The dyntree package*

nsetzer

August 18, 2006

The dyntree package is intended for users needing to typeset a dykin tree—a group theoretical construct consisting of cartan coefficients in boxes connected by a series of lines. This package makes it easy for the user to generate these objects by allowing the user to specify only the cartan coefficients and the root number(s) that they connect to below.

This package requires the coollist package, which is not a standard LaTeXpackage but is available at CTAN.

1 Basics

`dyntree` `\dynbox`

To create a Dynkin Tree Diagram, the syntax is as follows :

```
\start{dyntree}{⟨num_roots⟩}
\dynbox{⟨cartan_coefficients⟩}{⟨cvs_descendant_root_list⟩} \lend
⋮
\finish{dyntree}
```

where

⟨*num_roots*⟩ is an integer indicating the number of simple roots

⟨*cartan_coefficients*⟩ is an *ampersand* (&) delimited list of cartan coefficients (the number of which *must* be equal to ⟨*num_roots*⟩

⟨*cvs_descendant_root_list*⟩ is a comma delimited list of integers indicating which simple root can be lowered from this box. The simple roots are numbered from left to right starting at 1 and ending at ⟨*num_roots*⟩

Thus, if the group of interest had 3 simple roots, each `\dynbox` would have a ⟨*cartan_coefficients*⟩ with three entries; that is, it would be a list with three integers as in

```
1 & 0 & 0
```

and the list ⟨*cvs_descendant_root_list*⟩ would have at most three entries (but it need not have exactly three entries) with the entries being between 1 and 3. The entry 1 would correspond to the first (left-most) simple root being lowered, the entry 2 the second simple root, and 3 the third:

*This document corresponds to dyntree v1, dated 2006/08/14.

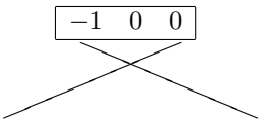
simple root:

1	2	3
1	0	0

So, finally, an entry such as

```
\dynbox{-1 & 0 & 0}{1,3}
```

would specify that the program should draw two lines below the box $\boxed{\begin{array}{ccc}1&0&0\end{array}}$; one for the first simple root, and another for the third simple root. The resulting portion of the diagram would look like



1.1 Quirks

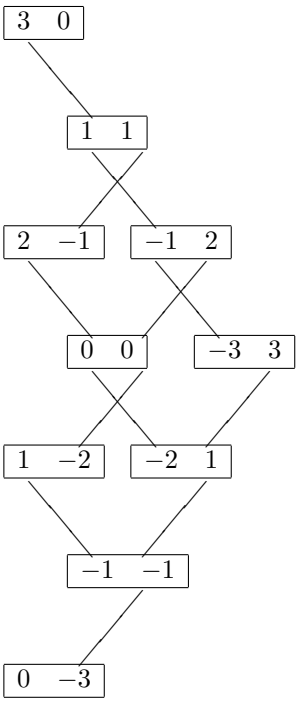
There are two quirks with this package, and they are

- if there are multiple Dynkin Tree Diagrams required for your document, you *must* enclose each tree in braces
- The lowest state (bottom-most dynbox) must have a non-empty entry in ⟨*cvs_descendant_root_list*⟩ even though no lines are to be drawn from it. To meet both criteria, place a zero in this spot.

1.2 Examples

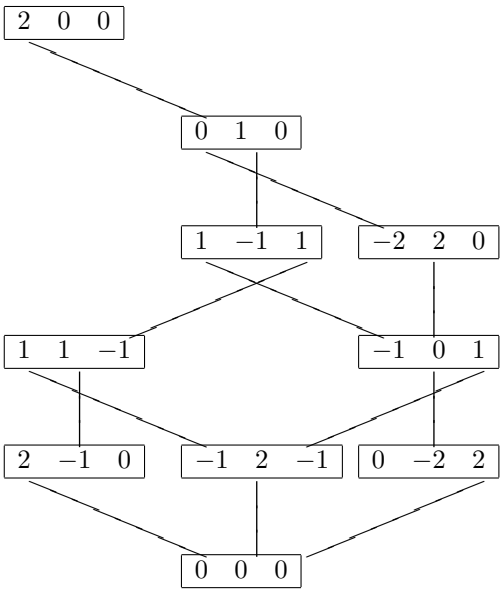
1.2.1 Example 1

```
{
\start{dyntree}{2}
\dynbox{3 & 0}{1} \lend
\dynbox{1 & 1}{1,2} \lend
\dynbox{2 & -1}{1}
\dynbox{-1 & 2}{1,2} \lend
\dynbox{0 & 0}{1,2}
\dynbox{-3 & 3}{0,2} \lend
\dynbox{1 & -2}{1}
\dynbox{-2 & 1}{0,2} \lend
\dynbox{-1 & -1}{0,2} \lend
\dynbox{0 & -3}{0} \lend
\finish{dyntree}
}
```



1.2.2 Example 2

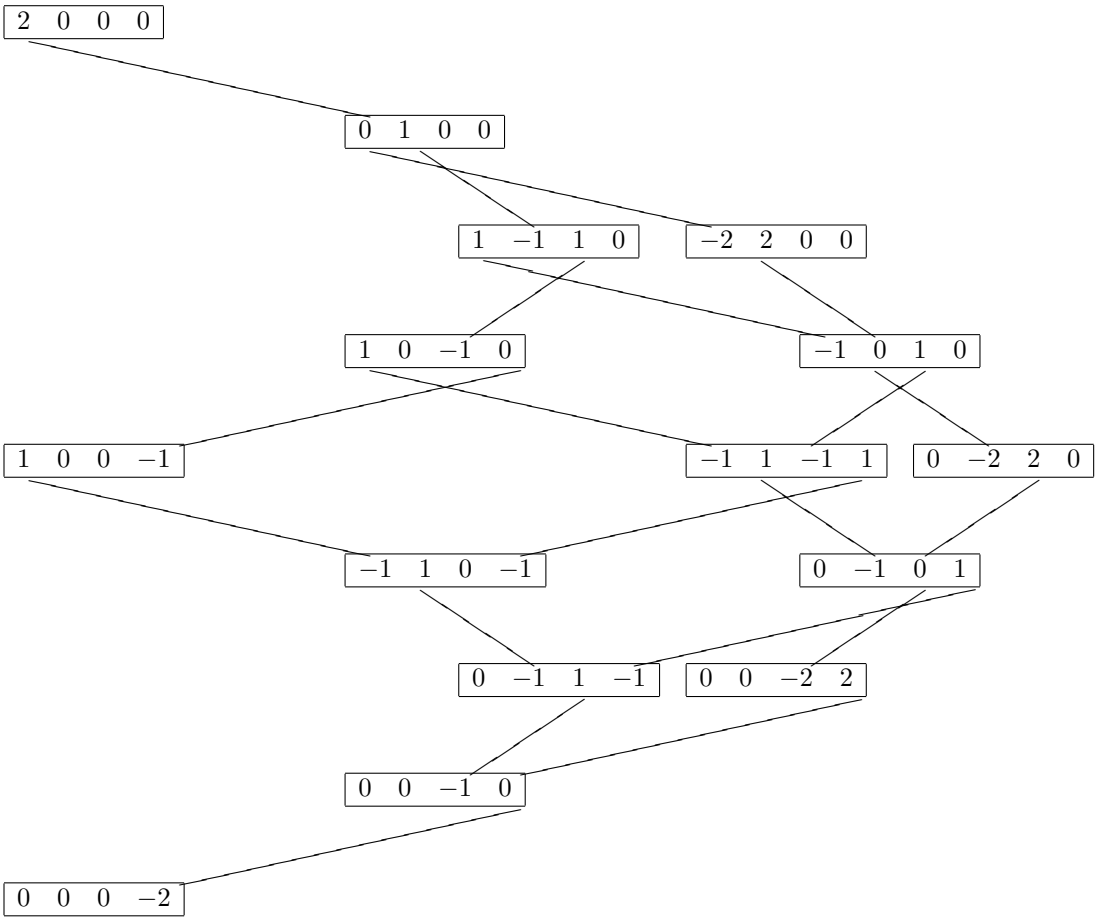
```
\begin{center}
{
\start{dyntree}{3}
\dynbox{2 & 0 & 0}{1} \lend
\dynbox{0 & 1 & 0}{1,2} \lend
\dynbox{1 & -1 & 1}{1,3}
\dynbox{-2 & 2 & 0}{2} \lend
\dynbox{1 & 1 & -1}{1,2}
\dynbox{-1 & 0 & 1}{2,3} \lend
\dynbox{2 & -1 & 0}{1}
\dynbox{-1 & 2 & -1}{2}
\dynbox{0 & -2 & 2}{3} \lend
\dynbox{0 & 0 & 0}{0} \lend
\finish{dyntree}
}
\end{center}
```



1.2.3 Example 3

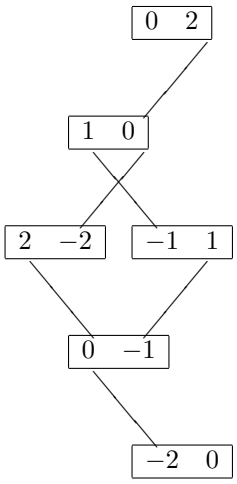
```
This is a $15$ of $SU(5)$

\begin{center}
{
\start{dyntree}{4}
\dynbox{2 & 0 & 0 & 0}{1} \lend
\dynbox{0 & 1 & 0 & 0}{1,2} \lend
\dynbox{1 & -1 & 1 & 0}{1,3}
\dynbox{-2 & 2 & 0 & 0}{2} \lend
\dynbox{1 & 0 & -1 & 0}{1,4}
\dynbox{-1 & 0 & 1 & 0}{2,3} \lend
\dynbox{1 & 0 & 0 &-1}{1}
\dynbox{-1 & 1 & -1 & 1}{2,4}
\dynbox{0 & -2 & 2 & 0}{3} \lend
\dynbox{-1 & 1 & 0 & -1}{2}
\dynbox{0 & -1 & 0 & 1}{3,4} \lend
\dynbox{0 & -1 & 1 & -1}{3}
\dynbox{0 & 0 & -2 & 2}{4} \lend
\dynbox{0 & 0 & -1 & 0}{4} \lend
\dynbox{0 & 0 & 0 & -2}{0} \lend
\finish{dyntree}
}
\end{center}
```

This is a 15 of $SU(5)$



1.2.4 Example 4

```
\begin{center}
{
\start{dyntree}{2}
\dynbox{0 & 2}{2} \lend
\dynbox{1 & 0}{1,2} \lend
\dynbox{2 & -2}{1}
\dynbox{-1 & 1}{2} \lend
\dynbox{0 & -1}{1} \lend
\dynbox{-2 & 0}{0} \lend
\finish{dyntree}
}
\end{center}
```



2 Implementation

2.1 Variables and Constants

The dyntree package utilizes a picture environment to create the tree. To do this it requires several constant length values, as well as calculated length values and counters. These are define below.

2.1.1 Cartan Coefficients Box Variables

The first thing to do is declare the length variables associated with a cartan coefficients box—dynbox for short. These variables are

`\DYNTREE@widechar`	the width of a -1
`\DYNTREE@thinchar`	the width of a 1
`\DYNTREE@cartancoefwidth`	the width of a cartan coefficient (a combination of -1 and 1)
`\DYNTREE@marginwidth`	the width of the margin of the cartan coefficients box
`\DYNTREE@colsepwidth`	the width between columns of the cartan coefficients box
`\DYNTREE@dynboxheight`	the height (baseline to top) of the cartan coefficients box
`\DYNTREE@dynboxdepth`	the depth (baseline to bottom) of the cartan coefficients box
`\DYNTREE@dynboxvlen`	the full vertical height of a cartan coefficients box
`\DYNTREE@dynboxwidth`	the width of a dynbox (calculated based on numroots)

```
1 \newlength{\DYNTREE@widechar}%
```

```
\newlength{\DYNTREE@thinchar}%
\newlength{\DYNTREE@cartancoefwidth}%
\newlength{\DYNTREE@marginwidth}%
\newlength{\DYNTREE@colsepwidth}%
\newlength{\DYNTREE@dynboxheight}%
\newlength{\DYNTREE@dynboxdepth}%
\newlength{\DYNTREE@dynboxvlen}%
\newlength{\DYNTREE@dynboxwidth}%
```

Now that they are declared, initialize the "exterior" ones

```
\settowidth{\DYNTREE@widechar}{$-1$}%
\settowidth{\DYNTREE@thinchar}{$1$}%
\setlength{\DYNTREE@cartancoefwidth}%
{\DYNTREE@widechar*1/2 + \DYNTREE@thinchar*1/2}%
\settowidth{\DYNTREE@marginwidth}%
{$\begin{array}{|c|}\hline 1 \\ \hline \end{array}$}%
\addtolength{\DYNTREE@marginwidth}{-\DYNTREE@thinchar}%
\settowidth{\DYNTREE@colsepwidth}%
{$\begin{array}{|cc|}\hline 1 & 1 \\ \hline \end{array}$}%
\addtolength{\DYNTREE@colsepwidth}{-\DYNTREE@marginwidth - \DYNTREE@thinchar*2}%
\settoheight{\DYNTREE@dynboxheight}%
{$\begin{array}{|c|}\hline 1 \\ \hline \end{array}$}%
\settodepth{\DYNTREE@dynboxdepth}%
{$\begin{array}{|c|}\hline 1 \\ \hline \end{array}$}%
\setlength{\DYNTREE@dynboxvlen}{\DYNTREE@dynboxheight + \DYNTREE@dynboxdepth}%
```

and now for convenience and error testing, print them out

widechar:	12.77782pt
thinchar:	5.00002pt
cartancoefwidth:	8.8889pt
marginwidth:	10.0pt
colsepwidth:	10.0pt
dynboxheight	8.9pt
dynboxdepth	3.9pt
dynboxvlen	12.79999pt

2.1.2 Dynkin Tree Variables

These variables are specific to the actual creation of the tree structure.

counters

DYNTREE@numlevel	the number of levels in the tree
DYNTREE@nextlevel	the number of the next level
DYNTREE@numboxes	counter for counting number of boxes in a row
DYNTREE@nextnumboxes	counter for counting number of boxes in the next row
DYNTREE@target	used to record the 'targeted' array element when sorting
DYNTREE@listlen	the length of the list of descendents
DYNTREE@xCoord	the x coordinate in scaled points
DYNTREE@yCoord	the y coordinate in scaled points
DYNTREE@xPos	the x coordinate in scaled points
DYNTREE@yPos	the y coordinate in scaled points
DYNTREE@xComp	the x coordinate in scaled points
DYNTREE@yComp	the y coordinate in scaled points
DYNTREE@leftX	the left most x coordinate in scaled points
DYNTREE@ct	generic counter
DYNTREE@counter	generic counter
DYNTREE@index	generic counter
DYNTREE@root	the root number

lengths

\DYNTREE@dynboxsep	the distance between dynkin boxes
\DYNTREE@levelsep	the distances between each level (from dynkin box bottom to top of next layer's dynbox)
\DYNTREE@leftmostX	the left most x value
\DYNTREE@rightmostX	the right most x value
\DYNTREE@unitlen	the unit length value before altering (to allow it to be restored)
\DYNTREE@templen	temporary length storage
\DYNTREE@holdlen	temporary length storage

commands

\DYNTREE@treestop	indicates the point where the gobbler stops reading (to allow LaTeXto properly read all the data. Its value is \&\&\&
\DYNTREE@treeend	indicates the end of the tree. It is in the definition of the gobbler. It has a value of \%\%\%

\lend
\dynbox

And there are two external commands:

\lend	indicates the end of one tree level. The value is never used: this token is used as a delimiter by the user and the code. Its value, which should never be typed, is \&\%\&\%
\dynbox	indicates the start of a dynbox. The value is never used: this token is purely for delimiting the start of the dynbox by the user; it is only defined to satisfy the \ifthenelse statement

```
25 \newcounter{DYNTREE@numlevel}%
```

```
\newcounter{DYNTREE@nextlevel}%
\newcounter{DYNTREE@numboxes}%
\newcounter{DYNTREE@nextnumboxes}%
\newcounter{DYNTREE@target}%
\newcounter{DYNTREE@listlen}%
\newcounter{DYNTREE@xCoord}%
\newcounter{DYNTREE@yCoord}%
\newcounter{DYNTREE@xPos}%
\newcounter{DYNTREE@yPos}%
\newcounter{DYNTREE@xComp}%
\newcounter{DYNTREE@yComp}%
\newcounter{DYNTREE@leftX}%
\newcounter{DYNTREE@ct}%
\newcounter{DYNTREE@counter}%
\newcounter{DYNTREE@index}%
\newcounter{DYNTREE@root}%
\newlength{\DYNTREE@dynboxsep}%
\newlength{\DYNTREE@levelsep}%
\newlength{\DYNTREE@leftmostX}%
\newlength{\DYNTREE@rightmostX}%
\newlength{\DYNTREE@unitlen}%
\newlength{\DYNTREE@templen}%
\newlength{\DYNTREE@holdlen}%
\newcommand{\DYNTREE@treestop}{\&\&\&}%
\newcommand{\DYNTREE@treeend}{\%\%\%}%
\newcommand{\lend}{\&\%\&\%}%
\newcommand{\dynbox}{}%
```

Now that they are declared, initialize the "exterior" ones

```
\setlength{\DYNTREE@dynboxsep}{\DYNTREE@colsepwidth}%
\setlength{\DYNTREE@levelsep}{1cm}%
```

2.2 The Tree Eater

While the dynkin tree structure resembles that of an environment, it actually consists of a LaTeXcommand that consumes the data, sorts it, and displays the proper items. The basis of this consumption are several "gobblers"—the first of which eats the tree one level at a time

```
\def\DYNTREE@gobbletree#1\lend#2\DYNTREE@treeend{%
```

Before the level can be processed, several things must be adjusted. First, since a new level is beginning the counter must be incremented by one

```
\addtocounter{DYNTREE@numlevel}{1}%
```

initialize the number of boxes for this level

```
\setcounter{DYNTREE@numboxes}{0}%
```

initialize the number of boxes for the next level

```
\setcounter{DYNTREE@nextnumboxes}{0}%
```

Process the level:

```
\DYNTREE@gobbledynboxes#1\lend%
```

Record the number of boxes for this level

```
\expandafter\xdef%
\csname DYNTREE@level@\roman{DYNTREE@numlevel}@numbox\endcsname%
{\arabic{DYNTREE@numboxes}}%
```

Now check for the signal to stop processing

```
\ifthenelse{ \equal{#2}{\DYNTREE@treestop} }%
{%
```

The End—just do nothing

```
}%
% Else
{%
```

continue processing levels until the end of the tree

```
\DYNTREE@gobbletree#2\DYNTREE@treeend%
}%
}%
```

2.3 The Box Eater

The second "gobbler" eats the boxes one by one:

```
\def\DYNTREE@gobbledynboxes#1\dynbox#2#3#4{%
```

increment the number of boxes

```
\addtocounter{DYNTREE@numboxes}{1}%
```

Store the boxes for this level in registers

```
\expandafter\newbox%
\csname DYNTREE@box@\roman{DYNTREE@numlevel}@\roman{DYNTREE@numboxes}\endcsname%
\expandafter\setbox%
\csname DYNTREE@box@\roman{DYNTREE@numlevel}@\roman{DYNTREE@numboxes}\endcsname=%
\hbox{$\begin{array}{|*{\DYNTREE@numroots}{c}|}\hline #2 \\ \hline \end{array}$}%
```

Calculate the X value for each descendent and place in sorted list
    * Get the length of the list

```
\listlenstore{DYNTREE@listlen}{#3}%
\expandafter\xdef
\csname DYNTREE@childline@\roman{DYNTREE@numlevel}@\roman{DYNTREE@numboxes}@boxnum\endcsname
{\arabic{DYNTREE@listlen}}%
\ifthenelse{\value{DYNTREE@listlen} > \DYNTREE@numroots}%
{%
\PackageError{dyntree}%
{%
Length of descendant of \arabic{DYNTREE@numboxes}%
  on level \arabic{DYNTREE@numlevel} exceeds number of roots%
  (\DYNTREE@numroots)%
  }%
```

```
}%
% Else
{}%
```

* store the list in a temp variable for convenience in typing, store it more permanently for use later on.

```
\liststore{#3}{DYNTREE@templist@}%
\liststore{#3}%
{DYNTREE@childlist@\roman{DYNTREE@numlevel}@\roman{DYNTREE@numboxes}@}%
```

* go through the list and generate the sorted 'array' `DYNTREE@level@`⟨*level*⟩`@`⟨*boxnum*⟩`@X`

```
\setcounter{DYNTREE@ct}{1}%
\whiledo{ \NOT \(\value{DYNTREE@ct} > \value{DYNTREE@listlen}\) \AND
\NOT\(\value{DYNTREE@ct} > \DYNTREE@numroots\) }%
{%
```

need to store the 'root' value in a counter to retrieve the data

```
\setcounter{DYNTREE@counter}{\csname DYNTREE@templist@\roman{DYNTREE@ct}\endcsname}%
```

check that the number submitted is within the allowed range

```
\ifthenelse{ \(\value{DYNTREE@counter} > \DYNTREE@numroots\) \OR
\(\value{DYNTREE@counter} < 1\)  }%
{%
\ifthenelse{\value{DYNTREE@counter} = 0}%
{%
```

Do nothing - this is the last level

```
}%
% Else
{%
\PackageError{dyntree}%
{%
Descendant root of \arabic{DYNTREE@numboxes} on level%
 \arabic{DYNTREE@numlevel} out of bounds%
 (\arabic{DYNTREE@counter} > \DYNTREE@numroots)%
}%
}%
}%
% Else
{%
```

temporarily store the length

```
\setlength{\DYNTREE@templen}%
{\csname DYNTREE@level@\roman{DYNTREE@numlevel}@\roman{DYNTREE@numboxes}@X\endcsname}%
\addtolength{\DYNTREE@templen}%
{\csname DYNTREE@rootX@\roman{DYNTREE@counter}\endcsname*(-1)}%
```

adjust the left most length

```
\ifthenelse{\DYNTREE@templen < \DYNTREE@leftmostX}%
{%
\setlength{\DYNTREE@leftmostX}{\DYNTREE@templen}%
}%
```

```
% Else
{}%
```
adjust right most length
```
\setlength{\DYNTREE@holdlen}{\DYNTREE@templen}%
\addtolength{\DYNTREE@holdlen}{\DYNTREE@dynboxwidth}%
\ifthenelse{ \DYNTREE@holdlen > \DYNTREE@rightmostX }%
{%
\setlength{\DYNTREE@rightmostX}{\DYNTREE@holdlen}%
}%
% Else
{}%
```
now store the x value for the line
```
\setlength{\DYNTREE@holdlen}%
{\expandafter\csname DYNTREE@dynboxX@\roman{DYNTREE@counter}\endcsname}%
\addtolength{\DYNTREE@holdlen}{\DYNTREE@templen}%
\setcounter{DYNTREE@xPos}{\DYNTREE@holdlen}%
```
`counter` has served its purpose and may be used in another context
add the length to the sorted 'array'
* initialize the counter to the END of the array
```
\setcounter{DYNTREE@counter}{\value{DYNTREE@nextnumboxes}}%
```
* get the value of the next level
```
\setcounter{DYNTREE@nextlevel}{\value{DYNTREE@numlevel} + 1}%
```
* Check for array elements
```
\ifthenelse{\value{DYNTREE@counter} = 0}%
{%
```
set the first element as the length
```
\expandafter\xdef%
\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@i@X\endcsname%
{\the\DYNTREE@templen}%
```
increment the number of elements
```
\addtocounter{DYNTREE@nextnumboxes}{1}%
}%
% Else
{%
```
there is at least one element
```
\edef\DYNTREE@temparray%
{\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@\roman{DYNTREE@counter}@X\endcsname}%
\edef\DYNTREE@tempinsert{\the\DYNTREE@templen}%
```
find where element should be inserted
```
\whiledo{ \(\value{DYNTREE@counter} > 0\) \AND
\lengthtest{\DYNTREE@tempinsert < \DYNTREE@temparray} }%
{%
\edef\DYNTREE@temparray%
{\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@\roman{DYNTREE@counter}@X\endcsname}%
```

```
\addtocounter{DYNTREE@counter}{-1}%
}%
```
the thing needs to be inserted at
```
\setcounter{DYNTREE@target}{\value{DYNTREE@counter} + 1}%
```
if they aren't equal, move from target to nextnumboxes up to target+1 to nextnumboxes+1
```
\ifthenelse{ \NOT \lengthtest{\DYNTREE@tempinsert = \DYNTREE@temparray} }%
{%
\setcounter{DYNTREE@counter}{\value{DYNTREE@nextnumboxes} + 1}%
\whiledo{ \value{DYNTREE@counter} > \value{DYNTREE@target} }%
{%
```
get the value in the array spot one before
```
\addtocounter{DYNTREE@counter}{-1}%
\edef\DYNTREE@tempswap%
{%
\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@\roman{DYNTREE@counter}@X\endcsname%
}%
```
store this value in the next array spot
```
\addtocounter{DYNTREE@counter}{1}%
\expandafter\xdef%
\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@\roman{DYNTREE@counter}@X\endcsname%
{\DYNTREE@tempswap}%
\addtocounter{DYNTREE@counter}{-1}%
}%
```
insert the original
```
\expandafter\xdef%
\csname DYNTREE@level@\roman{DYNTREE@nextlevel}@\roman{DYNTREE@target}@X\endcsname%
{\DYNTREE@tempinsert}%
```
increment the number of boxes in the next level
```
\addtocounter{DYNTREE@nextnumboxes}{1}%
}%
% Else
{%
```
Do nothing
```
}%
}%
}%
\addtocounter{DYNTREE@ct}{1}%
}%
\ifthenelse{\equal{#4}{\lend}}%
{%
}%
% Else
{%
```

Eat the boxes until there are no more

```
\DYNTREE@gobbledynboxes#4%
}%
}%
```

2.4 The Dyntree Environment

Dynkin Tree Environment

```
\def\start#1#2#3\finish#4%
{%
\ifthenelse{\equal{#1}{#4} \AND \equal{#4}{dyntree}}%
{%
\providecommand{\DYNTREE@numroots}{#2}%
```

Initialize Interior Dynkin Box Variables

```
\setlength{\DYNTREE@dynboxwidth}
{%
\DYNTREE@marginwidth +
\DYNTREE@cartancoefwidth*\DYNTREE@numroots +
\DYNTREE@colsepwidth*(\DYNTREE@numroots-1)
}%
```

Initialize Interior Dynkin Tree Variables

```
\setlength{\DYNTREE@leftmostX}{0pt}%
\setlength{\DYNTREE@rightmostX}{\DYNTREE@dynboxwidth}%
```

the highest left starts at zero, so initialize this value

```
\expandafter\gdef\csname DYNTREE@level@i@i@X\endcsname{0pt}%
```

There are no levels, so initialize `numlevel` to zero

```
\setcounter{DYNTREE@numlevel}{0}%
```

Determine the root lines and dynkin box offsets

```
\setcounter{DYNTREE@ct}{1}%
\whiledo{\NOT \(\value{DYNTREE@ct}>\DYNTREE@numroots\)}%
{%
```

Calculate the length and store it in a temporary length

```
\setlength{\DYNTREE@templen}%
{%
  (\DYNTREE@dynboxwidth + \DYNTREE@dynboxsep)*\value{DYNTREE@ct}%
 -(\DYNTREE@dynboxwidth + \DYNTREE@dynboxsep)*1/2%
 -(\DYNTREE@dynboxwidth + \DYNTREE@dynboxsep)*\DYNTREE@numroots/2%
}%
```

store the actual length in a command

```
\expandafter\xdef\csname DYNTREE@rootX@\roman{DYNTREE@ct}\endcsname%
{\the\DYNTREE@templen}%
```

Calculate the dynkin box x offset and store it in a temporary length

```
\setlength{\DYNTREE@templen}%
{%
```

```
\DYNTREE@dynboxwidth/2/\DYNTREE@numroots +
\DYNTREE@dynboxwidth*(\value{DYNTREE@ct}-1)/\DYNTREE@numroots
}%
```
store the actual length in a command
```
\expandafter\xdef\csname DYNTREE@dynboxX@\roman{DYNTREE@ct}\endcsname%
{\the\DYNTREE@templen}%
\addtocounter{DYNTREE@ct}{1}%
}%
```
Eat the tree
```
\DYNTREE@gobbletree#3\DYNTREE@treestop\DYNTREE@treeend%
```
Now the data has been stored, Draw The Tree:
store the current unit length to restore when finished
```
\setlength{\DYNTREE@unitlen}{\unitlength}%
\setlength{\unitlength}{1sp}%
```
get the width of the picture
```
\setcounter{DYNTREE@xCoord}{\DYNTREE@rightmostX - \DYNTREE@leftmostX}%
```
store the leftmost point as a counter
```
\setcounter{DYNTREE@leftX}{\DYNTREE@leftmostX}%
```
get the height of the picture
```
\setcounter{DYNTREE@yCoord}%
{%
\DYNTREE@dynboxvlen*\value{DYNTREE@numlevel} +
\DYNTREE@levelsep*(\value{DYNTREE@numlevel} - 1)
}%
\begin{picture}%
(\arabic{DYNTREE@xCoord},\arabic{DYNTREE@yCoord})%
(\value{DYNTREE@leftX},0)%
\setcounter{DYNTREE@ct}{1}%
\whiledo{\NOT \(\value{DYNTREE@ct} > \value{DYNTREE@numlevel}\)}%
{%
\setcounter{DYNTREE@counter}{1}%
```
get the y coordinate as a length
```
\setlength{\DYNTREE@templen}%
{%
  \DYNTREE@dynboxvlen*\value{DYNTREE@numlevel}
- \DYNTREE@dynboxvlen*\value{DYNTREE@ct}
+ \DYNTREE@levelsep*\value{DYNTREE@numlevel}
- \DYNTREE@levelsep*\value{DYNTREE@ct}
+ \DYNTREE@dynboxdepth
}%
```
convert the length to an integer in scaled points (sp)
```
\setcounter{DYNTREE@yCoord}{\DYNTREE@templen}%
\def\DYNTREE@tempboxnum{\csname DYNTREE@level@\roman{DYNTREE@ct}@numbox\endcsname}%
\whiledo{\NOT \( \value{DYNTREE@counter} > \DYNTREE@tempboxnum \)}%
{%
```

grab the value of the x coordinate (it's a length but not stored as one)

```
\xdef\DYNTREE@tempxCoord%
{%
\expandafter%
\csname DYNTREE@level@\roman{DYNTREE@ct}@\roman{DYNTREE@counter}@X\endcsname%
}%
```

convert x coordinate to a length

```
\setlength{\DYNTREE@holdlen}{\DYNTREE@tempxCoord}%
```

convert length to an integer in scaled points (sp)

```
\setcounter{DYNTREE@xCoord}{\DYNTREE@holdlen}%
```

place each dynkin box

```
\put(\arabic{DYNTREE@xCoord},\arabic{DYNTREE@yCoord})%
{%
\expandafter%
\copy\csname DYNTREE@box@\roman{DYNTREE@ct}@\roman{DYNTREE@counter}\endcsname%
}%
```

go through the descendants and place the lines

```
\setcounter{DYNTREE@listlen}%
{%
\expandafter%
\csname DYNTREE@childline@\roman{DYNTREE@ct}@\roman{DYNTREE@counter}@boxnum\endcsname%
}%
\setcounter{DYNTREE@index}{1}%
\whiledo{\NOT \(\value{DYNTREE@index} > \value{DYNTREE@listlen}\)}%
{%
\xdef\DYNTREE@childroot%
{%
\expandafter%
\csname DYNTREE@childlist@\roman{DYNTREE@ct}@\roman{DYNTREE@counter}@\roman{DYNTREE@index}\e
}%
\ifthenelse{\NOT \DYNTREE@childroot = 0 }%
{%
\setcounter{DYNTREE@root}{\DYNTREE@childroot}%
\setcounter{DYNTREE@xPos}{\value{DYNTREE@xCoord}}%
\setcounter{DYNTREE@xComp}{\value{DYNTREE@xCoord}}%
\setcounter{DYNTREE@yPos}{\value{DYNTREE@yCoord}}%
\setcounter{DYNTREE@yComp}{\value{DYNTREE@yCoord}}%
\xdef\DYNTREE@temprootX%
{\expandafter\csname DYNTREE@rootX@\roman{DYNTREE@root}\endcsname}%
\setlength{\DYNTREE@templen}{\DYNTREE@temprootX}%
\addtocounter{DYNTREE@xPos}{\DYNTREE@templen*(-1)}%
\xdef\DYNTREE@dynoffset%
{\expandafter\csname DYNTREE@dynboxX@\roman{DYNTREE@root}\endcsname}%
\setlength{\DYNTREE@templen}{\DYNTREE@dynoffset}%
\addtocounter{DYNTREE@xPos}{\DYNTREE@templen}%
\addtocounter{DYNTREE@xComp}{\DYNTREE@templen}%
\setlength{\DYNTREE@templen}{1pt}% for frame thickness
```

```
\addtocounter{DYNTREE@yPos}%
{\DYNTREE@levelsep*(-1) - \DYNTREE@dynboxdepth - \DYNTREE@templen}%
\addtocounter{DYNTREE@yComp}{\DYNTREE@dynboxdepth*(-1) - \DYNTREE@templen}%
\put(0,0)%
{%
\drawline%
(\arabic{DYNTREE@xComp},\arabic{DYNTREE@yComp})%
(\arabic{DYNTREE@xPos},\arabic{DYNTREE@yPos})%
}%
}%
% Else
{%
}%
\addtocounter{DYNTREE@index}{1}%
}%
\addtocounter{DYNTREE@counter}{1}%
}%
\addtocounter{DYNTREE@ct}{1}%
}%
\end{picture}%
```

restore the unit length to original value

```
\setlength{\unitlength}{\DYNTREE@unitlen}%
}%
% Else
{%
\PackageError{dyntree}{Invalid start(#1)/finish(#4) call}%
}%
}%
```

Change History

v1.0

General: Initial Release 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

E

F

H

L

S