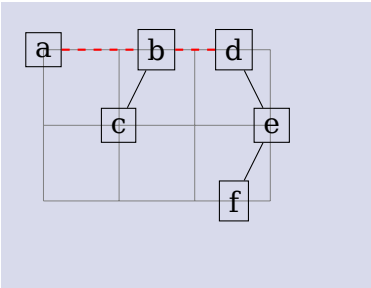
tikz graph 的节点对齐 (component align)

宿宝臣<subaochen@126.com>

2017 年 5 月 19 日

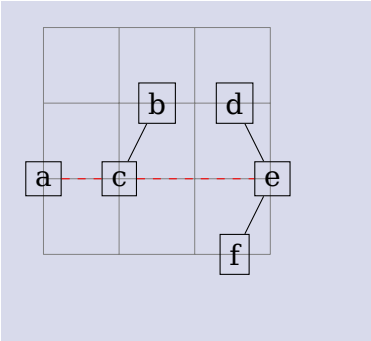
1 基本概念

tikz graph 在绘制图形的时候，首先从左向右放置节点，然后再根据横向对齐线进行垂直方向的对齐，比如下例中，首先横向从左向右放置好 a、b、d 三个节点，然后根据 a、b、d 三个节点决定其他节点的方位。为了清楚起见，在图中增加了网格线和红色的对齐线：



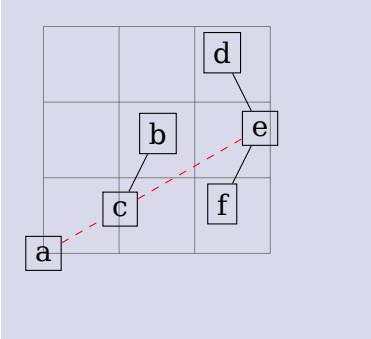

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,0)grid(3,-2);
   \graph [binary tree layout, nodes={draw}]{
       a, b -- c, d -- e[second] -- f
   };
   \draw[red,thick,dashed] (a) -- (b) -- (d);
\end{tikzpicture}
```

如果我们希望 a、c、e 三个节点在一个水平线上，如何操作呢？可以借助于 align here 命令，只要使用 align here 声明 c、e 两个节点即表示 c、e 节点位于对齐线上，或者说，对齐线横穿 c、e 两个节点的中心。由于整个图形是从 a 节点开始绘制的，即 a 节点的坐标为（0,0），对齐线即为穿过 a 节点的横线，如下图所示：




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,-1)grid(3,2);
   \graph [binary tree layout, nodes={draw}]{
       a, b -- c[align here], d -- e[second,align
         here] -- f
   };
   \draw[red,thin,dashed] (a) -- (c) -- (e);
\end{tikzpicture}
```

对齐线不一定是水平的，也不一定是垂直的，这和图形的 component direction 设置有关。component direction 的默认值是 0 即水平方向，这就是为什么大部分情况下（comoponent dirction=0 时）对齐线是水平线的原因。观察下例：




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,0)grid(3,3);
   \graph [binary tree layout, nodes={draw},
     component direction=30]{
       a, b -- c[align here], d -- e[second,align
         here] -- f
   };
   \draw[red,thin,dashed] (a) -- (c) -- (e);
\end{tikzpicture}
```

我们设置了 component direction=30，即对齐线是一条 30 度的斜线，如图中红色虚线所示。通过配合使用 align here，将 c、e 两个节点放置到对齐线上，可以看出红色对齐线正好穿过 c、e 两个节点的中心。

定义 1. 对齐线（alignment line） 根据 component direction 的不同设置，对齐线指从图中第一个节点出发的一条直线。当 component direction 取默认值 0（或者不设置 component direction）时，对齐线是一条水平直线；当 component direction=90 时，对齐线是一条垂直直线；对齐线可以是任意角度的直线。

2 对齐命令

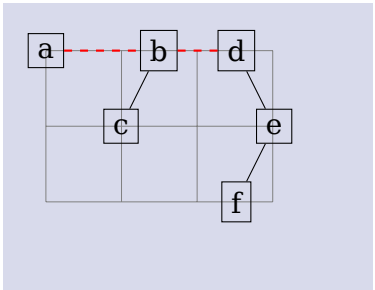
2.1 align here

align here 命令的效果参见节 1 中的示例。也就是说，align here 强制移动节点到对齐线上。一般来说，这种“强制”的手段在 tikz 绘图中应该尽量避免，让 tikz 的绘图策略自动发挥作用即可。

2.2 component align

component align 声明了自动对齐的策略，取值范围为：

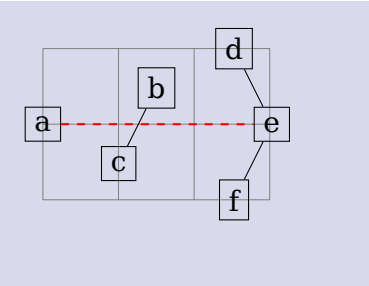
- 默认值 first node，即穿过第一个节点的水平线作为对齐线，因此所有 component 的第一个节点都部署在对齐线上。在下例中，a、b、d 作为每个 component 的第一个节点，可以看出都位于对齐线上。不声明 component align 和声明 component align=first node 效果是一样的，参见下例：




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,0)grid(3,-2);
   \graph [binary tree layout, nodes={draw},
     component align=first node]{
       a, b -- c, d -- e[second] -- f
   };
   \draw[red,thick,dashed] (a) -- (b) -- (d);
\end{tikzpicture}
```

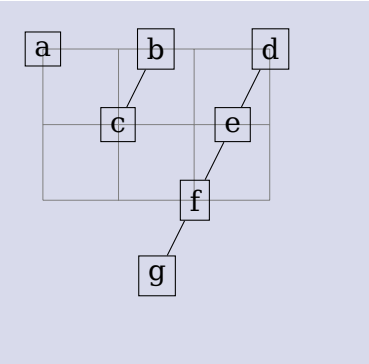
- center 对齐线水平穿过第一个节点，其他节点在对齐线两边平均分布。tikz 确定“平均分布”的方法是画若干垂直于对齐线的 shift line，如果该 shift line 穿过节点的中心，则节点中心到交点的距

离（称为投影高度）相等的节点即为平均分布的节点。作为比如：



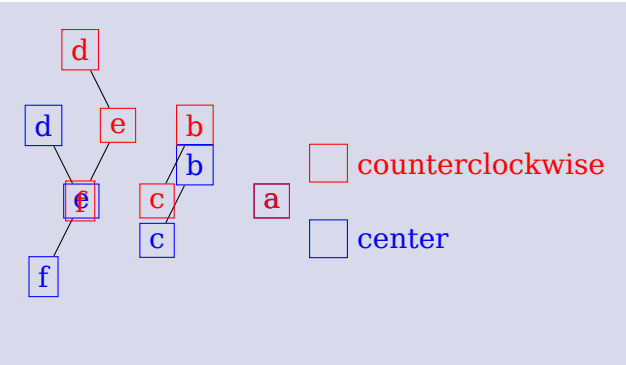

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,-1)grid(3,1);
   \graph [binary tree layout, nodes={draw},
     component align=center]{
       a, b -- c, d -- e[second] -- f
   };
   \draw[red,thick,dashed] (a) -- (e);
\end{tikzpicture}
```

- counterclockwise。counterclockwise 稍微有些费解，其布局过程如下：首先按照 center 布局，然后将投影高度最大的节点移动到 component direction 指定的对齐线上，其他节点根据相对关系也随之移动，示例：



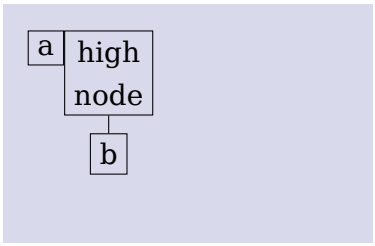

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}
   \draw[help lines] (0,-2)grid(3,0);
   \graph [binary tree layout, nodes={draw},
     component align=first node]{
       a, b -- c, d -- e -- f --g
   };

\end{tikzpicture}
```

下面把 counterclockwise 和 center 施加到在同一组 component 上，结果更清晰。可以看出，由于 counterwiseclock 是根据 center 布局中投影高度最大的节点来整体布局，因此 counterclockwise 布局是不对称的，这里 counterclockwise 是指如果从起点开始逆时针遍历各个节点，最后访问到的节点即为投影高度最大的那个节点。



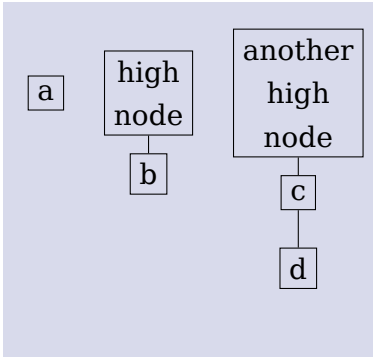
```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\begin{tikzpicture}

   \graph [binary tree layout, nodes={draw,blue},component align=center,component
     direction=180]{
       a, b -- c, d -- e[second] -- f
   };

   \graph [binary tree layout, nodes={draw,red},component align=counterclockwise,
     component direction=180]{
       a, b -- c, d -- e[second] -- f
   };
   \draw[red] (.5,.25)rectangle+(.5,.5) node[anchor=west,red] at (1,.5) {
     counterclockwise};
   \draw[blue] (.5,-.75)rectangle+(.5,.5) node[anchor=west,blue] at (1,-.5) {center};
\end{tikzpicture}
```

- clockwise，理解了 counterclockwise，clockwise 就不难理解了：顺时针遍历各个节点，最后访问到的即为投影高度最大的节点。
- counterclockwise bounding box。在 counterclockwise 布局中，对齐线是穿过节点中心的，即节点根据对齐线是垂直居中的；counterclockwise bounding box 布局和 counterclockwise 的唯一区别是，对齐线穿过节点的外边框，即，对齐线上的节点在垂直方向上是顶端对齐的，比如：



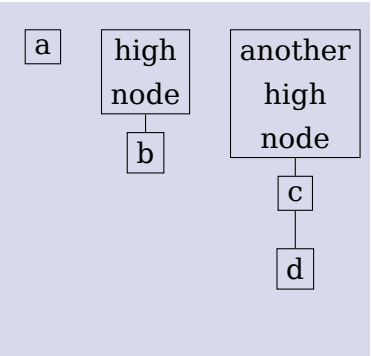

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\tikz \graph [tree layout, nodes={draw, align=
  center},
component sep=0pt,
component align=counterclockwise bounding box]
{ a, "high\\node" -- b};
```

- clockwise bounding box，请参考 counterclockwise bouding box。
- components go right top aligned，相当于 components direction=right,component align=counterclockwise，即节点从左向右布局，顶部节点垂直居中对齐。




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\tikz \graph [tree layout, nodes={draw, align=
  center},
components go right top aligned]
{ a, "high\\node" -- b,"another\\high\\node"--c--d
  };
```

- components go right absolute top aligned，相当于 components direction=right,component align=counterclockwise bounding box，即节点从左向右布局，顶部节点垂直顶部对齐。




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\tikz \graph [tree layout, nodes={draw, align=
  center},
components go right absolute top aligned]
{ a, "high\\node" -- b,"another\\high\\node"--c--d
  };
```

- components go right bottom aligned，不再详述，以下同。
- components go right absolute bottom aligned
- components go right center aligned
- components go right，相当于 component direction=right,component align=first node。
- components go left top aligned
- components go left absolute top aligned
- components go left bottom aligned
- components go left absolute bottom aligned
- components go left center aligned
- components go left
- components go down right aligned
- components go down absolute right aligned
- components go down left aligned
- components go down absolute left aligned
- components go down center aligned
- components go down
- components go up right aligned
- components go up absolute right aligned
- components go up left aligned
- components go up absolute left aligned
- components go up center aligned
- components go up