Graphviz

Nathan Warner

Computer Science
Northern Illinois University
United States

Contents

Getting started

Install graphiz with your package manager

```
sudo pacman -Sy graphviz
```

Create a graphviz file with the .dot extension, generate pngs from your graphs with

```
dot -Tpng mygraph.dot -o outputname.png
```

Directed and undirected graphs

2.1 Directed graphs

Create directed graphs with the following syntax

```
graph G {
    ...
}
```

Where G is the name of the graph

Create an edge connecting two nodes with

```
graph G {
    A -- B
}
```

Connect a vertex to multiple vertices with

```
graph G {
    A -- {B1, B2, B3}
}
```

2.2 Digraphs

Create digraphs with

```
digraph G {
    ...
}
```

Connect vertices with

```
digraph G {
    A -> B
    B -> {C,D}
}
```

Styles

3.1 Node styles

Define node styles with

```
graph G {
    node [ key=value, ... ]
}
```

For example,

```
graph G {
    node [ style=filled, color=red ]
}
```

3.1.1 Shape

Change the shape of the nodes with

```
graph G {
    node [ shape= ]
}
```

Options for shape include

- Ellipse
- Circle
- Box
- Diamond

3.1.2 Label

Specifies the text displayed within the node.

```
graph G {
    node [ label="..." ]
}
```

3.1.3 Color

Sets the color of the node's border.

```
graph G {
    node [ color=yellow ]
}
```

3.1.4 fillcolor

Determines the interior fill color (effective when combined with a style such as filled).

```
graph G {
    node [ style=filled, fillcolor=red ]
}
```

3.1.5 Style

Alters the appearance (e.g., filled, dashed, rounded).

```
graph G {
    node [ style=dashed ]
}
```

3.1.6 fontname, fontsize, fontcolor

Customize the font type, size, and color for the label.

3.1.7 width and height

Control the dimensions of the node.

3.1.8 Penwidth

Adjusts the thickness of the node border.

3.1.9 margin

Provides spacing between the label and the node border.

3.2 Styling specific nodes

In Graphviz, you can override the default node attributes by specifying attributes directly for that node. For example, if you want most nodes to have one style but a specific node to have a different style, you can define the default attributes at the top and then override them for the specific node.

```
digraph Example {
    // Global node attributes
    node [shape=box, style=filled, fillcolor=lightblue];

    // Default nodes
    A;
    B;
    C;

    // Override node B's style
    B [fillcolor=yellow, style=rounded];

    // Create some connections
    A -> B;
    B -> C;
}
```

3.3 Edges

- **label:** Sets a text label on the edge.
- **color:** Specifies the color of the edge line.
- **style:** Defines the style of the edge (e.g., solid, dashed, dotted, bold, invis).
- **penwidth:** Determines the thickness of the edge line.
- **arrowhead:** Sets the style of the arrow at the head end (e.g., normal, vee, dot, none).
- **arrowtail:** Sets the style of the arrow at the tail end.
- **dir:** Controls the direction of the arrow; can be forward, back, both, or none.
- **fontsize:** Specifies the size of the font used for the label.
- **fontname:** Sets the font used for the label text.
- **fontcolor:** Specifies the color of the label text.
- **weight:** Provides a relative weight that influences the layout; heavier edges tend to pull nodes closer.
- **constraint:** A boolean that, when true, makes the edge influence the layout positioning.
- **headlabel / taillabel:** Allows you to add labels near the head or tail of the edge.
- **headport / tailport:** Specify which part of the node boundary the edge should connect to.

3.4 Layouts

Graphviz offers several layout engines, each suited for different types of graphs and visual styles

- **dot (default):** Uses a hierarchical (layered) approach, making it ideal for directed graphs and flowcharts.
- **neato:** Implements a spring model (force-directed) layout suitable for undirected graphs.
- **fdp:** Similar to neato but uses a different force-directed algorithm; good for undirected graphs.
- **sfdp:** A scalable force-directed layout engine designed for larger graphs.
- **twopi:** Produces radial layouts by positioning nodes in concentric circles around a central point.
- **circo:** Generates circular layouts, particularly effective for cyclic graphs.

```
graph G {
    layout=neato
}
```