Skidmore College

The Monkey See, Monkey Do Guide to Tikz

Travis McVoy

January 19, 2024

Introduction

* put intro here lol *
Some tricks (I don't know where to put the tricks yet).
Desmos is helpful. Remembering to partition a circle is helpful for regular polygons.

Contents

1 General Tikz Information

Tikz can do just about anything that you want, but much of it builds on simpler concepts, like shapes and nodes. In order to make anything work, though, we will need to include packages in our preamble.

1.1 Packages

I use several packages throughout this document that are necessary to generate specific types of diagrams and/or plots. I've specified what each package is used for, but you can also just copy and paste the GRAPHS/PLOTS PACKAGES section in my preamble.

Whenever one is making graphs (graph theory graphs, not plots of functions), one should include `\usetikzlibrarygraphs,graphs.standard` in their preamble in addition to the tikz package.

If one desires to have rather fancy nodes, they will likely need `\usetikzlibraryshapes` in their preamble in addition to the tikz package.

1.2 Shapes

* STILL NEED TO DO SHAPES *

For rectangles, you specify two opposite corners, and that's how it knows where to place the rectangle. E.g. a rectangle with corners 0,0 and 2,2 is a square, centered at 1,1 with side length 2.

1.3 Nodes

Nodes enable us to label, and even construct, a diagram with ease. A node is essentially just a place in space on the page that can contain words or shapes. To create a node, we use the following code

```
\begin{center}
\begin{tikzpicture}
\node (node1) at (0,0) {This is a node};
\end{tikzpicture}
\end{center}
```

which has the output

This is a node

1.3.1 Lines between nodes

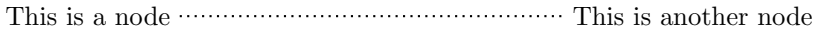
When creating/defining a node, there are a few basic elements we wish to understand. When we define a node, we use the `node` command, we place the name of the node in parentheses, we specify a location, and we place any text or symbols we wish to display in curly braces. The naming of the node helps us refer to it later. For example, suppose we wish to draw a thick dashed line between two nodes. We can do that with

```
\begin{center}
\begin{tikzpicture}

\node (node1) at (0,0) {This is a node};
\node (node2) at (8,0) {This is another node};

\draw[thick, dashed] (node1) -- (node2);

\end{tikzpicture}
\end{center}
```

which produces

This is a node - This is another node

We can also do dotted lines by changing `dashed` to `dotted`:



Admittedly, the dotted line isn't very thick, but it's thicker than it would be had we used the default settings:

If we so wish, we can specify the size of the dots or dashes by using the `line width` parameter. We write

```
\begin{center}
\begin{tikzpicture}

\node (node1) at (0,0) {This is a node};
\node (node2) at (8,0) {This is another node};

\draw[line width = 1.5, dotted] (node1) -- (node2);

\end{tikzpicture}
\end{center}
```

which produces

1.3.2 Nodes and shapes

In the previous section, I used words to specify nodes, but we can easily use shapes, too. We can put shapes around text, like so

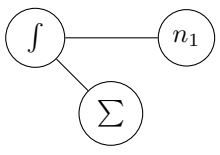
To achieve the above we barely change our previous code. Instead of defining a plain node, we add paramters:

```
\begin{center}
\begin{tikzpicture}

\node[draw, rectangle] (node1) at (0,0) {This is a rectangle node};
\node[draw, ellipse] (node2) at (8,0) {This is an ellipse node};

% NOTE: The ellipse shape won't work without the shapes package.
% See the packages section for more info.

\draw[line width = 1.5, dotted] (node1) -- (node2);

\end{tikzpicture}
\end{center}
```

There are many other shapes one can use, but I won't cover them all. See here for more information.

1.3.3 Nodes and Math

I've only used words so far, but nodes can contain mathematical symbols:



For reference, I achieved the diagram above with the code

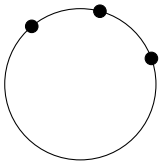
```
\begin{center}
\begin{tikzpicture}
\node[draw, circle] (n1) at (0,1)  {$n_1$};
\node[draw, circle] (n2) at (-2,1) {$\int$};
\node[draw, circle] (n3) at (-1,0)  {$\sum$};

\draw (n1) -- (n2);
\draw (n2) -- (n3);
\end{tikzpicture}
\end{center}
```

Notice that to put mathematical symobls in nodes, we place the math we wish to display in the curly braces.

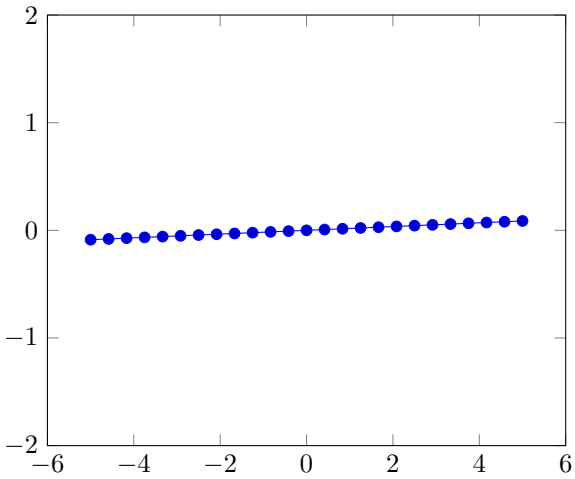
1.3.4 Cartesian vs. Radial Coordinates

So far, I've specifed the location of a node using cartesian coordinates, but this is not always ideal. Suppose we wish to draw a shape in which finding cartesian coordinates is unpleasant (points on a circle, perhaps). We can draw points using radial coordinates instead:

```
\begin{center}
\begin{tikzpicture}

% Radial cood: Theta is in degrees.  Radius in cm.

% nodes at 20, 75, and 130 degrees
\node[draw, circle, fill=black, scale=0.5] (n1) at (20:1cm) {};
\node[draw, circle, fill=black, scale=0.5] (n2) at (75:1cm) {};
\node[draw, circle, fill=black, scale=0.5] (n1) at (130:1cm) {};

% circle to visually verify degrees
\draw circle (1cm);

\end{tikzpicture}
\end{center}
```



2 Functions



As you can see, I still haven't figured out functions. It turns out plotting functions in Tikz is doable, but not easy. Python or desmos might be better.

3 Graph Theory

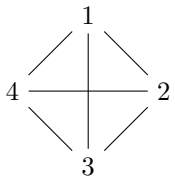
We can categorize graphs by those that have existing packages and those that do not. Cycle graphs, for example, are quite easy to draw as there exists commands that will draw them for us. Bipartite graphs, on the other hand, will need to be coded by hand (don't worry, it's not too hard).

3.1 Complete Graphs and Cycle Graphs

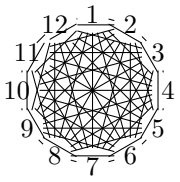
If we include `\usetikzlibrary{graphs, graphs.standard}` in our preamble, we can easily graph cycle and complete graphs with the following code. For a complete graph, we would use

```
\begin{center}
\begin{tikzpicture}
\graph { subgraph K_n [n=4, clockwise, radius=1cm] };
\end{tikzpicture}
\end{center}
```

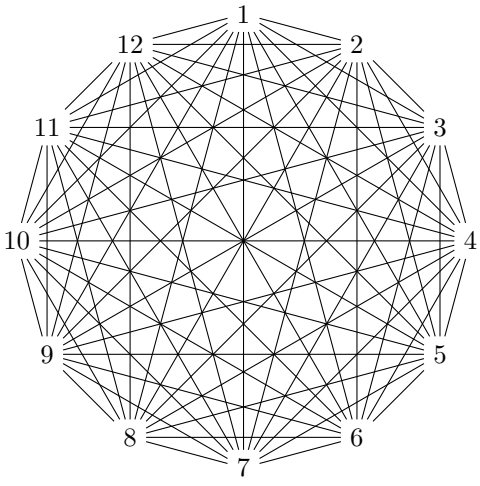
which generates the following graph:



The paramter `n=4` tells us how many verticies there are in the complete graph. We can up that number as we choose. Setting `n=12` yields



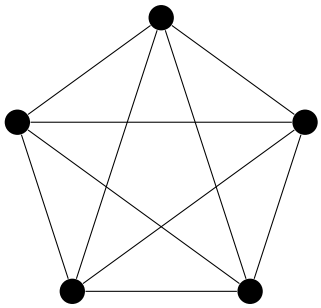
which clearly doesn't look very nice. We can fix this by adjusting the `radius` parameter. If we set `radius=3cm`, the scale of the graph will increase, which will declutter everything:



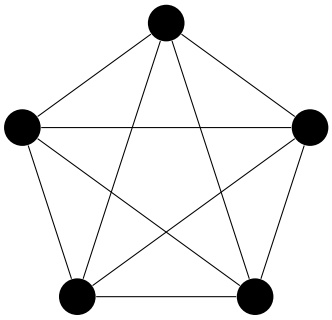
The `clockwise` parameter is just for the numbering of the verticies. Setting the parameter to `counterclockwise` will reverse the direction. I leave it to the reader to verify. One might wonder if we can replace the numbered verticies with circular nodes. Indeed we can with the following code

```
\begin{center}
\begin{tikzpicture} [every node/.style={fill,circle, inner sep=0.5pt}]
\graph {subgraph K_n [n=5, clockwise, radius=2cm]};
\end{tikzpicture}
\end{center}
```

which produces the graph below.



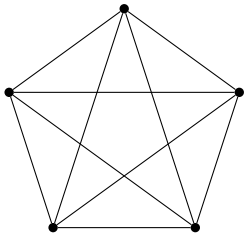
If we wish to make our nodes larger, we can do so by adjusting the `sep` parameter. By setting `sep=2pt` we achieve the following:



I find that there is a limit to how small we can make the nodes when using the `sep` parameter, but there is a way around this. Instead of using the `inner sep = 2pt` command, we can scale the entire tikz drawing. When we do that, the nodes will change in size, as will the entire drawing. To fix this, we can simply crank up the radius, which will effectively decrease the node size. We achieve that with the code

```
\begin{center}
\begin{tikzpicture} [every node/.style={fill,circle, scale = 0.2}]
\graph {subgraph K_n [n=5, clockwise, radius=8cm]};
\end{tikzpicture}
\end{center}
```

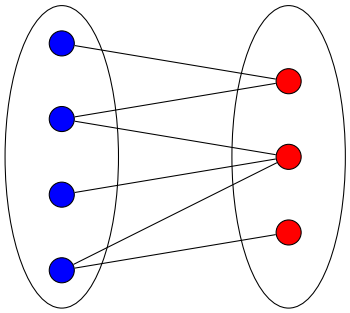
which produces the graph



One might notice that the section includes cycle graphs but thus far we have only done complete graphs. Everything we've learned about complete graphs applies to cycle graphs, only we change `K_n` to `C_n`.

3.2 Partite Graphs

So far as I can tell, there is not a nice package for bipartite (or multi partite) graphs as there is for cycle and complete graphs. Fortunately, partite graphs aren't hard to make. Effectively we will graph it using coordinates and ellipses. For an example, suppose we wish to make the following graph:



We could make a rough sketch of the above graph in desmos by assigning coordinates to each vertex and centering an ellipse around those coordinates. Click here to see how I constructed the graph in desmos. The power of desmos is in the sliders. Adjusting the sliders makes it really easy to adjust the ellipse to your liking. Once you know the coordinates of the verticies and the paramters for the ellipse, we use the following code:

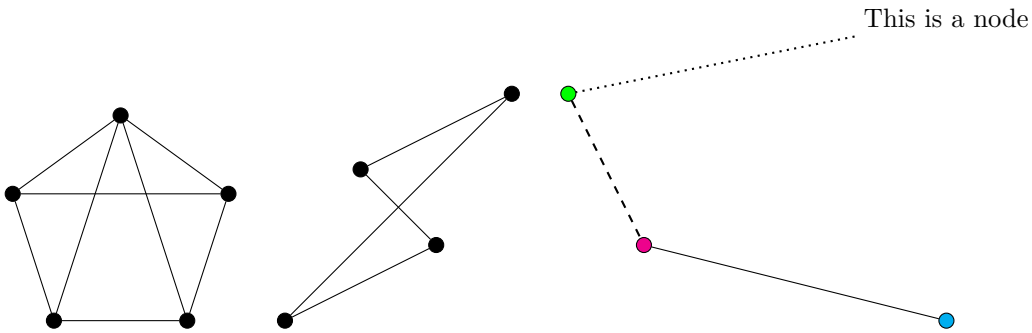
```
\begin{center}
\begin{tikzpicture}

% LEFT PARTITION VERTICIES:
\node[circle, draw, fill=blue] (v1) at (-2,0) {};
\node[circle, draw, fill=blue] (v2) at (-2,1) {};
\node[circle, draw, fill=blue] (v3) at (-2,2) {};
\node[circle, draw, fill=blue] (v4) at (-2,3) {};

% RIGHT PARTITION VERTICIES:
\node[circle, draw, fill=red] (u1) at (1,0.5) {};
\node[circle, draw, fill=red] (u2) at (1,1.5) {};
\node[circle, draw, fill=red] (u3) at (1,2.5) {};

% CONNECTING VERTICIES
\graph {
(v1) -- (u2);
(v1) -- (u1);
(v3) -- (u3);
(v3) -- (u2);
(v2) -- (u2);
(v4) -- (u3);
};

% ELLIPSES
\draw (-2,1.5) ellipse (0.75cm and 2cm);
\draw (1,1.5) ellipse (0.75cm and 2cm);

\end{tikzpicture}
\end{center}
```

Remarks.

1. If you don't include the `graphs` package in your preamble, connecting the verticies as I've done won't work.
2. If you want to make the node circles bigger or smaller, you can add a scale parameter. So, instead of `[circle, draw, fill=red]` you would type `[circle, draw, fill=red, scale = 0.5]`.
3. If you want to change the lines connecting verticies, you can, but I only know how to do that using the `draw` command. See the nodes section for more information.

3.3 Arbitrary Graphs

Suppose we wish to draw a graph that doesn't necessarily fit a particular family of graphs. A few examples may be



To draw arbitrary graphs, we will simply use what we know about nodes. We specify a node's location, and then connect accordingly.