

# tikz 的 graph 读书笔记

2017 年 5 月 16 日

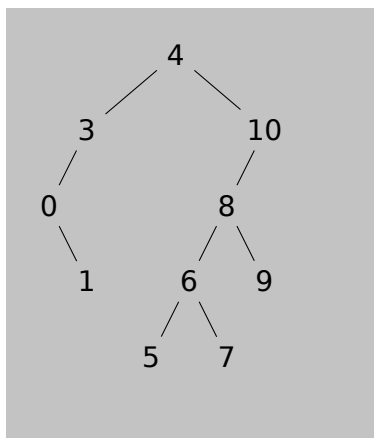
## 1 开发环境配置

参见:<http://dz.sdut.edu.cn/blog/subaochen/2017/05/tikz%E7%9A%84graph%E5%BC%80%E5%8F%91%E7%8E%AF%E5%A2%83/>。另外,最好升级 texlive 到 2016 版本以支持 lualatex 1.0,参见: <http://dz.sdut.edu.cn/blog/subaochen/2017/05/ubuntu-16-04%E5%AE%89%E8%A3%85texlive-2016/>。

## 2 graph 的布局

类似于 graphviz 的自动布局, tikz 支持若干种图形布局,也可以通过 lua 编写扩展的布局器。下面是一些常见 graph 布局的示例,先有一个感性认识,对相应的 tikz 代码目前可以不用深究。

- binary tree layout, 二叉树布局, 学过数据结构的应该很清楚:



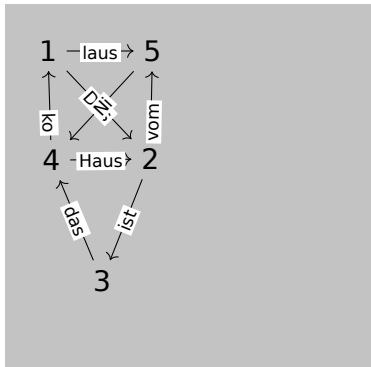
---

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees,layered}

\tikz \graph [binary tree layout, level distance=10mm]
{
    4 -- {
        3 -- 0 -- 1[second],
        10 -- {
            8 -- {
                6 -- {5,7},
                9
            }
        }
    }
}
};
```

---

- spring layout

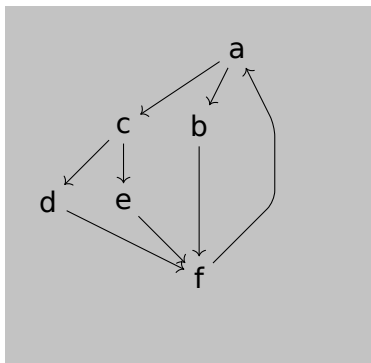



---

```
\usetikzlibrary{graphs,graphdrawing,quotes}
\usegdlibrary{force}
\tikz \graph [spring layout,edge quotes mid,
  edges={nodes={font=\scriptsize, fill=white, sloped,
    inner sep=1pt}}]
{
  1 ->["Das"] 2 ->["ist"] 3 ->["das"] 4 ->["Haus"]
  2 ->["vom" near start] 5 ->["Ni"] 4 ->["ko" near
    start]
  1 ->["laus", orient=right] 5;
};
```

---

- spring electrical layout
- layered layout

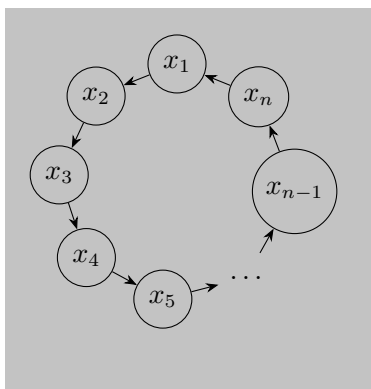



---

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees,layered}
\tikz [rounded corners]
\graph [layered layout, sibling distance=10mm, level
  distance=10mm]
{
  a -> {
    b,
    c -> { d, e }
  } ->
  f ->
  a
};
```

---

- simple necklace layout




---

```
\usetikzlibrary{graphs,graphdrawing,quotes}
\usegdlibrary{trees,circular}
\tikz[>=Stealth]
\graph [simple necklace layout, grow'=down, node sep=1
  em,
  nodes={draw,circle}, math nodes]
{
  x_1 -> x_2 -> x_3 -> x_4 ->
  x_5 -> "\dots"[draw=none] -> "x_{n-1}" -> x_n -> x_1
};
```

---

- phylogenetic tree layout.

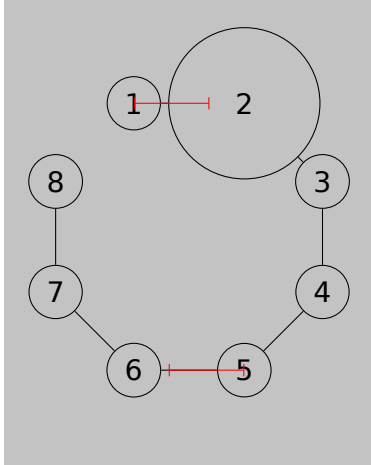
### 3 graph 程序的基本结构

`\usetikzlibrary{graphdrawing}`加载了基本的 `graph` 功能,`\usetikzlibrary{graphs}`加载了`\graph` 命令, `\usegdlibrary{force}`加载 `spring layout` 布局管理器, 以此类推。

## 4 节点的属性

### 4.1 node distance

`node distance` 影响节点（中心）之间的距离。如果 `node distance` 的设置小于节点之间的实际距离则以实际距离为准，默认值是 1cm。



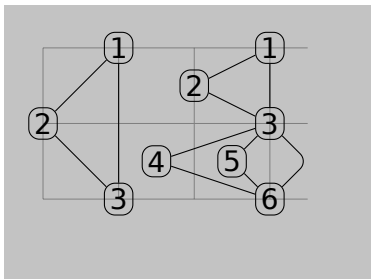

---

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees,circular}
\begin{tikzpicture}
\graph [simple necklace layout, node distance=1.5cm,
node sep=0pt,nodes={draw,circle}]
{
1 -- 2 [minimum size=2cm] -- 3 --
4 -- 5 -- 6 -- 7 --[orient=up] 8
};
\draw [red,- ] (1.center) -- ++(0:1cm);
\draw [red,- ] (5.center) -- ++(180:1cm);
\end{tikzpicture}
```

---

### 4.2 level distance/layer distance

`level distance` 影响节点层次之间的距离，尤其是 `layered layout` 等布局，默认值是 1cm。



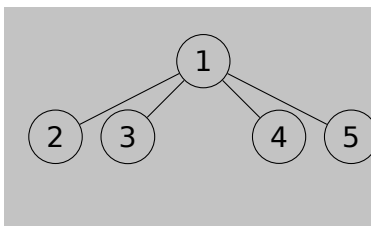

---

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{layered}
\begin{tikzpicture}[inner sep=2pt,rounded corners,nodes
=draw]
\draw [help lines] (0,0) grid (3.5,2);
\graph [layered layout, level distance=1cm]
{ 1 [x=1,y=2] -- 2 -- 3 -- 1 };
\graph [layered layout, level distance=5mm,level sep=0]
{ 1 [x=3,y=2] -- 2 -- 3 -- 1, 3 -- {4,5} -- 6 -- 3
};
\end{tikzpicture}
```

---

### 4.3 sibling distance

同一个层次的节点（中心）之间的距离，默认值是 1cm。




---

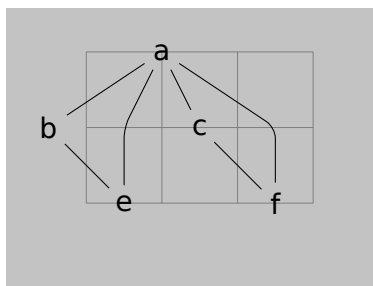
```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{trees}
\tikz \graph [tree layout, sibling distance=0cm,nodes={
circle,draw}]
{
1--{2,3[sibling distance=2cm],4,5}
};
```

---

## 4.4 pre sep/post sep

node、level、sibling 都有两个类似的属性：pre sep 和 post sep，分别表示节点前后距离节点边界的 padding 尺寸，组合起来就是 node pre sep, node post sep, level pre sep, level post sep, sibling pre sep, sibling post sep，默认值是 0.333em。为了设置方便起见，也支持 node sep, level sep, sibling sep。如果设置 node sep=1cm，则等价于设置 node pre sep=0.5cm, node post sep=0.5cm。

如下例所示，由于左图的 level distance=0，因此 3 和 5 节点紧贴在一块了，而尽管 3 的 level pre sep=1mm，但是由于同一个层次的 4 的 level pre sep=5mm，因此这一个层的节点和上一层节点的间距即为 level distance 定义的 0 和节点 4 的 level pre sep 定义的 5mm 之和，即 5mm。右图请读者自行分析。




---

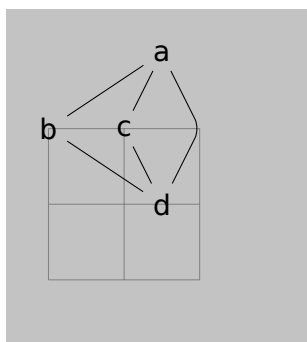
```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force,trees,layered}
\begin{tikzpicture}[rounded corners]
\draw [help lines] (0,0) grid (3,2);
\graph [layered layout]
{
  a [x=1,y=2] -- { b, c } -- {e, f} -- a
};
\end{tikzpicture}
```

---

## 5 节点的定位

可以通过固定图形的某个或者某几个节点的方式影响整个布局，定位节点的命令为：

- desired at，表示节点的坐标，可以简化为 [x=1, y=2] 的形式。
- anchor at，表示 anchor here 定位的节点的坐标。
- anchor here，将整个图形按照 anchor here 指定的节点布局。如果没有声明 anchor at 的坐标，则 anchor here 定位到原点。
- anchor node，指定哪个节点作为锚点。




---

```
\usetikzlibrary{graphs,graphdrawing}
\usegdlibrary{force,trees,layered}
\begin{tikzpicture}
\draw [help lines] (0,0) grid (2,2);
\graph [layered layout, edges=rounded corners, anchor at
={ (1,2) }]
{ a -- {b, c [anchor here] } -- d -- a};
\end{tikzpicture}
```

---

## 6 连接线的属性

连线的创建方式有如下几种：

- 如果加载了 graphs 库，则可以使用 graphs 库提供的直观的画线命令：--、->、<-、<->，参见代码清单??和代码清单??。

- 如果没有加载 **graphs** 库，则可以通过 **edge**、**child** 命令画线。so，为什么不使用 **graphs** 提供的直观画线命令呢？