Graphs in LaTeX

Robert A. Beeler*

January 8, 2017

1 Introduction

This document is to provide a "quick and dirty" guide for building graphs in LaTeX. Much of the document is devoted to examples of things that you can do with graphs in LaTeX. No doubt that there are a number of things that I have neglected to include in this document. Mainly, this is because there are things that I have never had reason to do (yet) and haven't figured out. This being said, if you figure something out that I haven't figured out, then I would be honored if you would let me know. This way, I can add these document next time I update it. In most cases, I have simply included output in this document. However, as I am also including the corresponding Tex code, you can generate it yourself.

First of all, you will need to add several packages: **pstricks, pst-node,** and **pst-plot**. These can be added in the preamble using the command `\usepackage{pstricks,pst-node,pst-plot}`.

Next, a word about compiling your code. I use the WinEdt compiler (MiKTeX). When compiling, I get the best result if I use the LaTeXcompiler (this creates a dvi file). I then convert the dvi file to a ps (postscript) file. Finally, I convert the postscript to a pdf.

*Department of Mathematics and Statistics, East Tennessee State University, Johnson City, TN 37614-1700 USA email: beelerr@etsu.edu

2 Basic commands

First of all, you need to create an environment for your graph. To begin this, use the command `\pspicture(a,b)`. You end this environment with the command `\endpspicture`. In the `\pspicture(a,b)`, you fill in specific numbers for a and b. This creates a "box" for you to work in. This box is bounded by the lines $x = 0$, $x = a$, $y = 0$, and $y = b$.

NOTE: LaTeXis so powerful that you can break the rules of LaTeXwithin LaTeX. So you "can" place points outside of the box that you created. However, you may get unexpected results.

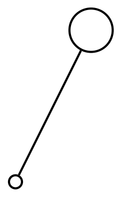
Next, you want to populate the box with vertices. This is done with the command `\cnode[](x,y){size}{label}`. The brackets can be filled with options, which we will talk about later. By default, you have a hollow vertex. The (x, y) gives the Cartesian coordinate where you will place the vertex. Instead of *size*, you will place a numeric value which determines the size of the vertex. Finally, you replace *label* with a mnemonic that you can refer to later. So, `\cnode[](1.5,0){.3}{ne}` creates a hollow vertex of size .3 at point (1.5,0) with label ne.

After you have placed several vertices in your box, you can connect them. This is done with the command `\ncline[]{vertex1}{vertex2}`. The brackets can be filled with options that we will discuss later. "vertex1" you will replace with the mnemonic of the first endpoint of the edge. Likewise, "vertex2" will be replaced with the label of the second endpoint of the edge.

Now, let's pull it all together. Suppose that we want a box of width 2 and height 4. There are two vertices. The first is size .1 and is at (1,1) (label v1). The second is at (2,3) and is size .3 (label v2). They are connected by an edge. Here is the required code:

```
\pspicture(2,4)
\cnode[](1,1){.1}{v1}
\cnode[](2,3){.3}{v2}
\ncline[]{v1}{v2}
\endpspicture
```

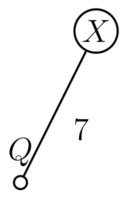
Here is the result:



Notice that the assigned labels don't actually appear in your graph. For this reason, you may also want to add labels to your graph. This is accomplished with the command `\rput(x,y){text}`. Again, (x, y) gives the coordinates where you will place the label. Replace *text* with your desired label. Suppose that we want to modify the above graph to place label Q above the vertex at (1,1), label X inside of the vertex at $(2, 3)$, and label 7 below and to the right of the edge. This is accomplished with the code:

```
\pspicture(2,4)
\cnode[](1,1){.1}{v1}
\cnode[](2,3){.3}{v2}
\rput(1,1.4){$Q$}
\rput(2,3){$X$}
\ncline[]{v1}{v2}
\rput(1.8,1.7){7}
\endpspicture
```

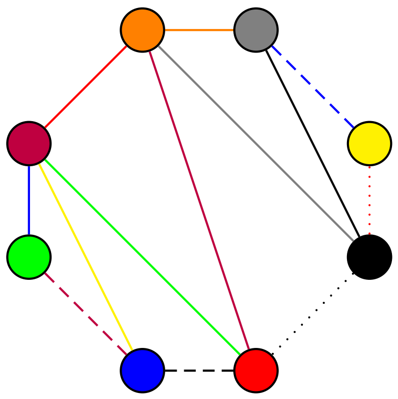
Here is the result:



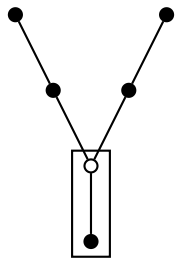
NOTE: When using `\rput` it sometimes takes a bit of trial and error to get your labels right where you want them. This is especially true when labeling edges or graphs with a lot of vertices or edges.

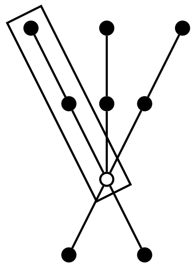
3 Options and variations

One option that you may want to take advantage of is color. Color can be added to be added to both edges and vertices. There are various colors available, including blue, red, yellow, orange, purple, green, black, and gray. For vertices, add the line `fillcolor=color, fillstyle=solid` inside the braces. Likewise, we can add color to edges using the command `linecolor=color` inside of the braces. As usual, you replace *color* with your chosen color. Often, journals want to print in black and white, but you still want to distinguish types of vertices. One way is to use dotted and dashed lines. This is done by adding the lines `linestyle=dotted` and `linestyle=dashed`, respectively, within the braces of `\ncline`. A sample output that mixes the different options is below.

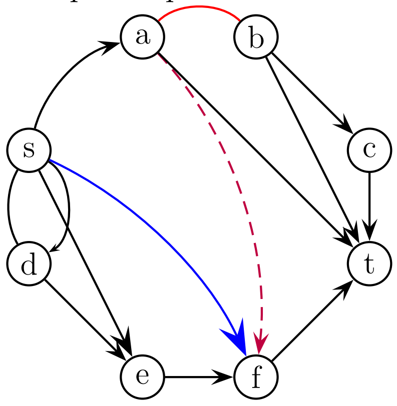


Another way to distinguish vertices is with the `\ncbox` command. Two example outputs are below.

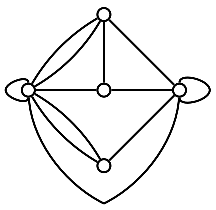




Another option is to use multiple edges or directed edges. Normally, LaTeXcreates straight lines between vertices. To give it a bit of curve, use the command `\ncarc[arcangle=a]{v1}{v2}`. Here, *a* is the angle of arc, relative to the straight between the vertices labeled v1 and v2. Note that you can add the various color and style options to the `\ncarc` command. To create a directed edge, we can modify either the `\ncline` or the `\ncarc` command by including `{->,arrowsize=size}` after the braces. Note that you need to replace *size* with how big you want the arrow to be (say 6pt). A sample output is below.



Making loops can be tricky. One option is to use the `\ncarc` command along with an “invisible vertex”. To create an invisible vertex, use the command `\cnode*(x,y){0}{label}`. This is useful if you need a loop or to make an edge go around a group of vertices. An example output is given below.



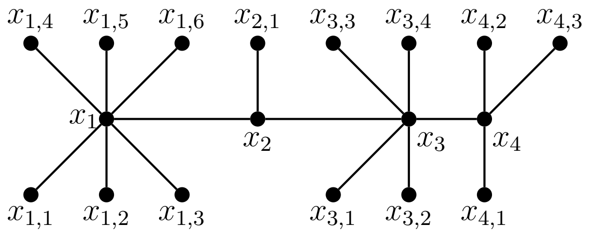


Figure 1: The caterpillar $P_4(6, 1, 4, 3)$

4 Figures and tables

Usually, when we include a graph in either a paper or a thesis, we place it in a figure environment. This allows us to caption it and refer to it later. To enter into the figure environment, use `\begin{figure}`. To end the figure environment, use `\end{figure}`. You can add captions and labels as usual. For best results, I usually place the graph itself in the center environment. To enter this environment, use `\begin{center}`. To end it, use `\end{center}`. By default, figures are placed on the next available page, as we see with Figure 1 (which appears immediately after this paragraph in the code).

Other options for the figure environment are `\begin{figure}[t]` (places the figure at the top of the current page), `\begin{figure}[h]` (places the figure HERE in the text), and `\begin{figure}[b]` (places the figure at the bottom of the current page). In my experience, it is usually best to avoid these qualifiers and let LaTeXdetermine the best place to put your figure.

Often times, we want multiple graphs to appear next to each other or on top of each other in the same figure. This is accomplished using the tabular environment. An example output is given below in Figure 2.

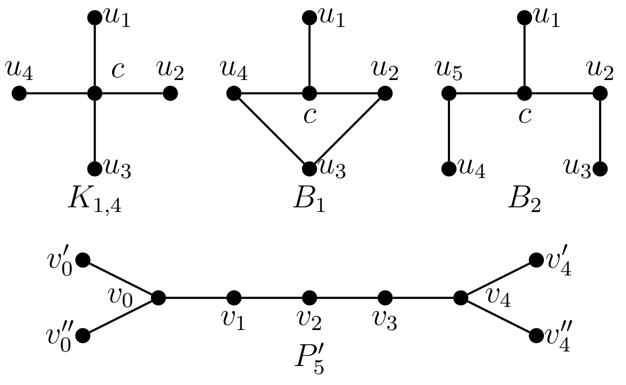
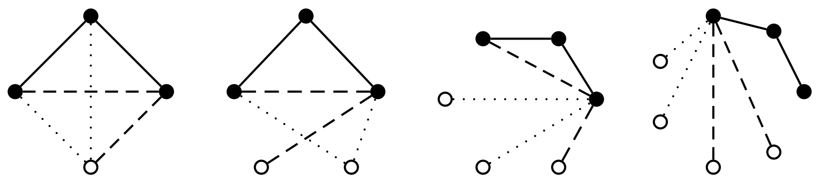
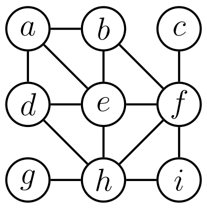


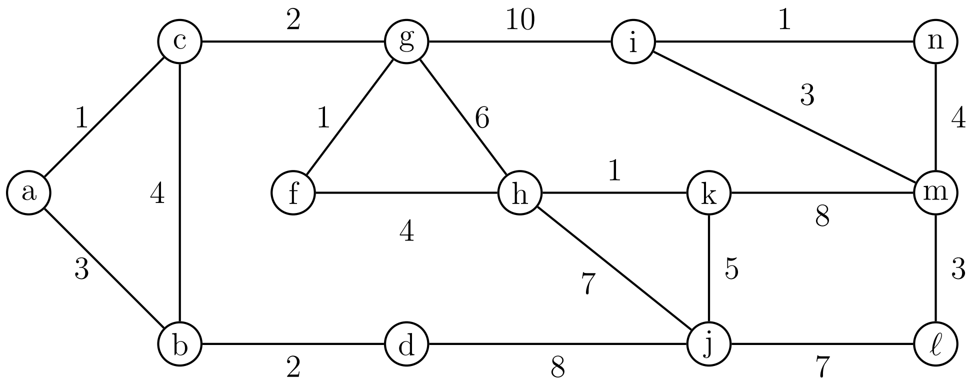
Figure 2: A tabular example

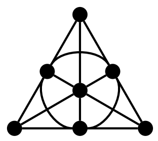
5 Additional examples

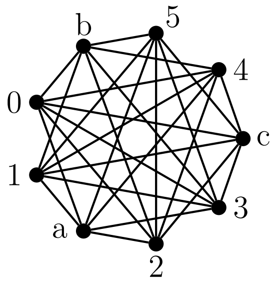
We end this manual by providing several templates for graphs of various sizes. As usual,I simply provide the output in this document and refer you to the associated tex code.






a
b
c
d
f
g
h
i
j
k
ℓ
m
n
1
3
4
2
2
1
6
4
10
1
3
4
1
8
7
5
8
3
7




b
5
4
0
c
1
3
a
2