

其实也就几点要素而已。

首先这个Path类是泛型的，因此你要继承这个Path类就需提供一个类型。这个类型就是Path里State的类型。也就是每一步的状态的类型。比如八数码问题的每一步都是一个int[3][3]；而吃豆人的每一步类型都是一个std::pair<int, int>，两个值分别是当前所处的y和x值。

要使用astar算法，你只需要继承Path类，也就是个性化定制里面的那些状态和函数。我上面截图的那些函数和状态都是需要在子类里重写的。

State state标定当前节点的状态。  
achievement标定要达到的目标状态。  
ID是我用来标识每一个节点的东西（因为我用哈希表来保存每个节点的前驱，因此必须对每个节点有唯一标识。）  
get\_next()返回站在当前节点（路径），能够衍生出的下一步的节点的表（因为可以有多个后继节点。）。  
is\_goal()判断当前节点是否目标（即和achievement是否相同。）

因为astar有两个代价函数，因此我这里有一个cost\_fn和一个cost\_left\_fn。在八数码里，cost\_fn是1，也就是每一步都把cost加上1那么语义就是当前的深度; cost\_left\_fn是和目标状态之间的差距，在八数码问题里就是放错的方块的数目。

另外需要重载<和==两个运算符。<号是用在哈希表比较上的。通常就是比较ID的大小。

==通常是用来比较两个节点（Path）的状态是否相同的。