DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

T.E/SEM V/CBCGS/AIML
Academic Year: 2022-23

NAME	**CHIKANKAR PRATHAMESH SHIVAJI**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML11**
SUBJECT	**ARTIFICIAL INTELLIGENCE LAB**
COURSE CODE	**CSL502**
PRACTICAL NO.	
DOP	
DOS	

The AI Chip Race

Reference link:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9779606

Publication: IEEE

DOI No.: 10.1109/MIS.2022.3165668

Abstract:

The strong demand for computing power for artificial intelligence (AI) and machine learning is accelerating the race to develop cheaper and faster AI chips. The AI chip market was valued 10.6billionin2021andthetotalrevenueisexpectedtoreach79.8 billion by 2027.aa.[Online]. Available: https://www.maximizemarketresearch.com/market-report/global-artificial-intelligence-chipset-market/66849/. To be part of the market, tech giants from different countries have been successively joining the race, while AI chip startups attracting billions of dollars are taking off like a rocket.

AI method used:

AI chips are hardware accelerators specifically designed to accelerate AI and machine learning-based applications. They generally include graphics processing units (GPUs), field-programmable gate arrays (FPGAs), and certain types of application-specific integrated circuits (ASICs) specialized for AI calculations.[2] Deep neural networks (DNNs) are the cutting-edge, computationally intensive AI systems that these accelerators are tailored to. As a popular machine learning approach, DNNs consist of two key stages—training and inference—DNN models are fed with large-scale data to extract useful patterns during training, and they are then used to make predictions for unseen data during inference.

General-purpose chips, such as central processing units (CPUs), have strong sequential operation capability, but they cannot provide sufficient performance for techniques like DNNs that require intensive parallel computation and high-bandwidth memory. Specialized AI chips can be up to thousands of times faster than CPUs for training and inference of DNNs.[1],[2]

Nvidia recently released H100 GPUs, offering about an order-of-magnitude leap compared to its precedent A100 GPUs.b GPUs have been a dominant hardware tool to accelerate AI systems, especially excelling at training computationally costly DNN models. It enjoys the strong support of the parallel computing platform to compute unified device architecture, and it is a type of widely commercialized AI chips.

Result:

For years, the semiconductor world seemed to have settled into a quiet balance: Intel vanquished virtually all of the RISC processors in the server world, save IBM's POWER line. Elsewhere AMD had self-destructed, making it pretty much an x86 world. Then Nvidia mowed down all of it many competitors in the 1990s. Suddenly only ATI, now a part of AMD, remained. It boasted just half of Nvidia's prior market share.

On the newer mobile front, it looked to be a similar near-monopolistic story: ARM ruled the world. Intel tried mightily with the Atom processor, but the company met repeated rejection before finally giving up in 2015.

Then just like that, everything changed. AMD resurfaced as a viable x86 competitor; the advent of field gate programmable array (FPGA) processors for specialized tasks like Big Data created a new niche. But really, the colossal shift in the chip world came with the advent of artificial intelligence (AI) and machine learning (ML). With these emerging technologies, a flood of new processors has arrived—and they are coming from unlikely sources.

Intel got into the market with its purchase of startup Nervana Systems in 2016. It bought a second company, Movidius, for image processing AI.

Microsoft is preparing an AI chip for its HoloLens VR/AR headset, and there's potential for use in other devices.

Google has a special AI chip for neural networks call the Tensor Processing Unit, or TPU, which is available for AI apps on the Google Cloud Platform.

Amazon is reportedly working on an AI chip for its Alexa home assistant.

Apple is working on an AI processor called the Neural Engine that will power Siri and FaceID.

ARM Holdings recently introduced two new processors, the ARM Machine Learning (ML) Processor and ARM Object Detection (OD) Processor. Both specialize in image recognition.

IBM is developing specific AI processor, and the company also licensed NVLink from Nvidia for high-speed data throughput specific to AI and ML.

Even non-traditional tech companies like Tesla want in on this area, with CEO Elon Musk acknowledging last year that former AMD and Apple chip engineer Jim Keller would be building hardware for the car company.

Future scope:

While deep learning and neural networks are advancing the state of AI technology rapidly, there are many researchers who believe that there is still a need for fundamentally new and different approaches if the most fantastic goals of AI are to be met. Most AI chips are being designed to implement ever-improving versions of the same ideas published by LeCun and Hinton and others more than a decade past, but there is no reason to expect even exponential progress along this path will lead to AI that can think like a human being. AI as we know it today cannot apply the deep learning about one task that it acquires with such great effort to a new, different task. Also, neural networks do not have a good way of incorporating prior knowledge, or rules like "up vs down" or "children have parents." Lastly, AI based on neural networks requires huge numbers of examples in order to learn, while a human can learn not to touch a hot stove given only one highly memorable experience. It is not clear how to apply current AI techniques to problems that don't come with huge labeled datasets.

Conclusion:

Given the tremendous market value and the strategic significance to each country, the AI chip race among tech giants, startups, and countries is expected to further accelerate in the future. The market share among GPUs, FPGAs, and ASICs is likely to change, as well.

PEAS ANALYSIS OR RENEWABLE ENERGY SYSTEM

Theory on PEAS

We know that there are different types of agents in AI. PEAS System is used to categorize similar agents together. The PEAS system delivers the performance measure with respect to the environment, actuators, and sensors of the respective agent. Most of the highest performing agents are Rational Agents.

Rational Agent: The rational agent considers all possibilities and chooses to perform a highly efficient action. For example, it chooses the shortest path with low cost for high efficiency. PEAS stands for a Performance measure, Environment, Actuator, Sensor.

PEAS Description

>Performance measure:

1.Converting energy in different form

2.Solar energy, wind energy into electrical error

>Environment:

Wind and sunlight

>Actuators:

Screen display, data set, machine learning libraries for moment of detection panel & fans.

>Sensors:

Solar panel, wind fans, wind mills.

->State, Space, Description

>State:

Any state from feature extraction to model training is a state,

Or in this case all state are the steps for converting solar or wind energy into electrical energy.

-initial state: the energy available in the environment i.e solar or wind

-goal state : converted electrical energy.

>Action:

State required conversion of solar wind energy into electrical energy

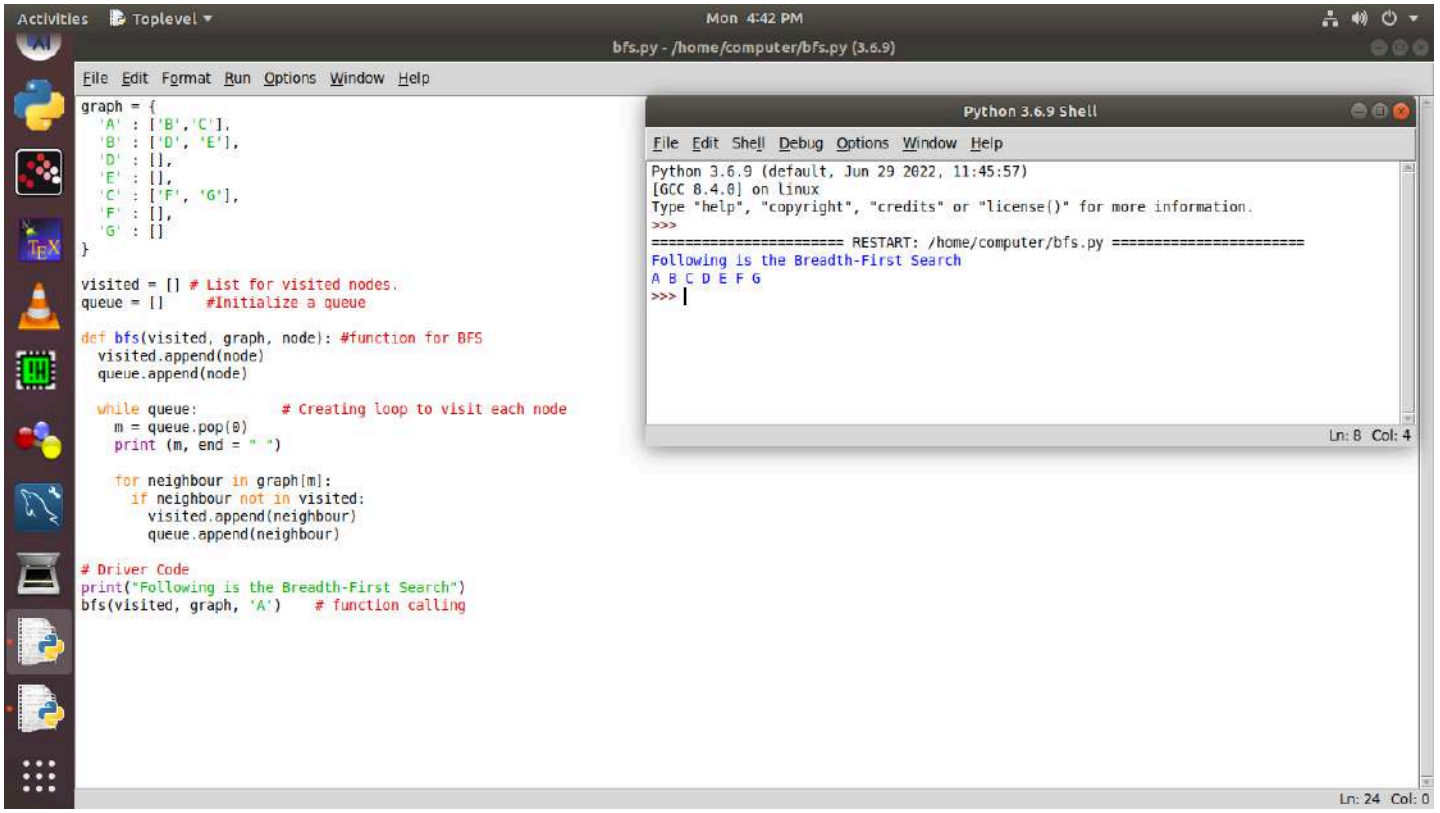
>Cost:

Accuracy must increase in every state.

Code -

```
graph = {
  'A' : ['B','C'],
  'B' : ['D', 'E'],
  'D' : [],
  'E' : [],
  'C' : ['F', 'G'],
  'F' : [],
  'G' : []
}

visited = [] # List for visited nodes.
queue = []     #Initialize a queue

def bfs(visited, graph, node): #function for BFS
  visited.append(node)
  queue.append(node)

  while queue:          # Creating loop to visit each node
    m = queue.pop(0)
    print (m, end = " ")

    for neighbour in graph[m]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
print("Following is the Breadth-First Search")
bfs(visited, graph, 'A')    # function calling
```

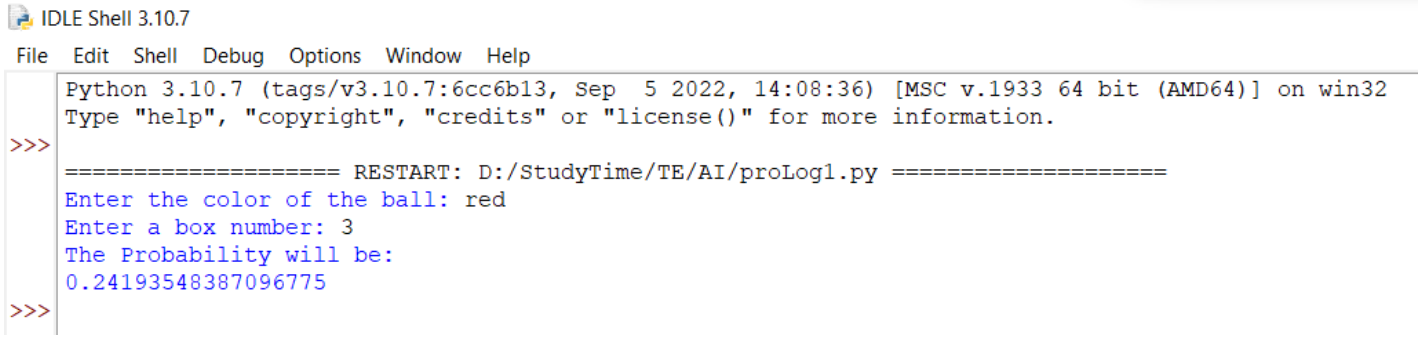
Output (ScreenShot) -



Code -

```
def probabilityOfRed(a):
    return a[0]/a[-1]

def probabilityOfBlue(b):
    return b[1]/b[-1]

def numerical(pa,pb,pc,pall,boxinput,colorinput):

    if boxinput == 1 and colorinput == 'red':
        return (pall * probabilityOfRed(pa))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 1 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pa))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

    if boxinput == 2 and colorinput == 'red':
        return (pall * probabilityOfRed(pb))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 2 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pb))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

    if boxinput == 3 and colorinput == 'red':
        return (pall * probabilityOfRed(pc))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 3 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pc))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

box1 = [3,2,5]
box2 = [4,5,9]
box3 = [2,4,6]
pofall = 1/3
colorinputs = input("Enter the color of the ball: ")
boxinputs = int(input("Enter a box number: "))
print("The Probability will be: ")
print(numerical(box1,box2,box3,pofall,boxinputs,colorinputs))
```

Output -


IDLE Shell 3.10.7
File Edit Shell Debug Options Window Help
Python 3.10.7 (tags/v3.10.7:6cc6b13, Sep 5 2022, 14:08:36) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:/StudyTime/TE/AI/proLog1.py ==========
Enter the color of the ball: red
Enter a box number: 3
The Probability will be:
0.24193548387096775
>>>

CODE:

```
def probabilityOfRed(a):
    return a[0]/a[-1]

def probabilityOfBlue(b):
    return b[1]/b[-1]

def numerical(pa,pb,pc,pall,boxinput,colorinput):

    if boxinput == 1 and colorinput == 'red':
        return (pall * probabilityOfRed(pa))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 1 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pa))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

    if boxinput == 2 and colorinput == 'red':
        return (pall * probabilityOfRed(pb))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 2 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pb))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

    if boxinput == 3 and colorinput == 'red':
        return (pall * probabilityOfRed(pc))/((pall * probabilityOfRed(pa))+(pall *
probabilityOfRed(pb))+(pall * probabilityOfRed(pc)))

    if boxinput == 3 and colorinput == 'blue':
        return (pall * probabilityOfBlue(pc))/((pall * probabilityOfBlue(pa))+(pall *
probabilityOfBlue(pb))+(pall * probabilityOfBlue(pc)))

box1 = [3,2,5]
box2 = [4,5,9]
box3 = [2,4,6]
pofall = 1/3
colorinputs = input("Enter the color of the ball: ")
boxinputs = int(input("Enter a box number: "))
print("The Probability will be: ")
print(numerical(box1,box2,box3,pofall,boxinputs,colorinputs))
```

OUTPUT:

```
PS E:\Program\hanzala> python .\bayes.py
Enter the color of the ball: red
Enter a box number: 2
The Probability will be:
0.32258064516129037
```

Program- database.pl

```
%teaches(X, Y): person X teaches in course Y
teaches(sudhir, course001).
teaches(tapas, course002).
teaches(pranab, course003).
teaches(joydeb, course004).


%student(X, Y): student X studies in course Y
studies(suparna, course001).
studies(santanu, course001).
studies(sudip, course002).
studies(sudip, course003).
studies(srobona, course003).
studies(subir, course003).
studies(swarup, course003).
```

Output -

```
true.

?- ['/home/computer/Documents/CSE-AIML/TE/AIML03_PRATHAMESH/AIL/prolog/database.pl'].
[1]+  Stopped                 swipl
(base) computer@computer:~$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
```

```
?- ['/home/computer/Documents/CSE-AIML/TE/AIML11_PRATHAMESH/AIL/prolog/database.pl'].
true.

?- teaches(sudhir,X).
X = course001.

?- teaches(X,Y).
X = sudhir,
Y = course001 .

?- teaches(X, course001).
X = sudhir.
```

Program- monkey.pl

```
%monkey wants to eat banana
on(floor,monkey).
on(floor,box).
in(room,monkey).
in(room,box).
in(room,banana).
at(ceiling,banana).
strong(monkey).
grasp(monkey).
climb(monkey,box).
push(monkey,box):-
   strong(monkey).
under(banana,box):-
   push(monkey,box).
canreach(banana,monkey):-
   at(floor,banana);
   at(ceiling,banana),
   under(banana,box),
   climb(monkey,box).
canget(banana,monkey):-
   canreach(banana,monkey),grasp(monkey).
```

Output -

```
?- ['/home/computer/Documents/CSE-AIML/TE/AIML11_PRATHAMESH/AIL/prolog/monkey.pl'].
true.

?- on(floor,box).
true.

?- in(room,monkey).
true .

?- trace.
true.

[trace]  ?- canget(banana,monkey).
   Call: (8) canget(banana, monkey) ? creep
   Call: (9) canreach(banana, monkey) ? creep
   Call: (10) at(floor, banana) ? creep
   Fail: (10) at(floor, banana) ? creep
   Redo: (9) canreach(banana, monkey) ? creep
   Call: (10) at(ceiling, banana) ? creep
   Exit: (10) at(ceiling, banana) ? creep
   Call: (10) under(banana, box) ? creep
   Call: (11) push(monkey, box) ? creep
   Call: (12) strong(monkey) ? creep
   Exit: (12) strong(monkey) ? creep
   Exit: (11) push(monkey, box) ? creep
   Exit: (10) under(banana, box) ? creep
   Call: (10) climb(monkey, box) ? creep
   Exit: (10) climb(monkey, box) ? creep
   Exit: (9) canreach(banana, monkey) ? creep
   Call: (9) grasp(monkey) ? creep
   Exit: (9) grasp(monkey) ? creep
   Exit: (8) canget(banana, monkey) ? creep
true.
```

CODE (HILL CLIMB) -

```
import random

def randomSolution(tsp):
    cities = list(range(len(tsp)))
    solution = []

    for i in range(len(tsp)):
        randomCity = cities[random.randint(0, len(cities) - 1)]
        solution.append(randomCity)
        cities.remove(randomCity)

    return solution

def routeLength(tsp, solution):
    routeLength = 0
    for i in range(len(solution)):
        routeLength += tsp[solution[i - 1]][solution[i]]
    return routeLength

def getNeighbours(solution):
    neighbours = []
    for i in range(len(solution)):
        for j in range(i + 1, len(solution)):
            neighbour = solution.copy()
            neighbour[i] = solution[j]
            neighbour[j] = solution[i]
            neighbours.append(neighbour)
    return neighbours

def getBestNeighbour(tsp, neighbours):
    bestRouteLength = routeLength(tsp, neighbours[0])
    bestNeighbour = neighbours[0]
    for neighbour in neighbours:
        currentRouteLength = routeLength(tsp, neighbour)
        if currentRouteLength < bestRouteLength:
            bestRouteLength = currentRouteLength
            bestNeighbour = neighbour
    return bestNeighbour, bestRouteLength
```

```
def hillClimbing(tsp):
    currentSolution = randomSolution(tsp)
    currentRouteLength = routeLength(tsp, currentSolution)
    neighbours = getNeighbours(currentSolution)
    bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

    while bestNeighbourRouteLength < currentRouteLength:
        currentSolution = bestNeighbour
        currentRouteLength = bestNeighbourRouteLength
        neighbours = getNeighbours(currentSolution)
        bestNeighbour, bestNeighbourRouteLength = getBestNeighbour(tsp, neighbours)

    return currentSolution, currentRouteLength

def main():
    tsp = [
        [0, 400, 500, 300],
        [400, 0, 300, 500],
        [500, 300, 0, 400],
        [300, 500, 400, 0]
    ]

    print(hillClimbing(tsp))

if __name__ == "__main__":
    main()
```

OUTPUT -

```
python -u "d:\StudyTime\TE\AI\hillClimb1.py"
([2, 3, 0, 1], 1400)
```

CODE:

```
print("""List of all events occurring in this network:
Burglary (B)
Earthquake(E)
Alarm(A)
John Calls(J)
Merry calls(M)

The Conditional probability of Alarm A depends on Burglar and earthquake:
Burglary    Earthquake    P(A= True)    P(A= False)
True        True          0.94          0.06
True        False         0.95          0.04
False       True          0.31          0.69
False       False         0.001         0.999

The Conditional probability of John that he will call depends on the probability of Alarm.
Alaram      P(J= True)      P(J= False)
True        0.91            0.09
False       0.05            0.95

The Conditional probability of Merry that she calls is depending on its Parent Node "Alarm."
Alarm       P(M= True)      P(M= False)
True        0.75            0.25
False       0.02            0.98

probability of burglary = 0.001
probability of earthquake = 0.002

""")
print("-----------")
print(" ")
print("ii) what is the probablity the alarm has sounded but neither burglary nor a earthquake has
occured and both john and merry called")
print(" ")
print("Probability that the alarm has sounded but neither burglary nor a earthquake has occured
and both john and merry called")
def totalprobability(pJA,pM,pAnotBE,pnotE,pnotB):
    return(pJA*pM*pAnotBE*pnotE*pnotB)
print("=
p(John|Alaram)*P(M|Alaram)*(p(Alaram|~Burglary,~Earthquake)*P(~Earthquake)*P(~Burglary)")
print("=",totalprobability(0.90,0.70,0.001,0.999,0.998))
print(" ")
print("-----------")
print(" ")
```

```
print("ii) what is the probablity that John calls")
print(" ")
print("Probability that jhon calls")
print("= p(John|Alaram)*P(Alaram)+P(John|~A).P(~Alaram)")
print("""=
p(John|Alarm)*{(P(Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(Alaram|~Burglary,
Earthquake)*P(~Burglary,Earthquake))+
(P(Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(Alaram|~Burglary,~Earthquake)
*P(~Burglary,~Earthquake))}+p(John|~Alaram)*
{(P(~Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(~Alaram|~Burglary,Earthquake)*
P(~Burglary,Earthquake))+
(P(~Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(~Alaram|~Burglary,~Earthquak
e)*P(~Burglary,~Earthquake))}""")

bayesian_probability = ((0.90*0.00252)+(0.05*0.9974))
print("=",bayesian_probability)
```

OUTPUT:

```
D: > StudyTime > TE > AI > bbn.py > totalprobability
    print("""List of all events occurring in this network:
    Burglary (B)
    Earthquake(E)
    Alarm(A)
    John Calls(J)
    Merry calls(M)

    The Conditional probability of Alarm A depends on Burglar and earthquake:
    Burglary    Earthquake    P(A= True)    P(A= False)
    True        True          0.94          0.06
    True        False         0.95          0.04
    False       True          0.31          0.69
    False       False         0.001         0.999

    The Conditional probability of John that he will call depends on the probability of Alarm.
    Alaram      P(J= True)      P(J= False)
    True        0.91            0.09
    False       0.05            0.95

    The Conditional probability of Merry that she calls is depending on its Parent Node "Alarm."
    Alarm       P(M= True)      P(M= False)
    True        0.75            0.25
    False       0.02            0.98

    probability of burglary = 0.001
    probability of earthquake = 0.002

    """)
```

```python
print("-----------")
print(" ")
print("ii) what is the probablity the alarm has sounded but neither burglary nor a earthquake has occured and both john and merry called")
print(" ")
print("Probability that the alarm has sounded but neither burglary nor a earthquake has occured and both john and merry called")
def totalprobability(pJA,pM,pAnotBE,pnotE,pnotB):
    return(pJA*pM*pAnotBE*pnotE*pnotB)
print("= p(John|Alaram)*P(M|Alaram)*(p(Alaram|~Burglary,~Earthquake)*P(~Earthquake)*P(~Burglary)")
print("=",totalprobability(0.90,0.70,0.001,0.999,0.998))
print(" ")
print("-----------")
print(" ")
print("ii) what is the probablity that John calls")
print(" ")
print("Probability that jhon calls")
print("= p(John|Alaram)*P(Alaram)+P(John|~A).P(~Alaram)")
print("""= p(John|Alarm)*{(P(Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(Alaram|~Burglary,Earthquake)*P(~Burglary,Earthquake))+
(P(Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(Alaram|~Burglary,~Earthquake)*P(~Burglary,~Earthquake))}+p(John|~Alaram)*
{(P(~Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(~Alaram|~Burglary,Earthquake)*P(~Burglary,Earthquake))+
(P(~Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(~Alaram|~Burglary,~Earthquake)*P(~Burglary,~Earthquake))}""")

bayesian_probability = ((0.90*0.00252)+(0.05*0.9974))
print("=",bayesian_probability)
```

```
PS D:\StudyTime\practiseStuff> python -u "d:\StudyTime\TE\AI\bbn.py"
List of all events occurring in this network:
Burglary (B)
Earthquake(E)
Alarm(A)
John Calls(J)
Merry calls(M)

The Conditional probability of Alarm A depends on Burglar and earthquake:
Burglary    Earthquake    P(A= True)    P(A= False)
True        True          0.94          0.06
True        False         0.95          0.04
False       True          0.31          0.69
False       False         0.001         0.999

The Conditional probability of John that he will call depends on the probability of Alarm.
Alaram      P(J= True)      P(J= False)
True        0.91            0.09
False       0.05            0.95

The Conditional probability of Merry that she calls is depending on its Parent Node "Alarm."
Alarm       P(M= True)      P(M= False)
True        0.75            0.25
False       0.02            0.98

probability of burglary = 0.001
probability of earthquake = 0.002


-----------

ii) what is the probablity the alarm has sounded but neither burglary nor a earthquake has occured and both john and merry called

Probability that the alarm has sounded but neither burglary nor a earthquake has occured and both john and merry called
= p(John|Alaram)*P(M|Alaram)*(p(Alaram|~Burglary,~Earthquake)*P(~Earthquake)*P(~Burglary)
= 0.0006281112599999999

-----------
```

```
ii) what is the probablity that John calls

Probability that jhon calls
= p(John|Alaram)*P(Alaram)+P(John|~A).P(~Alaram)
= p(John|Alarm)*{(P(Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(Alaram|~Burglary,Earthquake)*P(~Burglary,Earthquake))+
(P(Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(Alaram|~Burglary,~Earthquake)*P(~Burglary,~Earthquake))}+p(John|~Alaram)*
{(P(~Alaram|Burglary,Earthquake)*P(Burglary,Earthquake))+(P(~Alaram|~Burglary,Earthquake)*P(~Burglary,Earthquake))+
(P(~Alaram|Burglary,~Earthquake)*P(Burglary,~Earthquake))+(P(~Alaram|~Burglary,~Earthquake)*P(~Burglary,~Earthquake))}
= 0.052138
```

CODE:

```
def ConstBoard(board):
    print("Current State Of Board : ")
    for i in range (0,9):
        if((i>0) and (i%3)==0):
            print("")
        if(board[i]==0):
            print("- ",end=" ")
        if (board[i]==1):
            print("O ",end=" ")
        if(board[i]==-1):
            print("X ",end=" ")
    print("")

def User1Turn(board):
    pos=input("Enter X's position from [1...9]: ")
    pos=int(pos)
    if(board[pos-1]!=0):
        print("Wrong Move!!!")
        exit(0)
    board[pos-1]=-1

def User2Turn(board):
    pos=input("Enter O's position from [1...9]: ")
    pos=int(pos)
    if(board[pos-1]!=0):
        print("Wrong Move!!!")
        exit(0)
    board[pos-1]=1

def minimax(board,player):
    x=analyzeboard(board)
    if(x!=0):
        return (x*player)
    pos=-1
    value=-2
    for i in range(0,9):
        if(board[i]==0):
            board[i]=player
            score=-minimax(board,(player*-1))
```

```
            if(score>value):
                value=score
                pos=i
            board[i]=0

    if(pos==-1):
        return 0
    return value

def CompTurn(board):
    pos=-1
    value=-2
    for i in range(0,9):
        if(board[i]==0):
            board[i]=1
            score=-minimax(board, -1)
            board[i]=0
            if(score>value):
                value=score
                pos=i

    board[pos]=1

def analyzeboard(board):
    cb=[[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]

    for i in range(0,8):
        if(board[cb[i][0]] != 0 and
           board[cb[i][0]] == board[cb[i][1]] and
           board[cb[i][0]] == board[cb[i][2]]):
            return board[cb[i][2]]
    return 0

def main():
    print("Game start: ")
    board=[0,0,0,0,0,0,0,0,0]
    for i in range (0,9):
        if(analyzeboard(board)!=0):
            break
        if((i)%2==0):
```

```
            ConstBoard(board)
            User1Turn(board)
        else:
            ConstBoard(board)
            User2Turn(board)

    x=analyzeboard(board)
    if(x==0):
        ConstBoard(board)
        print("Draw!!!")
    if(x==-1):
        ConstBoard(board)
        print("X Wins!!! Y Loose !!!")
    if(x==1):
        ConstBoard(board)
        print("X Loose!!! O Wins !!!!")

main()
```

OUTPUT:

```
(base) computer@computer:~$ /usr/bin/python3.9
/home/computer/Documents/tictactoe_minmax.py
Game start:
Current State Of Board :
-  -  -


-  -  -


-  -  -
Enter X's position from [1...9]: 1
Current State Of Board :
X  -  -


-  -  -


-  -  -
Enter O's position from [1...9]: 3
Current State Of Board :
```

```
X  -  O

-  -  -

-  -  -
Enter X's position from [1...9]: 2
Current State Of Board :
X  X  O

-  -  -

-  -  -
Enter O's position from [1...9]: 6
Current State Of Board :
X  X  O

-  -  O

-  -  -
Enter X's position from [1...9]: 4
Current State Of Board :
X  X  O

X  -  O

-  -  -
Enter O's position from [1...9]: 9
Current State Of Board :
X  X  O

X  -  O

-  -  O
X Loose!!! O Wins !!!!
```

Name :- Prathamesh b. Chikankar

Roll No. :- AIML11 Branch :- CSE- (AI & ML)

Year :- TE Subject :- AI (Artificial Intelligence)

Topic :- Assignment No. 01

Sign :- Prathm Date :- August'22

Q.1. What is AI, Define AI with respect to
i) Acting humanly ii) Acting Rationally

1) AI was a term coined by John McCarthy in 1956 and he defined AI as the science & engineering of making intelligent machine
2) AI (Artificial Intelligence) is relevant to any intellectual task where the machine needs to take some decision to make/choose the next action based on the current state of the system, in short act intelligently and rationally.
3) In simple words AI system works like a human being brain where a machine or s/w shows intelligent while performing the given task, such system are called as intelligent system.
4) AI is one of the new fields in science & engineering and has a wide variety of application field.
5) AI application ranges from general fields like learning, perception and prediction to specific field such as writing stories, mathematical theorems, driving a bus, etc..
6) AI is the study of how to make machine do things which the moment people do better.
7) There are 2 main goal of AI:-
i) Building Intelligent Machine.
ii) understanding the nature of intelligence.
8) Different approaches to define AI is:-
i) Acting Humanly - [Turing test approach]
ii) Thinking Humanly - [cognitive modelling approach]
iii) Thinking Rationally - [laws of thought approach]
iv) Acting Rationally - [rational agent approach]

> Acting Humanly :- [Turing test approach]
• the art of making machine that performs function that require intelligence if performing by human [Kurzweil 1990].

PTO

- The study of how to make computer do things at which at the moment people are better (Rich & Knight, 1991)

> Acting Rationally :- (Rational Agent approach)

- Computational intelligence is the ability (study of design of intelligent AI agent (Poole et al 1998)
- AI is concerned with the intelligent behaviour of artifacts (Nilson, 1998)

Q.2. Explain various applications of AI in various areas.

Artificial intelligence is revolutionizing the industries with its applications and helping solve complex problems

① AI in Robotics:

Robotics is additionally field where AI applications are commonly used. Robots powered by AI use real-time updates to sense obstacles in its path and replan its journey instantly.

② AI in agriculture:

AI is used to detect defects and nutrient deficiency in soil. This is done using computer vision, robotics & machine learning.

③ AI in gaming:

Another sector where AI applications have found prominence is the Gaming sector.

It also helps in predicting human behaviour using game designs.

④ AI in automobiles:

AI used to build self-driving vehicles. AI can be used along with the vehicle's camera, radar, cloud service, GPS and control signals to operate the vehicle.

AI can also improve emergency braking, blind spot monitoring and assist steering.

⑤ AI in Social Media:

AI used in various social media sectors such as Instagram, facebook

PTO

and Twitter.
It helps to translate the post in diff. languages automatically.

⑤ Fuzzy logic:
The system which selies on degree of truth and changes in states along with rate of inputs & outputs, whose output depends on reading of the input, its state and rate of change of this state.

④ AI in finance:
The finance industry is implementing automation, chatbot, adaptive intelligence, algorithmic trading & machine learning in financial process.

⑤ AI in education:
AI can automate grading so that the tutor can have more time to learn. AI chatbot also help to communicate as teaching assistant.

Q.3. Define P, E, A, S and make PEAS analysis for the Hospital Management System.

PEAS:

① Performance measure: It is the unit to define the success of an agent performance various with agent based on their different aspects. Highest performance agents are rational agents.

② Environment: Environment is a surrounding of agent is set in every instant it keeps changing with time if agent is set in multiple different types of Environment: i) static & dynamic
ii) observable & unobservable
iii) single agent & multiple agent
iv) discrete & continuous
v) deterministic & stochastic.

③ Actuators: Actuator is a part of an agent that observes the output of action to the environment.

④ Sensors: sensors are the perceptive part of the agent that takes in

PTO

the input of the agent.

PEAS analysis of Hospital management system:-

> performance: patient Health, admission protocol, payment.
> Environment: Hospital, Doctor, patients.
> Actuators: percepts diagnosis, resources, scan report.
> sensor: symptom, patient, response.

seflex.

Q.4. what is AI agent. Explain the architecture of simple agent.

① AI agent is a person program that can make decision or performs a service based on its environment, user input & resources.
② It carries out actions with the best outcomes after considering past & current percepts.
③ An AI system is composed of an agent & its environment.

> simple seflex agent:

In AI simple reflex agent is type of 'intelligent agent' that performs action based on solely on current situation.

The 'simple reflex agent' works on condition rule, which means it maps the current state action. Since reflex action is based on the present condition so it called as current condition action rule for the current percept it checks the current required in the condition action rules given and based on its takes appropriate action.

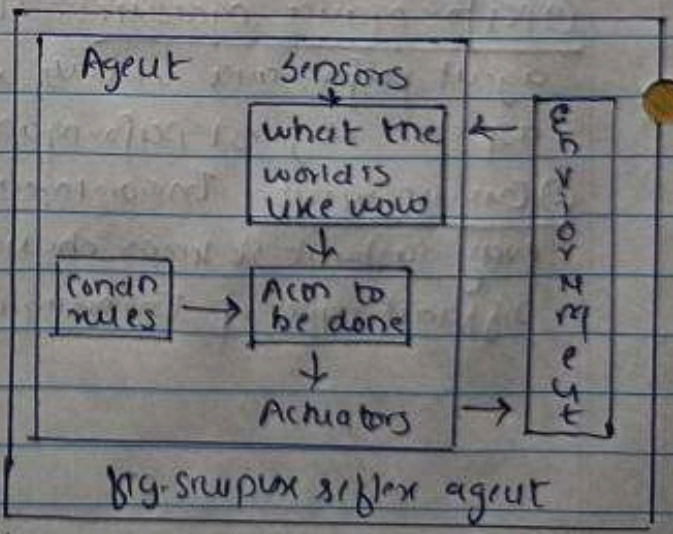


fig: simple reflex agent

Q.5. List various types of environment in AI. Explain any one of the types of environment in brief with suitable example.

Different types of environment are :-

i) static & dynamic
ii) observable and partially observable
iii) single agent & multiple agent
iv) discrete and continuous.
v) deterministic & stochastic.

> static & dynamic :

- Static : An environment that remain always unchanged by action of agent is called static environment. A static environment is the simplest one which is easy to deal with since the agent doesn't need to keep track of world during an action.

eg :- crossword puzzle.

The environment must remain constant doesn't expand or shrink.

- Dynamic : An environment is said to be dynamic if it changes by the action of agent. An dynamic environment keeps is constantly changing.

eg :- Roller coster

The environment keeps changing for every instant it set in motion.

Q.6. what is AI agent, Explain the architecture of goal based agent.

1) An AI agent is a program that makes decision or perform a service based on its environment, user i/p & experiences.
2) It carries out an action with the best outcomes after considering past and current percepts.
3) An AI system is composed of an agents & its environment.

> Goal-based agent

PTO

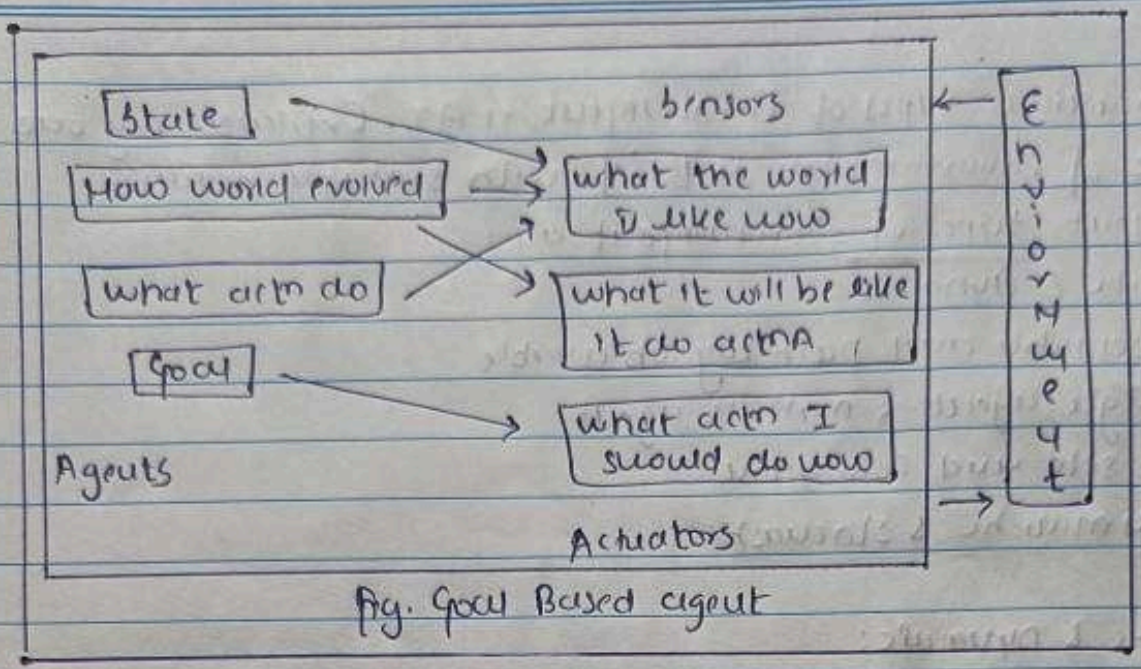


fig. Goal Based agent

- Goal Based reflex agent has a goal and how a strategy to reach the goal. All actn are taken to reach the goal.
- goal based agents expand the capability of the model-based agent by having the 'goal' information.
- these agents may be consider a long sequence of possible actn, before deciding whether goal is achieved or not.
- A goal based algorithm uses searching & planning to act in the most-efficient way to achieve the goal.

Q.7. Explain learning agents in details.

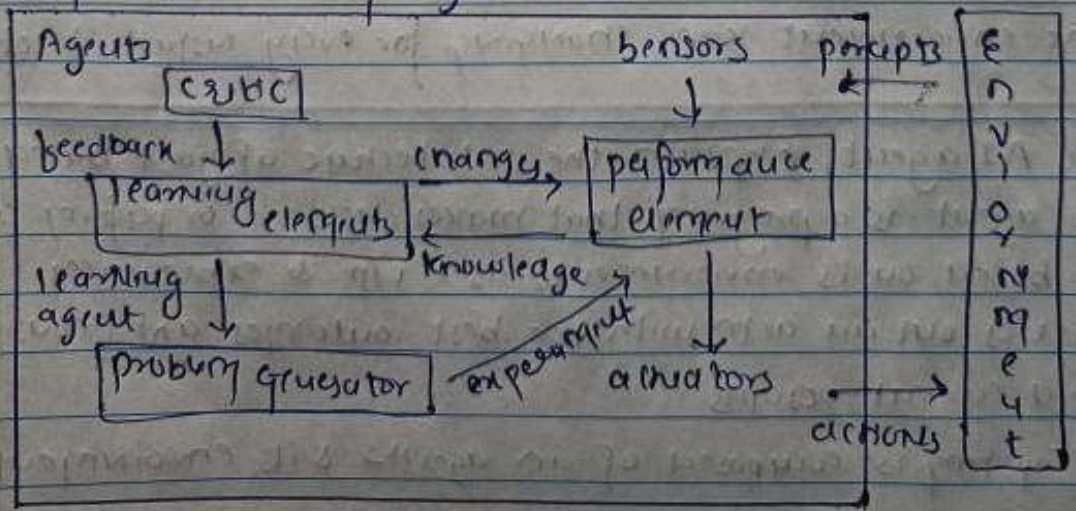


fig. learning agent

PTO

A learning agent in AI is the type of the agent that can learn from its past experiences or it has learning capabilities. It starts to act with basic knowledge and then is able to act & adapt automatically through learning.

A learning agent has mainly four components, which are:

1) learning element: It is responsible for making improvements by learning from environment.

2) Critic: the learning element takes feedback from critic which describes how is the agent is doing w.r.t a fixed performance standard

3) performance element: Responsible for selecting external action.

4) problem generator: This component is responsible for suggesting actions that will lead to new and informative experiments.

Q.8 What is uniformed searching? List various types of uniformed search techniques & Explain any one type with suitable graph tree and in details.

1) The uniformed search techniques have no additional information about states other than those provided in the problem defn.

2) They can only generate successor and distinguish b/w goal state and non-goal state.

3) They do not have any clue in specific knowledge.

> Types:

i) Depth first search
ii) Breadth first search
iii) Uniform-cost search
iv) Depth-limited search
v) Iterative deepening search
vi) Bidirectional search.

Q.10 → 1) Breadth first search:

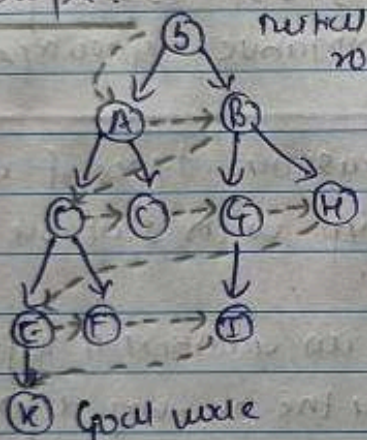
Breadth first searching algorithm is the most common or informed search strategy for traversing a tree or a graph.

PTO

This algorithm searches breadth wise
This algo. start search from root node of tree & already on successor node of the current layer before moving to nodes to next level.

Algorithm:
Step 1: Start by putting any one of the graph's vertices at the back of the queue.
Step 2: Take a front item of queue and add it to the visited list.
Step 3: Enroll a list of that vertex's adjacent nodes. queue
Step 4: Keep continuing step two & three till the queue is empty

Example:



Vertex node/ root node → level 0
— level 1
— level 2
— level 3
K Goal node — level 4

Traversed path →
S → A → B → C → D → G → H → E → F → I → K. //

Applications of BFS:-
i) unweighted graphs
ii) peer to peer network
iii) web crawlers
iv) Navigator systems
v) Network broadcasting.

Advantages:
i) soln will be definitely found out by BFS if there exist soln.
ii) BFS will never get trapped in blind valley, (unwanted nodes)
iii) If node more than one soln, then find soln with minm steps.

Disadvantages:
i) memory contraints (too much use of memory)
ii) consumes very large time

PTO

performance measures:
i) completeness: BFS is complete, if there is a goal node.
ii) complexity: i) time - $T(b) = 1 + b^2 + b^3 + \dots b^d$ ∴ b = node
$T(b) = O(b^d)$ d = depth
ii) space = $O(b^d)$

Q.9. What is informed search Technique? List various types of informed search techniques & explain any one with suitable graph tree and algorithms.

i) Informed search decide whether one non-goal state is more promising than another non-goal state.
2) It helps to find soln efficiently.
3) It has the prior knowledge about goal node.

> Types:
i) best first search
ii) Greedy best first search
iii) A* search
iv) memory bonded heuristic search.

Q.10 → i) Best-first search:
Best first search algorithm always select the path which appears best at the moment.
It is the combination of depth first search & breadth-first searching
It uses an heuristic methods for searching.
It also called as best first Greedy search
Costis associated with it in this method.

Algorithm:
Step 1: put the initial node on a list START
Step 2: If [START is empty] or [START = GOAL], then terminate search.
Step 3: Remove the first node from START, call this as Node A.
Step 4: If [a = goal], then terminate search with success.

PTO

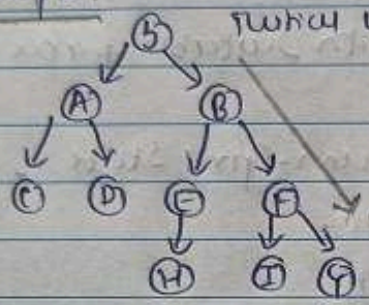
Step5: Else if node a has successor, generate all of them. Find out how many far they from Goal node. Sort them, by removing distance from the Goal.

Step6: Name the list as START 1.

Step7: Replace START with START 1

Step8: Goto Step2.

example:



node	H(n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
G	0
S	13

final path is:- S → B → F → G //

Advantages: It has advantage of both BFS and DFS more efficient than BFS & DFS.

Disadvantages: i) Time complexity: $T(n) = O(b^d)$
ii) Space complexity: $S(n) = O(b^q)$
iii) completeness: Incomplete if given state is infinite.
iv) optimality: Not optimal
} performance

It has behave an unguided depth first search in the worst case scenario.

Q.12. Explain A* Algorithm in details & solve for search path for the tree given.

i) A* is the most common form of best first search.

ii) It uses heuristic func(n) h(n) & cost to reach the node n from the start state g(n).

iii) A* search algorithm finds the shortest path through the search space using the heuristic function.

iv) This search algorithm expand less search tree & provides optimal result faster.

Algorithm:

Step 1: put the initial node on a list START

Step 2: If (START is empty) or (START = GOAL), then terminate search.

Step 3: Remove the first node from START call this node a.

Step 4: If (a = GOAL), then terminate search with success.

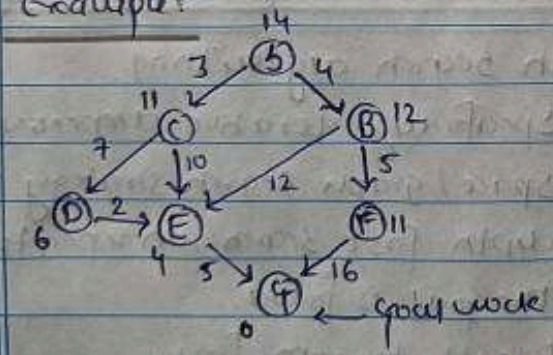
Step 5: Else, if node a has successor, generate all of them. Estimate the fitness no. of success by totaling the evaluation function value & the cost function value. Sort the list by fitness numbers.

Step 6: Name this new list as start 1

Step 7: Replace START with START 1

Step 8: Go to step 2.

Example:



node(n)	h(n)
S	14
B	12
C	11
D	6
E	4
F	11
T	0

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

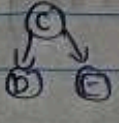
$f(C) = 11 + 3 = 14$

$f(B) = 4 + 12 = 16$

$F(E) = 12 + 4 = 16$

$F(G) = 16 + 11 = 27$

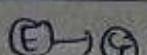
$f(D) = 6 + 7 = 13$



$F(E) = 4 + 10 = 14$

$F(E) = 4 + 2 = 6$

$f(G) = 5 + 0 = 5$



PTO

∴ final goal node path:-

$F(F) = 4+5+16 = 25$

$F(G) = 4+12+5 = 21$

$F(G) = 3+7+2+5 = (17)$ ... optimal

$F(G) = 3+10+5 = 18$

∴ path ⇒ b → C → D → E → G is the optimal path with cost 17.

(b → C → D → E → G) //

Advantages: i) Best algorithm than other methods.

ii) optimal & complete

iii) can solve very complex problems.

Disadvantages: i) It may not produce the shortest path, but path depends upon the fitness value.

ii) A* has some complexity issue.

iii) requires large memory as it requires all generated nodes in memory.

Q.13 Explain in detail iterative depth search algorithm.

i) Iterative deepening search or more specifically iterative deepening depth-first search (IDS) is a state space / graph search strategy in which a depth-limited version of depth-first search run repeatedly with increasing depth limits until the goal is found.

ii) The node which is first visited is efficiently breadth-first.

iii) It is optimal when path cost is non-decreasing function of depth of node.

Algorithm:

iterative-deepening-search (problem) returns a soln or failure.

inputs: problem, a problem

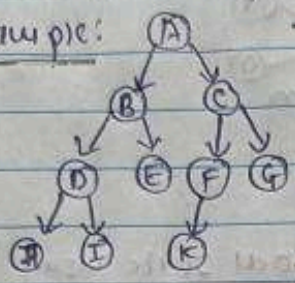
for depth: ← 0 to ∞ do

result ← Depth-limited search (problem, depth)

if result ≠ cut off then return result.

It terminates when a soln is found or if depth-limit search returns failure meaning that no soln exist.

Example:



— level 0
— level 1
— level 2
— level 3

1st iteration → A
2nd iteration → A, B, C
3rd iteration → A, B, D, E, C, F, G
4th iteration → A, B, D, H, I, E, C, F, K, G
in 4 iteration → it will find goal node.

Performance: i) Completeness: the algorithm is complete is branching infinite
ii) Time complexity: $T(n) = O(b^d)$ · b = node
iii) Space complexity: $T(n) = O(b^d)$ d = depth
iv) Optimal: Non-uniform path function

Advantages: It combines benefits of BFS & DFS; memory efficient
Disadvantages: Repetition of previous steps.

Q.14. Explain details about bidirectional search algorithm:

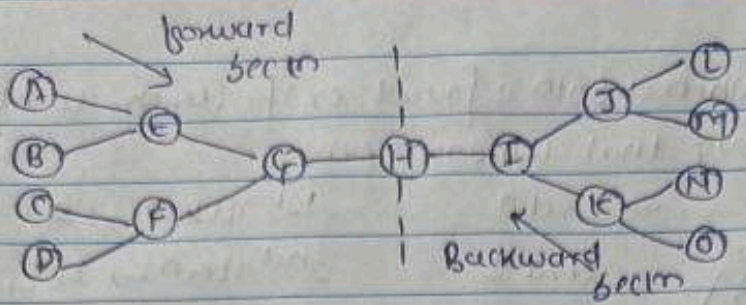
1) The principle used in bidirectional heuristic search algorithm is to find the shortest path from the current node to goal node.
2) The only difference being the two simultaneous searches from the initial point & from goal vertex.
3) The main idea behind bidirectional searches is to reduce the time taken for search drastically.
4) This takes place when both searches happen simultaneously from the 'initial node point' or 'breadth-first' and 'backwards' from goal node.

Algorithm:

Step 1: Let initial node & goal node be the intersection node.
Step 2: We start searching simultaneously from start to goal node & backward from goal node to start node.
Step 3: When the forward search & backward search intersect at one node, then searching stops.

PTO

Example:



i) Front to back: BFEA

ii) Front to front: BFFA

i) Front to back: It is estimated as node to Goal state using forward search & second, node to start state using reverse action.

ii) Front to front: Here the distance of all nodes is calculated and h is calculated as the maximum of all heuristic distance from the current node.

Performance measure:

1) Completeness: Bidirectional search is complete if Breadth first search (BFS) is used in both searches.

2) Optimality: It is optimal if BFS is used for search & paths have uniform cost.

3) Complexity: Time = $T(n) = O(b^{d/2})$ ∴ b = node

Space = $S(n) = O(b^{d/2})$ d = depth

Advantages: 1) Bidirectional search is fast search method

2) It require less memory space.

Disadvantages: 1) Implementation of bidirectional search is difficult.

2) In bidirectional search, one should know that the Goal state in advance.

Name :- Prathamesh S. Chikankar

Roll No. :- AIML11 Branch :- CSE-(AI&ML)

Year :- TE Subject :- AI

Topic :- Assignment No. 02

Sign :- Pratmy Date :- October'22

Q.1 What is Knowledge Representation?

Knowledge consists of facts, concepts, rules, and so on.
It can be represented in different forms, as mental images in one's thoughts, as spoken or written words in some language, as graphical or other pictures, and as character strings or collection of magnetic spots stored in a computer.
Knowledge Representation is a part of AI which concerned with AI agents thinking and how thinking contributes to intelligent behaviour of agent.
Responsible for representing information about the real world and a way which describes how we can represent knowledge in AI.
Explain different levels of knowledge Representation.

> Declarative knowledge: To know about something includes concepts, facts and objects. Simple than procedural language. Also called as descriptive knowledge & expressed in declarative sentence

> procedural knowledge: known as imperative knowledge. Type of knowledge which is responsible for knowing how to do something, can be directly applied to any task. includes rules, strategies, procedures, agendas, etc..

> Meta-knowledge: knowledge about the other types of knowledge

> Heuristic knowledge: representing knowledge of some experts in a field or subject. Rules of thumb based on previous experiences, awareness of approaches, and which are good to work but not guaranteed.

> Structural knowledge: Basic knowledge to problem solving. Describes relationship blw concepts & objects that exists.

- Knowledge level: Knowledge base & Inference system
- logical level: how knowledge represents knowledge is stored.
- Implementation level: physical representation of logic & knowledge

declarative approach & procedural approach.

Q.2. Difference b/w procedural & declarative knowledge

	Procedural knowledge	Declarative knowledge
1)	follow black box	follow white box
2)	Based on process orientation.	Based on data orientation.
3)	possible to faster usage	Based on knowledge in the process of system design
4)	knowing how to do something	It is knowledge about something
5)	Simple data type can be used	large data type can be used
6)	followed M C++ and cobol	followed by SQL
7)	programmers should understand execution	programmer should not interact with SQL
8)	initially faster but later it can be slow	initially slower but possibly faster later.
9)	work on interprites of language	work on data engine with DBMS

Q.3. Explain reasoning under uncertainty.

In a reasoning system, there are several types of uncertainty. Reasoning under uncertainty research in AI is 'focused on uncertainty of truth value' in order to find values other than True or false

> To develop a system that reasons with uncertainty means to provide the following:

1. An explanation about the origin & nature of uncertainty
2. To represent uncertainty in formal language.
3. A set of inference rules that deriver uncertain conclusions.
4. efficient memory-control mechanism for uncertainty management.

> when an inconsistency is assigned, only the truth value of the last type is changed.

PTO

Explain the following terms

① Mutually exclusive events:

→ events are said to be mutually exclusive if the happening of any one of them precludes the happening of all the others. i.e., no two or more can happen simultaneously in the same trial.

→ for example, in tossing a die all the 6 faces numbered 1 to 6 are mutually exclusive, since any one of the faces comes, the possibility of others is ruled out.

② Independent events:

→ several events are said to be independent if the happening (or non-happening) of an event is not affected by the happening (or non-happening) of the remaining events.

→ for example, when a die is thrown twice, the result of the first throw doesn't affected the result of the second throw.

Q.4. what is probability?

If an event E happens in m ways out of n possible likely ways, the probability of occurence of the event E is defined as

$$\Rightarrow \boxed{P(E) = p = m/n}$$

the probability of non-occurence is defined by

$$\Rightarrow P(\text{not } E) = q = \frac{n-m}{n} = 1 - \frac{m}{n} = 1 - p$$

$$\therefore \boxed{p + q = 1}$$

Solve the below example:

A, B, C are bidding for a contract. A has exactly half the chance that B has; B in turn is 4/5 as likely as C to win the contract. what is probability for each to win the contract, what is the contract is to be given to one of them.

Soln:-

PTO

> step 1): Since the events A, B and C are exclusive,

$\therefore P(A) + P(B) + P(C) = 1$ → eqn 1)

Now, $P(A) = 1/2\ P(B)$ and → eqn 2)

$P(B) = 4/5\ P(C)$ → eqn 3)

> step 2): let p be the probability of C, from eqn 1)

$1/2 \cdot 4/5 \cdot P(C) + 4/5 \cdot P(C) + P(C) = 1$

$\therefore \left(\frac{2}{5} + \frac{4}{5} + 1\right) P = 1 \Rightarrow P = 5/11$

$\therefore$ **$P(A) = 2/11$, $P(B) = 4/11$, $P(C) = 5/11$** → Ans.

Q. 5. What is probabilistic reasoning?

Probabilistic reasoning is a way of knowledge representation where we apply the concept of probability to indicate the uncertainty in knowledge. We combine probability theory with logic to handle the uncertainty. We use probability in probabilistic reasoning because it provides a way to handle the uncertainty that is the result of someone's laziness and ignorance.

There are two ways to solve problems with uncertain knowledge:

1. Bayes' rule 2. Bayesian statistics

explain in details Bayes theorem.

> Bayes' theorem; in simple words, determines the conditional prob. of an event A given that event B already occured.

Bayes theorem is also known as the Bayes rule or Bayes law.

> It is a method to determine the probability of an event based on the occurrence of prior events. It is used to calculate conditional probability. (Based on hypothesis - calculation).

> Bayes theorem states that the conditional probability of an event A, given the occurrence of another event B, is equal to the product of the likelihood of B, given A & probability of A.

PTO

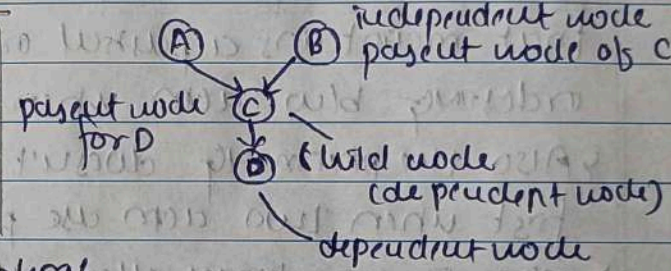
> Given as,

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- P(A) = how likely A happens
- P(B) = how likely B happens
- P(B/A) = how likely B happens given that A has happened (likelihood)
- P(A|B) = how likely A happens given that B has happened (posterior)

Q. 6. Explain Bayesian belief networks in details

- Multiple causes
- Multiple probability



> To solve the causes and outcome based probability problems, Bayesian belief Nlw are used.

> Bayesian Nlw are build based on probability distribution and joint probability distribution.

> This are also called bayes Nlw, belief Nlw, decision Nlw and Bayesian model.

> Real world problems are probabilistic and interconnected in nature. So Bayesian belief Nlw is largely used.

> Bayesian Belief Nlw consist of two parts.
① directed acyclic graph
② table of conditional probability.

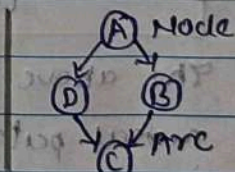


fig. Directed acyclic graph.

> The structured form of Bayesian Nlw that represent & so on distance problems under the uncertain knowledge is known as influence diagram.

> A Bayesian Nlw are made up of nodes and arcs.

> Node represent random variable of graph & arc represent the caused relationship b/w random variables or nodes.

Q.7. what is planning in AI?

planning → order of acm to perform a task.

The task coming up with a sequence of acm that will achieve a goal is called planning.

planning is typically viewed as a generic term of problem solving because it deals with search in an abstract level.

Explain in details and with suitable examples

① partial order planning:

↳ partial order planning is an approach to automated planning that maintains a partial ordering b/w acm and only commits ordering b/w acm when the acm are parties.

↳ Also the planning doesn't specify which acm will come out first when two acm are processed.

↳ This is sometimes also called a non-linear planner.

↳ The partial ordering is a less-than relation that is transitive and assymmetric.

↳ Example:

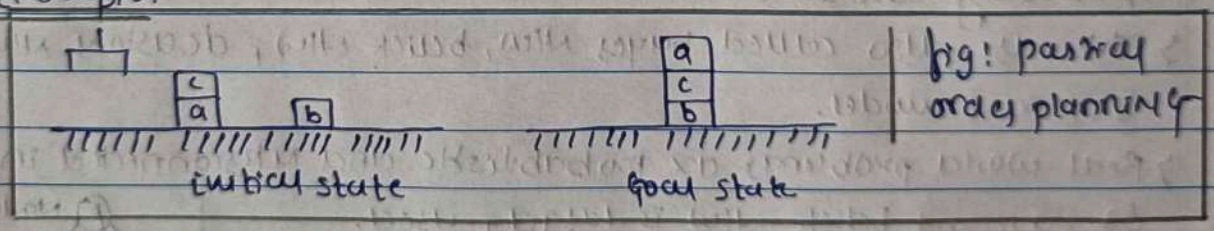


fig: partial order planning

The above initial state can be represented in POP as the following initial path. plan.

- plan (STEPS: {S1: Op (ACTION: start, EFFECT: clear(b) ∧ clear(c) ∧ on(c,a) ∧ ONTABLE(a) ∧ ONTABLE(b) ∧ ARMEMPTY), S2: Op (ACTION: finish, PRECOND: on(c,b) ∧ on(c,a))}, ORDERINGS: {S1<S2}, LINKS: {})

↳ This initial path is refined using POP's plan refinement operators. As we apply them, they will take us from an unfinished plan to a less and less unfinished plan, and ultimately to a soln plan.

PTO

② Total order planning

Fixed order planning where order of action are fixed, if an action gets failed if particular task is performed or goal / destination may not get reached, but sometimes it is advantageous at order of planning is fixed.

Forward state space search and Backward state space search are example of Total order planning. They only explore linear sequences of action from start to goal state, they cannot take advantage of problem decomposition, ie. Splitting the problem into smaller sub-problems and solving them individually. open set of three conditions, which specified which three conditions are not fulfilled by any action in partial order plan.

for example, suppose the goal is to deliver a specific piece of cargo to mumbai! This suggests the action unload (C, P, Mumbai), Thus, Action (unload (C, P, Mumbai))

precondition: In (C, P) ∧ (P, Mumbai) ∧ cargo (C) ∧ plane (P) ∧ Airport (Mumbai)

Effect: At (C, Mumbai) ∧ ¬ In (C, P)

Q.8. Explain supervised learning in details.

> learning that takes place based on a class of example is referred to as supervised learning. It is learning based on labelled data. In short, while learning, the system has knowledge of a set of labelled data. This is one of the most common & frequently used learning methods.

> The supervised learning method is comprised of a series of algorithms that build mathematical models of certain data sets that are capable of containing both inputs and the desired outputs for that particular machine.

> supervised learning uses classification and regression techniques to

PTO

develop predictive models. classification techniques predict, responses.

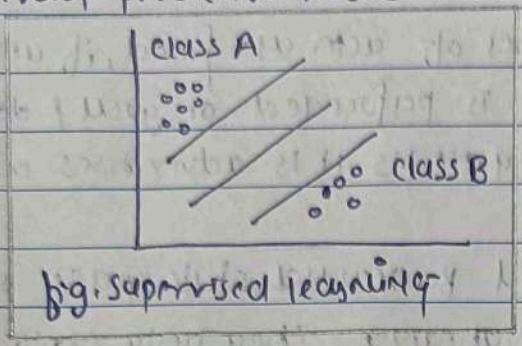


fig. supervised learning

> there are no. of challenges in supervised classification such as generalization, selection of input data for learning, & dealing with the variation.

Labeled examples are used for training in case of supervised learning. The set of labelled examples provide to learning algorithm is called the training set.

> Supervised learning is not just about classification, but is overall process that with guidelines maps to most appropriate decision.

Q.9. Explain Unsupervised learning in details.

> Unsupervised learning refers to learning from unlabelled data. It is based more on similarity and differences than on anything else. In this type of learning, all similar items are clustered together in a particular class where the label of class is not known.

> The task of hierarchical clustering is to arrange a set of objects into a hierarchy such that similar objects are grouped together. Non-hierarchical clustering seeks to partition the data into some no. of disjoint clusters. The process of clustering is depicted in figure given.

> A learner is fed with a set of scattered points, & it generates two clusters with respective representative centroids after learning. clusters show that points with similar properties and closeness are grouped together.

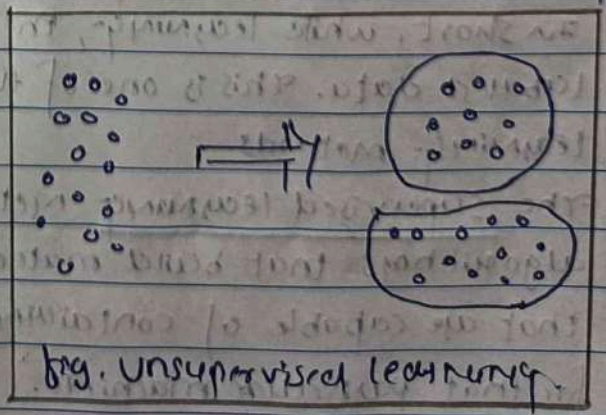


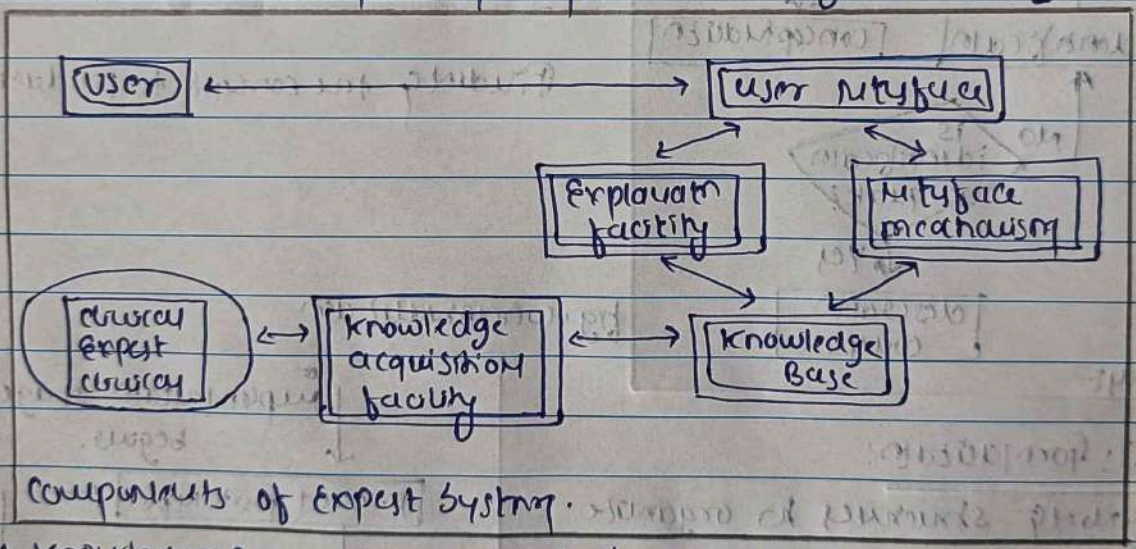
fig. Unsupervised learning.

PTO

> Unsupervised learning is a set of algorithms where the only information being uploaded is inputs. The device itself from there, is responsible for grouping together and creating ideal outputs based on the data it discovers. Often, unsupervised learning algorithms have certain goals, but they are not controlled in any manner.

Q.10. Explain expert system in details with suitable diagram.

Expert system is a AI based system that converts the knowledge of an expert in a specific subject into a software code, this code can be merged with other such codes based on the knowledge of other experts, and used for answering queries submitted through a computer. ✱ Expert system = knowledge + Inference engine.



Components of expert system.

1. Knowledge Base: Core module of any expert system (KB)
2. Inference engine: also called as 'rule interpreter' (IE)
   - major task of IE is to trace its way through a forest of rules.
3. User interface: provides needed facilities for users to communicate with system.
4. Knowledge acquisition facility: Creates a congenial atmosphere for expert.
5. External interface: communicates user b/w ES & external environment.
6. Explanation facility: ES reaches a conclusion may not be obvious to human users. Answers to users are available - Basics.

Q.11. Explain in details development stages of an expert system.

Expert system life cycle has five stages -

Step 1: Identification:

Determining the characteristics of the problem.

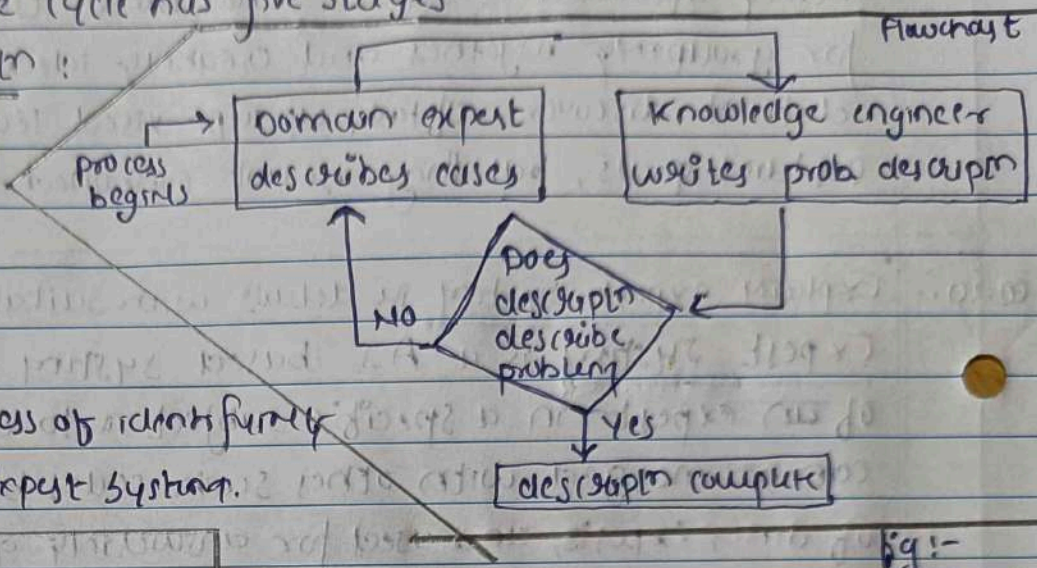


fig. Iterative process of identifying the problem in expert system.

fig:- identification

Step 2: Conceptualization:

Finding the concept to produce the solution.

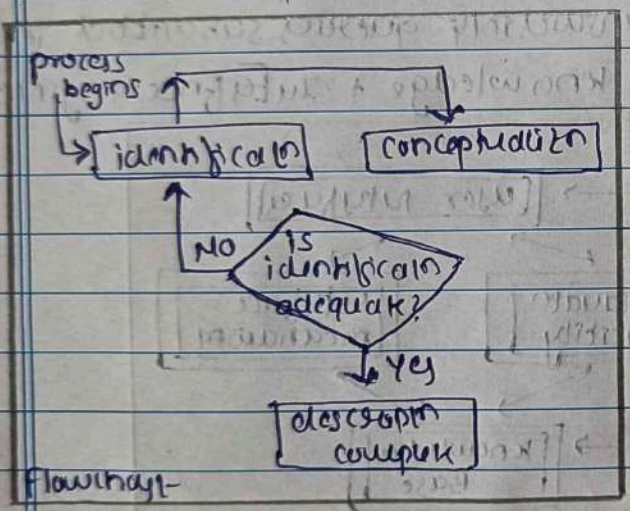


fig. conceptualization

Step 3: Formalization:

designing structures to organize the knowledge.

Step 4: Implementation:

formulating rules which embody the knowledge.

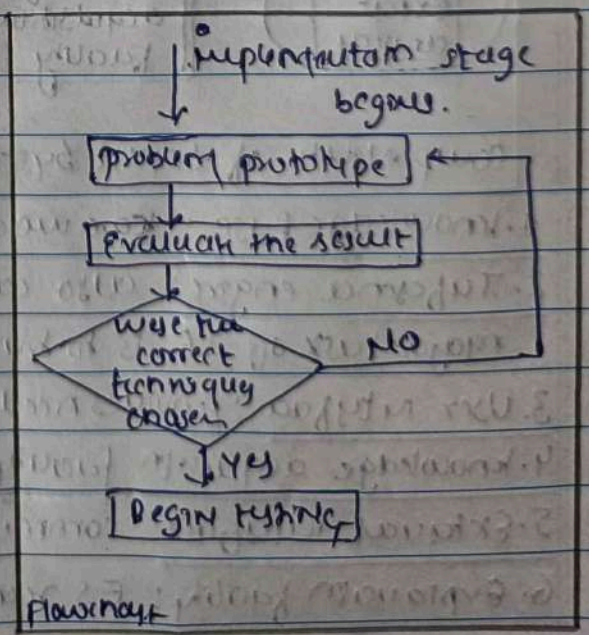


fig. Implementation

Step 5: Testing:

validating the rules.