Coding Hacking 题解

如果写的有问题，欢迎指正，也不要局限在这些方法，甚至可能会有更好的方法

有效字符串

可以考虑使用栈结构。先定义一个栈（用数组也可以），发现一个左括号（"("、"[]"、"{"）中的某一个，就把它放进栈中，碰到右括号，就将栈顶元素弹出，看是否配对。如果不配对或者栈顶没有元素可以弹出，那么这时候一定就是False，并且不用再继续找了，程序也可以结束。最后程序默认输出True就可以了。

索引字符串

trie树为啥过不了还没研究

因为这个题目放在这里了，然后还没接触过其他的知识点的话，实际上就纯暴力就能过

用一个变量$maxL$记录当前字符串的最长公共前缀的长度。将一个字符串s和其他的每一个字符串$tmpS$进行比较。我们假设现在检查到的下标是i，那么如果$s[i]$和$tmpS[i]$相同，i继续增加，一直到不相同（或者超出某个字符串的长度为止）。$maxL = max(maxL, i)$，然后最后取一下$[0, maxL]$的前缀就行。（i的边界，包括超界没超界的情况要考虑一下，这里没仔细写。）

逃出升天

可以使用一个新的数组，存的值是数字。一开始将所有位置设置为一个极大值INF。然后首先，将冲击波的落点的格子的值设置成冲击波出现的时间。我们发现因为扩散很固定，因此用bfs很符合直觉（当然dfs也可以）。那我们将这些点也加入到队列中，然后开始对外扩散，更新出每一个点的时间戳。(java的话可以用类来存储每个点的x，y和时间戳，c++可以用struct，其他语言应该也有类似的)

接下来，再将这个图中所有`@`的位置的时间戳设置成0，表示一开始就不能走。之后，从(0, 0)开始进行搜索，并且此刻时间为0。如果下一个要移动到的格子的时间戳是大于当前站立的格子的时间戳+1的话，那么就可以走，否则是走不了的。按照这个规则搜索一遍。bfs的话，如果当前取出来的点是终点格子，那么直接输出时间就可以。dfs的话，如果可以走到终点，维护一个变量用来更新最小花费时间，最后输出这个时间。

最后的希望

我们使用一个三维数组$dp[i][j][k]$，表示，在第i天结束的时候，剩下j个敌人和k层护盾的时候，最多剩下的花朵的数量。那么，其实可以分成两种大的状态，就类似于01背包的“取”或者“不取”，这里就是“用”或者“不用”，只是“用”的时候，要考虑子情况。如果今天没有用花，那么我们知道，$j + k$一定要小于等于初始的总护盾数量，因为现在是发动进攻结束以后的时间，如果是大于初始护盾数量，那么就被攻破了。如果今天是用掉花朵了，我们就要考虑到底是用1朵还是2朵还是3朵……

一开始将dp数组清空，全为0，那么，第一天结束的时候，必定要剩下两朵花。（前面可以预先看一下，如果第一天结束的时候都活不下来，直接-1就好了。）

```
// n是要坚持的天数，m是初始敌人数量，hp是护盾初始值

dp[1][m][hp-m] = 2;
for (int i = 1; i<= n; ++ i) {
    for (int j = 0; j <= m; ++j) {
        for (int k = 0; k <= hp;++k) {
            if (k+j <= hp) {
                dp[i][j][k] = max(dp[i][j][k], dp[i-1][j][k+j]*2);
            }
            for (int killNum = 0; killNum <= m; ++killNum) {
                // 必须要满足使用的花朵加上剩下的敌人小于等于一开始的敌人数量，并且现在剩下的护盾数量加上消灭敌人数量要小于等于初始值，用掉这些花朵的话最起码有剩余
                if (j+killNum <= m && k+j <= hp && dp[i-1][j+killNum][k+j] > killNum) {
                    dp[i][j][k] = max (dp[i][j][k], dp[i-1][j+killNum][k+j]-killNum);
                }
            }
        }
    }
}

int maxF = 0;
for (int i = 0; i <= m; ++i) {
    for (int j = 0; j <= hp; ++j) {
        maxF = max(maxF, dp[n][i][j]);
    }
}
```