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

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

NAME	**SINGH SUDHAM DHARMENDRA**
BRANCH	**CSE-(AI&ML)**
ROLL NO.	**AIML57**
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

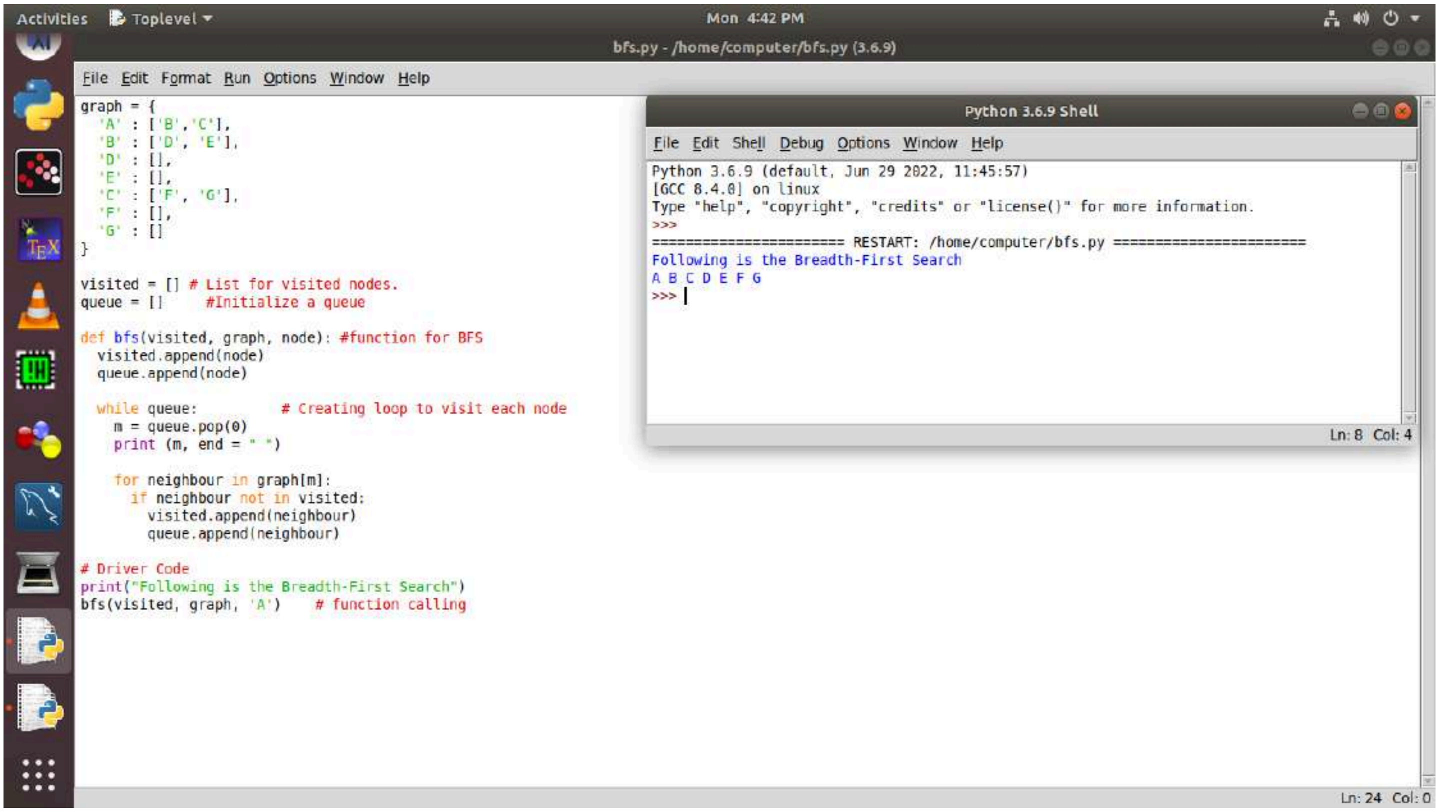
>Cost:

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 (HILL CLIMB) -

```
def findLocalMaxima(n, arr):
    mx = []
    if(arr[0]> arr[1]):
        mx.append([0,arr[0]])
    for i in range(1, n-1):
        if(arr[i-1] < arr[i] > arr[i + 1]):
            mx.append([i,arr[i]])
    if(arr[-1]> arr[-2]):
        mx.append([n-1,arr[n-1]])
    if(len(mx) > 0):
        for i in mx:
            print("Local maxima at position :", i[0], "is" ,i[1])
    else:
        print("There are no points of Local maxima.")
if __name__ == '__main__':
    n = 9
    arr =[10, 10, 15, 14, 13, 25, 50, 3]
    findLocalMaxima (n, arr)
```

OUTPUT -

```
python -u "d:\StudyTime\TE\AI\hillClimb.py"
Local maxima at position : 2 is 15
Local maxima at position : 6 is 50
```

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
>>>
```

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 -

```
?- ['/home/computer/Documents/CSE-AIML/TE/AIML57_SUDHAM/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/AIML57_SUDHAM/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 !!!!
```