Practical No-1

Aim: Implement Breadth first Search algorithm for Ramanian map problem.

Theory:

- BFS is a traversal approach we first walk through all nodes on the same level before making to the next.
- BFS was the first in first out approach in queue.
- BFS algorithm is non-recursive algorithm that uses of no concept.
- BFS used to find the shortest path in various graph.
- BFS is slow as compared to DFS.
- All neighbours first and therefore not suitable for decision making tree used in games.
- BFS algorithm visit sibling before child.
- Time complexity TC= O(V+E), Adjancy list is used and $O(V^2)$ where as v-vertius and E-edges.

Conclusion: We have successfully implemented Breadth First Search algorithm.

Practical No-2

Aim: Implement Iterative deep depth first search for Romanian Map Problem.

Theory:

Inverse

- Iterative Deep Depth First Search algorithm is combination of Breadth First Search & Depth First Search
- IDDFS calls DFS for different depth starting from initial value.
- In every call DFS is restricted from going beyond given depth.
- So basically we do DFS in BFS fashion.
- An important thing to note is we visit top level nodes multiples times.
- The last node is visited once, second last node visited twice and so on.
- It has two type of illustrations.

a) When the graph has no cycle
   - This case is simple, we can use DFS algorithm multiple times with different height limits.

b) When the graph has cycle
   - This is interesting as there is no visited flag in IDDFS.

800

Algorithm:

```
bool IDDFS (src, tar, max-depth):
    for limit from 0 to max-depth:
        if DLS (src, tar, limit) == true:
            return true
    return false

bool DLS (src, tar, limit):
    if (src == tar):
        return true:
    if (limit <= 0):
        return false
    for each adjacent i for sure
        if DLS (i, tar, limit ?1)
    return False
```

Conclusion: We have successfully implement iterative deep depth first search.

Practical No-3

Aim: Implement A* search algorithm for Romanian Map Problem.

Theory:

- The information about the path cost and distance of the goal node and path to reach goal node helps in effectionately finding the solution to problem statemer
- In this algorithm the open list and close list are maintained where the close list represents the the node which are already expanded and the open list represents the nodes which are yet to expanded.
- A star algorithm represents the algorithm which include the feature of BFS and DFS.
- Each node with lowest heuristic values is expanded a generates its successor and the algorithm continuous the goal state is reached.

Conclusion: We successfully implemented A* search algorithm.

200

Practical No.-4

Aim: Implement recursive best first search algorithm for Romanian Map Problem.

Theory:

- In BFS and DFS when we are at a node we can consider only adjacent as the next node.
- So both BFS and DFS blindly explore paths without considering any cost function.
- Best first Search falls under category of informed search.
- Implementation of Best First Search
  - It use a priority queue or heap to store the cost of nodes that have the latest evaluation function value.
  - So the implementation is a variation of BFS are just need to change Queue to priority Queue.

- The worst case time complexity for BFS is $O(n * \log n)$ where n is the member of nodes.
- In the worst case we may have to visit all nodes before we reach goal.
- The peformance of the algorithm depends on how well the cost or evaluation function is designed.

Algorithm

```
BFS (Graph g, Node)
i. Create an empty Priority Queue (pq)
ii. Insert 'start' in pq.
iii. Until pq is empty
        u= pq delete min
        if u is the goal
            Exit
        Else
            Foreach neighbour v of u
            if v is in unvisited
                mark v is visited
                pq.insert(v)
            Mark u as examined
        End procedure
```

Conclusion : We have successfully implemented the recursive best first search.

700

Practical No. 5

Aim: Implement the decision tree learning algorithm for restaurant waiting problem.

Theory:

In decision tree algorithm, it has the non-terminal & terminal nodes and it returns the class labels. Non-terminal node use as a test function for making a decision and represents the set of decision rules. various algorithm exists in machine learning to choose the best algorithm from the given set concept of decision tree is employed.

Here the learning is continues and based on feedback. We need to identify the attributes selection and the process are known as attributes selection and measures of attributes selection are:

i) Information Gain:-

It provides the information when the use of nodes in the decision tree to partition the training instances into small subsets.

ii) Gini Index:-

Is measured by substracting the sum of squared probabilities from each class from the value of while the information gain is obtained by multiplying the probability of the given by log of class probability.

200

Practical No-6

Aim: Implement Feed Forward back propogation neural network algorithm for given Dataset.

Theory:

→ The process of adding more layers to a neural Network called as Deep learning.

→ The main difference in the code from the single layer network is that two layers influence the calculations for the error, therefore the adjustments of weights.

→ The errors from the second layer of neurons need to the propagated backwards to the final layer. This is called back propogation.

→ The Back Propogation algorithm performs learning on a multiplayer feed-forward neural network.

→ A multilayer feed-forward network consists of an input layer, are or more hidden layers & an output layer.

Algorithm:

Step 1: Set the intial of weight set a smaller random non-zero number as co-efficient of each layer but set a co-efficiently initially.

Step 2: Enter a sample, as the corresponding desired output as well.

Step 3: Calculate the output at all layers.

Step 4: find the learning error for all layers.

Step 5: Modify weights & threshold

Step 6: When the weights of different layers are calculated the quality indicators can be set to determine whether the requirements are met or not. If the requirements are met, the algorithm ends; or return to step 3. The learning process for every given sample & desired output should be implemented, until all the input & output requirements are met.

Conclusion: Hence we completed the experiment with the desired output.

Practical No-7

Aim: Implement the Adaboost ensemble learning Algorithm.

Theory:

→ Boosting is an ensemble modelling technique that attempts to build a strong classifier from the number of weak classifiers.

→ It is done by building a model by using weak models in series.

→ Firstly, a model is built from the training data.

→ Then the record model is built which tries to correct the errors present in the first model.

→ This procedure is continued & models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

→ AddaBoost was the first really successful boosting algorithm developed for the purpose of binary classification.

→ AddBoost is short for Adaptive Boosting & is a very popular Boosting technique that combines multiple "weak classifiers" into a single "strong classifier."

Algorithms

Step 1: Initialise the dataset & assign equal weight to each of the data point.

Step 2: Provide this as input to the model & identify the wrongly classified data points.

Step 3: Increase the weight of the wrongly classified data points.

Step 4: if (got required results)
go to step 5
else
go to step 2

Step 5: End

Conclusion: Hence we completed the experiement with desired symbol.