# 使用IDA实现8-puzzle算法与RBFS代码阅读实验论文

171491207 王辛宇 2017硬件一班

# 引言:

上次实验中作者学习了github上某制作者的RBFS算法以企图学会如何实现，但是RBFS编写者在编写RBFS程序的同时也借助了他所编写的各种类。因此为了不让作者本次实验论文显得过于苍白，作者打算在本次实验论文中用IDA\*算法实现8-puzzle算法并伴随着RBFS代码的继续解读。

# 正文：

## IDA\*的实现：

### 实验原理：

1.迭代加深搜索算法IDS是一种将DFS与BFS结合起来的算法。对于DFS来说，一旦搜索空间深度无限或者过大，那么DFS将很难得到答案。而IDS先限制搜索的depth，若在深度内没有得到答案，那么便再次扩大搜索深度，可以很好的节约时间与资源。

2.而作者想要实现的IDA\*即为A\*搜索的迭代加深版本，IDA\*算法的核心是：设置每次可达的最大深度depth，若没有到达目标状态则加深最大深度，采用估值函数，剪掉f(n)大于depth的路径。

3.IDA\*算法的优点：使用回溯方法，不用保存中间状态，大大节省了空间。

缺点：重复搜索，回溯过程中每次depth变大都要再次从头搜索。

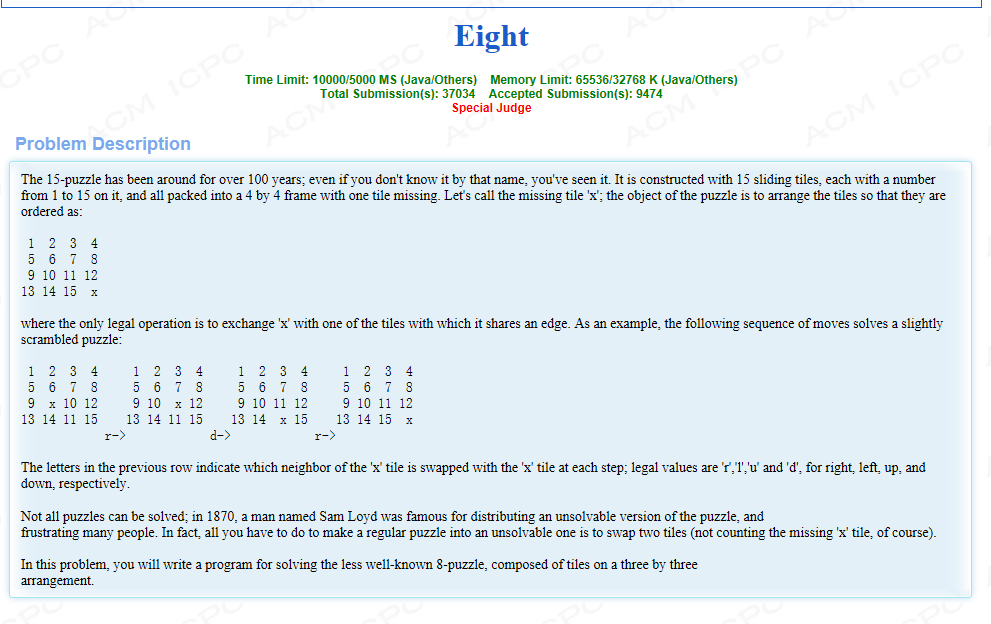
### 实验设备：

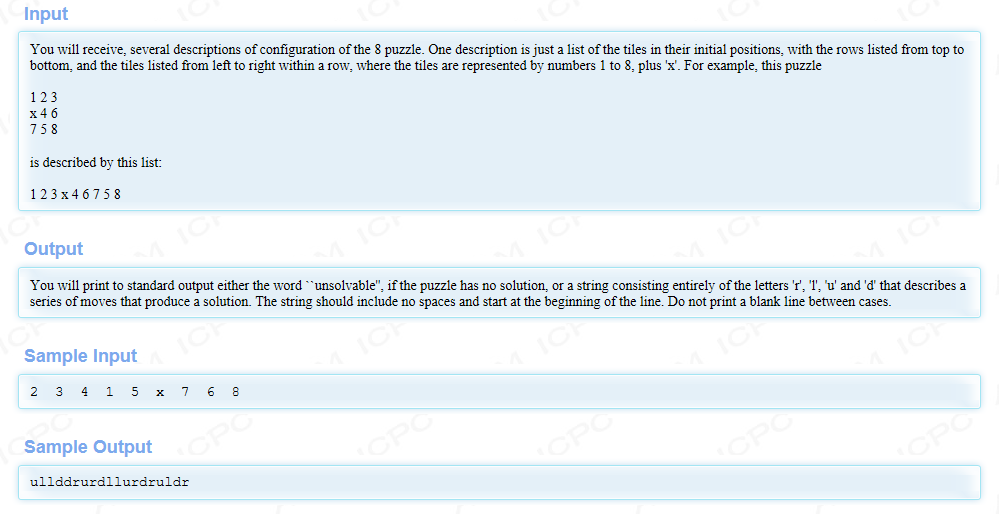
c-Free这个c++编辑平台

### 实验过程：

#### 关于8—puzzle算法:

作者使用的是杭州电科大网站的题目作为依据：图如下：





通过该图作者确定了输入输出格式。

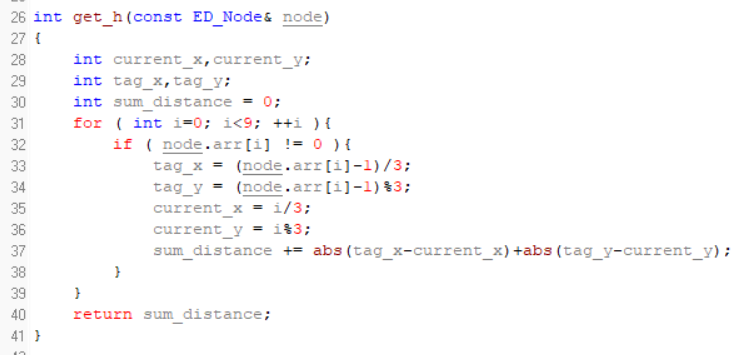
其中“r”表示x与右邻数交换，“l”表示向左，同理“u”与“d”分别表示向上与向下。

#### 在设定函数前作者定义的各种静态全局变量：

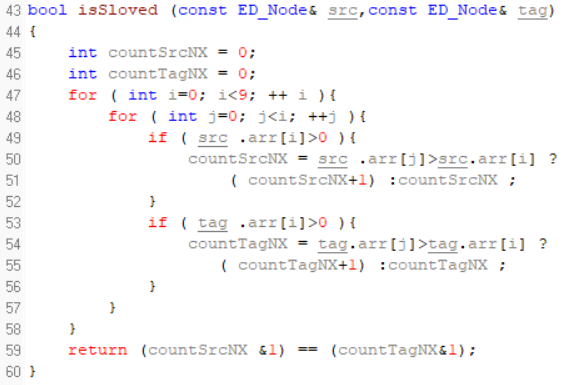
其中包括地图定义与结点结构体定义等变量



#### 3.获取h值的函数：

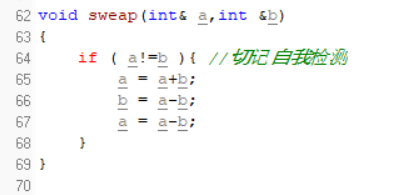


#### 4.判断问题是否能够解决的函数：



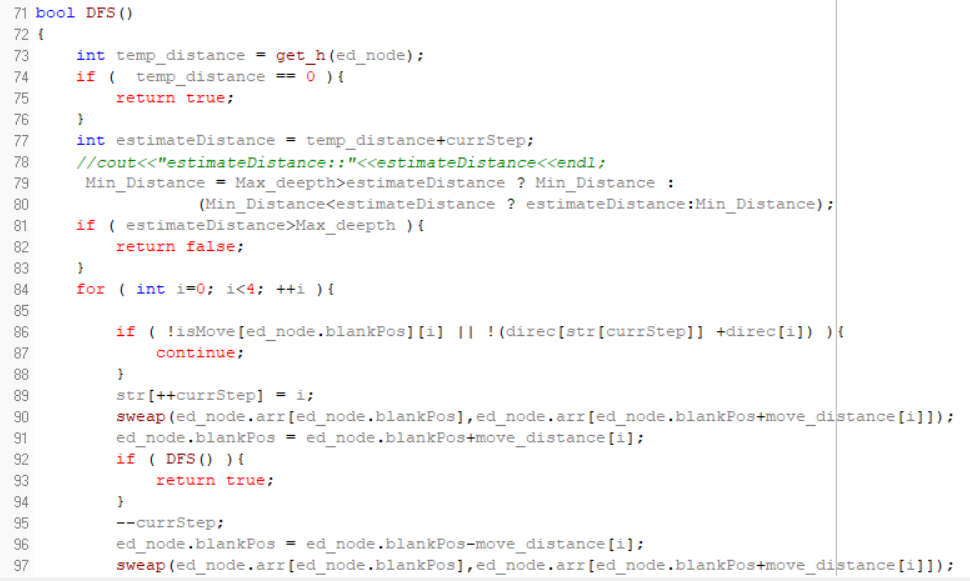
#### 交换两者内值：

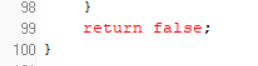
该算法中作者使用了比较巧妙的思维来交换两个变量的值，此外需要注意，两个值交换内含值必须将函数参数设置为地址。



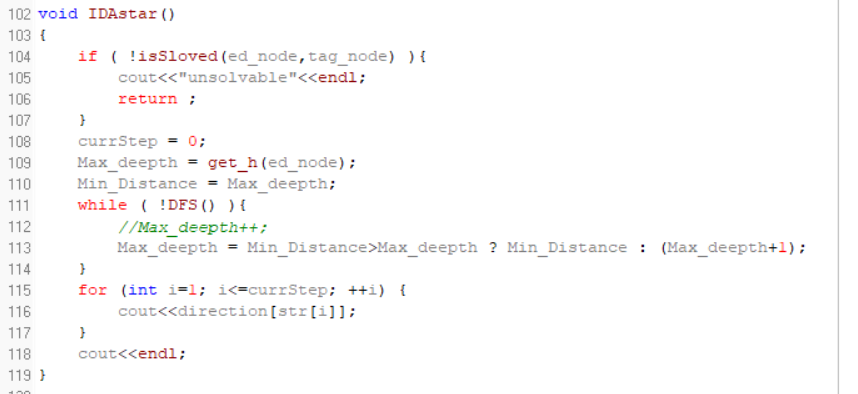
#### 深度搜索函数：

因为IDA本身为“带上层数限制”的DFS，所以需要以DFS为基础创建函数

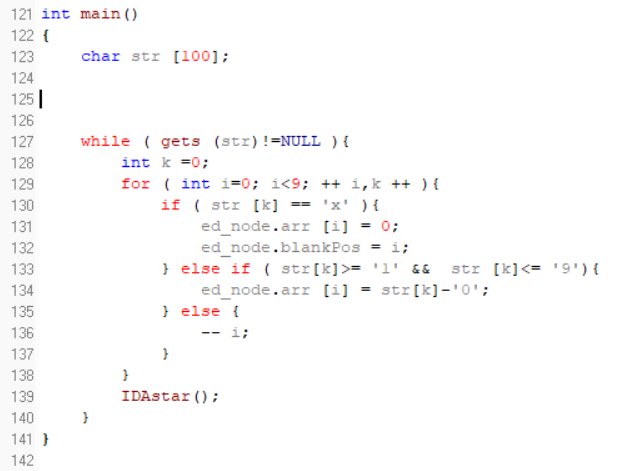




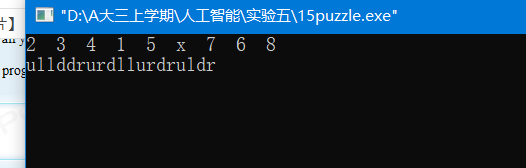
#### 7.IDA\*算法函数：



#### 8.主函数：

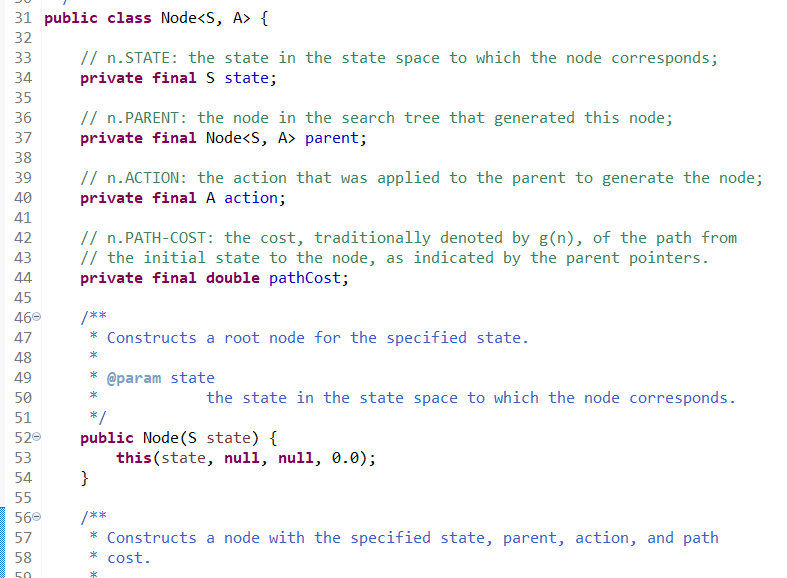


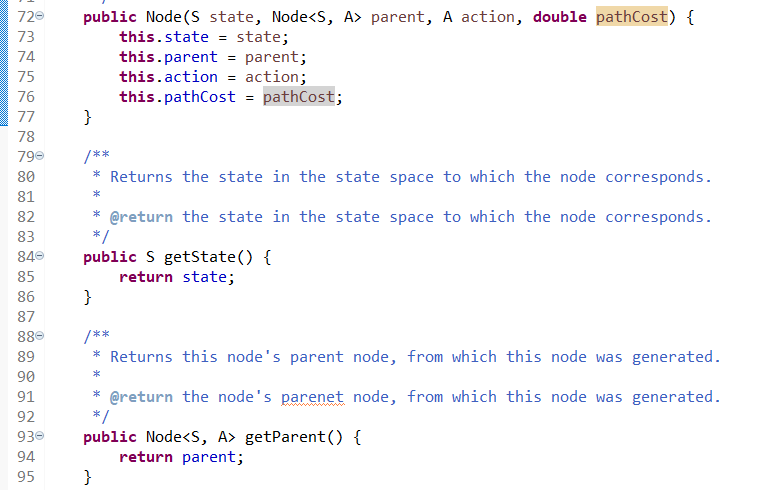
1. 运行结果截图：

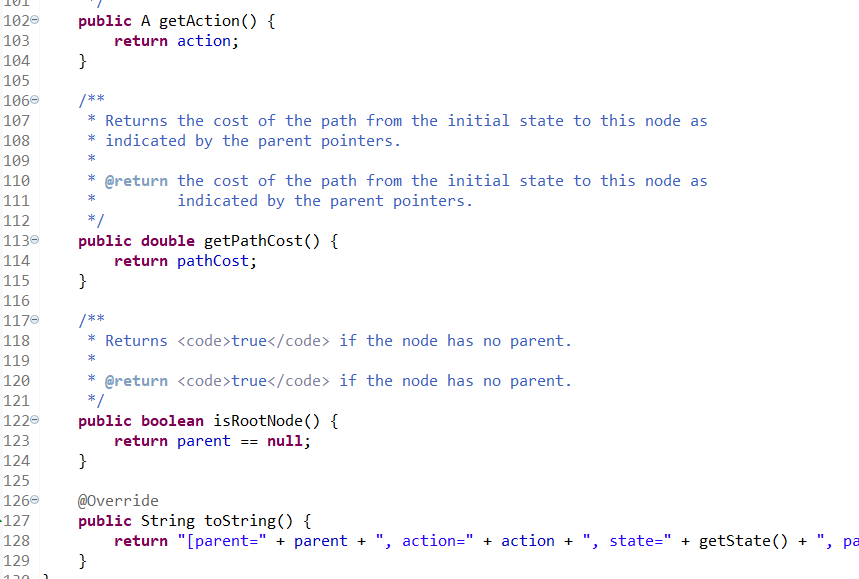


## RBFS算法的继续阅读掌握（结合作者上次实验论文观看更佳）

1. Node类的定义







Node类主要使用了“泛型定义”，作者在经过网上查找多次仍然未能掌握泛型的真正用法，作者将在下次实验前与java语言老师进行沟通以解答作者当前困惑。

# 结语：

本次实验论文作者展示了IDA\*如何解决八数码问题以及对于RBFS示范代码集的进一步深入理解，相信在下次实验论文作者能够实现RBFS的实现。