Comp472 Assignment 2 Demo

Team: anonymous

Yifan Yang 40038814

Node

generateNextSteps(self): Depends on the 0’s position, it will generate several possible operations in self.nextSteps[]

generateSuccessors(self): Loop self.nextSteps[],generate possible successors. And return successors[], which will be later used in search algorithm.

\_\_lt\_\_, the priorityQueue() use heapq to heapsort. Decides comparison relationship.

Heuristics

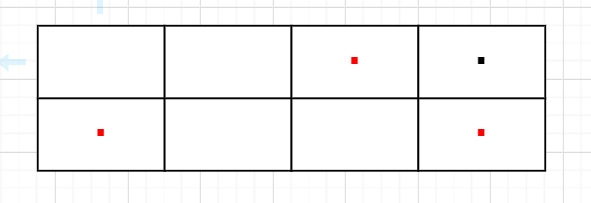
1.Hamming distance

2.myHeuristics: cause wrapping and diagonal ,h2 h3 in slides are not admissible.

H2 is partly depends on Manhattan distance

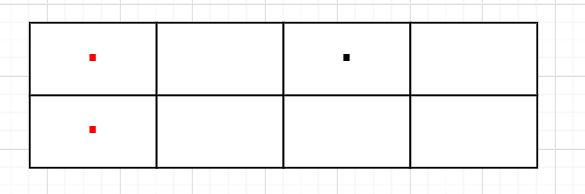
Cost1: Abs(tile1-tile2)=1 or 4 , when tile difference is 1, mainly regular move

it may also be diagonal, but 1<=3 ,still admissible

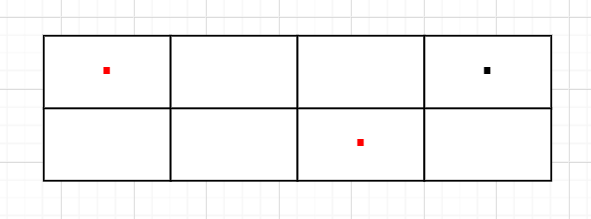


Cost2: Abs(tile1-tile2)= 2 or 3 or 5 ,

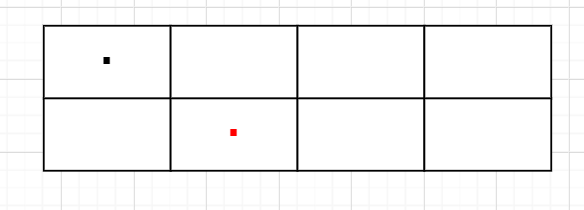
When 2: in same line, 2 regular moves. In different rows, 3 regular moves, 2<=3, admissible



When 3: in same line, wrapping, In different rows. Diagonal or 2 move, 2<=3 admissible



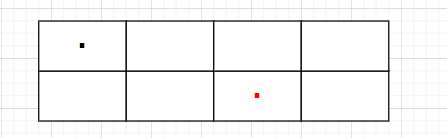
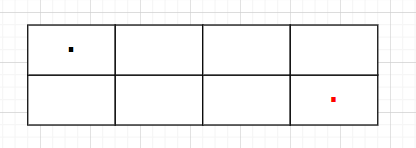
When 5: in different rows. Diagonal or 2 move, 2<=3,admissible



Cost 3: Abs(tile1-tile2)=6 or 7

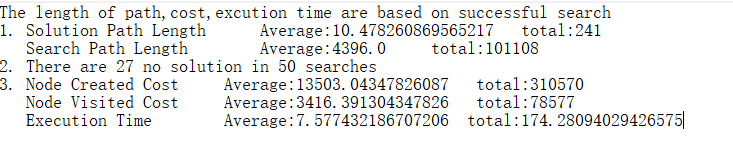
When6: 3 regular move

When7: Diagonal

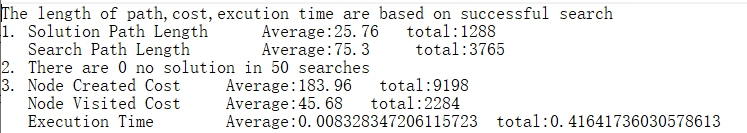


Analysis

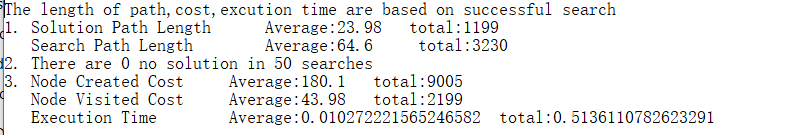
UCS



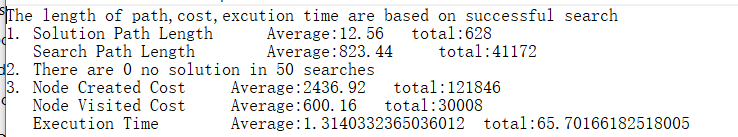
GBFS\_h1



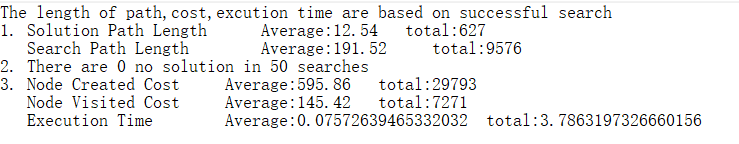
GBFS\_h2



aStar\_h1



aStar\_h2



Cost Analysis

As uninformed search, UCS is slowest. In 60 seconds, nearly half of puzzles cannot be solved in time. It has to visit all smaller g(n) states that not in closed list until reach goal. Which means its search path is very long

In informed searches. **GBFS is faster than aStar** . Cause it like DFS, once reach goal state, it doesn't need to backtrack again. Which means it has smaller Search Path Length. Secondly, GBFS only need to compare new node with open list. aStar needs to compare both open list and closed list. **But aStar can generates optimal solution.**

If optimal solution isn't required, GBFS can be seemed as a good-enough solution in puzzle problems.

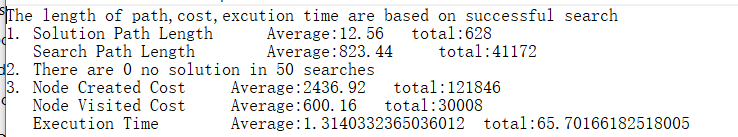
Heuristics influence:

in GBFS, cause the search path of GBFS is shorter, the h(n) hasn't been called too many times.

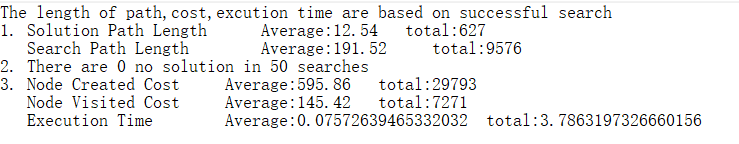
Both h(n) is quick, so doesn't influence too much.

in aStar, h(n) play a more important role because its long search path. The more informed heuristics , the less backtracking, the shorter the search path. Which optimizes the running time a lot

h1



h2



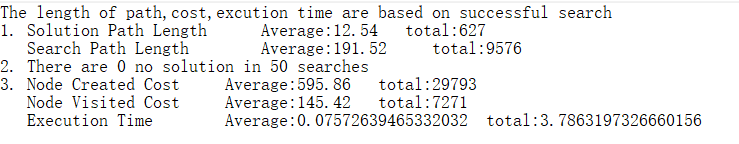
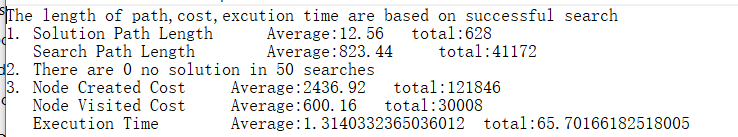
Optimality

Theoretically, UCS and aStar will be optimal

In implementation.

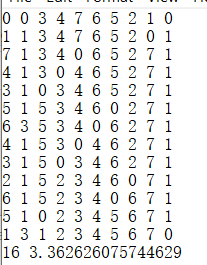
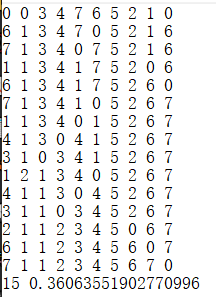
UCS can still hold optimality cause it is only depends on g(n)

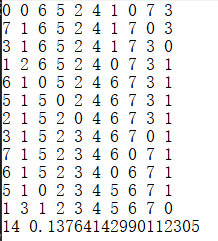
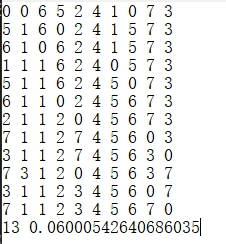
aStar, most puzzles can be done with shortest path. However, in some rare case, different heuristics will generate different solution path length.

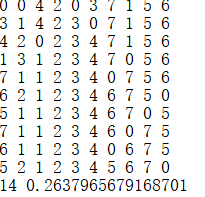
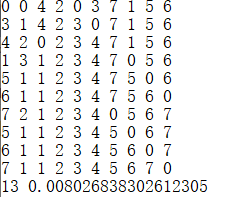


After observing several random puzzles.

H1 H2

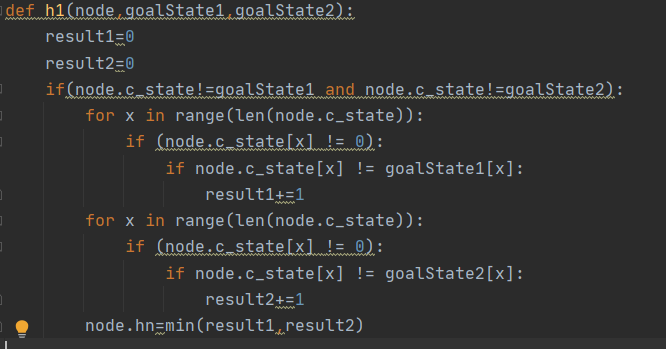
Observation . If different paths are generated,

1.the last step of longer path will have step bigger than 1.

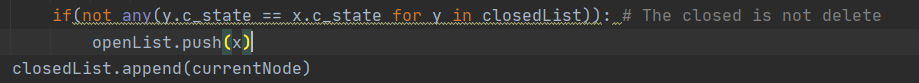
2.The longer path will have more big steps

Possible Reason:

1. hn is not always <= h\*(n): Possibility:low . The one cannot generate shortest path is h1, whose heuristics is Hamming distance. Hamming distancce should be admissible



2.Algorithm Problem: Possibility:low . Cause the hn of goal state is 0. fn=gn, And the heap is sorted by fn. Shortest path should be popped earlier than other goal state. Unless it is not pushed into Open.

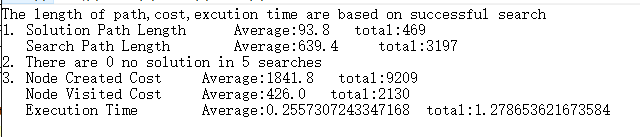


However, the node will be pushed into closed list after checking goal state. Once reach goal state, the function will return. So there is no goal state in closed state. So goal state should be pushed into openList

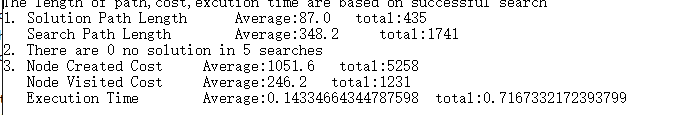
Scaling

I use 3\*4 puzzles with GBFS\_h1,GBFS\_h2, Astar2

GBFS\_h1

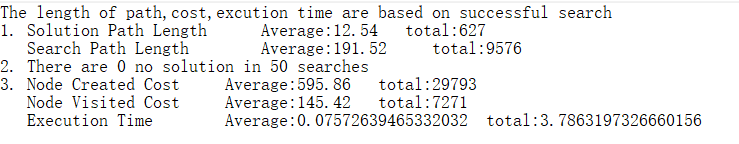


h2

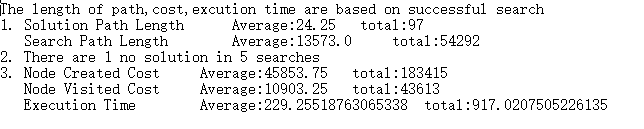


With larger puzzles. The search path of GBFS is longer. Quantitative change produces qualitative change. The influence of hn becomes more obvious

a\_star\_2\*4



a\_star\_3\*4



However, aStar are way more influenced by size of puzzle compared to GBFS. Because it is harder to estimate actual hn. The backtrack and searchPath length increase vastly.

Besides, the larger depth of tree creates more nodes. which means that with larger solution path length, the growth rate of node created and node visited increases sharply.

