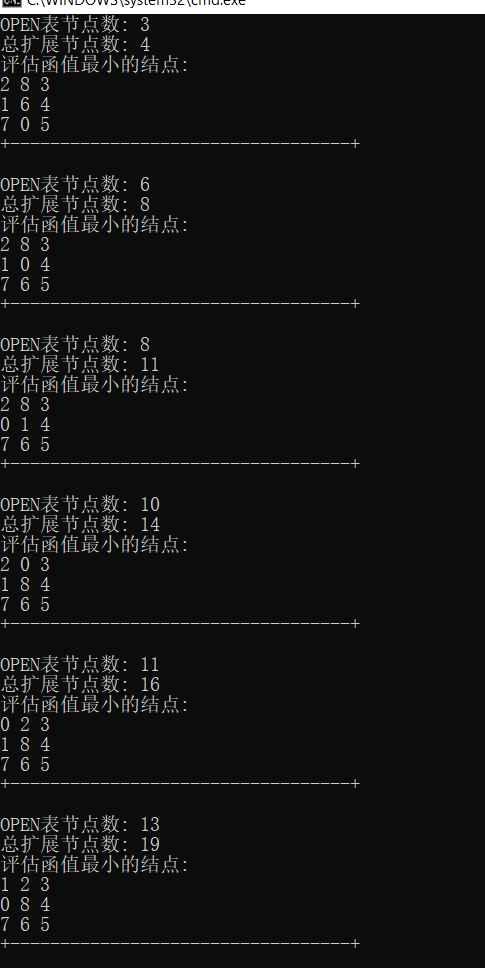
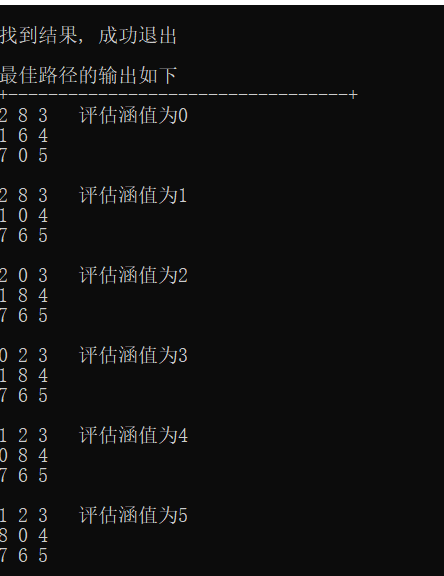
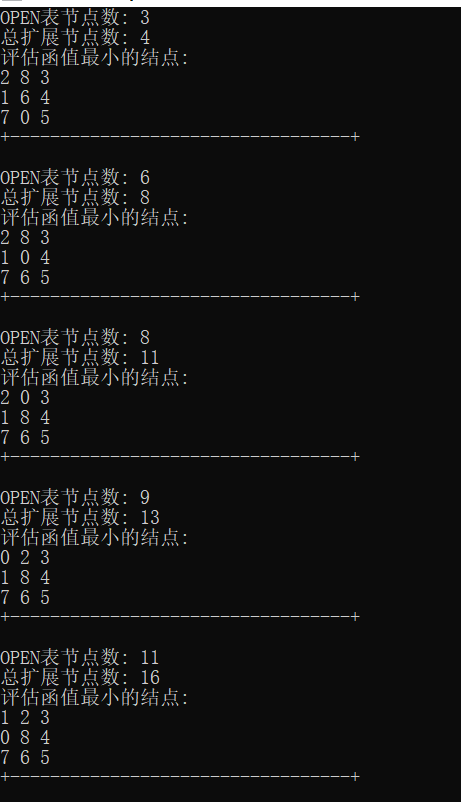
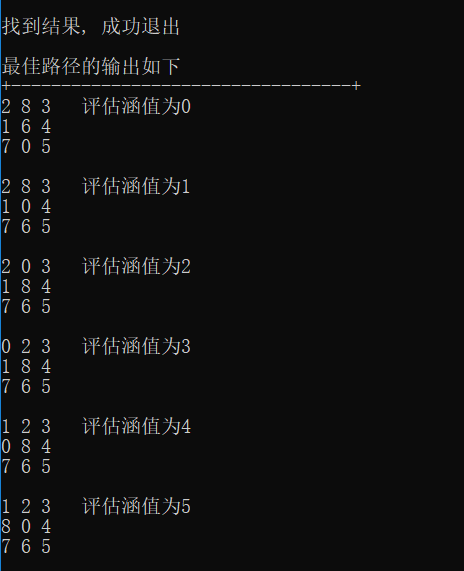
1. **使用启发式函数h1, 动态显示OPEN表的结点数、总扩展的结点数和评估函值最小的结点. 若de出结果, 则输出OPEN表中在最佳路径上的结点及其评估函数值**



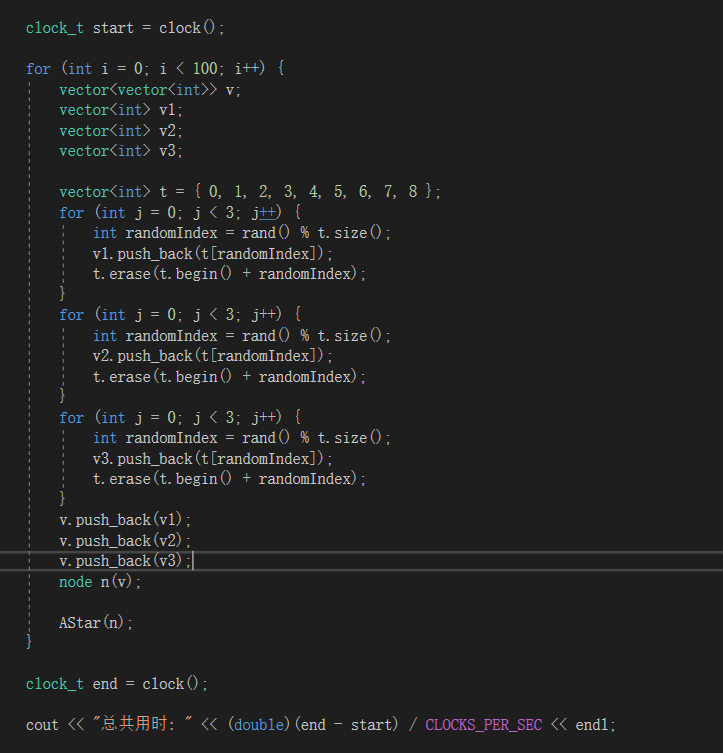


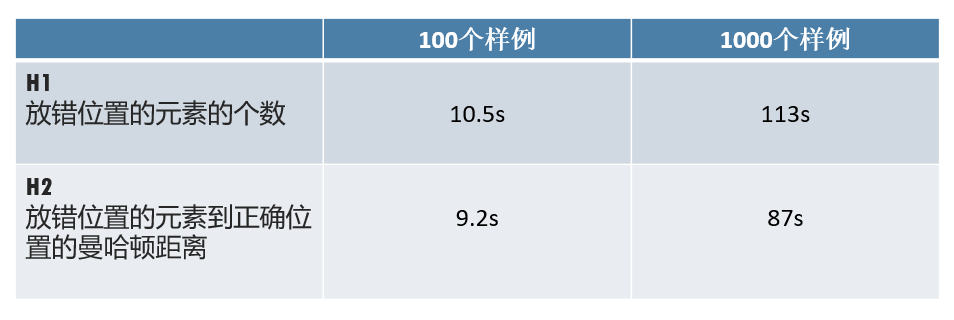
1. **使用启发式函数h2, 动态显示OPEN表的结点数、总扩展的结点数和评估函值最小的结点. 若de出结果, 则输出OPEN表中在最佳路径上的结点及其评估函数值**





1. **比较启发函数h1和h2的搜索效率**
2. 由上述的结果可以看出, 为了搜索到结果, h1执行了6步, h2执行了5步, 因此h2的搜索效率相对较高
3. 使用100个例子



 **因此可以看出, h2的搜索效率更高**

**4. 验证凡A\*算法挑选出来求后继的点n必定满足: f(n）≤f\*(S0）**

令S0 = n0, n1, n2, …, nk =Sg为一条最优路径, 设n’  path(n0, n1, n2, …, nk)中最后一个出现在Open表上的元素. 显然, n’一定存在, 因为至少有S0 = n0必然在Open表上, 只考虑当nk还未出现在Closed表中时, 因为若nk已在Closed表中时, 则nk = Sg, A\*算法将终止于成功退出.

由定义有

f(n’) = g(n’) + h(n’) = g\*(n’) + h(n’) (因为n’在最优路径上)

g\*(n’) + h\*(n’) = f\*(n’) = f\*(S0)

所以, f(n’) f\*(S0)成立

**5. 验证h1(n)的单调性，显示凡A\*算法挑选出来求后继的点ni扩展的一个子结点nj，检查是否满足: h(ni）≤1+h(nj）**

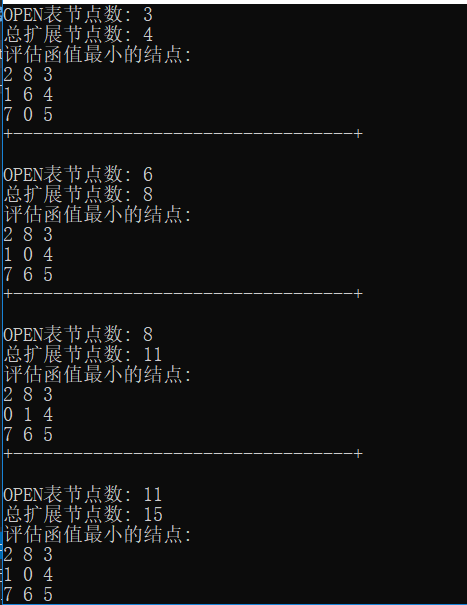
h(ni)表示从节点ni到目标节点的最佳路径的代价, c(ni, nj)表示节点ni到节点nj的最佳路径的代价, 根据三角不等式可知:

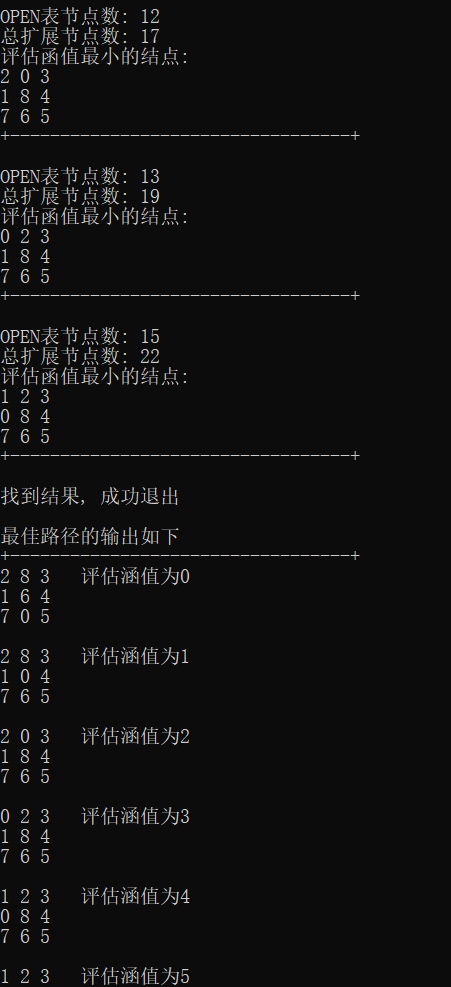
h(ni) h(nj) + c(ni, nj)

又因为nj是ni的后继节点, 所以c(ni, nj) = 1, 带入得

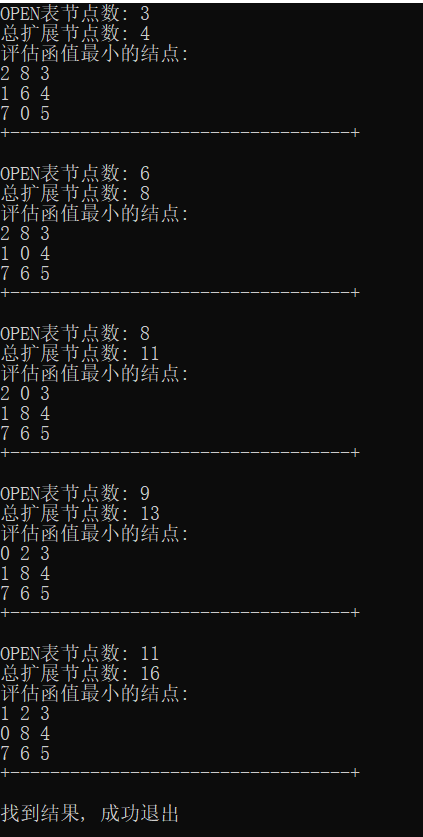
h(ni) h(nj) + 1

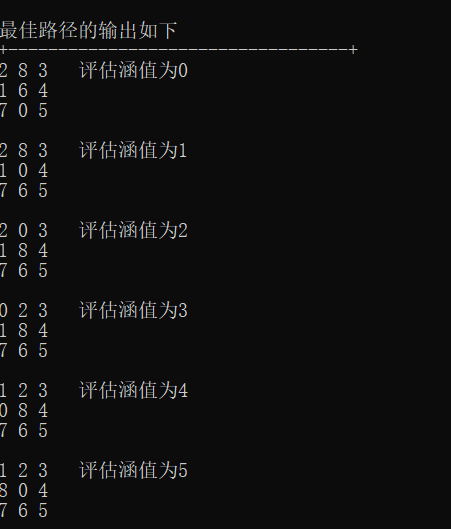
**6. 对于九数码问题, 启发函数h1(n)的结果如下**





**7. 对于九数码问题, 启发函数h2(n)的结果如下**





由上述的结果可以看出, 八数码和九数码问题的搜索图不同

8. 当除0外的数字的位置都没有错误时, 到达目标状态