

Lab File

Artificial Intelligence [CSE401]

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING



Submitted To:

Dr. S. K. Dubey
Associate Professor
CSE Department, ASET

Submitted By:

Shubham Periwal
A2305218241
B. Tech (CSE)
7CSE4-Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY UTTAR PRADESH
NOIDA-201301

INDEX

| Exp eri men t No. | Category of Assignment | Co de | Name of Experiment | Date of Allotment of Experimen t | Date of Evaluati on | Max. Mark s | Marks obtaine d | Sign. of Facult y |
|-------------------------------|------------------------------|-----------|---|--|---------------------------|-------------------|-----------------------|----------------------------|
| 1. | Mandatory Experiment | LR (0) | Write a Experiment to implement A* algorithm in python | 23-07-21 | 30-07-21 | 1 | | |
| 2. | Mandatory Experiment | | Write a Experiment to implement Single Player Game | 30-07-21 | 13-08-21 | 1 | | |
| 3. | Mandatory Experiment | | Write a Experiment to implement Tic- Tac-Toe game problem | 13-08-21 | 27-08-21 | 1 | | |
| 4. | Mandatory Experiment | | Implement Brute force solution to the Knapsack problem in Python | 27-08-21 | 03-09-21 | 1 | | |
| 5. | Mandatory Experiment | | Implement Graph coloring problem using python | 03-09-21 | 09-09-21 | 1 | | |
| 6. | Mandatory Experiment | | Write a Experiment to implement BFS for water jug problem using Python | 09-09-21 | 17-09-21 | 1 | | |
| 7. | Mandatory Experiment | | Write a Experiment to implement DFS using Python | 10-09-21 | 17-09-21 | 1 | | |
| 8. | Mandatory Experiment | | Tokenization of word and Sentences with the help of NLTK package | 17-09-21 | 24-09-21 | 1 | | |
| 9. | Mandatory Experiment | | Design an XOR truth table using Python | 24-09-21 | 1-10-21 | 1 | | |
| 10. | Mandatory Experiment | | Study of SCIKIT fuzzy | 01-10-21 | 22-10-21 | 1 | | |

Experiment 1

Date: 23-07-21

Software used: Python 3.8

Aim: Write a Experiment to implement A* algorithm in python.

Theory: A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals. Informally speaking, A* Search algorithms, unlike other traversal techniques, it has “brains”. What it means is that it is really a smart algorithm which separates it from the other conventional algorithms. This fact is cleared in detail in below sections. And it is also worth mentioning that many games and web-based maps use this algorithm to find the shortest path very efficiently.

Imagine a square grid which possesses many obstacles, scattered randomly. The initial and the final cell is provided. The aim is to reach the final cell in the shortest amount of time.

Here A* Search Algorithm comes to the rescue:

A* algorithm has 3 parameters:

- **g** : the cost of moving from the initial cell to the current cell. Basically, it is the sum of all the cells that have been visited since leaving the first cell.
- **h** : also known as the *heuristic value*, it is the **estimated** cost of moving from the current cell to the final cell. The actual cost cannot be calculated until the final cell is reached. Hence, h is the estimated cost. We **must** make sure that there is **never** an over estimation of the cost.
- **f** : it is the sum of g and h. So, **f = g + h**

The way that the algorithm makes its decisions is by taking the f-value into account. The algorithm selects the *smallest f-valued cell* and moves to that cell. This process continues until the algorithm reaches its goal cell.

Code:

```
def aStarAlgo(start_node, stop_node):
    sum1=1
    open_set = set(start_node)
    closed_set = set()
    g = {}
    parents = {}
    g[start_node] = 0
    parents[start_node] = start_node
    while len(open_set) > 0:
        n = None
        for v in open_set:
```

```

        if n == None or g[v] + heuristic(v) < g[n] + heuristic(n):
            n = v
    if n == stop_node or Graph_nodes[n] == None:
        pass
    else:
        for (m, weight) in get_neighbors(n):
            if m not in open_set and m not in closed_set:
                open_set.add(m)
                parents[m] = n
                g[m] = g[n] + weight
            else:
                if g[m] > g[n] + weight:
                    g[m] = g[n] + weight
                    sum1 = sum1 + g[m]
                    parents[m] = n
                    if m in closed_set:
                        closed_set.remove(m)
                    open_set.add(m)

    if n == None:
        print('Path does not exist!')
        return None
    if n == stop_node:
        path = []
        while parents[n] != n:
            path.append(n)
            n = parents[n]
        path.append(start_node)
        path.reverse()
        print('Shortest Path found for graph: {}'.format(path))
        print('Cost: ' + str(sum1))
        return path
    open_set.remove(n)
    closed_set.add(n)
    print('Shortest Path does not exist!')
    return None

def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None

def heuristic(n):
    H_dist = {'A': 10, 'B': 8, 'C': 5, 'D': 7, 'E': 3, 'F': 6, 'G': 5, 'H': 3,
'I': 1, 'J': 0}
    return H_dist[n]

Graph_nodes = {
    'A': [('B', 6), ('F', 3)],
    'B': [('D', 2), ('C', 3)],
    'C': [('B', 3), ('D', 1), ('E', 5)],
    'D': [('B', 2), ('C', 1), ('E', 8)],

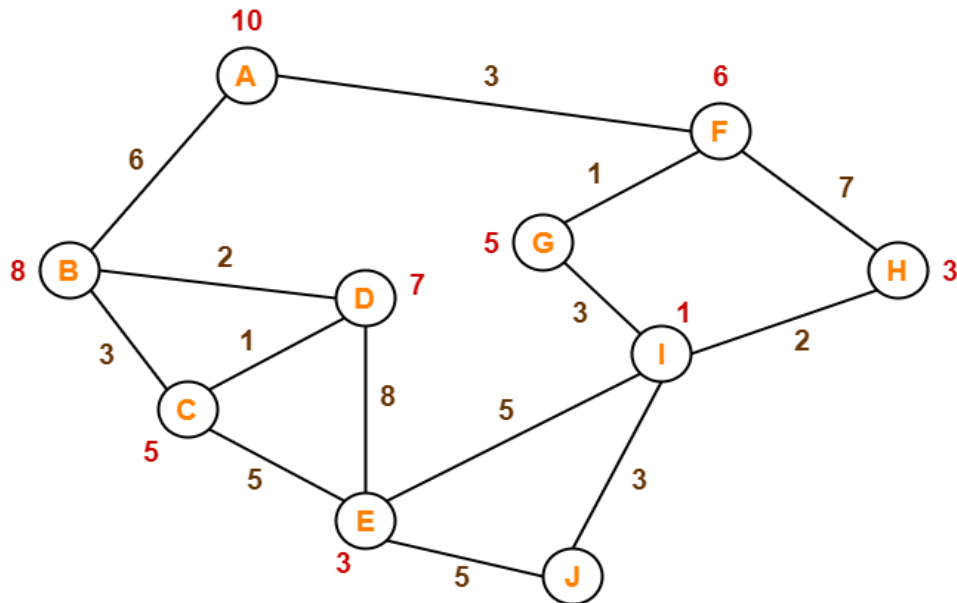
```

```

'E': [('C', 5), ('D', 8), ('I', 5), ('J', 5)],
'F': [('A', 3), ('G', 1), ('H', 7)],
'G': [('F', 1), ('I', 3)],
'H': [('F', 7), ('I', 2)],
'I': [('G', 3), ('E', 5), ('J', 3), ('H', 2)],
'J': [('E', 5), ('I', 3)]
}
print("Shubham Periwal 7CSE-4Y")
aStarAlgo('A', 'J')

```

Input:



Output:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe"

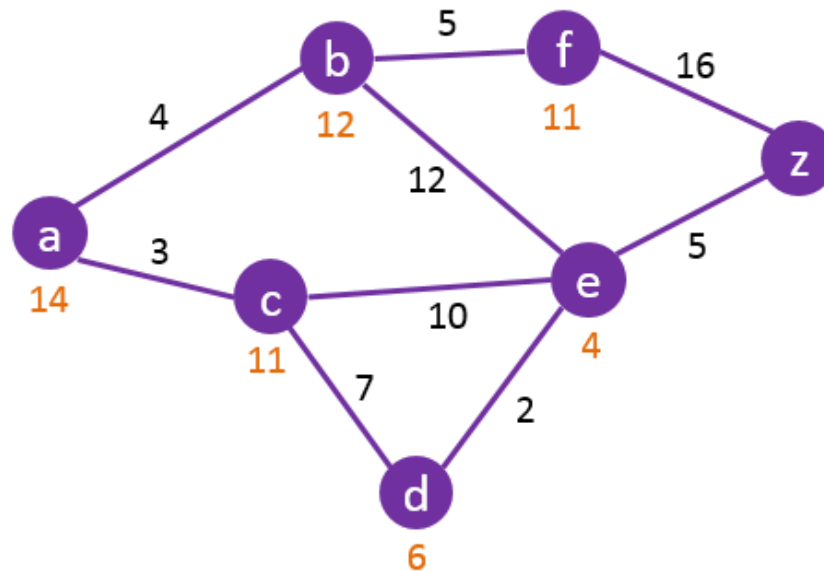
Shubham Periwal 7CSE-4Y

Shortest Path found for graph: ['A', 'F', 'G', 'I', 'J']

Cost: 10

PS C:\Users\SHUBHAM PERIWAL> █

Input:



Output:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```
PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/pyt
```

Shubham Periwal 7CSE-4Y

Shortest Path found for graph: ['S', 'C', 'D', 'E', 'G']

Cost: 13

```
PS C:\Users\SHUBHAM PERIWAL>
```

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment-2

Date: 30-07-21

Aim: Write a Experiment to implement Single Player Game.

Software Used: Python 3.8

Theory: In a single-person decision, there is only one player. *Player* is the term used in game theory for any entity capable of making a decision. Part of game theory is making assumptions about the behavior of the players. Our outcome is only as good as our assumptions. Therefore, a significant part of game theory requires us to challenge our assumptions. But for today's lesson, we'll keep it simple.

We all face multiple decisions every day. A congressman decides how to vote on a particular bill. A product manager decides to cut a feature before the release date. Even the trivial decision about which shirt to wear is a daily occurrence. Each decision we confront has three parts.

Actions, Outcomes, and Preferences

- Actions are all the options from which we can choose.
- Outcomes are consequences resulting from the actions.
- Preferences are the ordering of possible outcomes from most desirable to least desirable.

Code:

```
import random
print('Shubham Periwai 7CSE-4Y')
def computerChoiceGenerator():
    x=random.randint(1,3)
    return x

def WaterFireIceGame():
    choice={
        1: "Water",
        2: "Fire",
        3: "Ice"
    }
    computerInt=computerChoiceGenerator()
    userInt=int(input("Enter your choice 1.Water 2.Fire 3.Ice"))
    computerChoice=choice[computerInt]
```



```

userChoice=choice[userInt]

if(computerChoice==userChoice):
    return 0
elif((userChoice=="Fire" and computerChoice=="Ice") or (userChoice=="Ice" and
computerChoice=="Water") or (userChoice=="Water" and computerChoice=="Fire")):
    print(userChoice+" beats "+computerChoice+"! You Win")
    return 1
else:
    print(computerChoice+" beats "+userChoice+"! You Lost")
    return -1

def playGame():
    userScore=0
    computerScore=0
    nextGame=1
    while(nextGame==1):
        gamePoint=WaterFireIceGame()

        if gamePoint==0:
            print("Game Tied!")
        elif gamePoint==1:
            print("User Won")
            userScore+=1
        else:
            print("Computer Won")
            computerScore+=1
        print("User Score: "+str(userScore)+" Computer Score:
"+str(computerScore))

        nextGame=int(input("Enter 1 to continue, else 0"))

print("Rules of the Game-")
print("1. Water Beats Fire")
print("2. Fire Beats Ice")
print("3. Ice Beats Water")

playGame()

```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\SHUBHAM PERIWAL\Desktop> & 'C:\Users\SHUBHAM PERIWAL\AppData\Local\Programs\Python\Python36\python.exe' 'c:
10.1365161279\pythonFiles\lib\python\debugpy\launcher' '63893' '--' 'c:\Users\SHUBHAM PERIWAL\Desktop\import random.py'
Shubham Periwai 7CSE-4Y
Rules of the Game-
1. Water Beats Fire
2. Fire Beats Ice
3. Ice Beats Water
Enter your choice 1.Water 2.Fire 3.Ice 1
Water beats Fire! You Win
User Won
User Score: 1 Computer Score: 0
Enter 1 to continue, else 0 1
Enter your choice 1.Water 2.Fire 3.Ice 2
Water beats Fire! You Lost
Computer Won
User Score: 1 Computer Score: 1
Enter 1 to continue, else 0
```

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment-3

Date: 13-08-21

Aim: Write a Experiment to implement Tic-Tac-Toe game problem.

Software Used: Python 3.8

Theory: Tic-Tac-Toe is a very simple two-player game. So only two players can play at a time. This game is also known as Noughts and Crosses or Xs and Os game. One player plays with X and the other player plays with O. In this game we have a board consisting of a 3X3 grid. The number of grids may be increased.

Game Rules:

1. Traditionally the first player plays with "X". So you can decide who wants to go with "X" and who wants to go with "O".
2. Only one player can play at a time.
3. If any of the players have filled a square then the other player and the same player cannot override that square.
4. There are only two conditions that may match will be draw or may win.
5. The player that succeeds in placing three respective marks (X or O) in a horizontal, vertical, or diagonal row wins the game.

Winning condition:

Whoever places three respective marks (X or O) horizontally, vertically, or diagonally will be the winner.

Code:

Player vs Player

```
theBoard = {'7': ' ', '8': ' ', '9': ' ',
            '4': ' ', '5': ' ', '6': ' ',
            '1': ' ', '2': ' ', '3': ' '}
board_keys = []
for key in theBoard:
    board_keys.append(key)

def printBoard(board):
    print(board['7'] + '|' + board['8'] + '|' + board['9'])
    print('-+-+-')
    print(board['4'] + '|' + board['5'] + '|' + board['6'])
    print('-+-+-')
    print(board['1'] + '|' + board['2'] + '|' + board['3'])

def game():
```

```

turn = 'X'
count = 0
print("Shubham Periwal 7CSE-4Y\n")
for i in range(10):
    printBoard(theBoard)
    move = input("\nTurn:" + turn + "\tMove: ")
    print("")
    if theBoard[move] == ' ':
        theBoard[move] = turn
        count += 1
    else:
        print("Place already filled.\nMove: ")
        continue
    if count >= 5:
        if theBoard['7'] == theBoard['8'] == theBoard['9'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['4'] == theBoard['5'] == theBoard['6'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['1'] == theBoard['2'] == theBoard['3'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['1'] == theBoard['4'] == theBoard['7'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['2'] == theBoard['5'] == theBoard['8'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['3'] == theBoard['6'] == theBoard['9'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['7'] == theBoard['5'] == theBoard['3'] != ' ':
            printBoard(theBoard)
            print("\nGame Over.\n")
            print(" **** " + turn + " won. ****")
            break
        elif theBoard['1'] == theBoard['5'] == theBoard['9'] != ' ':

```

```
        printBoard(theBoard)
        print("\nGame Over.\n")
        print("**** " + turn + " won. ****")
        break
    if count == 9:
        print("\nGame Over.\n")
        print("It's a Tie!!")
    if turn == 'X':
        turn = 'O'
    else:
        turn = 'X'
    restart = input("Do want to play Again?(y/n)")
    if restart == "y" or restart == "Y":
        for key in board_keys:
            theBoard[key] = " "
        game()
if __name__ == "__main__":
    game()
```

Output:

X won:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (c) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python
Shubham Periwai 7CSE-4Y

| |
-+-+
| |
-+-+
| |

Turn:X  Move: 1

| |
-+-+
| |
-+-+
X| |

Turn:O  Move: 2

| |
-+-+
| |
-+-+
X|O|

Turn:X  Move: 5

| |
-+-+
|X|
-+-+
X|O|

Turn:O  Move: 9

| |O
-+-+
|X|
-+-+
X|O|
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
| |  
--+-+--  
|X|  
--+-+--  
x|o|
```

Turn:O Move: 9

```
| |o  
--+-+--  
|X|  
--+-+--  
x|o|
```

Turn:X Move: 7

```
x| |o  
--+-+--  
|X|  
--+-+--  
x|o|
```

Turn:O Move: 3

```
x| |o  
--+-+--  
|X|  
--+-+--  
x|o|o
```

Turn:X Move: 4

```
x| |o  
--+-+--  
x|X|  
--+-+--  
x|o|o
```

Game Over.

**** X won. ****

Do want to play Again?(y/n)y

Shubham Periwai 7CSE-4Y

O Won:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Do want to play Again?(y/n)y
Shubham Periwai 7CSE-4Y

| |
-+-+
| |
-+-+
| |

Turn:X  Move: 1

| |
-+-+
| |
-+-+
X| |

Turn:O  Move: 5

| |
-+-+
|O|
-+-+
X| |

Turn:X  Move: 2

| |
-+-+
|O|
-+-+
X|X|

Turn:O  Move: 3

| |
-+-+
|O|
-+-+
X|X|O

Turn:X  Move: 4

| |
-+-+
```

PROBLEMS

OUTPUT

TERMINAL

DEBUG CONSOLE

```
| |  
-+-+  
|o|  
-+-+  
x| |
```

Turn:X Move: 2

```
| |  
-+-+  
|o|  
-+-+  
x|x|
```

Turn:O Move: 3

```
| |  
-+-+  
|o|  
-+-+  
x|x|o
```

Turn:X Move: 4

```
| |  
-+-+  
x|o|  
-+-+  
x|x|o
```

Turn:O Move: 7

```
o| |  
-+-+  
x|o|  
-+-+  
x|x|o
```

Game Over.

**** O won. ****

Do want to play Again?(y/n)n

PS C:\Users\SHUBHAM PERIWAL>

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 4

Date: 27-08-21

Aim: Implement Brute force solution to the Knapsack problem in Python.

Software Used: Python 3.8

Theory: Knapsack is basically means bag. A bag of given capacity. We want to pack n items in your luggage.

- The i th item is worth v_i dollars and weight w_i pounds.
- Take as valuable a load as possible, but cannot exceed W pounds.
- V_i, w_i, W are integers.

1. $W \leq \text{capacity}$
2. $\text{Value} \leftarrow \text{Max}$

Input:

- Knapsack of capacity
- List (Array) of weight and their corresponding value.

Output: To maximize profit and minimize weight in capacity.

The knapsack problem where we have to pack the knapsack with maximum value in such a manner that the total weight of the items should not be greater than the capacity of the knapsack.

Code:

```
def knapsackBruteForce(w,price,weight,idx,currPrice,currKnap):
    if weight==0:
        print("Current Knapsack- ",currKnap)
        print("Current Price- "+str(currPrice))
        return 0
    if idx== -1:
        print("Current Knapsack- ",currKnap)
        print("Current Price- "+str(currPrice))
        return 0

    op1=0
    op2=0
    if (weight>=w[idx]):
        currKnap.append(w[idx])
        op1+=knapsackBruteForce(w,price,weight-w[idx],idx-
1,currPrice+price[idx],currKnap)+price[idx]
        currKnap.pop()
```

```

        op2+=knapsackBruteForce(w,price,weight,idx-1,currPrice,currKnap)

    return max(op1,op2)

n=int(input("Enter number of Items"))
w=[]
price=[]

for i in range(0,n):
    w.append(int(input("Enter weight of item no- "+str(i+1)+" ")))
    price.append(int(input("Enter price of item no- "+str(i+1)+" ")))

weight= int(input("Enter capacity of the knapsack - "))
currKnap=[]
print("Maximum Profit- " +str(knapsackBruteForce(w,price,weight,n-1,0,currKnap)))

```

Input:

Number of elements: 3
 Weights: 10 20 30
 Profits: 60 100 120
 Maximum Capacity: 50

Output:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python

Enter number of Items 3

Enter weight of item no- 1 10

Enter price of item no- 1 60

Enter weight of item no- 2 20

Enter price of item no- 2 100

Enter weight of item no- 3 30

Enter price of item no- 3 120

Enter capacity of the knapsack - 50

Current Knapsack- [30, 20]

Current Price- 220

Current Knapsack- [30, 10]

Current Price- 180

Current Knapsack- [30]

Current Price- 120

Current Knapsack- [20, 10]

Current Price- 160

Current Knapsack- [20]

Current Price- 100

Current Knapsack- [10]

Current Price- 60

Current Knapsack- []

Current Price- 0

Maximum Profit- 220

PS C:\Users\SHUBHAM PERIWAL> █

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 5

Date: 03-09-21

Aim: Implement Graph coloring problem using python

Software Used: Python 3.8

Theory: Graph coloring problem is to assign colors to certain elements of a graph subject to certain constraints.

Vertex coloring is the most common graph coloring problem. The problem is, given m colors, find a way of coloring the vertices of a graph such that no two adjacent vertices are colored using same color. The other graph coloring problems like *dge Coloring* (No vertex is incident to two edges of same color) and *ace Coloring* (Geographical Map Coloring) can be transformed into vertex coloring.

Chromatic Number: the smallest number of colors needed to color a graph G is called its chromatic number. For example, the following can be colored minimum 2 colors.

Code:

```
def inputGraph():
    v=int(input("Enter the number of nodes: "))
    e=int(input("Enter number of edges: "))
    node={}
    graph={}

    for i in range(0,v):
        u=input("Enter Node "+str(i+1)+" Name ")
        graph[u]=[]
        node[u]=-1

    for i in range(0,e):
        u=input("Edge "+str(i)+" (from)- ")
        v=input("Edge "+str(i)+" (to)- ")
        graph[u].append(v)
        graph[v].append(u)

    for i in node.keys():
        if i not in graph:
            graph[i]=[]

    print(node)
    print(graph)
    return node,graph

def checkValidColor(src,graph,nodes,i):

    for nbr in graph[src]:
```

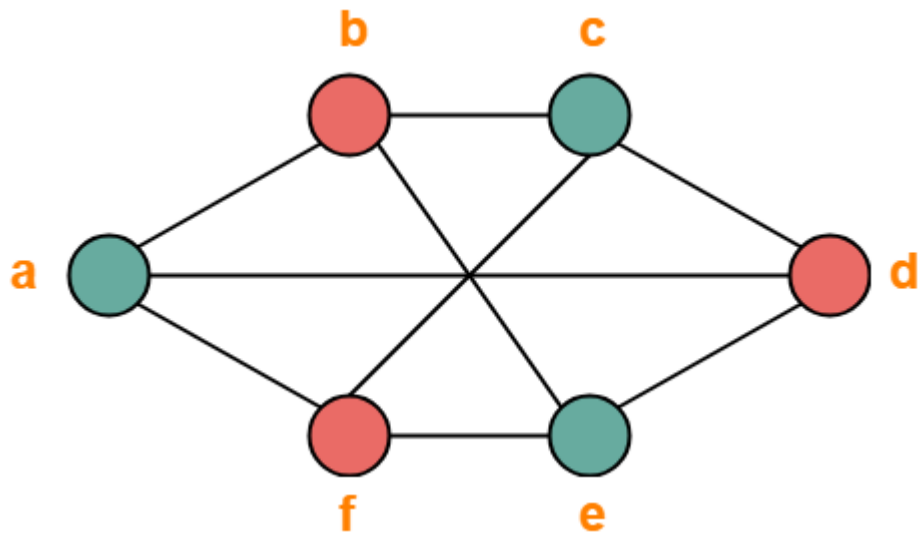


```

        if nodes[nbr]==i:
            return False
        return True
def colorGraph(color,src,nodes,graph):
    for i in color.keys():
        if checkValidColor(src,graph,nodes,i):
            nodes[src]=i
            for nbr in graph[src]:
                if nodes[nbr]==-1:
                    colorGraph(color,nbr,nodes,graph)
nodes,graph=inputGraph()
color={
    1: "Red",
    2: "Blue",
    3: "Yellow",
    4: "Green",
    5: "White",
    6: "Black",
}
src=input("Enter starting Node- ")
colorGraph(color,src,nodes,graph)
colorUsed=set()
for i in nodes.keys():
    print(i+" - "+color[nodes[i]])
    colorUsed.add(nodes[i],)
print(colorUsed)
print("Chromatic Number- " +str(len(colorUsed)) )

```

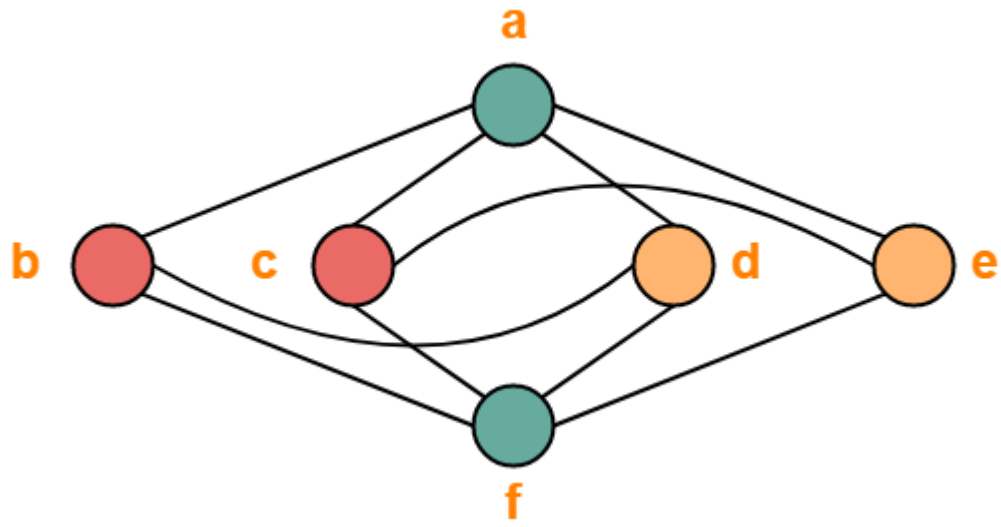
Input:



Output:

```
PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe" "c:/Users/SHUBHAM PERIWAL/Desktop/Untitled-1.py"
Enter the number of nodes: 6
Enter number of edges: 9
Enter Node 1 Name A
Enter Node 2 Name B
Enter Node 3 Name C
Enter Node 4 Name D
Enter Node 5 Name E
Enter Node 6 Name F
Edge 0 (from)- A
Edge 0 (to)- B
Edge 1 (from)- A
Edge 1 (to)- F
Edge 2 (from)- A
Edge 2 (to)- D
Edge 3 (from)- B
Edge 3 (to)- E
Edge 4 (from)- B
Edge 4 (to)- C
Edge 5 (from)- F
Edge 5 (to)- C
Edge 6 (from)- F
Edge 6 (to)- E
Edge 7 (from)- E
Edge 7 (to)- D
Edge 8 (from)- C
Edge 8 (to)- D
{'A': -1, 'B': -1, 'C': -1, 'D': -1, 'E': -1, 'F': -1}
{'A': ['B', 'F', 'D'], 'B': ['A', 'E', 'C'], 'C': ['B', 'F', 'D'], 'D': ['A', 'E', 'C'], 'E': ['B', 'F', 'D'], 'F': ['A', 'C', 'E']}
Enter starting Node- A
A - Black
B - White
C - Black
D - White
E - Black
F - White
{5, 6}
Chromatic Number- 2
PS C:\Users\SHUBHAM PERIWAL> █
```

Input:



Output:

```
PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe" "c:/Users/SHUBHAM PERIWAL/Desktop/Untitled-1.py"
Enter the number of nodes: 6
Enter number of edges: 10
Enter Node 1 Name A
Enter Node 2 Name B
Enter Node 3 Name C
Enter Node 4 Name D
Enter Node 5 Name E
Enter Node 6 Name F
Edge 0 (from)- B
Edge 0 (to)- A
Edge 1 (from)- B
Edge 1 (to)- F
Edge 2 (from)- B
Edge 2 (to)- D
Edge 3 (from)- A
Edge 3 (to)- C
Edge 4 (from)- A
Edge 4 (to)- D
Edge 5 (from)- A
Edge 5 (to)- E
Edge 6 (from)- C
Edge 6 (to)- E
Edge 7 (from)- C
Edge 7 (to)- F
Edge 8 (from)- D
Edge 8 (to)- F
Edge 9 (from)- F
Edge 9 (to)- E
{'A': -1, 'B': -1, 'C': -1, 'D': -1, 'E': -1, 'F': -1}
{'A': ['B', 'C', 'D', 'E'], 'B': ['A', 'F', 'D'], 'C': ['A', 'E', 'F'], 'D': ['B', 'A', 'F'],
'E': ['A', 'C', 'F'], 'F': ['B', 'C', 'D', 'E']}
Enter starting Node- B
A - White
B - Black
C - Black
D - Green
E - Green
F - White
{4, 5, 6}
Chromatic Number- 3
Shubham Periwai 7CSE-4Y
PS C:\Users\SHUBHAM PERIWAL> █
```

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 6

Date: 09-09-21

Aim: Write a Experiment to implement BFS for water jug problem using Python.

Software Used: Python 3.8

Theory: You are given a m liter jug and a n liter jug. Both the jugs are initially empty. The jugs don't have markings to allow measuring smaller quantities. You have to use the jugs to measure d liters of water where d is less than n.

(X, Y) corresponds to a state where X refers to amount of water in Jug1 and Y refers to amount of water in Jug2. Determine the path from initial state (xi, yi) to final state (xf, yf), where (xi, yi) is (0, 0) which indicates both Jugs are initially empty and (xf, yf) indicates a state which could be (0, d) or (d, 0).

The operations you can perform are:

1. Empty a Jug, (X, Y) \rightarrow (0, Y) Empty Jug 1
2. Fill a Jug, (0, 0) \rightarrow (X, 0) Fill Jug 1
3. Pour water from one jug to the other until one of the jugs is either empty or full, (X, Y) \rightarrow (X-d, Y+d)

Code:

```
class Waterjug:
    def __init__(self, am, bm, a, b, g):
        self.a_max = am;
        self.b_max = bm;
        self.a = a;
        self.b = b;
        self.goal = g;
    def emptyA(self):
        self.a = 0;
        print ('(', self.a, ',', self.b, ')', "- Applied RULE 1: Empty Jug 1")
    def emptyB(self):
        self.b = 0;
        print ('(', self.a, ',', self.b, ')', "- Applied RULE 2: Empty Jug 2")
    def fillA(self):
        self.a = self.a_max;
        print ('(', self.a, ',', self.b, ')', "- Applied RULE 3: Pour water in Jug 1")
    def fillB(self):
        self.b = self.b_max;
```

```

    print ('(', self.a, ',', self.b, ')', "- Applied RULE 4: Pour water in
Jug 2")
    def transferAtoB(self):
        while (True):
            self.a = self.a - 1
            self.b = self.b + 1
            if (self.a == 0 or self.b == self.b_max):
                break
        print ('(', self.a, ',', self.b, ')', "- Applied RULE 5: Pour water from
Jug 1 to Jug 2")
    def transferBtoA(self):
        while (True):
            self.a = self.a + 1
            self.b = self.b - 1
            if (self.b == 0 or self.a == self.a_max):
                break
        print ('(', self.a, ',', self.b, ')', "- Applied RULE 6: Pour water from
Jug 2 to Jug 1")
    def main(self):
        print("Initial State: (0,0)")
        print("Capacity: (",self.a_max,",",self.b_max,")")
        print("Required: (",self.goal,",",self.goal,")")
        print("\nStates:")
        while (True):
            if (self.a == self.goal or self.b == self.goal):
                break
            if (self.a == 0):
                self.fillA()
            elif (self.a > 0 and self.b != self.b_max):
                self.transferAtoB()
            elif (self.a > 0 and self.b == self.b_max):
                self.emptyB()
waterjug=Waterjug(4,3,0,0,2);
waterjug.main();

```

Output:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe"
Initial State: (0,0)
Capacity: ( 4 , 3 )
Required: ( 2 , y)

States:
( 4 , 0 ) - Applied RULE 3: Pour water in Jug 1
( 1 , 3 ) - Applied RULE 5: Pour water from Jug 1 to Jug 2
( 1 , 0 ) - Applied RULE 2: Empty Jug 2
( 0 , 1 ) - Applied RULE 5: Pour water from Jug 1 to Jug 2
( 4 , 1 ) - Applied RULE 3: Pour water in Jug 1
( 2 , 3 ) - Applied RULE 5: Pour water from Jug 1 to Jug 2
PS C:\Users\SHUBHAM PERIWAL> █
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe"
Initial State: (0,0)
Capacity: ( 5 , 2 )
Required: ( 1 , y)

States:
( 5 , 0 ) - Applied RULE 3: Pour water in Jug 1
( 3 , 2 ) - Applied RULE 5: Pour water from Jug 1 to Jug 2
( 3 , 0 ) - Applied RULE 2: Empty Jug 2
( 1 , 2 ) - Applied RULE 5: Pour water from Jug 1 to Jug 2
PS C:\Users\SHUBHAM PERIWAL> █
```


Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 7

Date: 10-09-21

Aim: Write a Experiment to implement DFS using Python.

Software Used: Python 3.8

Theory: The Depth-First Search is a recursive algorithm that uses the concept of backtracking. It involves thorough searches of all the nodes by going ahead if potential, else by backtracking. Here, the word backtrack means once you are moving forward and there are not any more nodes along the present path, you progress backward on an equivalent path to seek out nodes to traverse.

The only purpose of this algorithm is to visit all the vertex of the graph avoiding cycles.

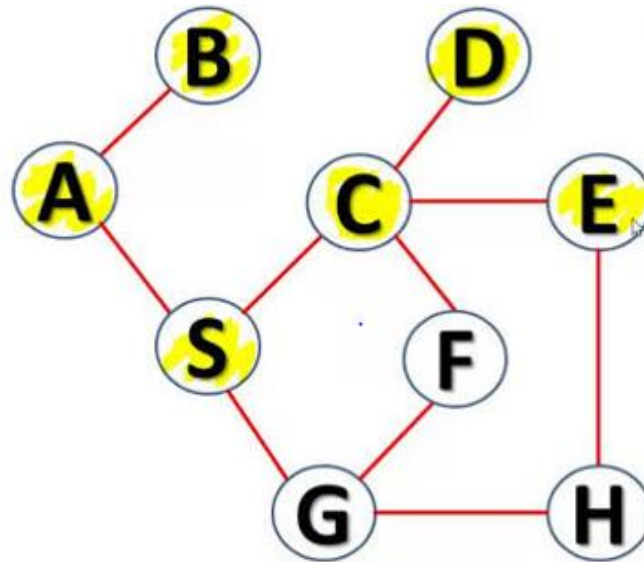
The DSF algorithm follows as:

- 1.We will start by putting any one of the graph's vertex on top of the stack.
- 2.After that take the top item of the stack and add it to the visited list of the vertex.
- 3.Next, create a list of that adjacent node of the vertex. Add the ones which aren't in the visited list of vertexes to the top of the stack.
- 4.Lastly, keep repeating steps 2 and 3 until the stack is empty.

Code:

```
graph = {'A' : ['B','C'], 'B' : ['D', 'E'], 'C' : ['F'], 'D' : [], 'E' : ['F'],
'F' : []}
visited = set()
def dfs(visited, graph, node):
    if node not in visited:
        print (node, "->", end=" ")
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)
print("Shubham Periwai 7CSE-4Y\n")
print("Graph:")
print(graph)
print("\nStarting Node: 'A'")
print("DFS Traversal: ")
dfs(visited, graph, 'A')
print("end")
```

Input:



Output:

```
PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe" "c:/Users/SHUBHAM PERIWAL/Desktop/Untitled-1.py"
Shubham Periwal 7CSE-4Y
```

Graph:

```
{'B': ['A'], 'A': ['B', 'S'], 'D': ['C'], 'C': ['B', 'E', 'S', 'F'], 'E': ['C', 'H'], 'F': ['C', 'G'], 'G': ['S', 'H'], 'S': ['G', 'C', 'A'], 'H': ['G', 'E']}
```

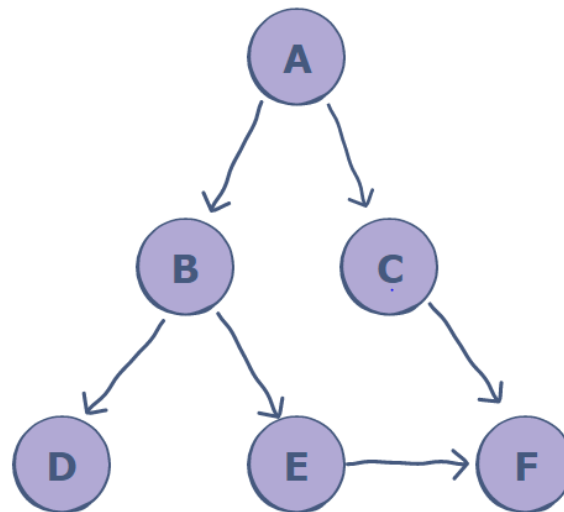
Starting Node: 'B'

DFS Traversal:

```
B -> A -> S -> G -> H -> E -> C -> F -> end
```

```
PS C:\Users\SHUBHAM PERIWAL> █
```

Input:



Output:

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

```
PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe" "c:/Users/Shubham Periwai 7CSE-4Y
```

```
Graph:
{'A': ['B', 'C'], 'B': ['D', 'E'], 'C': ['F'], 'D': [], 'E': ['F'], 'F': []}
```

Starting Node: 'A'

DFS Traversal:

A -> B -> D -> E -> F -> C -> end

```
PS C:\Users\SHUBHAM PERIWAL> █
```

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 8

Date: 17-09-21

Aim: Tokenization of word and Sentences with the help of NLTK package

Software Used: Python 3.8

Theory: Tokenization is one of the most common tasks when it comes to working with text data. Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.

Let's take an example. Consider the below string:

"This is a cat."

We get ['This', 'is', 'a', 'cat'].

There are numerous uses of doing this. We can use this tokenized form to:

- Count the number of words in the text
- Count the frequency of the word, that is, the number of times a particular word is present

Code:

```
import nltk

from nltk.tokenize import word_tokenize, sent_tokenize, WordPunctTokenizer

nltk.download('punkt')

text="Python is an interpreted high-level general-
purpose programming language. Its design philosophy emphasizes code readability with its use
of significant indentation. Its language constructs as well as its object-
oriented approach aim to help programmers write clear, logical code for small and large-
scale projects"

print("\n\nOutput of word tokenizer: \n",word_tokenize(text))


print("\n\nOutput of sentence tokenizer: \n",sent_tokenize(text))

print("\n\nOutput of punctuation tokenizer: \n",WordPunctTokenizer().tokenize(text))
```

Input:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Output:

 ShubhamPeriwal.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment Share Settings Profile

+ Code + Text

RAM Disk Editing

[1] import nltk
from nltk.tokenize import word_tokenize
nltk.download('punkt')

[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
True

[3] text="Our economy is badly weakened, a consequence of greed and irresponsibility on the part of some, but also our collective failure to make hard choices and prepare the nation for a
print("\nOutput of word tokenizer:\n",word_tokenize(text))

Output of word tokenizer:
['Our', 'economy', 'is', 'badly', 'weakened', ',', 'a', 'consequence', 'of', 'greed', 'and', 'irresponsibility', 'on', 'the', 'part', 'of', 'some', ',', 'but', 'also', 'our', 'collect

[4] from nltk.tokenize import sent_tokenize
print("\nOutput of sentence tokenizer:\n",sent_tokenize(text))

Output of sentence tokenizer:
['Our economy is badly weakened, a consequence of greed and irresponsibility on the part of some, but also our collective failure to make hard choices and prepare the nation for a new

from nltk.tokenize import WordPunctTokenizer
print("\nOutput of punctuation tokenizer:\n",WordPunctTokenizer().tokenize(text))

Output of punctuation tokenizer:
['Our', 'economy', 'is', 'badly', 'weakened', ',', 'a', 'consequence', 'of', 'greed', 'and', 'irresponsibility', 'on', 'the', 'part', 'of', 'some', ',', 'but', 'also', 'our', 'collect

0s completed at 12:57

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 9

Date: 24-09-21

Aim: Design an XOR truth table using Python

Software Used: Python 3.8

Theory: The Exclusive-OR Gate/ **XOR Gate** is a combination of all the [three basic gates](#) (NOT, AND, OR gates). It receives two or more input signals but produces only one output signal.

It results in a low '0' if the input bit pattern contains an even number of high '1' signals.

If there are an odd number of high '1' signals, it results in high. For this reason, the Exclusive-OR Gate(XOR Gate) is called the Anti-Coincidence Gate or Inequality Detector or Odd Ones Detector.



$$\begin{aligned} Y &= A \oplus B \\ &= (A \cdot B') + (A' \cdot B) \rightarrow 1 \\ &\quad \text{(or)} \\ &= (A + B) \cdot (A' + B') \end{aligned}$$

Code:

```
print("Shubham Perwal 7CSE-4Y")
con="y"
while(con=="y"):
    ch=int(input("\nEnter number of inputs (2/3): "))
    if ch==2:
        A=[0,1]
        B=[0,1]
        print("\nA", "B", "A.(~B)", "(~A).B", "A^B", sep=" ")
        for i in range(2):
            for j in range(2):
                print(str(A[i]), " " + str(B[j]), " " + str(A[i]&(~B[j])), " "
                    + str((~A[i])&B[j]), " " + str(A[i]^B[j]))
            )
        elif ch==3:
            A=[0,1]
```

```

B=[0,1]
C=[0,1]
print("\nA","B","C","ABC","A'B'C","A'BC'","AB'C'","A^B^C",sep=" ")
for i in range(2):
    for j in range(2):
        for k in range(2):
            print(str(A[i])," "+str(B[j])," "+str(C[k]),"
"+str(A[i]&B[j]&C[k])," "+str((~A[i])&(~B[j])&C[k]),"
"+str((~A[i])&B[j]&(~C[k])), " "
+str(A[i]&(~B[j])&(~C[k])), " "+str(A[i]^B[j]^C[k]))
con=input("\nContinue (y/n): ")

```

Input:

A,B,C

Output:

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\SHUBHAM PERIWAL> & "C:/Users/SHUBHAM PERIWAL/AppData/Local/Programs/Python/Python36/python.exe"
Shubham Periwai 7CSE-4Y

Enter number of inputs (2/3): 2

A   B   A.(~B)   (~A).B   A^B
0   0       0       0       0
0   1       0       1       1
1   0       1       0       1
1   1       0       0       0

Continue (y/n): y

Enter number of inputs (2/3): 3

A   B   C   ABC   A'B'C   A'BC'   AB'C'   A^B^C
0   0   0   0     0     0     0     0
0   0   1   0     1     0     0     1
0   1   0   0     0     1     0     1
0   1   1   0     0     0     0     0
1   0   0   0     0     0     1     1
1   0   1   0     0     0     0     0
1   1   0   0     0     0     0     0
1   1   1   1     0     0     0     1

Continue (y/n): 

```

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |

Experiment 10

Date: 1-10-21

Aim: Study of SCIKIT fuzzy

Software Used: Python 3.8

Problem Statement

Controlling the ore amount in the field of mineral processing using Fuzzy Control System.

Abstract

To get a solution regarding the challenge of managing ore amount in the field of mineral processing. Firstly, we will obtain a data on influencing factors, using the NumPy module library. The fuzzy control method designed using the SciKit Fuzzy module library screens out factors with a bigger influence and then selects them as the input. Then a motor frequency is generated as the output.

The fuzzy control variable's range of values is defined, the fuzzy membership function is constructed, and the fuzzy control rule is set up. Finally, the mine's fuzzy control is realized by activating the fuzzy controller. The results show that the hardness of the raw ore and the weight of the belt scale are the most important parameters in controlling the amount of ore, and they may both be employed as input variables in the fuzzy control system. The fuzzy controller created with the SciKit Fuzzy module library can be used in mineral processing.

Method Used

The influence of numerous parameters on the amount of ore is analyzed using a linear regression model created using the NumPy module library, and the most influential elements are chosen as input variables for the fuzzy control process.

The fuzzy inference method is then carried out using the SciKit Fuzzy module library.

The control variables are generated as the output after fuzzy rules and fuzzy sets are constructed and defuzzified.

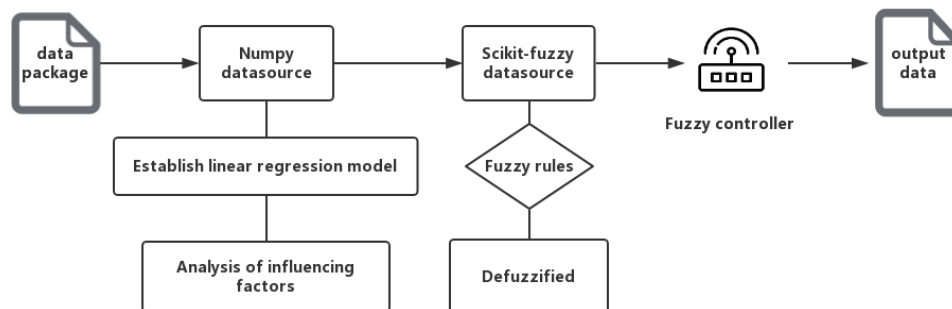


Fig. 1 Development of fuzzy control system for constant feeding by Python

Fuzzy control system based on SciKit fuzzification

Figure 2 depicts the expected use of SciKit Fuzzy in the fuzzy control system. The range and membership function for the fuzzy input variables are defined first. When the fuzzy control rules are defined, the input will be decomposed into different fuzzy sets. The data is then defuzzified, and the motor frequency is output to meet the goal of controlling the mining machine's motor.

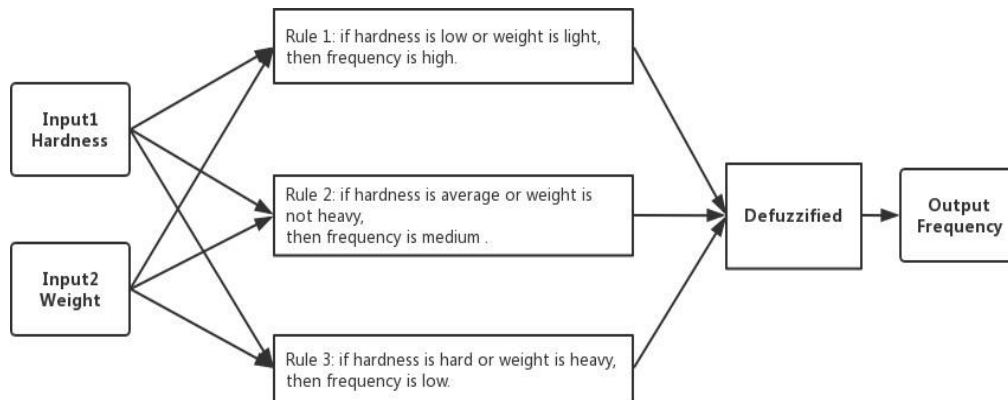


Fig. 2 Fuzzy reasoning process

Fuzzification

The term "fuzzification" refers to the process of determining the membership value of each input variable by comparing it to the membership function.

Conclusions

The SciKit Fuzzy module library was used to overcome the problem of various input and output systems in the beneficiation process. The intelligent fuzzy control constant feeding system can select the appropriate feeding rate based on various ore properties and automatically adjust the frequency of the feeding machine motor based on the weight of the ore conveyed, thereby increasing the lower mill's operating efficiency. This has a favorable effect on the factory's output and economic indices. In conclusion, SciKit Fuzzy make it easier to create intelligent systems for mineral processing and raise the industry's intelligence level.

Internal Assessment (Mandatory Experiment) Sheet for Lab Experiment
Department of Computer Science & Engineering
Amity University, Noida (UP)

| | | | |
|--------------------|-----------------|----------------|-------------------------|
| Experiment | B. Tech CSE | Course Name | Artificial Intelligence |
| Course Code | [CSE 401] | Semester | 7 |
| Student Name | Shubham Periwal | Enrollment No. | A2305218241 |
| Marking Criteria | | | |
| Criteria | Total Marks | Marks Obtained | Comments |
| Concept (A) | 2 | | |
| Implementation (B) | 2 | | |
| Performance (C) | 2 | | |
| Total | 6 | | |