实验题目

- 了解力导向图的原理
- 实现d3的力导向图layout

实验日期

- 2021.11.06

班级

- 19智能

姓名

- 倪诗宇

实验目的

- 了解力导向图的原理
- 实现d3的力导向图layout

实验过程

仿真模拟系统中存在多个节点和多种类型的力，通过力控制节点的运动，每个节点都在多个力的作用下不断发生移动，直到系统趋于平衡。中间会发生多次tick事件，每次tick，仿真系统都会更新节点的位置，且系统的能量(alpha)也会逐渐降低，直到达到某个数值(alphaMin)，整个图表就停止运动了。

第一步，构造/寻找数据

- 构造数据

```
// var nodes = [//节点集
//   {name:"安徽省"},
//   {name:"阜阳市"},
//   {name:"合肥市"},
//   {name:"蚌埠市"},
//   {name:"芜湖市"},
//   {name:"山东省"},
//   {name:"颍州区"},
//   {name:"颍泉区"},
//   {name:"颍东区"},
//   {name:'青岛市'},
//   {name:'济南市'},
//   {name:'历下区'},
//   {name:'历城区'},
//   {name:'市中区'},
//   {name:'长清区'},
//   {name:'天桥区'},
// ]
```

```
//
// var links = [//边集
//   {source:0,target:1,relation:"地级市"},
//   {source:0,target:2,relation:"省会"},
//   {source:0,target:3,relation:"地级市"},
//   {source:0,target:4,relation:"地级市"},
//   {source:1,target:6,relation:"下属区"},
//   {source:1,target:7,relation:"下属区"},
//   {source:1,target:8,relation:"下属区"},
//   {source:0,target:5,relation:"兄弟省",},
//   {source:5,target:9,relation:'地级市'},
//   {source:5,target:10,relation:'省会'},
//   {source:10,target:11,relation:'下属区'},
//   {source:10,target:12,relation:'下属区'},
//   {source:10,target:13,relation:'下属区'},
//   {source:10,target:14,relation:'下属区'},
//   {source:10,target:15,relation:'下属区'},
// ]
```

- 寻找数据（很大，不在此附加）
- 数据主要由 nodes 节点和 nodes 之间的连接信息 links ， links 的基本信息由 source 和 target 组成（source 和 target 为节点的索引信息，从 0 开始）

第二步，创建力仿真系统

```
const simulation = d3.forceSimulation(nodes)
    .force('charge', d3.forceManyBody())
    .force('link', d3.forceLink(links))
    .force('x', d3.forceX(width / 2))
    .force('y', d3.forceY(height / 2))
```

- d3.forceSimulation(nodes)用来创建力仿真系统，并添加进 nodes
- .force() 用来添加一些基本的力属性，位置属性和边

```
.force('x', d3.forceX(width / 2))
.force('y', d3.forceY(height / 2))
```

设置图的中心节点，将节点推向期望的点 (x,y)

第三步，设置排斥力和连接力的部分属性

```
simulation.alphaDecay(0.05) // 衰减系数，值越大，图表稳定越快

simulation.force('charge')
    .strength(-50) // 排斥力强度，正值相互吸引，负值相互排斥

simulation.force('link')
    .distance(0) // 连接距离
    .strength(1) // 连接力强度 0 ~ 1
    .iterations(1) // 迭代次数
```

- distance 的值与节点之间的距离成正比，可以通过修改 distance 的值，改变图的整体范围

第四步，绘制边

注意：前面所做的工作都是为了得到各个节点的坐标和速度，并不能画图

和初始值对比，转换过后多了5个属性。

- index 节点的索引
- x 节点当前x坐标
- y 节点当前y坐标
- vx 节点当前x速度
- vy 节点当前y速度

```
const simulationLinks = svg.append('g')
    .selectAll('line')
    .data(links)
    .enter()
    .append('line')
    .attr('stroke', d => '#c2c2c2')
```

第五步，绘制节点

```
 const simulationNodes = svg.append('g')
    .selectAll('circle')
    .data(nodes)
    .enter()
    .append('circle')
    .attr('r', 3.5)
    .attr('fill', d => d.children ? null : '#000') // 叶子节点黑底白边，父节点白底黑边
    .attr('stroke', d => d.children ? null : '#fff')
    .call(d3.drag()
        .on('start', started)
        .on('drag', dragged)
        .on('end', ended)
    )
```

- call函数设置了在发生三种不同事件的时候的效果

第六步，设置拖拽事件

力导向图布局的形成是一个异步的过程，而且需要一定时间

每次拖动开始，设置`alphaTarget`并重启仿真系统，alpha的值会从`alphaTarget`递减到`alphaMin`，所以如果你将`alphaTarget`的值设置的比`alphaMin`小，就会卡住，不会继续更新。

- 开始事件
- d3.event.active 代表的是除去当前事件，当前正在发生的拖动事件的个数
- 在 dragStart 的时候，如果没有其他的拖拽事件，那么 d3.event.active 的将会是 0，仿真模拟计算将会被启动，各个点的位置将依次被计算

```
function started(d) {
    if (!d3.event.active) {
        simulation.alphaTarget(0.2).restart()
    }
    d.fx = d.x
    d.fy = d.y
    // fx fy 表示下次节点被固定的位置
    // 每次tick结束node.x都会被设置为node.fx，node.vx设置为0
}
```

- 拖拽事件

```
function dragged(d) {
    d.fx = d3.event.x
    d.fy = d3.event.y
}
```

- 结束事件
    - 如果在dragended的时候，d3.event.active的如果是 0，说明计算的是最后一个点，此时可以关闭仿真模拟，不再计算。

```
function ended(d) {
    if (!d3.event.active) {
        // 设置为0直接停止，如果大于alphaMin则会逐渐停止
        simulation.alphaTarget(0)
    }
    d.fx = null
    d.fy = null
}
```

如果不在每次拖拽过后手动关闭仿真模拟，那么计算将会一直持续下去，再也不能完成第二次拖拽。而在拖拽开始时不判断开启仿真模拟，那么一次拖动也不能完成

第七步，设置 tick 事件

虽然仿真系统会更新节点的位置(`只是设置了nodes对象的x y属性`)，但是它不会转为svg内部元素的坐标表示，这需要我们自己来操作

```
simulation.on('tick', ticked)
function ticked() {
    simulationLinks.attr('x1', d => d.source.x )
        .attr('y1', d => d.source.y )
        .attr('x2', d => d.target.x )
        .attr('y2', d => d.target.y )

    simulationNodes.attr('cx', d => d.x )
        .attr('cy', d => d.y )
}
```

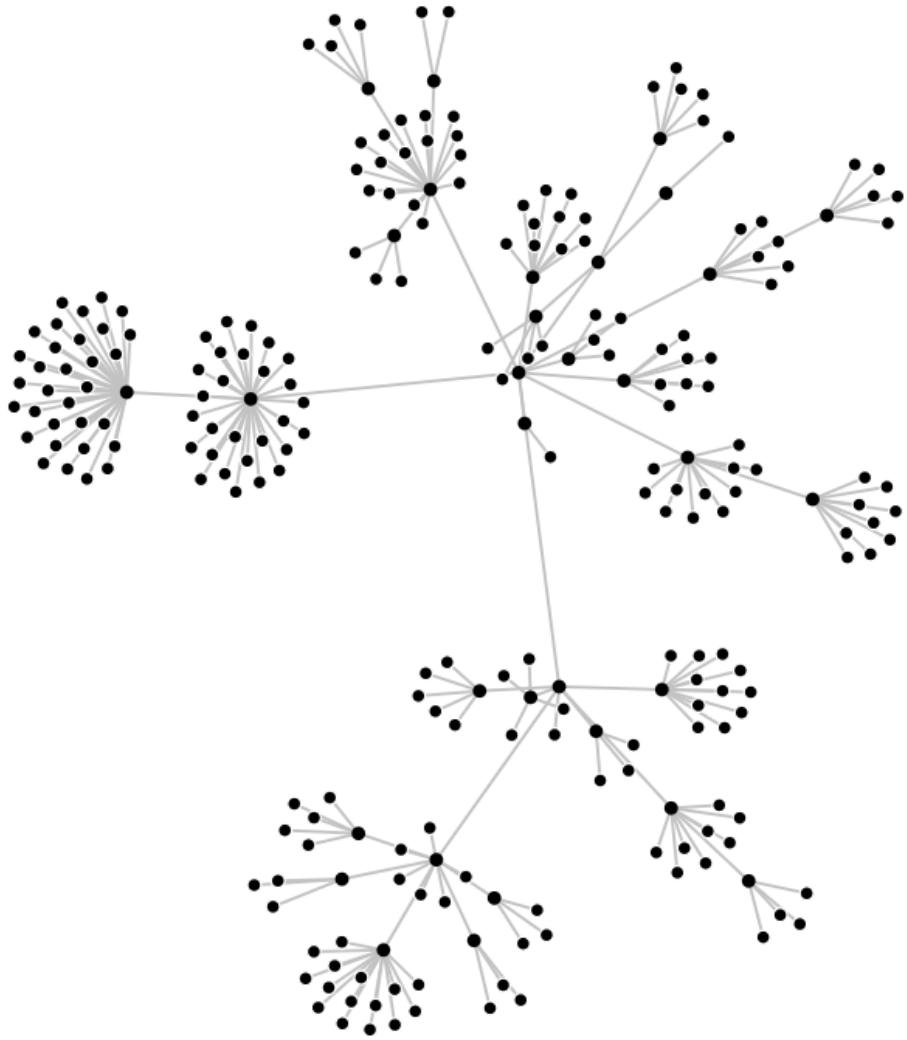
结论分析与体会

d3 的力导向图的算法的主要思想是减少计算的复杂度。

- 将离得远的点计算出一个重心，来模拟这些点的位置信息
    - 重心是通过四叉树的数据结构来计算的
- 然后利用距离公式来模拟粒子间的吸引和排斥效果

实验结果

- distance： 10



distance：50