8-Puzzle Solver

CS 293 Final Project Demo

28-11-2012

Janga Varun – 110050076

T Pradeep Kumar - 110050074

Outline <for a total of 30 mins>

- Aim of project (1 mins)
- Demo (5 mins)
- Teamwork Details (0.5 min)
- Design Details –Algorithm (5 mins)
- Design Details – Implementation (8 mins)
- Viva (9 mins)
- Transition time to next team (2 mins)

Aim of the project

- The aim of the project is to
  - Successfully understand and implement various algorithms using different heuristics to solve the 8-puzzle(N^2 -1 in general). Also study the working of the genetic algorithm.
  - Create a Graphical Application on Qt which makes use of the implemented algorithms.

Demo

- *Now explain the program further while doing the demo*
  - *Tip: one member can explain, one member can do demo*

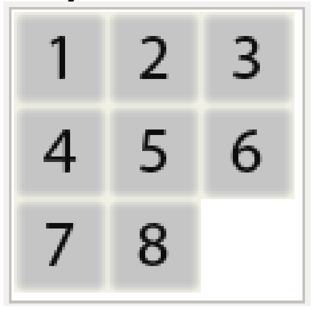
Teamwork Details

- Varun :
  - Implented IDA-star algorithm.
  - Implemented the graphics using Qt.
- Pradeep:
  - Implemented A-star algorithm.
  - Implemented the heuristics.

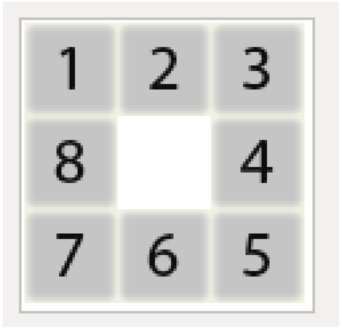
Overall Contribution by Janga Varun	Overall Contribution by Pradeep
60%	40%

The (N^2 – 1) Puzzle

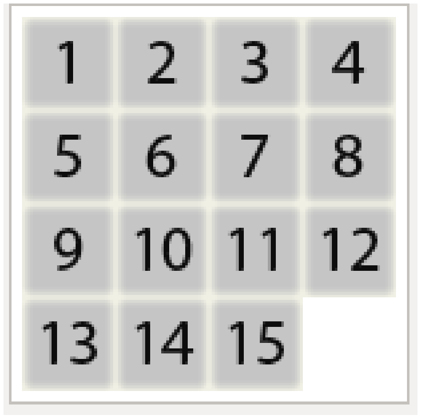
- The 8-puzzle is a square board with 9 positions, filled by 8 numbered tiles and one gap.The gap can be swapped with an adjacent tile. The objective in the game is to begin with an arbitrary configuration of tiles, and move them so as to reach goal states(as shown below)in fewest number of moves.



1	2	3
4	5	6
7	8	



1	2	3
8		4
7	6	5



1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

- It is solved using A* and IDA* algorithms.
- *IDA-star is effiecient in terms of memory.*

Design Details

- A-star Algorithm
  - A* finds one of the least-cost path from a given initial state to the goal state.
  - As A* traverses, it follows a path of the lowest known heuristic cost, using a sorted priority queue of alternate possibilities along the way.
- Implementation From
  - The initial state we generate different possibilities , give them scores based on some heuristics and they are sorted in the priority queue. From the queue we take the best move and check,
    - If goal is not reached, add its neighbours to the same priority queue
    - If goal is reached, return the path.

Design Details

- IDA* Algorithm
  - It is an informed search based on the idea of the uninformed Iterative Deepening Depth-First-Search.
- Implementation
  - Implemented using depth limited search
    - which has an iteratively increasing limit to the depth
    - If limit = k,DepthFirstSearch is done till it reaches depth =k.
  - Here comes the informed part,during this search some children at a node are skipped if their cost to reach goal is more than that of cost_limit(which is similar to k above).

Class Design – High level

- *We have used two classes Board,Solver for the algorithm implementation.*
- *QApplication and MainWindow are the main classes whereas many other classes from Qt have been used for the graphics implementation.*
- *Most of the implementation stick to the algorithms . Have not made any changes to the existing algorithms.*
- *But in the solver class function solvable has been implemented differently knowing the fact that only one of the twin boards has a solution. For any board its twin board is the board came from swaping two numbers(not including 0) if the orginal board. For example 1 2 3 4 6 5 7 8 0 and 1 2 3 4 5 6 7 8 0 are twin boards. Only one of them has a solution.*

Class Design - Details

Class Name	Brief Description
Board	*Stores the configuration of the numbers. (2-d array,direction queue, heuristics,helper functions,operator overloaders,hole positions,parent board…etc. These are the members of the class).*
Solver	Contains the details like which algorithm to be used and which heuristic to be used and checks whether the given board is solvable or not.(initstate,algo,heuristic,a-star and ida-star algos,solution steps,generate moves,depth limited search(...),isSolvable(),solve() ..etc. These the members of the class.)

Class Design - Details

Class Name	Brief Description
MainWindow	Displays the mainwindow and has methods to update solver entities like Algorithm, Heuristic,BoardSize and display the output after solving, also an animation is provided to view the step by step sliding to reach the goal

Data Structures Used

Purpose for which data structure is used	Data Structure Used	Whether Own Implementation or STL
boardQueue	*Priority queue*	*STL*
moves	multimap	*STL*
neighbours,path	queue	STL
solution_steps,direction	queue	STL

Data Structures Used

Purpose for which data structure is used	Data Structure Used	Whether Own Implementation or STL
Timer for animation	QTimeLine	Qt Library-QTimeLine
Animation Object	QGraphicsItemAnimation	Qt Library
	Many more similar to above used for graphics (inbuilt in Qt)	

Source Code Information

File Name	Brief Description	Author (Team Member)
Board.h	*Contains board class*	both
Board.cpp	*Contains the functions of the board class*	both
Solver.h	Contains solver class	both
Solver.cpp	*Contains the functions of the solver class*	both
Main.cpp	Initiates QApplication and MainWindow	-

Source Code Information

File Name	Brief Description	Author (Team Member)
MainWindow.h	*Contains MainWindow class*	*Varun*
MainWindow.cpp	*Contains the functions of the MainWindow class,which help in interacting with algorithm*	*Varun*
uimainwindow.h	*Contains Graphics Information*	*Qt IDE*

Brief Conclusion

- We here by conclude that, we have successfully implemented the 8-puzzle solver application on Qt using A* and IDA* algorithms with 3 heuristics common to 3x3 and 4x4 along with 2 other heuristics specific to 3x3

Thank You – Questions?

Back Up Slides

- *<Use as many slides as you need to keep as much information handy as you need to, to show in case I ask a question>*