Readme A*算法 cpp

2016年4月4日 14:01

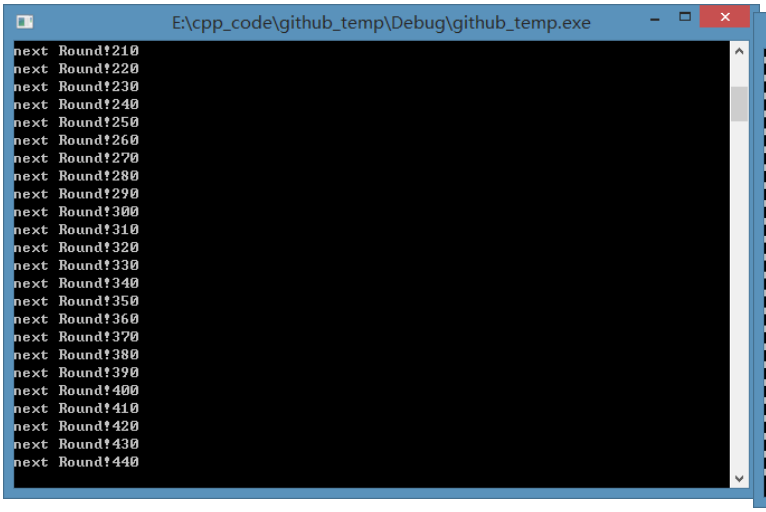
1. 思路
   a. 使用 f = g+h，作为估计值，f为已经走过的步数，h为距离拼图完成的估值；
   b. 其中，h使用了三种评估方式，对比一下性能
      - H1 9个格子中，不在自己的位置上的格子的个数；
      - H2 9个格子中，若移动到各自的目的位置，需要走过的最少步数
      - H3 9个格子中，若移动到格子的目的位置，绝对距离是多少（例如对角线视为1.414）
   c. openlist与closelist的维护，前者存放未探索的、且即将被探索状态，后者存放已探索的状态，直到openlist中出现了与目标状态完全一致的状态（即A*算法）
2. 代码审计
   a. 优点
      - 数据结构的选取，EightPuzzleState，构造得非常合适，通过pre_state、pre_move将initial_state到final_state连接起来
      - h的计算，在version2、version3中，使用矩阵的读取，省略大量计算内容来提高程序性能
   b. 缺点
      - 空格定义为-1，不合理，这样遍历起来，例如

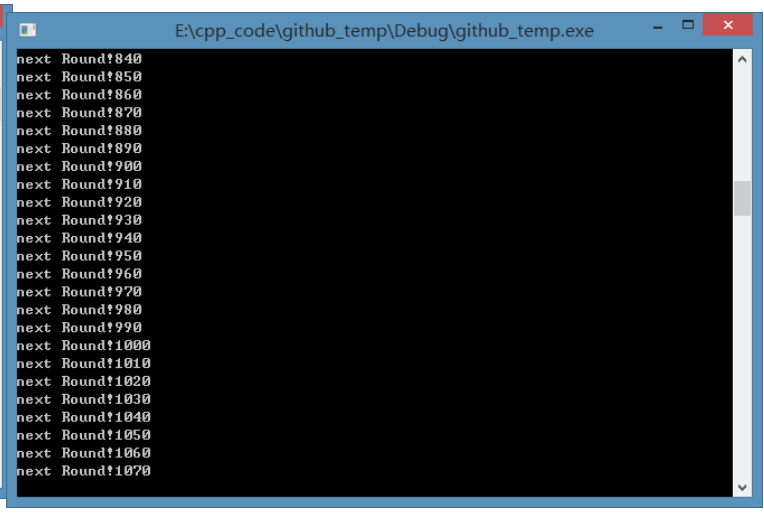
```
Int h = 0;
for (int i = 0; i < 9; i++) {
  h += (a[i/3][i%3] == (i+1));
}
```

这样就可以不用考虑 -1 是否在9的位置了，h的计算会方便一些

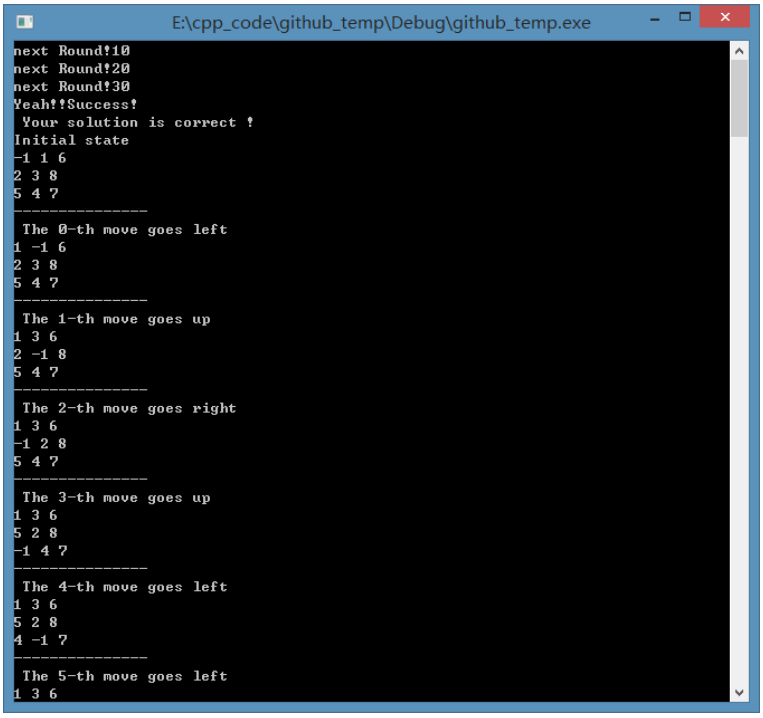
      - 每次遍历closelist 和 openlist的时候，太耗时了，越到后面数据多的时候，检验是否在list的时候，速度过慢
      - 内存泄露问题（已解决）每个successor，都要malloc EightPuzzleState，占用空间，当其不被加入openlist的时候，则造成内存泄露，解决方法，即及时free
3. 运行结果
   a. 不同h的计算方式的区别
      - H1效率极差，离目标较远的state也会被拉入其中
      - H2、H3效率相当，基本没什么区别
   b. 10、50、150的random_move下
      - 10----100%可以出结果
      - 50---很大几率出结果，否则运行很久，最慢的时候，进行了500+ round，仍然在解，但速度已经受限。最多可以推算20+个step得到的结果
      - 150---有可能出结果，在暂停查看的时候，100%是停在了检验openlist和closelist的时候，即速度受限的地方，如果要改进，可以定时清除closelist中，已经遍历过的、最早遍历过的一部分state
      - 综上，算法没有问题，数据结构也没有问题；是因为问题本身的复杂度导致短时间内没有出结果，在random move过多的时候，性能迅速下降。

Random = 150、运气不好的时候，运行到1000+也没有结果！！！！





Random = 150、运气好的时候，运行到30多次loop，即可出结果


E:\cpp_code\github_temp\Debug\github_temp.exe
next Round!10
next Round!20
next Round!30
Yeah!!Success!
Your solution is correct !
Initial state
-1 1 6
2 3 8
5 4 7

The 0-th move goes left
1 -1 6
2 3 8
5 4 7

The 1-th move goes up
1 3 6
2 -1 8
5 4 7

The 2-th move goes right
1 3 6
-1 2 8
5 4 7

The 3-th move goes up
1 3 6
5 2 8
-1 4 7

The 4-th move goes left
1 3 6
5 2 8
4 -1 7

The 5-th move goes left
1 3 6