布局规则代码分析

一、布局类型

中文名	LanguageID	enum MindMapLayoutType
思维导图	Mind Map Chart	MindMap
组织架构图（向下）	Organization Chart (Down)	OrganizationDown
组织架构图（向上）	Organization Chart (Up)	OrganizationUp
树状图（向左）	Tree Chart (Left)	TreeLeft
树状图（向右）	Tree Chart (Right)	TreeRight
逻辑图（向左）	Logic Chart (Left)	LogicLeft
逻辑图（向右）	Logic Chart (Right)	LogicRight

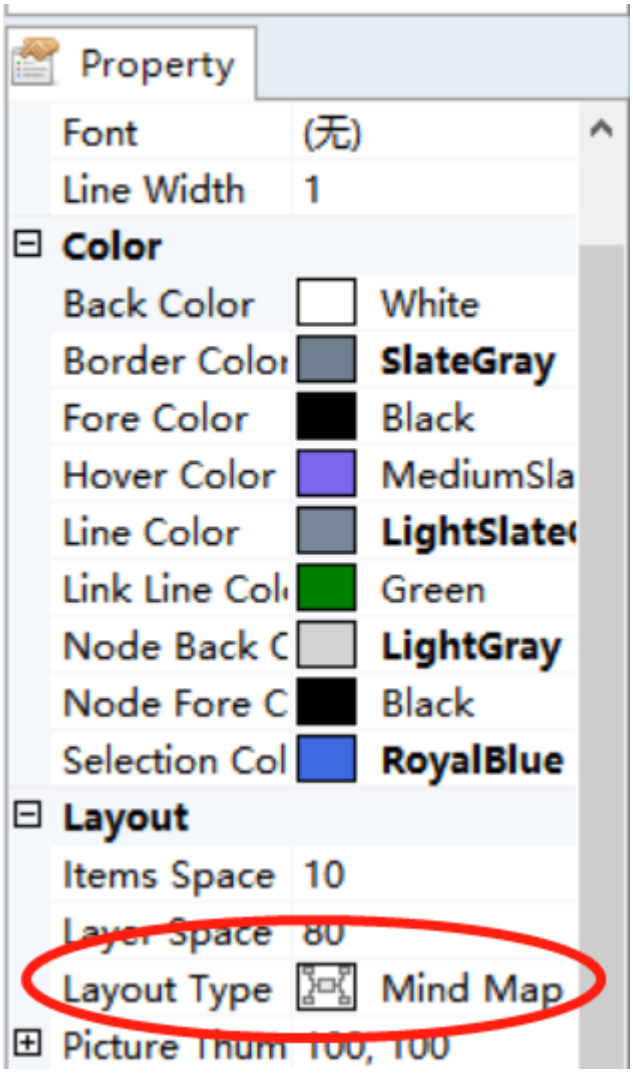
二、触发布局变化条件

(1) 改变文档布局属性 （Property -- Layout -- Layout Type）

(2) 元素插入、合并、折叠、删除、改名

三、改变文档布局属性

3.1 定位入口



通过Layout Type全局搜索，找到MindMap.cs中代码，当改变布局属性时，调用OnLayoutTypeChanged()方法。

```
[DefaultValue(MindMapLayoutType.MindMap), LocalDisplayName("Layout Type"),
LocalCategory("Layout")]
    public MindMapLayoutType LayoutType
    {
        get { return _LayoutType; }
        set
        {
            if (_LayoutType != value)
            {
                _LayoutType = value;
                OnLayoutTypeChanged();
            }
        }
    }
```

3.2调用步骤

类	名称	实现功能
MindMap.cs	OnLayoutTypeChanged	当LayoutTypeChanged时，触发事件LayoutTypeChanged(this, EventArgs.Empty);
MindMapView.cs	OnMapChanged	MindMap.LayoutTypeChanged += new EventHandler(Map_LayoutTypeChanged);
MindMapView.cs	Map_LayoutTypeChanged	新建对应布局类LayoutManage.GetLayouter(Map.LayoutType); 更新布局UpdateView(ChangeTypes.AllVisual)
MindMapView.cs	UpdateView(重写)	更新布局，有很多其他入口了ChangeTypes None=0 Data=2 Visual=4 Layout=8 ViewPort=16 NoData=32 AllVisual=Visual|Layout|ViewPort All=Data|Visual|Layout|ViewPort
MindMapView.cs	LayoutView	获得Layouter的区域大小size （Layouter.LayoutMap(Map, args);）
Layouter.cs	**LayoutMap**	**文档topic及link位置确定，每个topic位置及与其孩子节点的连线**

四、核心

4.1 整体调控 LayoutMap （***）

功能：根据不同布局规则，更新topic\link位置

代码分布描述：

(1) 当前模型topic矩形范围，每个topic位置及与其孩子节点的连线topic.Lines

```
Rectangle bounds = Layout(map.Root, e);
```

父类仅有Layout接口，会跳转到对应的布局类（继承Layouter），执行该类的Layout，具体规则见 4.2小节 。

(2) 当前模型Link矩形范围

```
Rectangle linesBounds = GetLinksFullBounds(map);
```

(3)Topic偏移量 (把模型的topic位置、连线按当前page大小移动)

```
var psx = Math.Max(0, (map.PageSize.Width - (bounds.Width + map.Margin.Horizontal)) / 2);
var psy = Math.Max(0, (map.PageSize.Height - (bounds.Height + map.Margin.Vertical)) / 2);
Offset(map.Root, map.Margin.Left - bounds.X + psx, map.Margin.Top - bounds.Y + psy);
```

(4) Link偏移量

```
map.GetLinks(true).RefreshLayout();
```

【补充，topic和link分开，个人认为是因为topic需要递归遍历，link直接调用map对象方法获取】

4.2 Topic布局方法 Layout

由于每一种布局规则存在差异性，详细布局规则在第五章展开。

4.2.1 统计大小 CalculateSizes

功能：

(1)从Root出发，深度优先递归遍历每一个topic

(2)对于每一个topic，可以分为两种大小：自身大小，所有孩子总大小

自身大小：文本+其他附加控件

所有孩子总大小：包含所有（展开的）孩子的最小区域大小

(3)哈希表layoutInfos，记录（2）对应的信息

```
layoutInfos[topic] = TopicLayoutInfo
TopicLayoutInfo.ChildrenSize        //包含所有孩子的最小区域大小
TopicLayoutInfo.FullSize            //按布局规则，包含自身及所有孩子的最小区域大小
```

4.2.2 确定位置 LayoutTopic

功能：

(1)确定每个topic的location

(2)确定topic的折叠按钮的location

(3)确定topic之间的线关系，线关系绑定在父topic.Lines上

4.3 Link布局方法 GetLinksFullBounds

link的区域与布局规则无关，只与它连接的两个topic位置有关。

功能：

(1)遍历所有link（link是可以直接从map里获取的，不需要再遍历topic了）

```
Link[] lines = map.GetLinks(true);
```

(2)得到包含所有link的最小区域

```
Rectangle rect = line.LayoutData.GetFullBounds();
result = Rectangle.Union(result, rect);
```

4.4 Topic偏移量 Offset

由于4.2中topic的location位置是基于画布原点的，需要根据当前page大小偏移到屏幕合适位置。

功能：

(1)递归遍历每个topic，偏移以下内容。

```
topic.Bounds.Offset(x,y)              //topic区域
topic.FoldingButton.Offset(x,y)       //topic折叠按钮
topic.line.Offset(x,y)                //topic与孩子节点连线
```

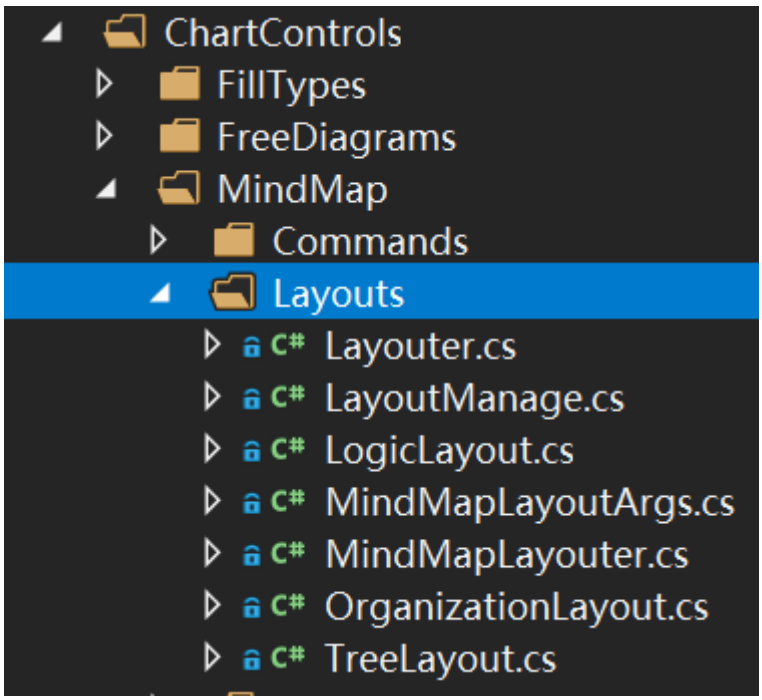
(2)偏移量计算

可以认为：移动布局，使模型的区域中心在page的中心

4.5 Link偏移

由于4.4 topic位置发生变化，因此link位置也要根据topic发生相应调整，调用line.RefreshLayout()即可。

五、不同布局规则



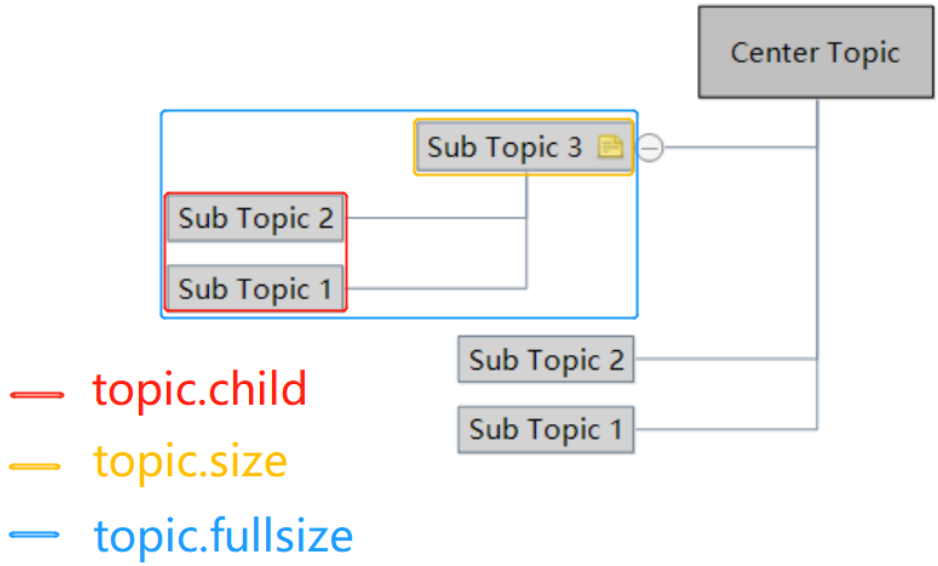
布局可以分为4个类，LogicLayout \ MindMapLayouter \ OrganizationLayout \ TreeLayout ，其中部分布局涉及上下左右方向，通过vetor参数来决定。

布局规则的不同，主要体现在 topic及其孩子topic的相对位置，以及画线的位置和形状上。

5.1 树状图 TreeLayout.cs

树状图，分向左和向右两种

5.1.1 统计区域大小



topic.ChildrenSize.Height= ∑ children.FullSize.Height+ (children.count - 1)*e.ItemsSpace

topic.ChildrenSize.Width = max(children.FullSize.Width)

可以理解成：

当前topic的ChildrenSize，其高度 = 所有孩子的FullSize的高度相加 + (孩子个数-1)*高度间距

其宽度 = 所有孩子的FullSize的宽度中最大值

topic.FullSize.Height= topic.ChildrenSize.height + e.ItemsSpace + topic.Height

topic.FullSize.Width = max(topic.ChildrenSize.Width + e.LayerSpace + topic.Width/2 , topic.Width)

可以理解成：

当前topic的FullSize，其高度 = ChildrenSize.height + 高度间距 + topic自身高度

其宽度 = max(ChildrenSize.width + 宽度间距 + topic自身宽度/2， topic自身高度)

5.1.2 确定位置

LayoutTopic(topic, layoutInfos, x, y, e);

5.1.2.1 树状图向右时

位置起点坐标为（0，0），从根节点开始往孩子节点定位。可以认为是画布原点从左往右，从上往下确定位置。

step1.定位节点位置

```
topic.Location = new Point(x, y);
```

step2.定位节点折叠按钮位置

```
topic.FoldingButton = new Rectangle(
  topic.Left-foldBtnSize,
  topic.Top+(int)Math.Round((topic.Height-foldBtnSize)/2.0f,MidpointRounding.AwayFromZero),
  foldBtnSize, foldBtnSize);
```

可以理解成：向右时，折叠按钮位置在当前节点左侧中间

step3.确定孩子区域的起点

```
x = topic.Left + topic.Width / 2 + e.LayerSpace;
y = y + topic.Height + e.ItemsSpace
```

step4.定位节点和孩子节点连线

```
CreateTopicLine(e, topic, subTopic, Vector4.Bottom, Vector4.Left)
topic.Lines.Add(line);
```

line包含 target,beginSide,endSide,beginRect,endRect。分别为终点topic、起点方向、终点方向、起点topic区域、终点topic区域。其中终点topic区域根据终点topic是否有折叠按钮做适当调整。

5.1.2.2 树状图向左时

位置起点坐标为（root.FullSize.w，0），可以认为是画布原点从右往左，从上往下确定位置。

step1.定位节点位置

```
topic.Location = new Point(x, y);
```

step2.定位节点折叠按钮位置

```
topic.FoldingButton = new Rectangle(
  topic.Right,
  topic.Top+(int)Math.Round((topic.Height-foldBtnSize)/2.0f,MidpointRounding.AwayFromZero),
  foldBtnSize,foldBtnSize
```

可以理解成：向左时，折叠按钮位置在当前节点右侧中间

step3.确定孩子区域的起点

```
x = topic.Left + topic.Width / 2 - e.LayerSpace;
y = y + topic.Height + e.ItemsSpace
```

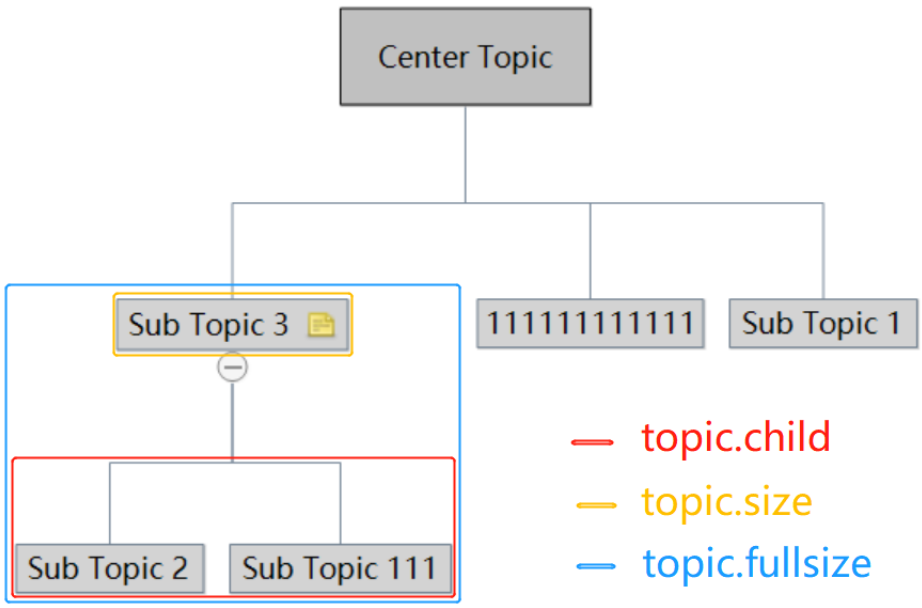
step4.定位节点和孩子节点连线

```
CreateTopicLine(e, topic, subTopic, Vector4.Bottom, Vector4.Right)
topic.Lines.Add(line);
```

line包含 target,beginSide,endSide,beginRect,endRect。分别为终点topic、起点方向、终点方向、起点topic区域、终点topic区域。其中终点topic区域根据终点topic是否有折叠按钮做适当调整。

5.2 组织架构图 OrganizationLayout.cs

5.2.1 统计区域大小



topic.ChildrenSize.Height = max(children.FullSize.Height)

topic.ChildrenSize.Width = ∑ children.FullSize.Width+ (children.count - 1)*e.ItemsSpace

可以理解成：

当前topic的ChildrenSize，其高度 = 所有孩子的FullSize的高度中最大值

其宽度 = 所有孩子的FullSize的宽度相加 + (孩子个数-1)*宽度间距

topic.FullSize.Height= ChildrenSize.Height + e.LayerSpace + topicFullSize.Height

topic.FullSize.Width = max(ChildrenSize.Width, topicFullSize.Width)

可以理解成：

当前topic的FullSize，其高度 = ChildrenSize.height + 高度间距 + topic自身高度

其宽度 = max(ChildrenSize.width + 宽度间距 + topic自身宽度/2， topic自身高度)

5.2.2 确定位置

LayoutTopic(topic, layoutInfos, x, y, e);

5.2.2.1 组织架构图向下时

位置起点坐标为（0，0），从根节点开始往孩子节点定位。可以认为是画布原点从上往下，从左往右确定位置。先子后兄弟。

step1.定位节点位置

```
topic.Location = new Point(x + (topic.FullSize.Width - topic.Width) / 2, y);
```

横坐标为当前区域的中间位置。

step2.定位节点折叠按钮位置

```
topic.FoldingButton = new Rectangle(
  topic.Left+(int)Math.Round((topic.Width-foldBtnSize)/2.0f, MidpointRounding.AwayFromZero),
  topic.Bottom,
  foldBtnSize,foldBtnSize)
```

可以理解成：向下时，折叠按钮位置在当前节点下侧中间

step3.确定孩子区域的起点

```
x = x + (tli.FullSize.Width - tli.ChildrenSize.Width) / 2
y = topic.Bottom + e.LayerSpace;
```

step4.定位节点和孩子节点连线

```
CreateTopicLine(e, topic, subTopic, Vector4.Bottom, Vector4.Top)
topic.Lines.Add(line);
```

line包含 target,beginSide,endSide,beginRect,endRect。分别为终点topic、起点方向、终点方向、起点topic区域、终点topic区域。其中终点topic区域根据终点topic是否有折叠按钮做适当调整。

5.2.2.2 组织架构图向上时

位置起点坐标为（0，tli.FullSize.Height - e.LayerSpace），从根节点开始往孩子节点定位。可以认为是画布原点从下往上，从左往右确定位置。

step1.定位节点位置

```
topic.Location = new Point(x + (tli.FullSize.Width - topic.Width) / 2, y - topic.Height);
```

step2.定位节点折叠按钮位置

```
topic.FoldingButton = new Rectangle(
  topic.Left+(int)Math.Round((topic.Width-foldBtnSize)/2.0f, MidpointRounding.AwayFromZero),
  topic.Top - foldBtnSize,
  foldBtnSize,foldBtnSize)
```

可以理解成：向右时，折叠按钮位置在当前节点左侧中间

step3.确定孩子区域的起点

```
x = x + (tli.FullSize.Width - tli.ChildrenSize.Width) / 2
y = topic.Top - e.LayerSpace
```

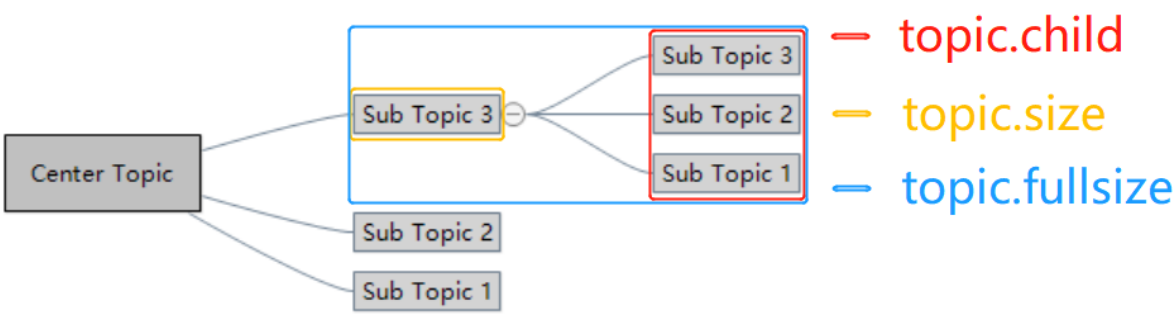
step4.定位节点和孩子节点连线

```
CreateTopicLine(e, topic, subTopic, Vector4.Top, Vector4.Bottom)
topic.Lines.Add(line);
```

line包含 target,beginSide,endSide,beginRect,endRect。分别为终点topic、起点方向、终点方向、起点topic区域、终点topic区域。其中终点topic区域根据终点topic是否有折叠按钮做适当调整。

5.3 逻辑图 LogicLayout.cs

与上面2种类似，不展开了



5.4 思维导图 MindMapLayout.cs

思维导图的孩子节点分两种：根节点的、非根节点的

对于根节点的孩子节点，其vector存在左右两种。而非根节点的孩子节点，其方向与其父节点一致。

整体步骤：

1、从根节点出发，一次将根节点的孩子节点等分（左可能比右多1个）成左右两边；

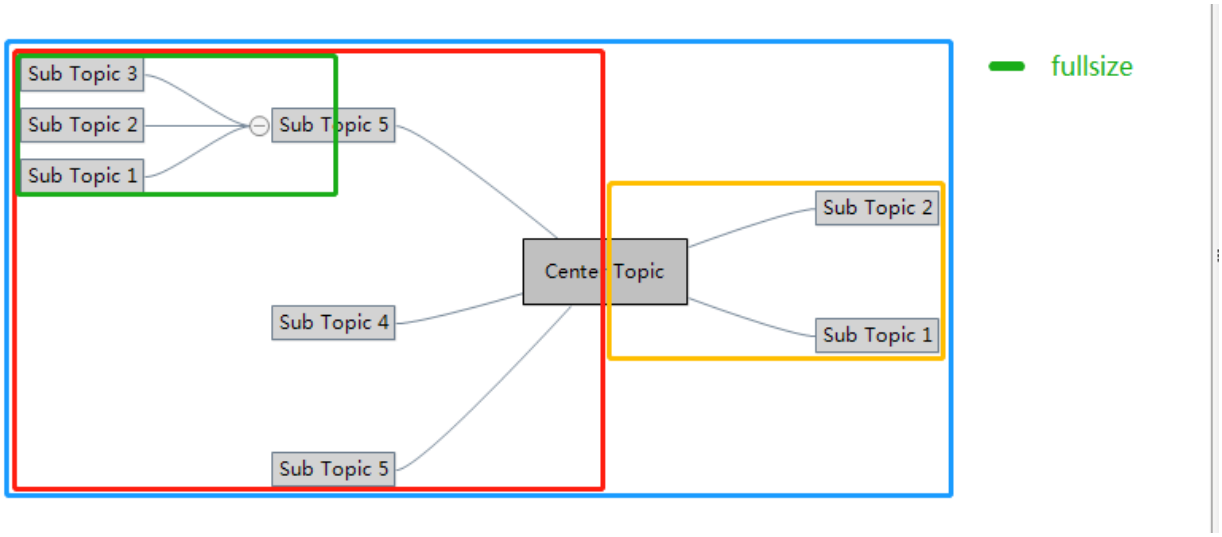
2、计算左侧孩子所占区域；

3、确定左侧孩子的位置；

4、计算右侧孩子所占区域；

5、计算右侧孩子的位置；

6、包括左右侧孩子区域的最小区域，为该模型的区域。



5.4.1 根节点的孩子节点划分

// sideCount商、def余数。可以理解为一边至少有sideCount个，左比右多0个或1个

int sideCount = Math.DivRem(root.Children.Count, vectors.Length, out def);

一次性把一侧的子节点放入subtopic，并执行CalculateSizes(root, rootFullSize, subTopics, args, vectors[vi], layoutInfos)来统计一侧孩子所占区域。

5.4.2 统计左侧孩子所占区域

【例外：没有ChildrenSize的值】

topic.FullSize.Height= ∑ children.FullSize.Height+ (children.count - 1)*e.ItemsSpace

topic.FullSize.Width = max(topic.Width, children.FullSize.Width) + e.LayerSpace + topic.Width / 2

可以理解成：

当前topic的FullSize，其高度 = 所有孩子的FullSize的高度相加 + (孩子个数-1)*高度间距

其宽度 = 所有孩子的FullSize的宽度中最大值 + 宽度间距 + 当前topic宽度的一半

5.4.3 子节点之间的合适距离

思维导图添加删除节点之所以间距会变的原因

```
int nodeSpace = e.ItemsSpace;
if (parent.IsRoot)
    nodeSpace = GetRootItemsSpace(parent, subTopics, layoutInfos, e);
```

布局时的间距，根节点的孩子节点之间，间距需要通过GetRootItemsSpace计算，非根节点的孩子节点之间，间距默认为e.ItemsSpace

```
int count = subTopics.Length;
double angle = Math.PI / (count + 1) * (count - 1);
int c1 = (int)Math.Ceiling(Math.Sin(angle / 2) * layerSpace * 2);

for (int i = 0; i < count; i++)
{
    int h = subTopics[i].Height;
    if (i == 0 || i == count - 1)
        c1 -= h / 2;
    else
        c1 -= h;
}
return Math.Max(c1, e.ItemsSpace);
```

sin(PI/2) =1，count越大，angle越趋近于PI，c1越趋近于 layerSpace * 2。

5.4.4 确定位置

【以左侧节点确定位置来举例，右侧类似】

step1.根节点位置确定，中心在画布原点

```
root.Bounds = new Rectangle(pt.X - size.Width / 2, pt.Y - size.Height / 2, size.Width,
size.Height);
```

step2.定位节点左侧孩子位置

```
//fullHeight = ∑ 左侧孩子.FullSize.Height +  (左侧孩子数-1) * nodeSpace
int fullHeight = 0;
for (int i = 0; i < subTopics.Length; i++)
{
    if (i > 0)
        fullHeight += nodeSpace;
        fullHeight += ((TopicLayoutInfo)layoutInfos[subTopics[i]]).FullSize.Height;
}

y  =  topic.Y + (topic.Height/2) - (fullHeight / 2)      //保证节点左侧区域的的中线与节点在一条线上
pt.X = parent.Bounds.Left - e.LayerSpace - subTopic.Bounds.Width
pt.Y =  y + (subTopic.FullSize.Height) / 2
subTopic.Location = new Point(pt.X, pt.Y - subTopic.Size.Height / 2);
```

step3.定位节点折叠按钮位置

向左时，折叠按钮位置在当前节点左侧中间

step4.定位节点和孩子节点连线

```
var line = CreateTopicLine(e, parent, subTopic, vector, GetReverseVector(vector));
//GetReverseVector为vector取反，左变右，上变下
```