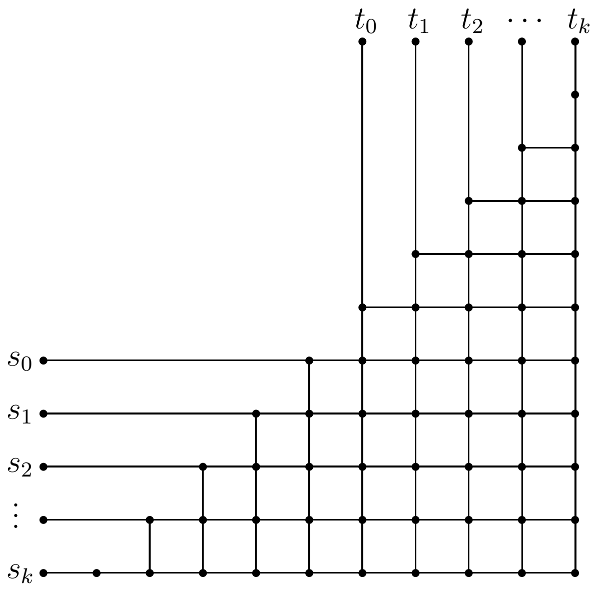
Drawing Pictures in LaTeX

Besides including pictures drawn in other software, there may be times when you want to draw a picture right in your LaTeX document. One of the advantages of this is that any labels in your pictures can be in the same fonts that LaTeX uses for everything else. This is tricky to manage, though not impossible, if your picture came from somewhere else.

The `picture` environment

LaTeX provides the `picture` environment for drawing simple pictures. Here's an example of what you can do:



This can get you, well, a little ways. The `picture` environment works by creating your picture as a series of characters, placed appropriately. For instance, if you want a line of slope $\frac{1}{2}$ and length 5, you say `\line(2,1){5}`. Somewhere in one of LaTeX's fonts is a character which is a piece of a line of slope $\frac{1}{2}$. When you compile your file, LaTeX creates a document with that character placed multiple times in appropriate places, and the effect is to draw a line.

This makes the approach very generic, so that it can be put into any format that can deal with characters. But of course you see the problem—LaTeX can't possibly have a character for every possible slope you might want to draw. In fact the denominator of your slope must be no greater than 6. So there is only so much you can do with the picture environment.

Postscript

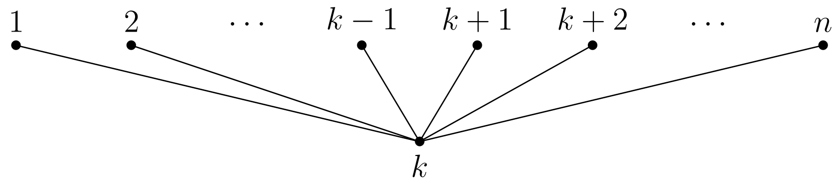
When you compile a LaTeX file, the compiler creates a `.dvi` file in the same directory. This is a "device independent" file, which simply contains instructions about where to put characters on a page. The intention of the creators of TEX and LATEX was to avoid marrying the system to a particular style of output, and therefore the idea was to allow anybody who favored a particular output format to write a driver program to read in a `.dvi` file and convert it appropriately. Thus, for example, there are drivers to convert `dvi` files to `pdf` files and `png` files. Wikipedia uses the `png` driver to display its formulas.

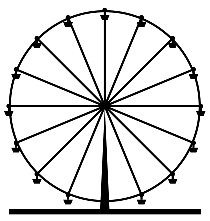
Most users of LATEX have settled on PostScript as their favorite output format. PostScript is a language created by Adobe Systems for their laser printers. It is very good at describing how to plot both pictures and text on a printed page. Most of the Postscript in the world is generated automatically by software from some higher-level format—for example if you have a postscript printer attached to your Microsoft Windows computer, you probably have a printer driver on your machine which takes output from Microsoft Word and converts it to Postscript just before sending it to the printer.

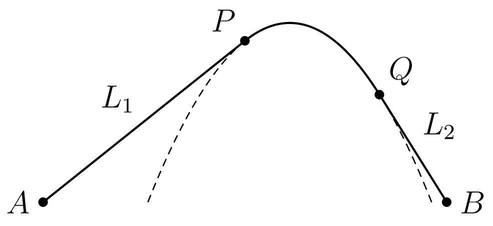
What most people don't know, though, is that Postscript is a full blown programming language, and a really cool one at that. It's a little hard to read at first, and the syntax is unusual, but once you get used to it you can do a lot of cool stuff.

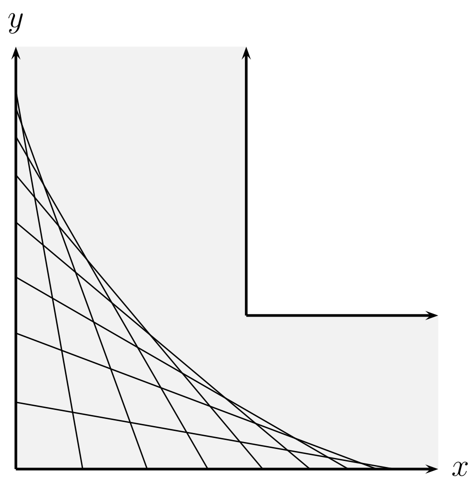
PSTricks

PSTricks is a LATEX package for drawing pictures using Postscript. It acts as a kind of intermediary so that you can take advantage of much of the power of Postscript without learning all the syntax. Here are some samples of what you can do with PSTricks:









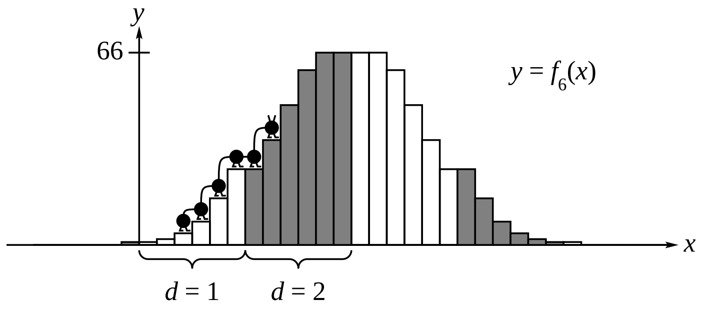
You should be able to work out how to do most simple things by looking at the code for these samples. The manual for PSTricks can be found at:

`http://www.tex.uniyar.ac.ru/doc/pst_ug.pdf`.

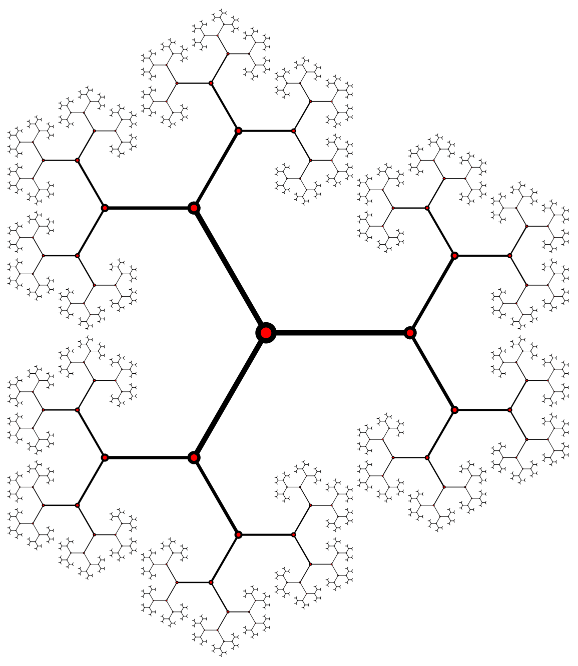
Making EPS pictures

Of course for the really coolest of pictures, nothing beats programming right in Postscript. Since Postscript is a real programming language, you can have it compute all the dimensions of your picture from a few parameters you specify at the beginning, then adjust the parameters to be exactly what you want. You can even have it compute entire mathematical functions and display them.

For instance this figure was written in Postscript, and saved as EPS (encapsulated Postscript), which is just the same as Postscript with dimensions specified for the box it occupies. It computes the values of the heights of the rectangles before displaying them. One can change a single line of code and have the graph for $n = 7$ display instead of $n = 6$.



And since it's a real programming language, you can even do recursion in Postscript. This allows you to create figures that would be nearly impossible in a WYSIWYG picture editor.



Postscript syntax looks more like assembly language than like a high level programming language, so it seems strange at first. It uses a stack to pass parameters and do most everything else; for instance, to translate the origin to the center of the page you might say:

```
306 396 translate
```

which means, "Put 306 on the stack, then put 396 on the stack, then pop two arguments off the stack and make them the new origin." By default the units of length are points ($\frac{1}{72}$ of an inch).

Learning raw Postscript might be beyond the scope of what you want to do in this course. But keep it in mind when you want to create a really impressive figure.