC*^{left}
    \TC*_{right}}
  \TC*^{above}
  \TC*_{below}}
```

You can change the character abbreviations with

\MakeShortTab{*char1*}{*char2*}

The \n*put commands can also give good results:




```
\psset{npos=.6,nrot=:U}
\pstree[treemode=R, thistreesep=1cm,
   thislevelsep=3cm]{\Tc{3pt}}{%
   \Tc{3pt}\naput{above}
   \Tc*{2pt}\naput{above}
   \Tc*{2pt}\nbput{below}}
```

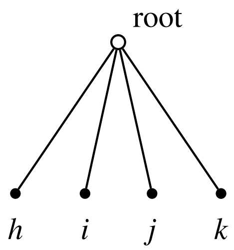
You can put labels on the nodes using **\nput**. However, **\pstree** won't take these labels into account when calculating the bounding boxes.

There is a special node label option for trees that does keep track of the bounding boxes:

~*[*par*]**{*stuff*}**

Call this a "tree node label".

Put a tree node label right after the node to which it applies, before any node connection labels (but node connection labels, including the short forms, can follow a tree node label). The label is positioned directly below the node in vertical trees, and similarly in other trees. For example:




```
\pstree[radius=2pt]{\Tc{3pt}\nput{45}{\pssucc}{root}}{
   \TC*~{$h$} \TC*~{$i$} \TC*~{$j$} \TC*~{$k$}}
```

Note that there is no "long form" for this tree node label. However, you can change the single character used to delimit the label with

\MakeShortTnput{*char1*}

If you find it confusing to use a single character, you can also use a command sequence. E.g.,

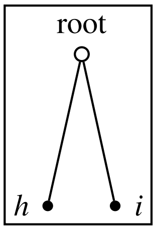
```
\MakeShortTnput{\tnput}
```

You can have multiple labels, but each successive label is positioned relative to the bounding box that includes the previous labels. Thus, the order in which the labels are placed makes a difference, and not all combinations will produce satisfactory results.

You will probably find that the tree node label works well for terminal nodes, without your intervention. However, you can control the tree node labels be setting several parameters.

To position the label on any side of the node (left, right, above or below), set:

tnpos=*l/r/a/b* **Default:**




```
\psframebox{%
  \pstree{\Tc{3pt}~[tnpos=a,tndepth=0pt]{root}}{
    \TC*~[tnpos=l]{$h$}
    \TC*~[tnpos=r]{$i$}}}
```

When you leave the argument empty, which is the default, PSTricks chooses the label position is automatically.

To change the distance between the node and the label, set

tnsep=*dim* **Default:**

When you leave the argument empty, which is the default, PSTricks uses the value of **labelsep**. When the value is negative, the distance is measured from the center of the node.

When labels are positioned below a node, the label is given a minimum height of

tnheight=*dim* **Default: \ht\strutbox**

Thus, if you add labels to several nodes that are horizontally aligned, and if either these nodes have the same depth or **tnsep** is negative, and if the height of each of the labels is no more than **tnheight**, then the labels will also be aligned by their baselines. The default is \ht\strutbox, which in most TEX formats is the height of a typical line of text in the current font. Note that the value of **tnheight** is not evaluated until it is used.

The positioning is similar for labels that go below a node. The label is given a minimum *depth* of

tndepth=*dim* **Default: \dp\strutbox**

For labels positioned above or below, the horizontal reference point of the label, i.e., the point in the label directly above or below the center of the node, is set by the **href** parameter.

When labels are positioned on the left or right, the right or left edge of the label is positioned distance **tnsep** from the node. The vertical point that is aligned with the center of the node is set by

tnyref=*num* **Default:**

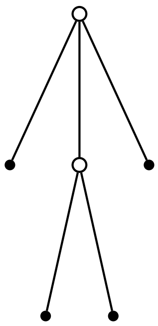
When you leave this empty, **vref** is used instead. Recall that **vref** gives the vertical *distance* from the baseline. Otherwise, the **tnyref** parameter works like the **yref** parameter, giving the fraction of the distance from the bottom to the top of the label.

18 Details

Both **\pstree**'s root node argument and successors argument are processed as LR-boxes, and so everything in Appendix **??**, including the treatment of math and verbatim text, applies, except the following. Because **\pstree** has two arguments, you cannot use **\pslongbox** to define a "long" version of **\pstree**. However, there is a variant **\psTree** of **\pstree** whose syntax is:

\psTree{*root node*} *successors* \endpsTree

For example:



```
\psTree{\Tc{3pt}}
   \TC*
   \psTree{\Tc{3pt}}
      \TC*
      \TC*
   \endpsTree
   \TC*
\endpsTree
```

LaTeX purists can write \begin{psTree} and \end{psTree} instead.

PSTricks does a pretty good job of positioning the nodes and creating a box whose size is close to the true bounding box of the tree. However, PSTricks does not take into account the node connections or labels when calculating the bounding boxes, except the tree node labels.

If, for this or other reasons, you want to fine tune the bounding box of the nodes, you can set the following parameters:

bbl=*dim*	**Default:**
bbr=*dim*	**Default:**
bbh=*dim*	**Default:**
bbd=*dim*	**Default:**
xbbl=*dim*	**Default:**
xbbr=*dim*	**Default:**
xbbh=*dim*	**Default:**
xbbd=*dim*	**Default:**

The x versions increase the bounding box by *dim*, and the others set the bounding box to *dim*. There is one parameter for each direction from the center of the node, **l**eft, **r**ight, **h**eight, and **d**epth.

These parameters affect trees and nodes, and subtrees that switch directions, but not subtrees that go in the same direction as their parent tree (such subtrees have a profile rather than a bounding box, and should be adjusted by changing the bounding boxes of the constituent nodes).

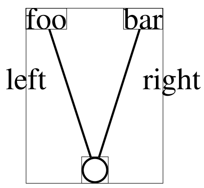
Save any fiddling with the bounding box until you are otherwise finished with the tree.

You can see the bounding boxes by setting the

showbbox=*true/false* **Default: false**

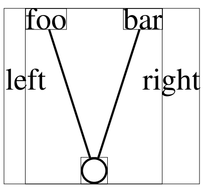
parameter to true. To see the bounding boxes of all the nodes in a tree, you have to set this parameter before the tree.

In the following example, the labels stick out of the bounding box:



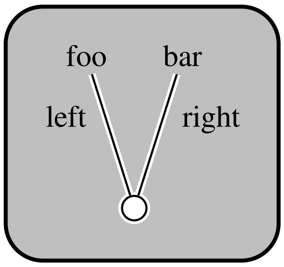

```
\psset{tpos=.6,showbbox=true}
\pstree[treemode=U]{\Tc{5pt}}{%
   \TR{foo}^{left}
   \TR{bar}_{right}}
```

Here is how we fix it:




```
\psset{tpos=.6,showbbox=true}
\pstree[treemode=U,xbbl=8pt,xbbr=14pt]{\Tc{5pt}}{%
   \TR{foo}^{left}
   \TR{bar}_{right}}
```

Now we can frame the tree:



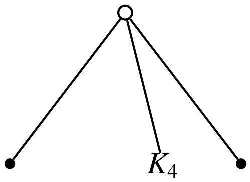

```
\psframebox[fillstyle=solid,fillcolor=lightgray,framesep=14pt,
  linearc=14pt,cornersize=absolute,linewidth=1.5pt]{%
  \psset{tpos=.6,border=1pt,nodesepB=3pt}
  \pstree[treemode=U,xbbl=8pt,xbbr=14pt]{%
    \Tc[fillcolor=white,fillstyle=solid]{5pt}}{%
    \TR*{foo}^{left}
    \TR*{bar}_{right}}}
```

We would have gotten the same result by changing the bounding box of the two terminal nodes.

You can also adjust the distance between successors with the

\tspace{*dim*}

command.



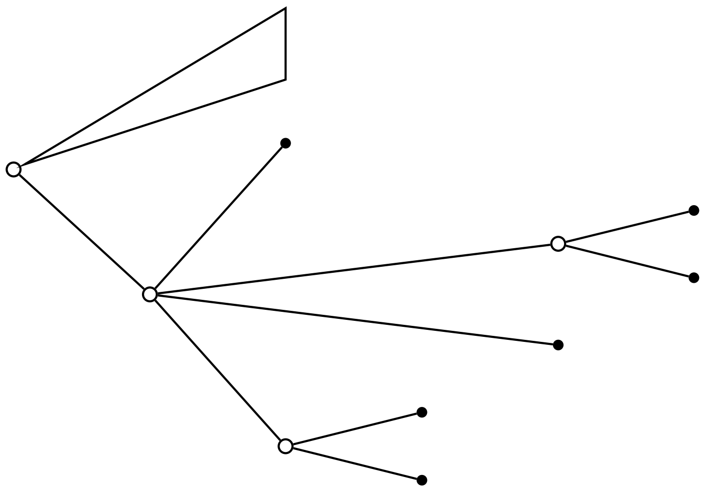

```
\pstree{\Tc{3pt}}{%
  \Tc*{2pt}%
  \tspace{1cm}
  \TR*{$K_4$}%
  \Tc*{2pt}}
```

To skip levels, use

\skiplevel*[*par*]**{*nodes or subtrees*}**
\skiplevels*[*par*]**{*int*}** ***nodes or subtrees*** **\endskiplevels**

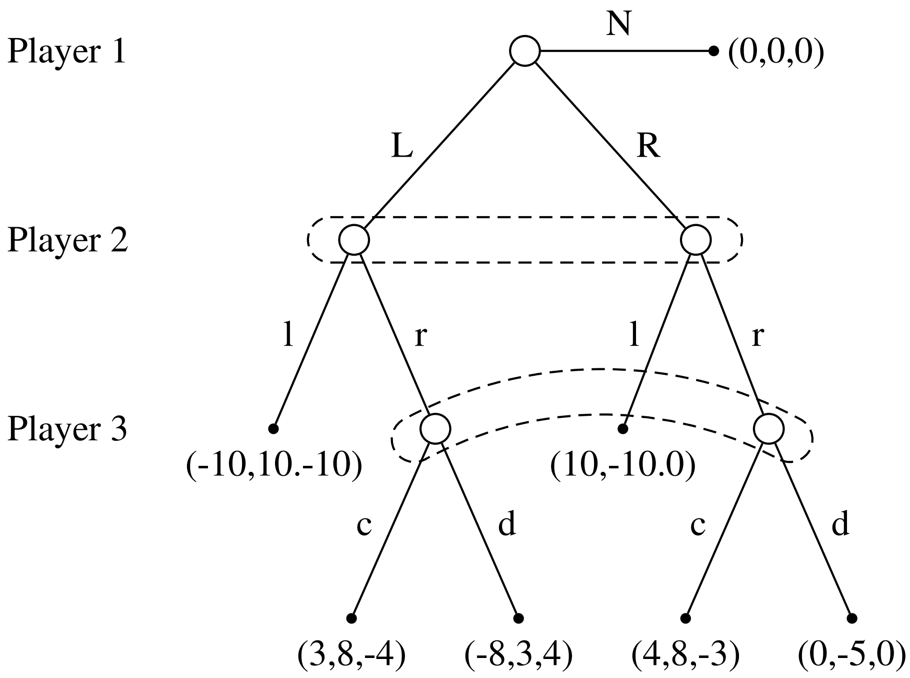
These are kind of like subtrees, but with no root node.

```
\pstree[treemode=R,levelsep=1.8,radius=2pt]{\Tc{3pt}}{
  \skiplevel{\Tfan}
  \pstree{\Tc{3pt}}{
    \TC*
     \skiplevels{2}
      \pstree{\Tc{3pt}}{\TC*  \TC*}
      \TC*
    \endskiplevels
    \pstree{\Tc{3pt}}{
      \TC*
      \TC*}}}
```



The profile at the missing levels is the same as at the first non-missing level. You can adjust this with the bounding box parameters. You get greatest control if you use nested **\skiplevel** commands instead of **\skiplevels**.

```
\large
\psset{radius=6pt, dotsize=4pt}
\pstree[thislevelsep=0,edge=none,levelsep=2.5cm]{\Tn}{%
  \pstree{\TR{Player 1}}{\pstree{\TR{Player 2}}{\TR{Player 3}}}
  \psset{edge=\ncline}
  \pstree
    {\pstree[treemode=R]{\TC}{\Tdot ~{(0,0,0)} ^{N}}}{%
    \pstree{\TC[name=A] ^{L}}{%
      \Tdot ~{(-10,10.-10)} ^{l}
      \pstree{\TC[name=C] _{r}}{%
        \Tdot ~{(3,8,-4)} ^{c}
        \Tdot ~{(-8,3,4)} _{d}}}
    \pstree{\TC[name=B] _{R}}{%
      \Tdot ~{(10,-10.0)} ^{l}
      \pstree{\TC[name=D]_{r}}{%
        \Tdot ~{(4,8,-3)} ^{c}
        \Tdot ~{(0,-5,0)} _{d}}}
    }}
\ncbox[linearc=.3,boxsize=.3,linestyle=dashed,nodesep=.4]{A}{B}
\ncarcbox[linearc=.3,boxsize=.3,linestyle=dashed,
  arcangle=25,nodesep=.4]{D}{C}
```


Player 1
Player 2
Player 3
N
(0,0,0)
L
R
l
r
l
r
(-10,10.-10)
(10,-10.0)
c
d
c
d
(3,8,-4)
(-8,3,4)
(4,8,-3)
(0,-5,0)