EXPERIMENT - 1

AIM: To implement a banana camel problem

ALGORITHM:

1. The most effective way to transfer bananas is to divide the path into smaller sub parts.

2. Suppose x is a breakpoint in the path, the optimal choice is to transfer all the bananas from initial point to x & then from x to final point.

3. The total number of trips the camel which can carry c bananas at a time has to make to transfer x bananas is calculated by formula $2.x/(c-1)$

Experiment - 2

Graph coloring

Aim: To implement vertex coloring

Algorithm:

1. Initiate all the nodes
2. Set the node for the first coloring, the priority is the node with largest degree.
3. Choose the color candidate with the selection color function with no adjacent node having the same color.
4. Check the eligibility of the color, if it's able to save to the solution set.
5. Repeat step 2 if solution is not complete

EXPERIMENT -3

IMPLEMENTATION OF CONSTRAIN SATISFACTION PROBLEM

AIM:

To solve the cryptarithmetic problem by implementing constraint satisfaction problem.

ALGORITHM:

1. Begin
2. if n letters are assigned, then
   for all digits i from 0 to 9, do
   if digit i is not used, then
   nodelist[n].value = i
   if isValid(nodelist, count, word1, word2, word3) = true
   for all items j in the nodelist, do
   show the letters & corresponding values.
   done
   return true
   done
   return false
3. For all digits i from 0 to 9, do
   if digit i is not used, then
   nodelist[n].value = i
   mark as i is used
   if permutations(nodelist, count, n+1, word1, word2, word3)
   return true
   otherwise mark i as not used.
   done
   return false
4. END

ALGORITHM : (BFS)

Step 1 :

As provided in the problem statement, at any given state we can do either of the following options.

1. Fill a Jug
2. Empty a Jug
3. Transfer water from one Jug to another until either of them get filled or empty.

Step 2 :

We start at an initial state in the queue where both jug ax empty.

Step 3 :

If we explore all the possible intermediate states, we can empty the other jug & return the current state's entire path.

Step-4: If the target is obtained, empty the other Jug

DFS

DDFS, Depth first search is implemented by filling a water Jug full.

2) Then empty the Jug slowly one by one

3) All the combinations till you get the target.

4) Queue the next, possibility till the target is reached.

5) If the target is not reached, return false continue the combination.

Experiment - 5

Best First Search and A* algorithm

Aim:- To implement best first search algorithm & A* algorithm

Algorithm:

Best first search algorithm.

Step 1: Place the starting node into the OPEN list.

Step 2: If the OPEN list is empty, Stop & return failure

Step 3: Remove the node n, from the OPEN list which has the lowest value of h(n), & places it in the CLOSED list.

Step 4: Expand the node n, & generate the successors of node n.

Step 5: Check each successor of node n, & find whether any node is a goal node or not. If any successor node is goal node, then return success & terminate the search, else proceed to Step 6.

Step 6: For each successor node, algorithm checks for evaluation function f(n), & then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.

Step 7: Return to Step 2.

Algorithm of A* search

Step 1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not. if the list is empty then return failure & stops

Step 3: Select the node from the open list which has the smallest value of evaluation function (g+h), if node n is goal node then return success & stop, otherwise.

Step-4: Expand nodes & generate all of its successors & put it in the closed list. For each successor 'n', check whether 'n' is already in the OPEN or CLOSED list, if not then compute evaluation function for 'n' & place into Open list

Step-5: Else if node 'n' is already in OPEN & CLOSED, then it should be attached to the back pointer which reflects the lowest g(n) value.

Step 6: Return to Step 2.

Experiment - 6

Implementation of uncertain methods for an application

Aim:- To implement Bayesian Belief Networks to model the problem of Monty Hall.

Algorithm:

1. Randomly assign the prize to one of three doors
2. Ask the player to choose one of the three doors
3. If the player chose the door with the prize, return "win".
4. Otherwise, have the host open one of the other two doors that does not have the prize behind it.
5. Give the player the option to switch their choice to the other unopened door or stick with their original choice.
6. If the player chooses to switch, return to step 3.
7. If the player chooses to stick with the original choice, return 'lose'.

Implementation of Unification & Resolution.

Aim:- To implement unification & resolution to solve real world problems.

Algorithm:

Unify(Ψ_1, Ψ_2)

Step 1 : If Ψ_1 or Ψ_2 is a variable or constant, then:

a) If Ψ_1 or Ψ_2 are identical, then return NIL.

b) Else if Ψ_1 is a variable

a. then if Ψ_1 occurs in Ψ_2, then return FAILURE

b. Else return $\{(\Psi_2/\Psi_1)\}$.

c) Else if Ψ_2 is a variable

a. If Ψ_2 occurs in Ψ_1 then return FAILURE.

b. Else return $\{(\Psi_1/\Psi_2)\}$

d) Else return FAILURE

Step 2: If the initial Predicate symbol in Ψ_1 & Ψ_2 are not same, then return FAILURE

Step 3: If Ψ_1 and Ψ_2 have a different number of arguments, then return FAILURE

Step 4: Set Substitution set(SUBST) to NIL.

Step 5: For i=1 to the number of elements in Ψ_1.

a) Call Unify function with the ith element of Ψ_1 & ith element of Ψ_2, and put the result into S.

b) If S= failure then return Failure

c) If S ≠ NIL then do,

a. Apply S to the remainder of both L1 & L2

b. SUBST = APPEND (S, SUBST).

Step 6 : Return SUBST,

Resolution

1. Conversion of facts into first-order logic
2. Convert FOL statement into CNF
3. Negate the statement which needs to prove (proof by contradiction)
4. Draw resolution graph.

Experiment -8

Implement a Learning Algorithm.

Aim :- To implement a learning algorithm for an application.

1. Logistic Regression
2. SVM
3. Naive Bayes

Algo:

1. Load the dataset from a CSV file
2. Split the dataset features(x) & target variable(y)
3. Split the data into training & testing sets using train_test_split() function scikit-learn
4. Create a Gaussian Naive Bayes classifier using GaussianNB() function from scikit-learn
5. Fit the classifier to the training data using fit() method
6. Make predictions on the test data using predict() method.
7. Evaluate the accuracy of the model using accuracy_score function from scikit-learn
8. Adjust the hyperparameters (eg. smoothing parameter) as necessary to improve model performance.

Experiment - 9

NLP PROGRAM

Aim:

To implement an NLP program.

Algorithm:

1. Import libraries required for stemming
2. Get the input from the user
3. Tokenization has to be done before stemming to break text into tokens for easy computation.
4. Stemming has to be done on all other words. Finally stem words are joined to make a sentence.
5. Lemmatization has to be done.
6. Get the output

Experiment 10

Deep Learning

Aim:. To apply deep learning to solve a problem

Algorithm:

1. Define the problem
2. Gather & preprocessing data
3. Define the model architecture
4. Train the model
5. Evaluate the model.
6. Fine-tune the model
7. Test the model
8. Deploy the model
9. Monitor & update the model.

Result:- Hence deep learning code was implemented.