Assignment . 1

Title : Implement depth first search algorithm and Breadth first search algorithm.

Problem statement : Implement depth first search algorithm and Breadth first search algorithm use an undirected graph and develop a recursive . algorithm for searching all the vertices of graph or tree data structure.

Objective

- Understand and implement BFS/ DFS algorithm.
- Analyze time complexity of search algorithm.

Theory.

DFS :

Depth first search or Depth first traversal is a recursive algorithm for searching all the vertices of a graph or tree data searching. Traversal means . visiting all the nodes of a graph.

→ Depth first Search Algorithm.

A standard DFS implementation puts each vertex of the graph into one of two categories

- Visited
- Not Visited

The purpose of the algorithm . is to mark each vertex as visited while avoiding cycles.

1. Start by putting any one of the graph's v on top of a stack.
2. Take the top item of stack and add it visited list.
3. Create a list of that vertex's adjacent Add the ones which aren't in the visited to the top of the stack.
4. keep repeating step 2 and 3 until the is empty.

- BFS.

Traversal means visiting all the nodes of Breadth first Traversal or Breadth first is a recursive algorithm for searching all vertices of graph or tree data structure.

→ Breadth first Search Algorithm

A standard BFS implementation puts eac vertex as visited to of graph into one of two categories.

- Visited
- Not visited.

The purpose of the algorithm is to mark eac vertex as visited while avoiding cycles.

The BFS algorithm works as follows:

1) Start by putting any one of the graph's ver at the back of queue.
2) Take the front item of the queue and add it to the visited list.

1. Start by putting any one of the graph's
on top of a stack.
2. Take the top item of stack and add it
visited list.
3. Create a list of that vertex's adjacent
Add the ones which aren't in the visited
to the top of the stack.
4. Keep repeating step 2 and 3 until the
is empty.

- BFS.

Traversal means visiting all the nodes of
Breadth first Traversal or Breadth first
is a recursive algorithm for searching all
vertices of graph or tree data structure.

→ Breadth first Search Algorithm

A standard BFS implementation puts ea
vertex as visited so of graph is to on
of two categories.

- Visited
- Not visited.

The purpose of the algorithm is to mark ea
vertex as visited while avoiding cycles.
The BFS algorithm works as follows:

1) Start by putting any one of the graph's ve
at the back of queue.
2) Take the front item of the queue and ad
it to the visited list.

3) Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4). Keep repeating step 2 and 3 until the queue is empty.

Complexity of Depth First Search

- Time complexity of the DFS algorithm : $O(V+E)$ where V is the number of nodes and E is the number of edges.
- Space complexity of the algorithm : $O(V)$.

Complexity of Breadth first Search.

- Time complexity of the BFS algorithm : $O(V+E)$ where V is the number of nodes and E is the number of edges.
- Space complexity of BFS algorithm : $O(V)$

Application of DFS Algorithm:

- For finding the path
- To test if the graph is bipartite
- For detecting cycles in graph.

Application of BFS algorithm

- To built index by search index.
- For GPS navigation

- In minimum spanning tree.

Conclusion:

Implemented depth first search algori and Breadth first search algorithm for undirected graph using recursive algorithm searching all the vertices of graph or tree data structure.

Assignment. 2

Title: Implement A star Algorithm for any game search problem.

Theory.

• A Star Algorithm :

① The A star algorithm is a widely used and effective technique for path finding and graph traversal particularly in search problems encountered in games.

②. It is an extension of Dijkstra's algorithm and uses heuristics to guide its search towards the goal effectively.

③. It combines the benefits of both uniform cost search (like Dijkstra's algorithm) and greedy search. It tries to find the shortest path from a starting node to a goal node by maintaining a priority queue of nodes to be explored.

④ The priority of a node is determined by combining the cost to reach that node from the start node (known as the "g" value) and an estimate of the cost to reach the goal from that node (known as the "h" value)

⑤ A* algorithm extends the path that minimizes the following function:

$$f(n) = g(n) + h(n)$$

Here,

- n is the last node on the path.
- g(n) is the cost of path from the node to node 'n'.
- h(n) is a heuristic function that estimates cost of the cheapest path node 'n' to the goal node.

A* Algorithm.

- The implementation of A* Algorithm involves maintaining two lists - OPEN and CLOSED
- OPEN contains those nodes that have been evaluated by the heuristic function but have not been expanded into successors yet.

A* Search Algorithm.

1) Initialize the open list
2) Initialize the closed list.

- Put the starting node on the open list (you leave its 'f' at zero)
  (F = g + h)

3) while the open list is not empty.

a) find the node with the least f on the open list call it "q"
b) pop 'q' off the open list
c) generate q's 4 successors and set their parents to 'q'
d) for each successor.
   i) If successor is the goal, stop search
   ii) else, compute both g and h for successor.

successor.h = distance from goal to successor
successor.g = q.g + distance between successor and q.
successor.f = successor.g + successor.h.

iii) if a node with the same position as successor is in the OPEN list which has a lower 'f'. than successor, skip this successor.

iv) If a node with same position as successor is in the CLOSED list which has a lower 'f'. than successor skip this successor otherwise. add the node to the open list.

end (for loop)

e) Push q on the closed list
end (while loop)

• Applications:

① Pathfinding in games: A* is extensive used in video games for determining shortest path between two points on a # helps non-player characters (NPC's) navigate obstacles and reach their destin efficiently.

②. Robotics and Autonomous Vehicles: A* is employed in robotics for planning the pa of a robot from one point to another avoiding obstacles in its environment.

③. Natural Language Processing: A* can be applied in various NLP tasks such as machine translation or speech recognition

④ Network Routing: A* can be used to find the shortest path in computer netw such as the Internet or transportation networks.

Conclusion:

We have successfully implemented A* algorithm for game search problem.

• Applications:

① Pathfinding in games: A* is extensive used in video games for determining shortest path between two points on a m It helps non-player characters (NPC's) navigate obstacles and reach their destina efficiently.

②. Robotics and Autonomous Vehicles: A* is employed in robotics for planning the pat of a robot from one point to another w avoiding obstacles in its environment.

③. Natural Language Processing: A* can be applied in various NLP tasks such as machine translation or speech recognition

④ Network Routing: A* can be used to find the shortest path in computer netwo such as the Internet or transportation networks.

Conclusion:

We have successfully implemented A* algorithm for game search problem.

Assignment - 3

Problem statement

Implement greedy search algorithm for any of the following application.

- Selection sort
- Minimum spanning tree
- Single source shortest path Problem
- Job scheduling problem
- Prim's Minimal spanning Tree Algorithm
- Kruskal's minimal spanning tree Algorithm
- Dijkstra's Minimal spanning tree Algorithm.

Software Required :-

Python

Theory.

- Kruskal Minimum Spanning Tree

A spanning tree is a subset to a connected graph G, where all the edge are connected i.e, we can traverse to any edge from a particular edge with or without intermediate Also spanning tree must not have any cycles in it.

Thus, we can say that if there are n vertices in connected graph then no of edges that a spanning tree may have n-1.

A minimum spanning tree has (V-1) edges
V is the number of vertices in the gi
graph.
A minimum spanning tree for a weigh
connected undirected graph is a spanning
with weight less than or equal to weigh
of every other spanning tree.

Step for MST using kruskal algorithm.

1. Sort all the edges in non-decreasing orde
their weight.
2. Pick the smallest edge. checks if it for
a cycle with spanning tree formed so f
If cycle is not formed include this edge
else discard it.
3. Repeat step 2 until there are (V-1) edg
spanning tree.

Prim's Minimum Spanning Tree Algorithm.

A group of edge that connects two set of
vertices in graph theory. So at every step
Prim's algorithm. Pick the minimum weig
edge from the cut. and include this verte
to MST. And they must be connected with
the minimum weight edge to make
a MST.

Algorithm

1. Create a set mstSet that keeps track of vertices already included in MST.
2. Assign a key value to all vertices in the input graph Initialize all key value as Infinite
3. While mstSet doesn't include all vertices
   a. Pick a vertex u which is not there in mstSet and has a minimum key value
   b. Include u to mstSet.
   c. Update key value of all adjacent vertices of u if the weight of edge u-v less than the previous key value of v, update the key value as weight of u-v.

Conclusion.

Thus, we have implemented greedy search Algorithm Prim's Minimal Spanning Tree Algorithm & Kruskal's Minimal Spanning tree Algorithm.

Assignment 4.

Problem statement.

Implement a solution for a constraint satisfaction Problem using Branch and Bound & Backtracking for a n-queens for a graph colouring problem.

Software Required

Python.

Theory

• N queen problem using Branch & Bound.

The N queen puzzle is the problem of placing N chess queens on N×N chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column or diagonal.

Backtracking Algorithm for N-queen is already discussed here. In backtracking solution we backtrack when we hit a dead end. In Branch & Brand solution after building a partial solution, that these is no point going any deeper as we going to hit a dead end.

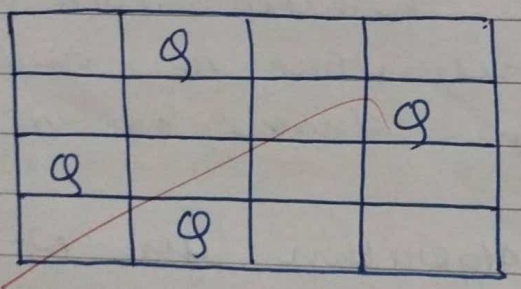
For a N×N matrix, fill slash code and blackslash code matrix using below formula.

slashcode [row][col] = row + col
blackslashcode [row][col] = row - col + (N-1)

The N-1 in the blackslash code is there to ensure that the code are never negative we will be using code as indices in array.

- N Queen Problem using Backtracking.

The N Queen is the problem of placing chess queens on N×N chessboard so no two queens attack each other.
For example, there is a 4 queen problem



The expected output is binary matrix has for the blocks where queens are placed for example, the following is output matrix for the 4 queen solution.

{ 0, 1, 0, 0 }
{ 0, 0, 0, 1 }
{ 1, 0, 0, 0 }
{ 0, 0, 1, 0 }

Backtracking Algorithm.

The idea is to place the queens one by one in different columns, starting from the leftmost column when we place a queen in column. we check for clashes with already placed queens. In current column if we find a row for which there is no clash. we mark this row and column as part of solution If we do not find a row due to clashes. then we backtrack and return false.

1. start in the leftmost column.
2. If all queens are placed, return true.
3. Try all rows in the current column
4. Do following for every tried row.

a) If the queen can be placed safely in row, then mark this [row, column] as part of solution & recursively check if placing queen here leads to a solution.
b) If placing the queen in [row, column] leads to a solution then return true.

c). If placing queen doesn't lead to a solution then unmark this (row, colu and go to sleep via to try other rows.

5) If all rows have been tried and not worked, return false to trigger backtr

Conclusion:

Thus we have implemented a solution for a constraint satisfication.

c). If placing queen doesn't lead to a solution then unmark this (row, colu and go to sleep a to try other rows.

5) If all rows have been tried and not worked, return false to trigger backtr

Conclusion:

Thus we have implemented a solution for a constraint satisfication.

DATE:

Assignment . No 5.

Title : Develop an elementary chatbot for any suitable customer interaction application

Theory :

- understanding the chatbot

A chatbot is an Artifical Intelligence - based software developed to interact with human in their natural languages. These chatbots are generally converse through auditory or textual methods and they can effortlessly mimic human language to communicate with human beings in human - like way. A chatbot is considered are of the best application of Natural language processing.

- Two category of chatbots :

1) Rule - based chatbots :

The Rule-based chatbot approach trains the chatbot to answer questions based on a list of pre - determined rules on which it was primarily trained. These set rules can either be pretty simple or quite complete, and we can use these rulebased chatbots to handle simple queries but not process more complicated requests.

DATE:

2) Self-learning chatbots.

Self-learning chatbot are chatbots that can learn on their own. These leverage advanced technologies such as Artificial Intelligence (AI) and machine learning (ML) to train themselves from behaviours & instance, generally these chatbot are quite smarter than rule-based bots. The self-learning chatbots are further categorized by into two categories.

a) Retrieval-based chatbots:

A retrival based chatbots works on predefined into pattern & sets response. one the question or pattern is instead, the chatbot utilizes a heuristic approach to deliver the relevant response. The model based on retrieval is extensively utilized to design & develop goal-oriented chatbot using custom feature such as the flow & tone of the bot in order to enhance the experience of customer.

b) Generative chatbots:

Unlike retrival-based chatbots, generative chatbots are not based on predefined response they leverage seq+ seq neural network. This is constructed on the concept of machine translation, where the source code

DATE:

is converted from one language to another language. In the seq2seq approach the input is changed into an output.

- Chatbot is present generation.

Today, we have smart chatbot powered by Artificial Intelligence that utilize natural language processing (NLP) in order to understand the commands from humans (text & voice) and learn from experience. Chatbots have become a stable customer interaction utility for companies and brands that have an action online existence with the help of python, chatbots are considered a nifty utility as they facilitate rapid messaging between brand & customer.

Benefits of chatbots :-

- 24x7 availability
- Instant answer to queries
- Multi-lingual support to ~~queries~~ enhance business
- Simple & easy to use UI to engage customers
- Cost effective & uses interactive
- Understand the customer behaviour
- Increase sales of business by offering promo codes or gifts.

DATE:

Conclusion :-

Thus we have develop an elementary chatbot for any suitable customer interaction application.

DATE:

Assignment. No 6.

Title : Implements any one of following expert system.

1. Information managements.
2. Hospital & medical facilities
3. Help desks management
4. Employee performance evaluation
5. Stock market analysis
6. Airline scheduling & cargo schedules.

Theory.

• Expert syetems.

AI is a piece of software that simulates the behaviour & judgements of a human or organisation that has experts in a particular domain is known as expert system. It does this by acquiring relevant knowledge from knowledge base & interpreting is according to user's problem.

It is widely used is many area such as material diagonsis, accordinig coding games etc. An expert system is AI software that uses knowledge stored is knowledge base to solve problem that would usually require human expert knowledge is its knowledge base

DATE:

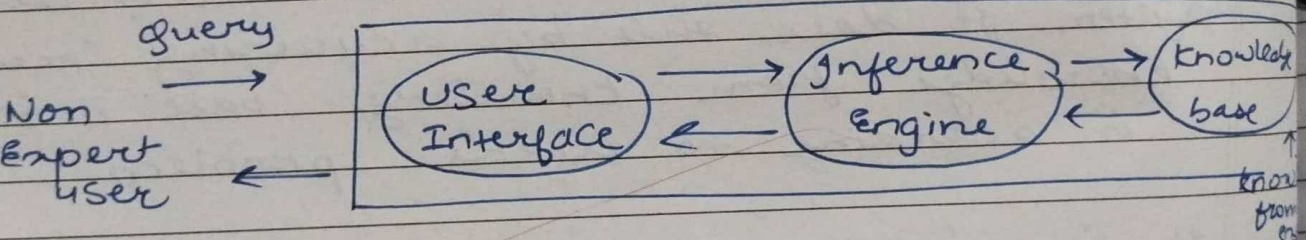
Example:

> MYCIN: One of the earliest expert s... based on backward chaining. It can identify various that can cause se... infections & can also recommend day... on person's weight.

> R1/XCON: It could select specific soft... to generate computer system wish... by user.

> CaDep: It is a clinical support system th... could suggest of variety of disease... based on finding doctor.

Components of Expert System -



Two strategies.

1) forward chaining

2) Backward chaining

DATE:

Advantages:

1) Low accessibility cost
2) fast response
3) Not effected by emotions
4) low error rate

Disadvantage:

1) No emotions
2) common sense
3) Developed for specific domain
4) Need manual updation.

Conclusion

Thus we have implemented expert system.