Practical No 1

Aim :- Implement Breadth first search algorithm for Romanian map problem or any other map

Theory :- Traversal means visiting all the nodes of a graph. Breadth first Search (BFS) is a recursive algorithm for searching all the vertices of a graph or tree data structure.

BFS algorithm :-

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

a.) Visited
b) Not-visited

The algorithm works as follow

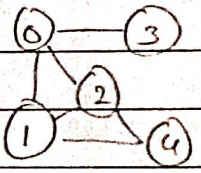
1) Start by putting any one of the graphs vertices at the back of a queue.

2) Take the front item of the queue and put it to the visited list

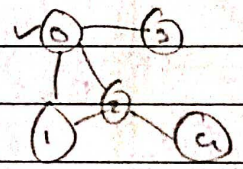
3. Create a list of that adjacent node. Add the ones which aren't in the visited list to the back of the queue.

4. Keep repeating steps 2 & 3 until the queue is empty.

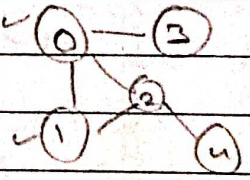
→ Initial queue : [ ]
visited : [ ]



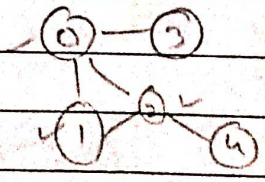
→ At 0 queue : [1, 2, 3]
visited [0]



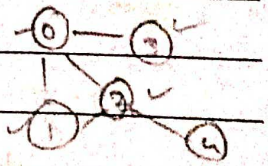
→ At 1 queue : [2, 3]
visited : [0, 1]



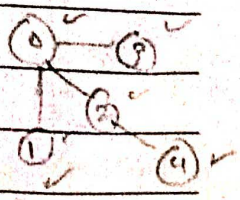
→ At 2 queue : [3, 4]
visited : [0, 1, 2]



→ At 3 queue : [4]
visited : [0, 1, 2, 3]



→ At 4 queue : [ ]
visited : [0, 1, 2, 3, 4]



Practical No. 2

Aim :- Implement iterative deep depth first search for Romanian map problem or any other map

Theory :-

Search algorithm such as breadth first or depth first search are not ideal as they have high space requirement or time requirements.

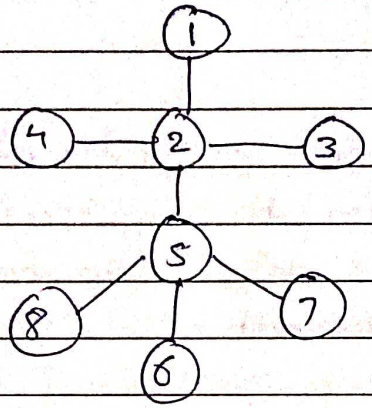
One easy to combine the space efficiency of depth first search with the optimality of breadth first searches is to use iterative deepening.

The idea is to recompute the element of the frontier rather than storing them. Each recomputation can be depth first search which thus uses less space.

Consider making a breadth. This is carried out by having a depth first searching, which ~~be~~ searches only to a limited depth. It can be done a depth first search 1 by building paths of length 1 in a depth first manner then it can build paths to depth 2, then

and so on. It can throw away all of the previous computation each time and start again. Eventually it will find a solution if one exists and as it is enumerating paths in order.

Example



Level 1
level 2
Level 3
level 4

Start : 1 Goal : 6

At level 2
visited : [1]

At level 2
visited : [1, 2]
Queue : [4, 3, 5]

At level 3
visited : [1, 2, 5]
Queue : [8, 7, 6]

Final path :- 1 → 2 → 5 → 6

CHIKITSAK SAMUHA'S
PATKAR - VARDE COLLEGE

PAGE No. :
DATE :

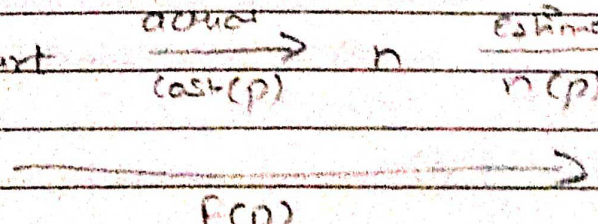
Practical No 3

Aim :- Implement A* Search algorithm for Romanian map problem or in any other map.

Theory :-

A* Search is a combination of lowest cost first and best first searches that consider both path cost and heuristic information in its selection of which path to expand. For each path on the frontier, A* uses an estimate of the total path cost from a start node to a goal node constrained to start along that path. It uses cost the cost(p), the cost of the path found as well as the heuristic function h(p) the estimated path cost from the end of p to the goal.

For any path p on the frontier, define f(p) = cost(p) + h(p). This is an estimate of the total cost to follow path p then go to the goal node.



If h(n) is an underestimate of the path cost from node n to a goal node then f(p) is an underestimate of a path cost of going from a start node to a goal node via p.

A* is implemented by treating the frontier as a priority queue ordered by f(p).

Practical No 4

Aim : Implement recursive best-first search algorithm for Romanian map problem or any other map.

Theory :

Best-First search :-

Best First search is a combination of heuristic search where the heuristic function always selects a path on the frontier with the lowest heuristic value. It usually does not search well, it can follow paths that look promising because they are close to the goal but the costs of the path may keep increasing.

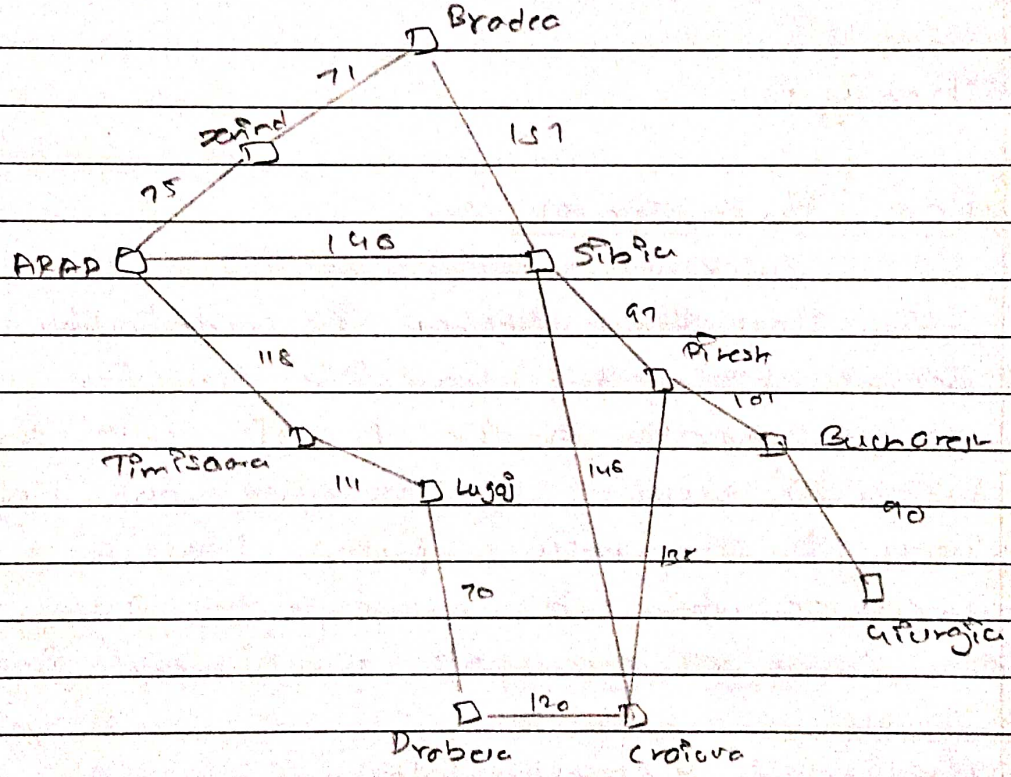
Romanian map problem

The problem arises from the complex map and topology of romania, where it is not possible to make pro active decision easily for finding shortest paths between cities Commonly Arad to Bucharest.

CHIKITSAK SAMUHA'S
PATKAR - VARDE COLLEGE

PAGE No. :
DATE :

We can use many search / traversal algorithm to solve such complex map problem, which is usually implemented by storing the map as a graph data type and traversal by the algorithm depending on its type



map of Romania.

PAGE No. : 9
DATE :

Practical No. 5

Aim :- Implement decision tree learning algorithm for the restaurant waiting problem or any other problem.

Theory :-

Decision Trees :-

A decision tree or a classification tree is a tree in which each internal root node is labelled with an input feature. Decision tree learning is one of the most successful techniques for supervised classification learning.

Restaurant waiting problem :-

This problem considers the fact that most of time people have to wait for a certain amount of time in some restaurant before they can be admitted and seated.

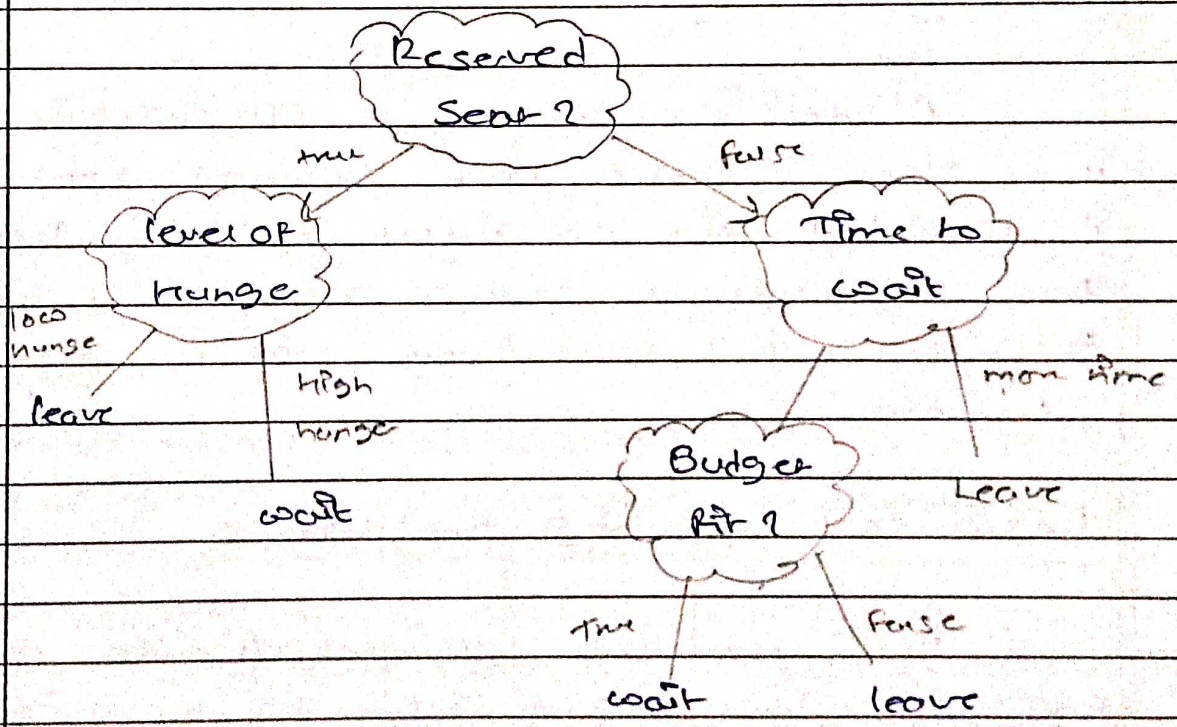
There are many factors the customer might think before waiting or leaving such as

CHIKITSAK SAMUHA'S
PATKAR - VARDE COLLEGE

PAGE No. : 10
DATE :

1. Time to wait
2. Weather condition
3. Type of food available
4. Reservation status.
5. price Band of items
6. level of Hunger

We can make a decision tree for such a problem and try to analyze.



PAGE No. : 11
DATE :

Practical No. 6.

Aim :- Implement feed forward back propagation neural network learning algorithm for the restaurant waiting problem

Theory :-

Feed Forward neural network :

Feed forward network can be seen as cascaded squashed linear function. The input feed into a layer of hidden units, which can feed into layers of more hidden units, which eventually feed into the output layer. Each of the hidden units is a squashed linear function of its inputs.

Back propagation learning :-

Back propagation learning is gradient descent search through the parameter space to minimize the sum of squares error.

The back propagation algorithm is similar to the linear perceptron. but it takes into account multiple layers and the

activation function for each example. It involves first the value of the hidden units then the value of the output units. It then passes the error back through the network computing the error on the output nodes and the error on the hidden nodes. It then updates all the weights based on the definition of the error

Gradient Descent Searches :

Gradient Descent Search involves repeated evaluation of the function to be minized In this case the error and its derivative An evaluation of the error involves looping through all of the example

Practical No 7

Aim :- Implement Adaboost ensemble learning algorithm for the restaurant waiting problem or any other problem

Theory :-

AdaBoost Classifier :

AdaBoost, like random forest classifier is another ensemble classifier. AdaBoost classifier combines weak algorithm may classify the object poorly, but if we combine multiple classifiers with selection of training set at every interation iteration and assigning right amount of weight in final voting, we can have good accuracy score for overall classifier.

$$h(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h(x)\right)$$

$h_t(x)$ is the output of weak classifier

α_t is weight assigned to classifier.

$$\alpha_t = 0.5 \times \ln\left(\frac{1-E}{E}\right)$$

weight are updated by :

$$D_{t+1}(i) = \frac{D_t(i) \cdot \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

D_t is the weight of previous level
Z_t is the sum of all weights
y_i is the y part of training Example.

Sklearn.ensemble.AdaBoostClassifier:

We use Sklearn package to use adaBoost classifier and apply it to the dataset managed by pandas.

PAGE No. : 15
DATE : 13

Practical No 8.

Aim :- Implement Naive Bayes learning algorithm for the restaurant waiting problem.

Theory :-

Bayesian classifie :

A Bayesian classifie is based on the idea that the role of a natural class is to predict the values of features for member of that class.

Bayesian classifie is that if an agent knows the class, it can predict the values of the othe features. If it does not know the class given some of the feature values. It then make the learning agent build a probabilities model of the features and uses that model to predict the classification of an example

The simplest case in the naive Bayesian classifie which makes the independence assumption that the input given the classification

The independence of the naive bayesian classifie is embodied in a particular belief network where the feature are the nodes, the target variable has no parents and the classification is the only parent of each input feature. This belief network requires the probability distribution P(y) for the target feature y and P(x_i/y) for each input feature x_i.

Bayes's Rule:

when P(e|k) ≠ 0.

$$P(h/e_{new}) = \frac{P(e(h_{new}) \times (P(h|k)}{P(e|k)}$$

If P(e|k) = 0

$$P(h|e) = \frac{P(e|h) \times P(h)}{P(e)}$$