Discrete Mathematics Project

Topic :
ToC and Ant colony Algorithms

Assigned by:

Prof. Manish Gupta & Prof. Manoj Raut

Contents

1 Introduction

1.1 What is theory of computation ?

- A subfield of computer science called theory of computation (TOC) studies how effectively and efficiently problems can be solved using algorithms. Real-world computers do calculations that, by their very nature, behave like mathematical models to find systematic solutions to problems.
- Basic Terminologies of Theory of computation are Symbol, Alphabets, String etc.

References : [4] [1]

1.2 What is Ant Colony Algorithms?

- The algorithmic world is delightful with diverse procedures and apparatuses being created nonstop to deliver to the requirement for superior execution figuring. As a matter of fact, when calculations are propelled by normal regulations, it are seen to intrigue results. Transformative calculations have a place with such a class of calculations. These calculations are planned in order to mirror specific ways of behaving as well as developmental characteristics of the human genome.
- Also, such algorithmic plan isn't simply obliged to people however can be motivated by the regular way of behaving of specific creatures too. The fundamental point of manufacturing such procedures is to give sensible, important but a few minimal expense answers for issues that are until recently unsolvable by regular means.
- Different enhancement methods have in this manner developed in view of such transformative calculations and subsequently opened up the area of metaheuristics. Metaheuristic has been gotten from two Greek words, specifically, Meta meaning one level above and heuriskein significance to find. Calculations, for example, the Particle Swarm Optimization and Ant Colony

Optimization are instances of multitude knowledge and meta-heuristics. The objective of multitude insight is to plan wise multi-specialist frameworks by taking motivation from the aggregate way of behaving of social bugs like insects, termites, honey bees, wasps, and other creature social orders like runs of birds or schools of fish.

References : [2] [3]

2 Maze Game

2.1 Application

How to get solution of any type of maze game?

2.2 formulate the mathematics

We can find the solution of the maze game using the concepts of NFA in discrete mathematics course. We can use different NFA state to demonstrate the working of maze game.

2.3 Solution

- In this maze game, we know that there will be some bunch of blocks so we have to make a path in which the player can go to the end state.
- So to make these type of path, player should go in different direction like up, down, right, left etc. By using distinct state of NFA we can make this.
- If the player choose different path then he/she can't reach till the end state.The Player will move ahead till the dead state.
- while changing the state we make the normal state all of them except the last one and when player reach to the end state he/she won the game.
- The best rule for traversing mazes is the wall follower, also known as either the left-hand rule or the right-hand rule.

- If the maze is simply connected, that is all its walls are connected together or to the maze's outer boundary, then by keeping one hand in contact with one wall of the maze the solver is guaranteed not to get lost and will reach a different exit if there is one.
- otherwise, the algorithm will return to the entrance having traversed every corridor next to that connected section of walls at least once.
- Another perspective into why wall following works is topological. If the walls are connected, then they may be deformed into a loop or circle.
- Then wall following reduces to walking around a circle from start to finish. To further this idea, notice that by grouping together connected components of the maze walls, the boundaries between these are precisely the solutions, even if there is more than one solution .
- If the maze is not simply-connected e.g if the start or endpoints are in the center of the structure surrounded by passage loops, or the pathways cross over and under each other and such parts of the solution path are surrounded by passage loops. This method will not necessarily reach the goal.

- Another concern is that care should be taken to begin wall-following at the entrance to the maze.
- If the maze is not simply-connected and one begins wall-following at an arbitrary point inside the maze, one could find themselves trapped along a separate wall that loops around on itself and containing no entrances or exits.
- it Should be the case that wall-following begins late, attempt to mark the position in which wall-following began.
- Because wall-following will always lead you back to where you started, if you come across your starting point a second time, you can conclude the maze is not simply-connected, and you should switch to an alternative wall not yet followed.

2.4 Algorithm

begin procedure

- Cursor(x,y), sizeofmaze(z,w), input W, A, S, D, Q
- Cursor(x, y) = (1, 0)
- sizeofmaze (z, w) = (20, 35)
- Enter 'W' for go to up direction
- Enter 'S' for go to downward direction
- Enter 'D' for go to right direction
- Enter 'A' for go to left direction
- Enter 'Q' for quit the game
- switch(input)
  case 'W' :
  x=x-1
  y=y

  case 'S' :
  x=x+1
  y=y

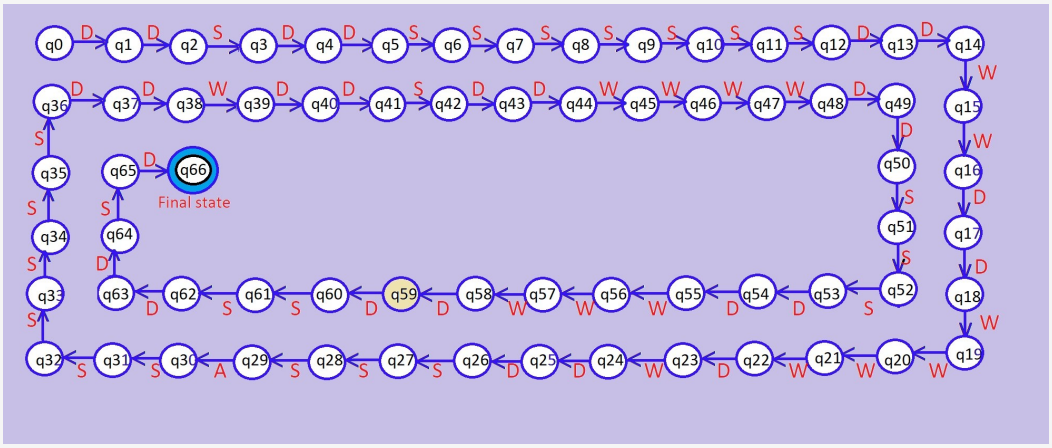
  case 'D' :
  x=x
  y=y+1

  case 'A' :
  x=x
  y=y-1

  case 'Q' :
  Exit

end procedure

2.5 Figure



NFA OF MAZE GAME

W = up direction
A=let direction
D=right direction
S=down direction
Q=quit

3 Snake Game

3.1 Application

How does the snake game work?

3.2 Formulates the mathematics

The snake game works on the basis of the concepts of DFA.Everything is controlled using the position of the head of snakes.

3.3 Solution

If the head is moving in the one direction than it cannot move in its opposite direction. We can only move perpendicular to that direction. Once a fruit is eaten the size of the snake is increasing and the game becomes more difficult. You have 2 ways to lose the

game. One is by hitting the wall and second is by impinging the snake body itself with its head. You can also press Q to quit the game any time you want.There is also the difficulty levels like slow, medium and fast.

3.4 Algorithm

begin procedure

- snakegame(char input, W, S, A, D, Q, direction)
- Enter 'W' to go upward direction
- Enter 'S' to go downward direction
- Enter 'D' to go right direction
- Enter 'A' to go left direction
- Enter 'W' for Quit the game

- while(!wall !own input!=Q)
  if(input = W)
  [if(input = W)
  continue]

  else if(input = A)
  [direction = left
  direction++]

  else if(input = D)
  [direction = right
  direction++;]

  else if(input = S)
  print (game is over)

  else
  print (Invalid Move)

- if(input = S)
  [if(input = S)
  continue]

  else if(input = A)
  [direction = left
  direction++;]
  else if(input = D)
  [direction = right
  ]direction++;]

  else if(input = W)
  print (game is over)

  else
  print (Invalid Move)
- if(input = A)
  [if(input = A)
  continue]

  else if(input = W)
  [direction = up
  direction++;]

  else if(input = S)
  [direction = down
  direction++;]

  else if(input = D)
  print (game is over)

  else
  print (Invalid Move)

- if(input = A)
  [if(input = A)

continue]

else if(input = W)
[direction = up
direction++]

else if(input = S)
[direction = down
direction++]

else if(input = D)
print (game is over)

else
print (Invalid Move)

- if(input = D)
[if(input = D)
continue]

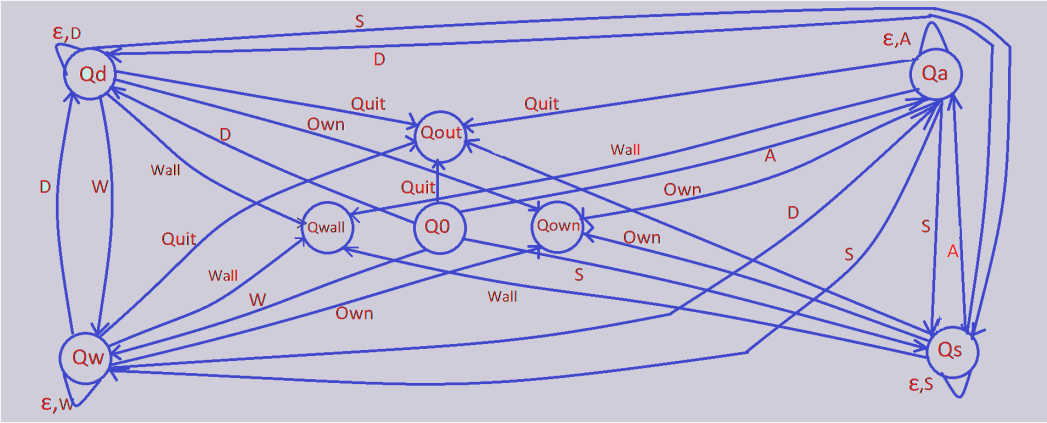
else if(input = W)
[direction = up
direction++]

else if(input = S)
[direction = down
direction++]

else if(input = A)
print (game is over)

else
print (Invalid Move)

end procedure



DFA OF SNAKE GAME

W = up direction
A=let direction
D=right direction
S=down direction
Q=quit

4 Pac-run

4.1 Application

Application of Ant Colony Algorithm in real life.

4.2 Solution

- We have made a game where you need to run from the enemy. The enemy here uses Ant Colony Algorithm.
- Everytime you move one step the enemy will use ACA and find the shortest distance and its shortest path respectively to catch you.
- It will not collide to walls as it will consider it a obstacle.

Concept

- Subterranean insects are eusocial bugs that favor local area endurance and supporting as opposed to as individual species. They speak with one another utilizing sound, contact and pheromone.

- Pheromones are natural substance intensifies discharged by the subterranean insects that trigger a social reaction in individuals from same species.
- These are synthetic compounds fit for behaving like chemicals outside the body of the discharging individual, to affect the way of behaving of the getting people.
- Since most insects live on the ground, they utilize the dirt surface to leave pheromone trails that might be followed by different subterranean insects.
- Insects live in local area homes and the basic rule of ACO is to notice the development of the insects from their homes to look for food in the most limited conceivable way.
- At first, insects begin to move haphazardly looking for food around their homes. This randomized inquiry opens up various courses from the home to the food source.
- Presently, in light of the quality and amount of the food, insects convey a part of the food back with fundamental pheromone focus on its bring way back.
- Contingent upon these pheromone preliminaries, the likelihood of choice of a particular way by the accompanying insects would be a directing variable to the food source. Clearly, this likelihood depends on the fixation as well as the pace of vanishing of pheromone.
- It can likewise be seen that since the dissipation pace of pheromone is additionally a game changer, the length of every way can without much of a stretch be represented.

4.3 Algorithms

- Before the game starts the enemy knows the map. Everytime it moves it gives back feed on the path and constantly try's to improve the distance between you and him. Everytime you make a move the following procedure happens.

- while not terminated do
  generateSolutions()
  daemonActions()
  repeat
  end procedure

- generateSolution()
  Will find distances between him and you for all paths that are available.

- daemonAction()
  It will take the action on which path to choose and will act upon it.

Made by

- Param Patel : 202101178
- Rushang Parmar : 202101240
- Jay Vakhariya : 202101149
- Sameer Odedara : 202101245

References

[1] Jean-Michel Autebert, Jean Berstel, and Luc Boasson. Context-free languages and pushdown automata. In *Handbook of formal languages*, pages 111–174. Springer, 1997.

[2] Anne-Cécile Caron. Linear bounded automata and rewrite systems: Influence of initial configurations on decision properties. In *Colloquium on Trees in Algebra and Programming*, pages 74–89. Springer, 1991.

[3] Marco Dorigo and Thomas Stützle. Ant colony optimization: overview and recent advances. *Handbook of metaheuristics*, pages 311–351, 2019.

[4] Brink van der Merwe, Hellis Tamm, and Lynette van Zijl. Minimal dfa for symmetric difference nfa. In *International Workshop on Descriptional Complexity of Formal Systems*, pages 307–318. Springer, 2012.