程序设计实习

郭炜　微博 http://weibo.com/guoweiofpku
http://blog.sina.com.cn/u/3266490431
刘家瑛 微博 http://weibo.com/pkuliujiaying

广度优先搜索

八数码问题进一步讨论

广搜与深搜的比较

- 广搜一般用于状态表示比较简单、求最优策略的问题
  - 优点：是一种完备策略，即只要问题有解，它就一定可以找到解。并且，广度优先搜索找到的解，还一定是路径最短的解。
  - 缺点：盲目性较大，尤其是当目标节点距初始节点较远时，将产生许多无用的节点，因此其搜索效率较低。需要保存所有扩展出的状态，占用的空间大
- 深搜几乎可以用于任何问题
  - 只需要保存从起始状态到当前状态路径上的节点
- 根据题目要求凭借自己的经验和对两个搜索的熟练程度做出选择

3

八数码问题 ： 如何加快速度

POJ 1077为单组数据

HDU 1043 为多组数据

裸的广搜在POJ 能过，在HDU会超时

八数码问题 ： 如何加快速度

1. 双向广搜(HDU能过）

从两个方向以广度优先的顺序同时扩展

2. 针对本题预处理 (HDU能过）

因为状态总数不多，只有不到40万种，因此可以从目标节点开始，进行一遍彻底的广搜，找出全部有解状态到目标节点的路径。

3. A* 算法(HDU能过）

双向广度优先搜索(DBFS)

- DBFS算法是对BFS算法的一种扩展。

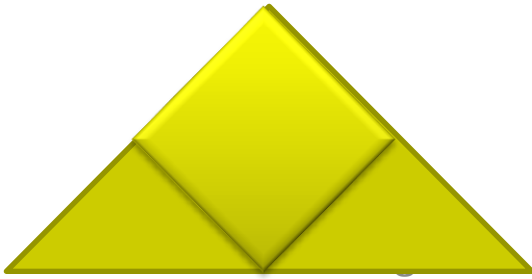
  - BFS算法从起始节点以广度优先的顺序不断扩展，直到遇到目的节点
  - DBFS算法从两个方向以广度优先的顺序同时扩展，一个是从起始节点开始扩展，另一个是从目的节点扩展，直到一个扩展队列中出现另外一个队列中已经扩展的节点，也就相当于两个扩展方向出现了交点，那么可以认为我们找到了一条路径。

双向广度优先搜索(DBFS)

- 比较
  - DBFS算法相对于BFS算法来说，由于采用了双向扩展的方式，搜索树的宽度得到了明显的减少，时间复杂度和空间复杂度上都有提高！

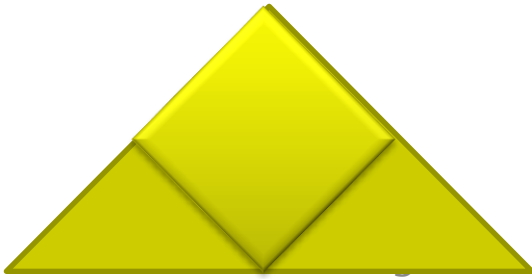
双向广度优先搜索(DBFS)

- 比较
  - DBFS算法相对于BFS算法来说，由于采用了双向扩展的方式，搜索树的宽度得到了明显的减少，时间复杂度和空间复杂度上都有提高！
  - 假设1个结点能扩展出n个结点，单向搜索要m层能找到答案，那么扩展出来的节点数目就是: $(1-n^m)/(1-n)$



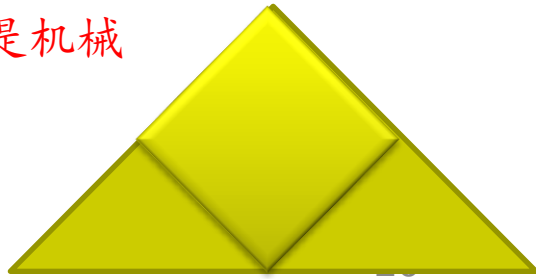
双向广度优先搜索(DBFS)

- 比较
  - DBFS算法相对于BFS算法来说，由于采用了双向扩展的方式，搜索树的宽度得到了明显的减少，时间复杂度和空间复杂度上都有提高！
  - 假设1个结点能扩展出n个结点，单向搜索要m层能找到答案，那么扩展出来的节点数目就是: $(1-n^m)/(1-n)$
  - 双向广搜，同样是一共扩展m层，假定两边各扩展出m/2层，则总结点数目 $2 * (1-n^{m/2})/(1-n)$



双向广度优先搜索(DBFS)

- 比较
  - DBFS算法相对于BFS算法来说，由于采用了双向扩展的方式，搜索树的宽度得到了明显的减少，所以在算法的时间复杂度和空间复杂度上都有较大的优势！
  - 假设1个结点能扩展出n个结点，单向搜索要m层能找到答案，那么扩展出来的节点数目就是: $(1-n^m)/(1-n)$
  - 双向广搜，同样是一共扩展m层，假定两边各扩展出m/2层，则总结点数目 $2 * (1-n^{m/2})/(1-n)$
  - 每次扩展结点总是选择结点比较少的那边进行扩展，并不是机械的两边交替。



DBFS的框架(1)

一、双向广搜函数:

void dbfs()

{

1. 将起始节点放入队列q_0,将目标节点放入队列q_1;
2. 当两个队列都未空时，作如下循环:
    1) 如果队列q_0里的节点比q_1中的少,则扩展队列q_0;
    2) 否则扩展队列q_1
3. 如果队列q_0未空，不断扩展q_0直到为空;
4. 如果队列q_1未空，不断扩展q_1直到为空;

}

DBFS的框架(2)

二、扩展函数

int expand(i) //其中i为队列的编号，0或1

{

取队列qi的头结点H;

对H的每一个相邻节点adj:

1 如果adj已经在队列q_i之中出现过，则抛弃adj;

2 如果adj在队列q_i中未出现过，则:

1） 将adj放入队列q_i;

2) 如果adj 曾在队列q_{1-i}中出现过, 则：输出找到的路径

}

需要两个标志序列，分别记录节点是否出现在两个队列中

八数码问题，单向广搜POJ 891MS 双向广搜 POJ 63MS HDU 通过!

//下面程序是1号字，所以看不见。请全选，拷贝粘贴到别处后再看