Boggle

姓名：吴侃 学号：14348134 邮箱：wkcn@live.cn

班别：2014级计算机系一班 日期：20160913

一. 算法描述

这次重要的算法为Boggle求解算法，由于一个单词是逐个相邻字母串联而成，并且需要对整个棋盘进行搜索．我使用了字典树(Trie)结构，在DFS（深度优先搜索算法）同时，移动字典树上的指针，若字典树上没有这个分支，则剪枝．若遇到isWord = true的节点，则认为构成了一个单词，将该单词加入解中．

我用非递归的方法实现了DFS，原理为使用一个结构Rec记录每一步搜索的状态．

建立字典树：

void AddWord(Node \*root, string word){

for (对于word的每一个单词c){

if (root.children[c] == 0)

root.children[c] = 新建一个Node节点，并返回地址；

root = root.children[c];

}

root->isWord = true; // 标记该节点构成一个单词

}

对于Boggle棋盘的每一格分别开始一次DFS搜索，在搜索的过程中不访问这次DFS已经访问的格子。当开始新的DFS搜索时，字典树中有一个指针P指向树的根部，每访问一个格子W，查找该格子对应的字母是否是P指向的节点N的儿子，分两种情况：

不是儿子：对搜索进行回溯，即不再从W进行进一步的搜索。

是儿子：判断节点N的isWord属性，如果isWord = true, 说明找到了一个单词；如果isWord = false, 说明没有找到单词，但可以从格子W开始进行更深一层的搜索（这条路径上可能存在单词）。

退出条件：访问了所有可能存在单词的路径。

二. 验证算法

测试方法一：

我也写了一个简单版的暴力算法BoggleSolverSimple，用DFS加递归实现。对于一个BoggleBoard, 测试字典树方法与暴力方法得到的解是否一致。随机生成多组BoggleBoard进行测试，结果一致。正确。

测试方法二：

使用题目描述下面的网址测试，得到的所有解累计的分数和答案一样。正确。

三．性能比较

对于4 X 4的Boggle Board

使用字典树版本只需0.15s, 而暴力版本需要10s

100 x 100的Boggle Board, 字典树版本求解需要10s左右