N-PUZZLE SOLVER

omar sayed mostafa

FACULTY OF COMPUTER AND INFORMATION SCIENCE **Ain shams University**

A Little Brief about N-Puzzle solver:

N-Puzzle Application is a soft-ware to solve N-Puzzle Game Using AI algorithm (A*) with some help from other Algorithms like Priority Queue and Map.

About Project Logic Steps To Solve N-Puzzle:

1-Reading a puzzle from file.

2-Parse the puzzle and Pass the puzzle details and data to Class Detect.

3- Class Detect Receive Puzzle details and Decide If this Puzzle solvable or Not.

4-After deciding If the puzzle solvable, Puzzle details and data are sent to A* class which use A* search Algorithm with help from priority Queue and Map.

5-After Find the Goal, Goal Node return its Parents (Steps Path).

A* Algorithm "How It works":

- First receive start node, Consider start node is the "Current node".
- Calculate [Heuristic Function], F(N)=G(N)+H(N); where G(N) is number of moves and H(n) is a Heuristic Function(Hamming / Manhattan).

- After calculate priority of current node, discover it's child and calculate their priority , nodes is "En-Queue" to Priority queue as their priorities , then De-Queue from Priority Queue the next Node to Process on it the same steps (recursion the same steps on each node) till to find goal node.

- When discover a node's child, should make sure first that child doesn't exist in Closed or Open sets.
  If exist in Closed Set and Priority of the current node is more than the one in closed set, then remove it from closed and push current node again to Queue.
- Else if it's exist in Open set and its priority is less than the current node, remove it from Open set and push the current node.

Reading from class “File”

- Reading Puzzle from a File: Actually Reading a puzzle coast **O(N*N)** which N= size of Puzzle.
- It reads char by char in file and parse it to integer and fill 2D array[N,N], So In all cases will Coast **O(N*N).**

Class “Detect” of puzzle solvable or not

- Actually Detecting Coast **O(N-Log(N))** where N in this case equal size of puzzle square .

- **Detecting is happening on 2 steps** :
  1. Calculating number of swapping to locate every number in his right place, that happening by “merge swapping” to Find number of swapping and it coast **O(N Log N).**

  2. Detect the solvability of puzzle by theory, if puzzle size odd, so in each move it generate even number of swaps.
     Else if the puzzle size is even, so in each move/slide it should be “number of swaps needed” + “zero| blank tile row” = odd number.
     Total Complexity of Detect O(N-Log(N)).

A* algorithm

A* coast depend on number of moves needed to solve the puzzle where as the number of moves increase as the coast increase.

Generally, it can be said total complexity equal to =

"size of priority queue" *Log(N)

Where **N** is the **MAX**(closed, open, priority queue) members number in each iteration at worst case.

Class "Node"

Complexity of each node:

- Each node need to discover its child.
- Discover child coast for every child **O(N*N)** while N is size of puzzle(coast for calculate **heuristic function**, initialize child).
- Every node after discover its child ,it must search first at (Open ,close) set to make sure that child isn't discovered before or the new child is best than the old one discovered before it's take **O(log(N))** where N is the maximum number of elements in(open | closed)set.
- **Total** complexity is **O(Log N)+O(M*M)** where N is the maximum number of elements in(open | closed)set, and M is the puzzle size.

Priority Queue

- Priority queue is based on heap sort algorithm which coast to **O(Log N)** [heaping sort always make the most priority on the top].

- Add element to priority queue coast **O(1)** ,then re-heaping the queue coast O(Log N) ,totally =**O(Log N).**

- Removing from priority queue the same as adding, coast = **O(Log N).**
