# tikz graph 之 orient 命令

宿宝臣<subaochen@126.com>

2017 年 5 月 17 日

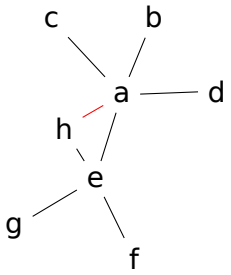
## 1 基本概念

tikz graph 总体上是根据不同的布局管理器自动决定节点的位置，但是有的时候我们希望其中的两个节点存在指定的相对关系，比如节点 a 在节点 b 的正下方，或者节点 a 在节点 b 的 11 点钟方向等等，在此基础上再决定整个图片的伸展方向和布局，此时 orient 命令就派上了用场。orient 的基本用法是指定一个角度，比如 orient=30 表示两个节点连线的角度为 30 度。orient 的取值除了角度值之外，还可以是 up、down、right、left、north、sourth、west、east 及其组合，更绝妙和形象的是，可以使用-表示 right，|表示 down。orient 配合 node distance 可以实现类似于“极坐标”的效果：以其中一个节点为原点，另外一个节点可以使用角度 orient 和长度 node distance 来表达。

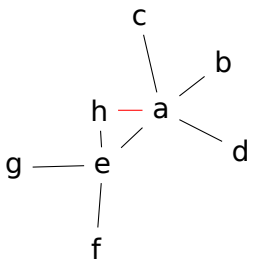
也就是说，orient 命令是通过局部（两个节点之间的相对关系）来影响全局的，因此如果在一个 graph 中存在两个 orient 声明，只有最后一个才会生效。

orient 有两种使用方法：

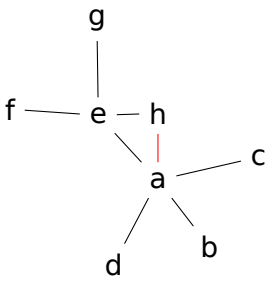
1. 定义连线的角度。当 orient 用于连线（edge）时表示两个节点连线的角度，直接在连线的末端使用 [orient=30] 这样的形式设置。比如 `h --[orient=30] a` 表示节点 h 到节点 a 的连线角度为 30 度，如下例红色线条：



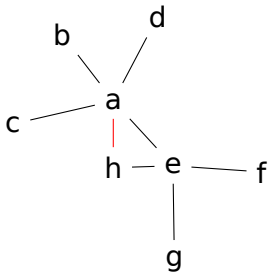

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm]
{
   a -- { b, c, d, e -- {f, g, h} };
   h -- [orient=30,red] a;
};
```




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.3cm]
{
a -- { b, c, d, e -- {f, g, h} };
h -- [orient=-,red] a;
};
```

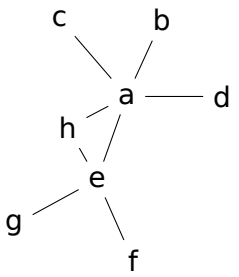



```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm]
{
   a -- { b, c, d, e -- {f, g, h} };
   h -- [orient=|,red] a;
};
```

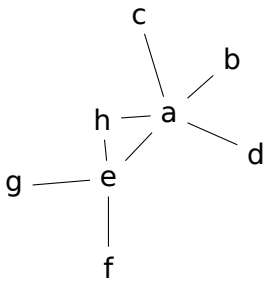



```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm]
{
a -- { b, c, d, e -- {f, g, h} };
h -- [orient=north,red] a;
};
```

2. 定义节点的方位。当 orient 用于节点时，表示该节点相对于上一级节点的方位。比如在节点右边使用 [>orient=right] 这样的形式表示该节点位于上一级节点的右边，注意其中的“>”符号，还不太理解为什么这样用，手册中似乎没有给出说明。比如：




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm]
{
   a -- { b, c, d[>orient=right], e -- {f, g, h} };
   h -- a;
};
```

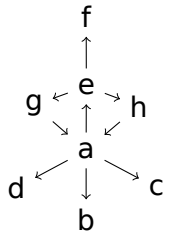



```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm]
{
   a -- { b, c, d[>orient=right], e -- {f[>orient=down
     ], g, h} };
   h -- a;
};
```

# 2 orient tail/orient head

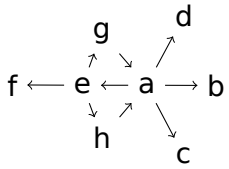
orient 是通过两个局部的节点影响全局的，在这两个局部节点中，orient tail 是指起始节点，orient head 是指开始节点。在没有明确声明 orient tail 或者 orient head 的时候，orient 是指**相邻**的两个上下级节点之间的位置关系。显然，可以通过指定 orient tail 所指向的节点或者 orient head 所指向的节点打破这种“默认设置”，重构整张图形。

先看一下默认的 spring layout 布局情况：



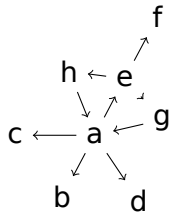

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout] {
   a -> { b, c, d, e -> {f, g, h} };
   { h, g } -> a;
};
```

假设我们希望节点 a 在 e 的右边，即将整个图片逆时针旋转 90 度，则可以这样写：



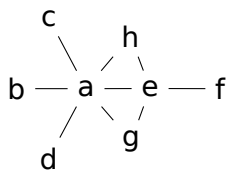

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout] {
   a[orient=right,orient tail=e] -> { b, c, d, e -> {f,
      g, h} };
   { h, g } -> a;
};
```

同样的，如果我们指定 orient tail=c，则结果为：

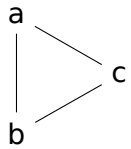



```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout] {
   a[orient=right,orient tail=c] -> { b, c, d, e -> {f,
      g, h} };
   { h, g } -> a;
};
```

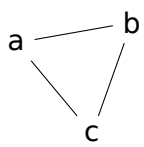
orient head 的意义和 orient tail 正好相反，参见：




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout]
{
   a [orient=right, orient head=f] -- { b, c, d, e -- {
     f, g, h} };
   { h, g } -- a;
};
```



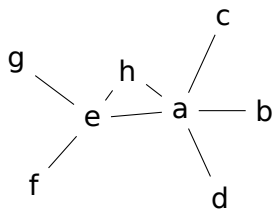

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout] { a -- b -- c -- a };
```



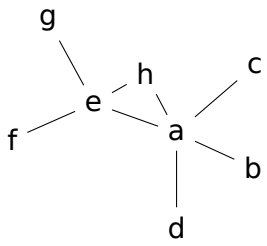

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout, orient=10,
orient tail=a, orient head=b] { a -- b -- c -- a };
```

# 3 horizontal/vertical

horizontal 实际上是 orient tail,orient head 和 orient=0 的缩写，virtical 实际上是 orient tail，orient head 和 orient=|的缩写，比如：



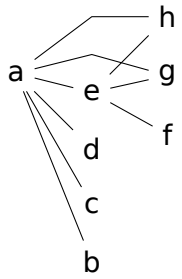

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm,
  horizontal=a to b]
{
   a -- { b, c, d, e -- {f, g, h} };
   h -- a;
};
```



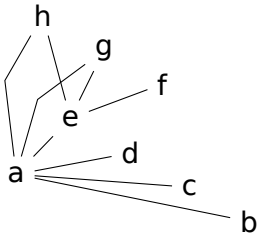

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force}
\tikz \graph [spring layout,node distance=1.4cm,
  vertical=a to d]
{
   a -- { b, c, d, e -- {f, g, h} };
   h -- a;
};
```

# 4 grow

grow 决定了相邻节点的“生长”方向，即当 tikz 放置了一个节点后，下一个节点应该出现在什么方向上。grow=down 往往意味着在下方布置下一个节点，grow=right 往往意味着在右边布置下一个节点。显然，grow 只对有树状层次结构的布局管理器有效，比如 tree layout, binary tree layout。需要注意的是，grow 是一个全局性的设置，即使在某个节点上设置了 grow 属性，也是针对全局的。




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{layered}
\tikz \graph [layered layout, grow=right, sibling
  distance=5mm]
{
a -- { b, c, d, e -- {f, g, h} };
{ h, g } -- a;
};
```




```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{layered,trees}
\tikz \graph [layered layout, grow=60, sibling distance
  =5mm]
{
   a -- { b, c, d, e -- {f, g, h} };
   { h, g } -- a;
};
```