Roll No: TCS2324012

SIES College of Arts, Science & Commerce
RISE WITH EDUCATION
NAAC REACCREDITED - 'A' GRADE
ISO 9001 : 2008

S.I.E.S College of Arts, Science and Commerce,

Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that Mr./ Miss. Vinay Srinivas Eligeti has successfully completed the necessary course of experiments in the subject of **Artificial Intelligence** during the academic year **2023– 2024** complying with the requirements of **University of Mumbai**, for the course of **T.Y.BSc. Computer Science [Semester-V]**

Head of the Department
(Computer Science)

-Dr. Manoj Singh

Prof. In-Charge

Mrs. Shivani Deopa

Date: 1th September 2023

College Seal

Index Page

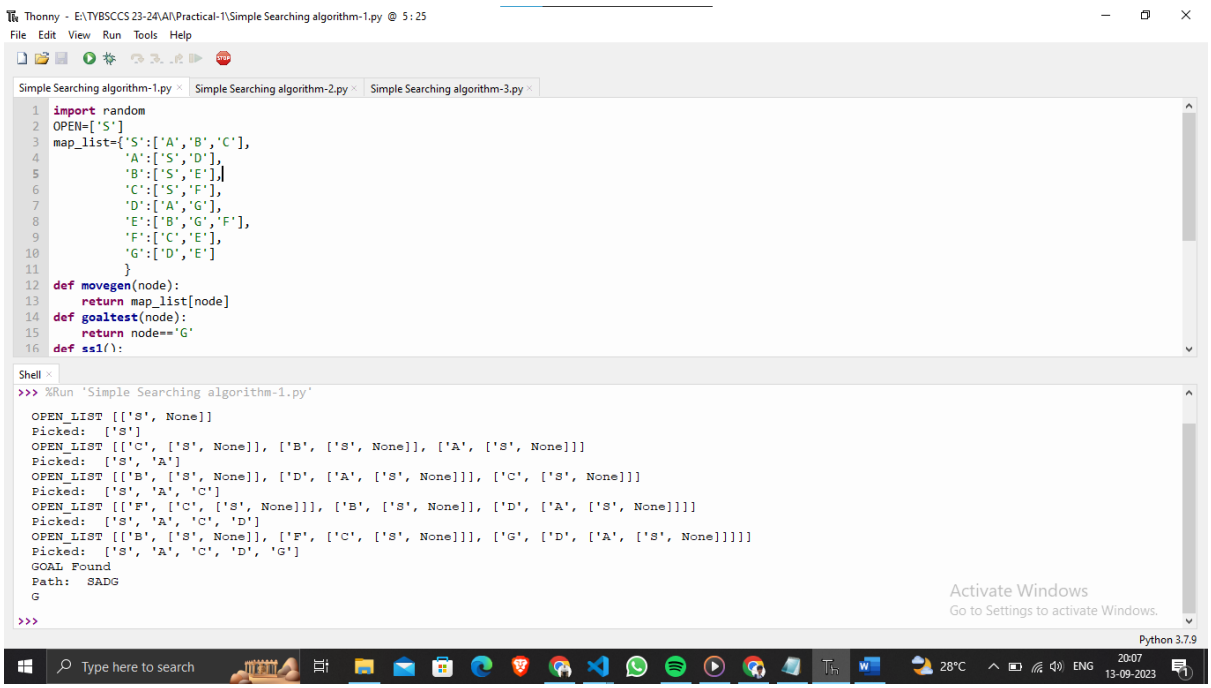
Practical-1

Implementing Simple Search 1 Algorithm.

Code snippet:

```
import random
OPEN=['S']
map_list={'S':['A','B','C'],
          'A':['S','D'],
          'B':['S','E'],
          'C':['S','F'],
          'D':['A','G'],
          'E':['B','G','F'],
          'F':['C','E'],
          'G':['D','E']
          }
def movegen(node):
    return map_list[node]
def goaltest(node):
    return node=='G'
def ss1():
    while len(OPEN)>0:
        random.shuffle(OPEN)
        N=OPEN.pop()
        if goaltest(N):
            return "Found"
        else:
            n=movegen(N)
            for i in n:
                if n not in OPEN:
                    OPEN.append(i)
                print("OPEN_LIST",OPEN)
    return "NOT FOUND"
print(ss1())
```

Output:


Thonny - E:\TYBSCCS 23-24\AI\Practical-1\Simple Searching algorithm-1.py @ 5 : 25
File Edit View Run Tools Help
Simple Searching algorithm-1.py
Simple Searching algorithm-2.py
Simple Searching algorithm-3.py
import random
OPEN=['S']
map_list={'S':['A','B','C'],
'A':['S','D'],
'B':['S','E'],
'C':['S','F'],
'D':['A','G'],
'E':['B','G','F'],
'F':['C','E'],
'G':['D','E']
}
def movegen(node):
return map_list[node]
def goaltest(node):
return node=='G'
def ss1():
Shell
>>> %Run 'Simple Searching algorithm-1.py'
OPEN_LIST [['S', None]]
Picked: ['S']
OPEN_LIST [['C', ['S', None]], ['B', ['S', None]], ['A', ['S', None]]]
Picked: ['S', 'A']
OPEN_LIST [['B', ['S', None]], ['D', ['A', ['S', None]]], ['C', ['S', None]]]
Picked: ['S', 'A', 'C']
OPEN_LIST [['F', ['C', ['S', None]]], ['B', ['S', None]], ['D', ['A', ['S', None]]]]
Picked: ['S', 'A', 'C', 'D']
OPEN_LIST [['B', ['S', None]], ['F', ['C', ['S', None]]], ['G', ['D', ['A', ['S', None]]]]]
Picked: ['S', 'A', 'C', 'D', 'G']
GOAL Found
Path: SADG
G
>>>
Activate Windows
Go to Settings to activate Windows.
Python 3.7.9
Type here to search
28°C
ENG
20:07
13-09-2023

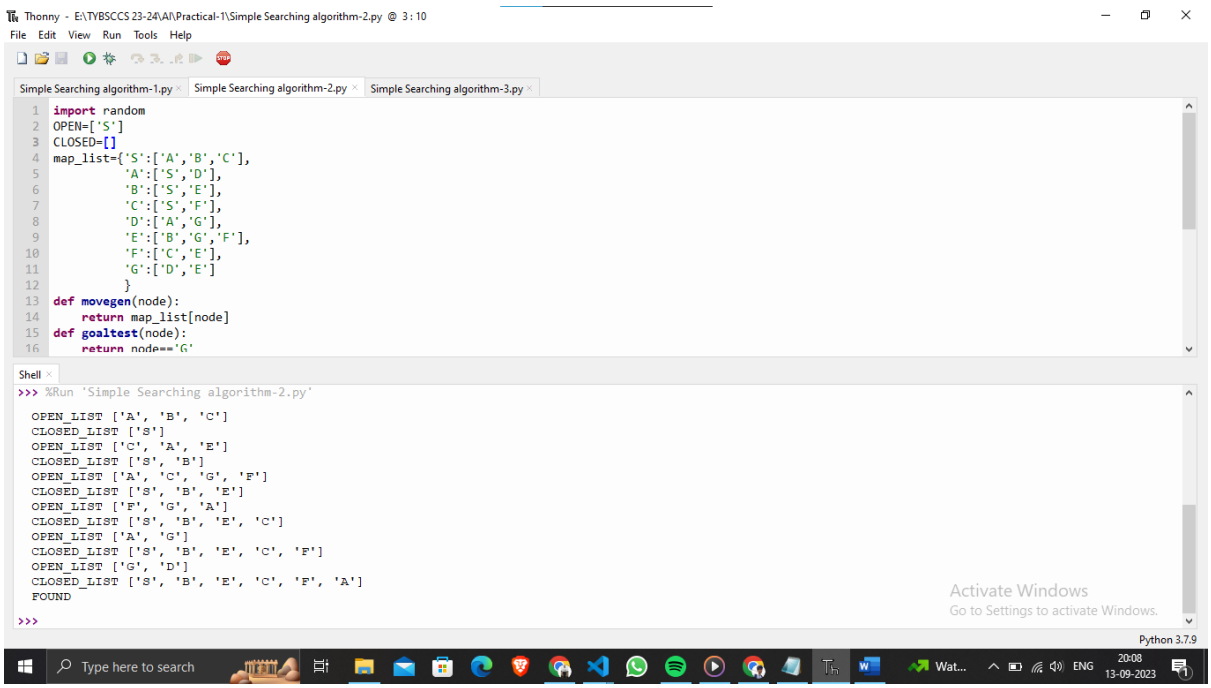
Practical-2

Implementing Simple Search 2 Algorithm.

Code snippet:

```
import random
OPEN=['S']
CLOSED=[]
map_list={'S':['A','B','C'],
          'A':['S','D'],
          'B':['S','E'],
          'C':['S','F'],
          'D':['A','G'],
          'E':['B','G','F'],
          'F':['C','E'],
          'G':['D','E']
          }
def movegen(node):
    return map_list[node]
def goaltest(node):
    return node=='G'
def ss2():
    while len(OPEN)>0:
        random.shuffle(OPEN)
        N=OPEN.pop()
        CLOSED.append(N)
        if goaltest(N):
            return "FOUND"
        else:
            n=movegen(N)
            for i in n:
                if i not in CLOSED and i not in OPEN:
                    OPEN.append(i)
            print("OPEN_LIST",OPEN)
            print("CLOSED_LIST",CLOSED)
    return "NOT FOUND"
print(ss2())
```

Output:


Thonny - E:\TYBSCCS 23-24\AI\Practical-1\Simple Searching algorithm-2.py @ 3 : 10
File Edit View Run Tools Help
Simple Searching algorithm-1.py
Simple Searching algorithm-2.py
Simple Searching algorithm-3.py
import random
OPEN=['S']
CLOSED=[]
map_list={'S':['A','B','C'],
'A':['S','D'],
'B':['S','E'],
'C':['S','F'],
'D':['A','G'],
'E':['B','G','F'],
'F':['C','E'],
'G':['D','E']
}
def movegen(node):
return map_list[node]
def goaltest(node):
return node=='G'
Shell
>>> %Run 'Simple Searching algorithm-2.py'
OPEN_LIST ['A', 'B', 'C']
CLOSED_LIST ['S']
OPEN_LIST ['C', 'A', 'E']
CLOSED_LIST ['S', 'B']
OPEN_LIST ['A', 'C', 'G', 'F']
CLOSED_LIST ['S', 'B', 'E']
OPEN_LIST ['F', 'G', 'A']
CLOSED_LIST ['S', 'B', 'E', 'C']
OPEN_LIST ['A', 'G']
CLOSED_LIST ['S', 'B', 'E', 'C', 'F']
OPEN_LIST ['G', 'D']
CLOSED_LIST ['S', 'B', 'E', 'C', 'F', 'A']
FOUND
>>>
Activate Windows
Go to Settings to activate Windows.
Python 3.7.9
Type here to search
ENG
20:08
13-09-2023

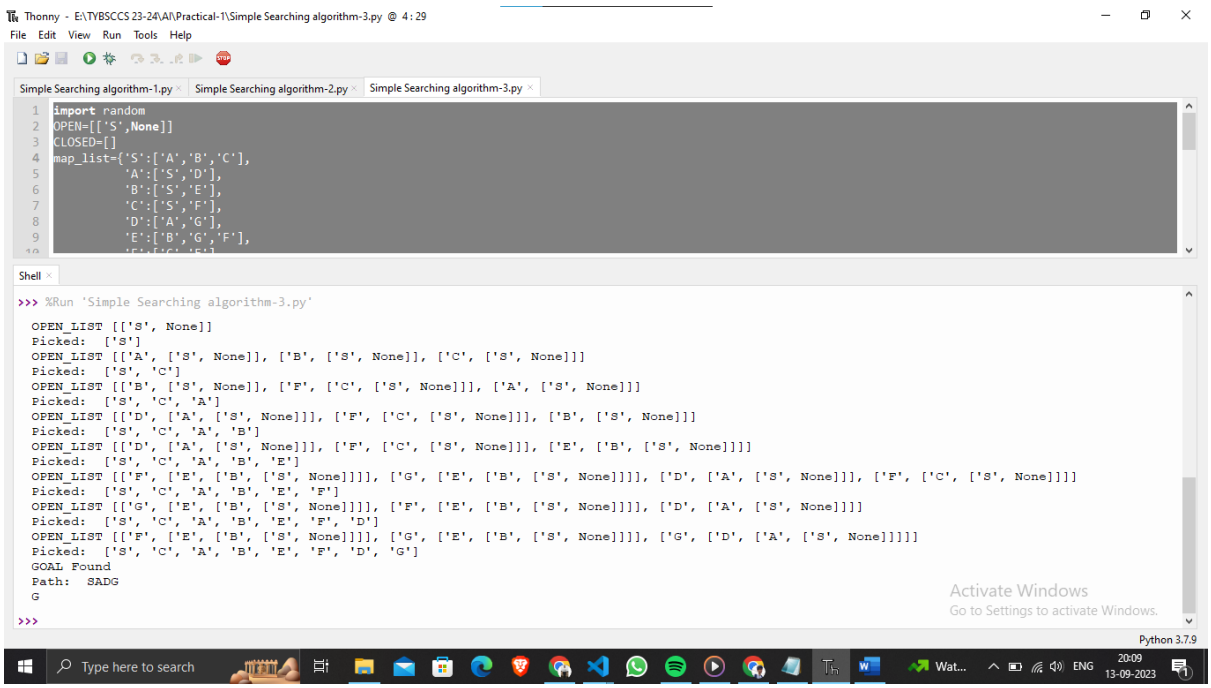
Practical-3

Implementing Simple Search 3 Algorithm.

Code snippet:

```
import random
OPEN=[['S',None]]
CLOSED=[]
map_list={'S':['A','B','C'],
          'A':['S','D'],
          'B':['S','E'],
          'C':['S','F'],
          'D':['A','G'],
          'E':['B','G','F'],
          'F':['C','E'],
          'G':['D','E']
          }
def movegen(node):
    return map_list[node]
def goaltest(node):
    return node=='G'
def returnpath(path):
    if path is not None:
        return str(path[0])+returnpath(path[1])
    else:
        return ""
def ss3():
    while len(OPEN)>0:
        random.shuffle(OPEN)
        print("OPEN_LIST",OPEN)
        M=OPEN.pop()
        N=M[0]
        CLOSED.append(N)
        print("Picked: ",CLOSED)
        if goaltest(N):
            print("GOAL Found")
            print("Path: ",returnpath(M)[::-1])
            return N
        else:
            neigh=movegen(N)
            for node in neigh:
                if node not in CLOSED and node not in OPEN:
                    new_list=[node,M]
                    OPEN.append(new_list)
    print("Not Found")
print(ss3())
```

Output:


Thonny - E:\TYBSCCS 23-24\AI\Practical-1\Simple Searching algorithm-3.py @ 4 : 29
File Edit View Run Tools Help
Simple Searching algorithm-1.py
Simple Searching algorithm-2.py
Simple Searching algorithm-3.py
import random
OPEN=[['S',None]]
CLOSED=[]
map_list={'S':['A','B','C'],
'A':['S','D'],
'B':['S','E'],
'C':['S','F'],
'D':['A','G'],
'E':['B','G','F'],
Shell
>>> %Run 'Simple Searching algorithm-3.py'
OPEN_LIST [['S', None]]
Picked: ['S']
OPEN_LIST [['A', ['S', None]], ['B', ['S', None]], ['C', ['S', None]]]
Picked: ['S', 'C']
OPEN_LIST [['B', ['S', None]], ['F', ['C', ['S', None]]], ['A', ['S', None]]]
Picked: ['S', 'C', 'A']
OPEN_LIST [['D', ['A', ['S', None]]], ['F', ['C', ['S', None]]], ['B', ['S', None]]]
Picked: ['S', 'C', 'A', 'B']
OPEN_LIST [['D', ['A', ['S', None]]], ['F', ['C', ['S', None]]], ['E', ['B', ['S', None]]]]
Picked: ['S', 'C', 'A', 'B', 'E']
OPEN_LIST [['F', ['E', ['B', ['S', None]]]], ['G', ['E', ['B', ['S', None]]]], ['D', ['A', ['S', None]]], ['F', ['C', ['S', None]]]]
Picked: ['S', 'C', 'A', 'B', 'E', 'F']
OPEN_LIST [['G', ['E', ['B', ['S', None]]]], ['F', ['E', ['B', ['S', None]]]], ['D', ['A', ['S', None]]]]
Picked: ['S', 'C', 'A', 'B', 'E', 'F', 'D']
OPEN_LIST [['F', ['E', ['B', ['S', None]]]], ['G', ['E', ['B', ['S', None]]]], ['G', ['D', ['A', ['S', None]]]]]
Picked: ['S', 'C', 'A', 'B', 'E', 'F', 'D', 'G']
GOAL Found
Path: SADG
G
>>>
Activate Windows
Go to Settings to activate Windows.
Python 3.7.9
Type here to search
ENG
20:09
13-09-2023

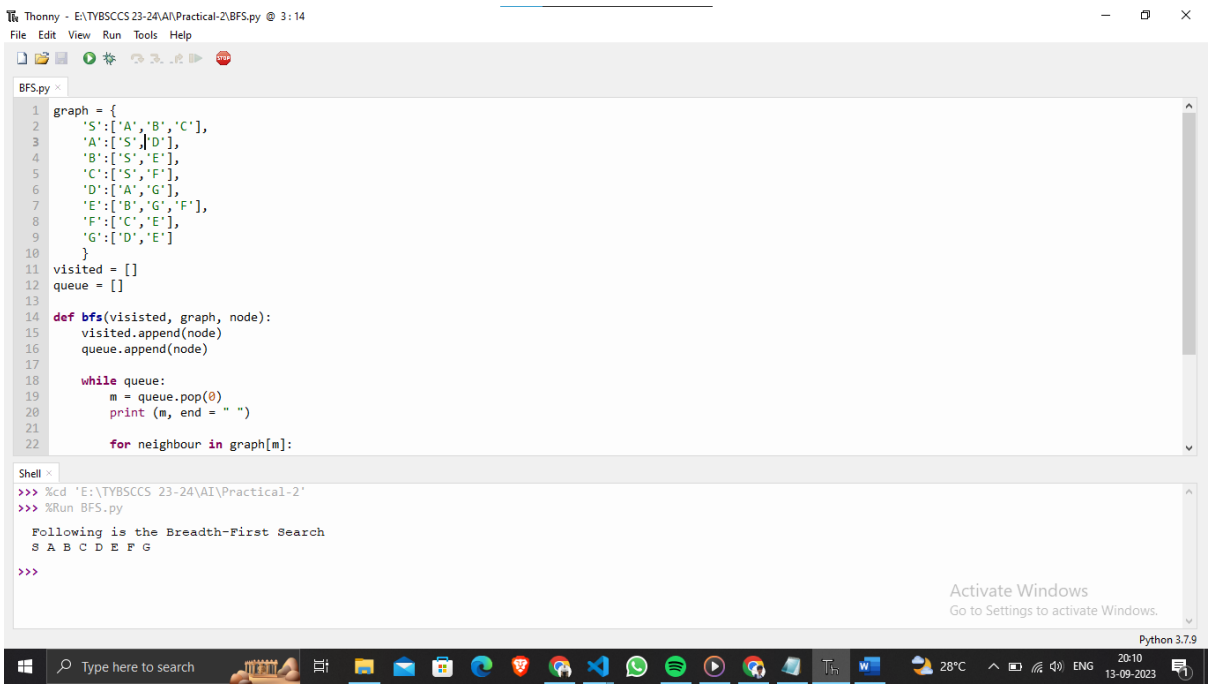
Practical-4

Implement Breadth first search algorithm for map problem.

Code snippet:

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

def bfs(visisted, graph, node):
    visited.append(node)
    queue.append(node)

    while queue:
        m = queue.pop(0)
        print (m, end = " ")

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

print("Following is the Breadth-First Search")
bfs(visited, graph, 'S')
```

Output:


Thonny - E:\TYBSCCS 23-24\AI\Practical-2\BFS.py @ 3 : 14
File Edit View Run Tools Help
BFS.py
graph = {
'S':['A','B','C'],
'A':['S','D'],
'B':['S','E'],
'C':['S','F'],
'D':['A','G'],
'E':['B','G','F'],
'F':['C','E'],
'G':['D','E']
}
visited = []
queue = []
def bfs(visisted, graph, node):
visited.append(node)
queue.append(node)
while queue:
m = queue.pop(0)
print (m, end = " ")
for neighbour in graph[m]:
Shell
>>> %cd 'E:\TYBSCCS 23-24\AI\Practical-2'
>>> %Run BFS.py
Following is the Breadth-First Search
S A B C D E F G
>>>
Activate Windows
Go to Settings to activate Windows.
Python 3.7.9
Type here to search
28°C
ENG
20:10
13-09-2023

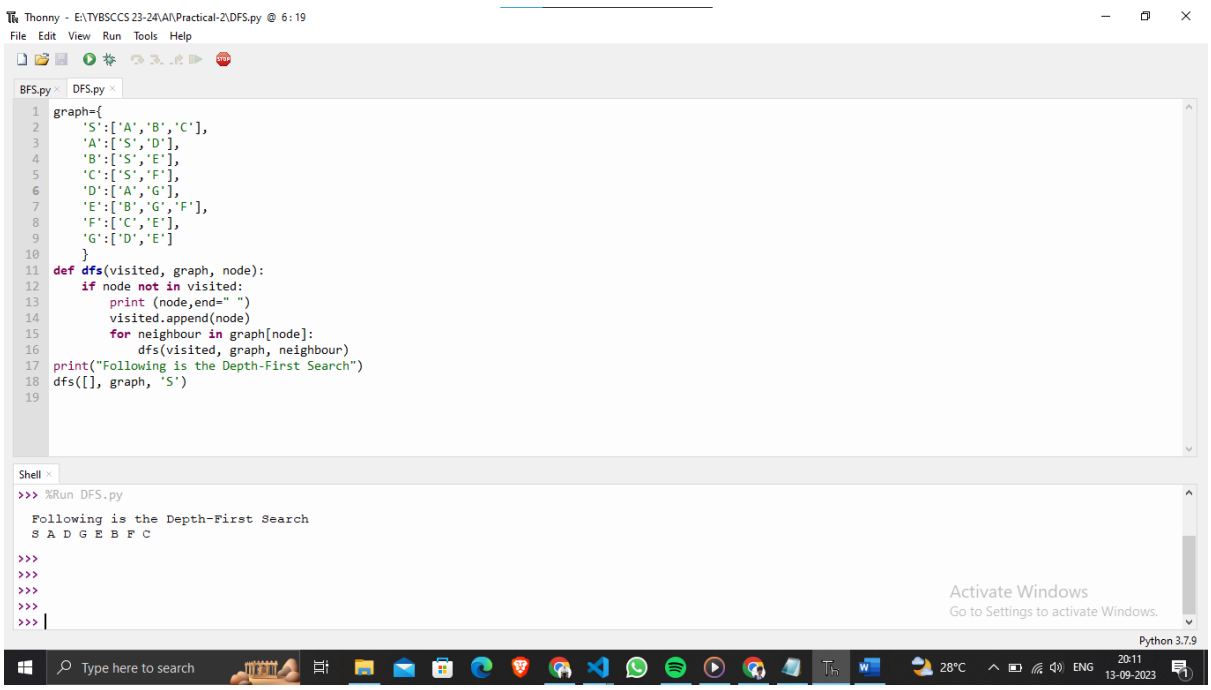
Practical-5

Implement Iterative deep depth first search for map problem.

Code snippet:

```
graph={
    'S':['A','B','C'],
    'A':['S','D'],
    'B':['S','E'],
    'C':['S','F'],
    'D':['A','G'],
    'E':['B','G','F'],
    'F':['C','E'],
    'G':['D','E']
    }
def dfs(visited, graph, node):
    if node not in visited:
        print (node,end=" ")
        visited.append(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)
print("Following is the Depth-First Search")
dfs([], graph, 'S')
```

Output:



Practical-6

Implement A* search algorithm for Romanian map problem.

Code snippet:

```
#A star
nodeList={'mumbai':[("delhi",1200),("nasik",350),("goa",800),("pune",130)],
          "delhi":[("nasik",375),("mumbai",1200)],
          "nasik":[("indore",600),("delhi",375),("mumbai",350),("nagpur",600)]
,
          "indore":[("nasik",400)],
          "nagpur":[("nasik",600),("pune",450)],
          "pune":[("mumbai",130),("nagpur",450),("blore",550)],
          "blore":[("hyd",110),("goa",750)],
          "goa":[("blore",750),("hyd",850),("mumbai",800)],
          "hyd":[("blore",750),("goa",750)]}
hd={"mumbai":790,"delhi":1515,"nasik":1140,"indore":1540,"nagpur":1110,"pune":
660,"blore":110,"goa":850,"hyd":0}
openList=[("mumbai",700)]
closedList=[]
def goalTest(node):
    return node=="hyd"
def moveGen(node):
    return nodeList[node[0]]
def sort(mylist):
    for i in range(len(mylist)):
        for j in range(0,len(mylist)-i-1):
            if (mylist[j][1]>mylist[j+1][1]):
                temp=mylist[j]
                mylist[j]=mylist[j+1]
                mylist[j+1]=temp
    return (mylist)
def AStar():
    while(len(openList)>0):
        sort(openList)
        print("Open List Contains:",openList)
        node=openList.pop(0)
        closedList.append((node[0],hd[node[0]]))
        print("Picked node:",node)
        if (goalTest(node[0])):
            return "Goal Found"
        else:
            neighbours=moveGen(node)
            print("Neighbours of ",node," are: ",neighbours)
            for node in neighbours:
                if node not in openList and node[0] not in closedList[0]:
                    tup=(node[0],(node[1]+hd[node[0]]))
                    openList.append(tup)
```

```
    return "Goal Not Found"
AStar()
'''
Goal='hyd'
finding hd->>selecting mumbai
hd1=mumbai->delhi(1200)->nasik(375)->indore(600)->nasik(375)loop hence
discarded
hd2=mumbai->nasik(350)->discarded
hd3=mumbai->goa(800)->blore(750)->hyd
hd4=
hd5=
hd6=
'''
```

Output:

Practical-7

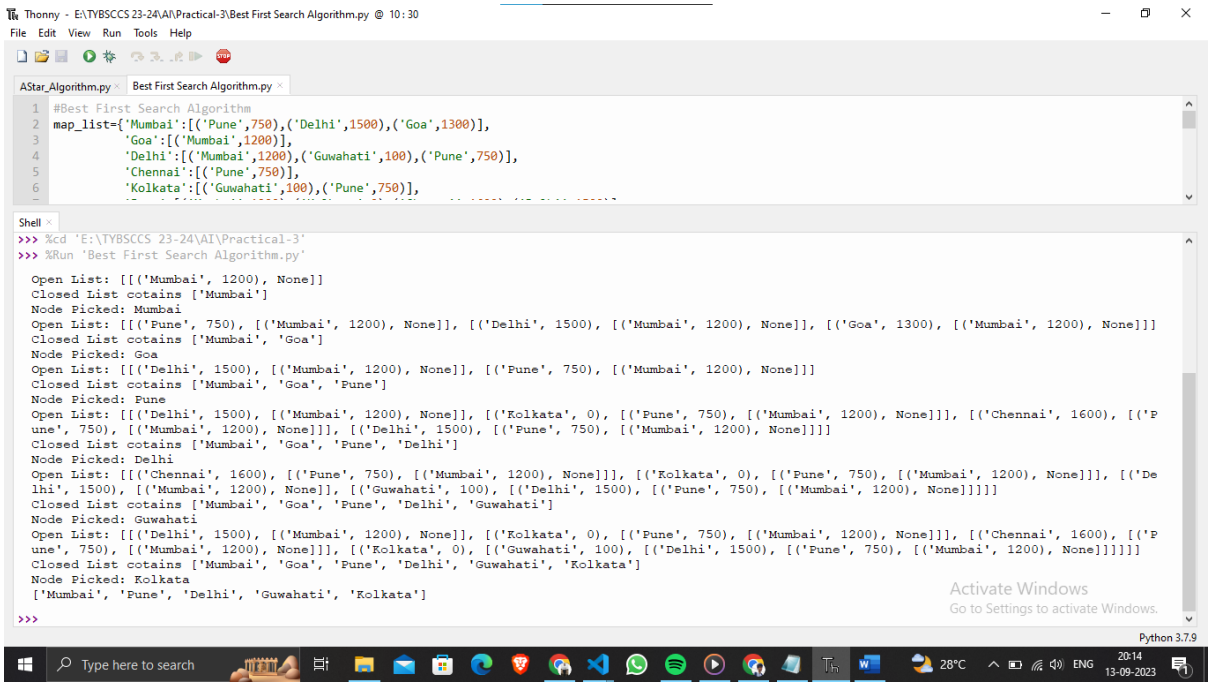
Implement recursive best-first search algorithm for Romanian map problem.

Code snippet:

```
#Best First Search Algorithm
map_list={'Mumbai':[('Pune',750),('Delhi',1500),('Goa',1300)],
          'Goa':[('Mumbai',1200)],
          'Delhi':[('Mumbai',1200),('Guwahati',100),('Pune',750)],
          'Chennai':[('Pune',750)],
          'Kolkata':[('Guwahati',100),('Pune',750)],
          'Pune':[('Mumbai',1200),('Kolkata',0),('Chennai',1600),('Delhi',1500
)],
          'Guwahati':[('Delhi',1500),('Kolkata',0)]
          }
OPEN=[[('Mumbai',1200),None]]
CLOSED=[]
def movegen(node):
    return map_list[node]
def goaltest(node):
    return node=='Kolkata'
final=[]
def reconstructpath(path):
    if path is None:
        return ""
    else:
        final.append(path[0][0])
        reconstructpath(path[1])
        return final
def sort(a):
    for i in range(len(a)):
        for j in range(0,len(a)-i-1):
            if ((a[j][0][1]),a[j+1][0][1]):
                (a[j],a[j+1])=(a[j+1],a[j])
    return a
def best():
    while len(OPEN)>0:
        print("Open List:",OPEN)
        x=sort(OPEN)
        seen=x.pop(0)
        N=seen[0][0]
        CLOSED.append(N)
        print("Closed List cotains",CLOSED)
        print("Node Picked:",N)
        if goaltest(N):
            print(reconstructpath(seen)[::-1])
            return "Found"
```

```
        else:
            neigh=movegen(N)
            for i in neigh:
                if i[0] not in CLOSED and i not in OPEN:
                    new=[i,seen]
                    OPEN.append(new)
    return "Not Found"
best()
```

Output:



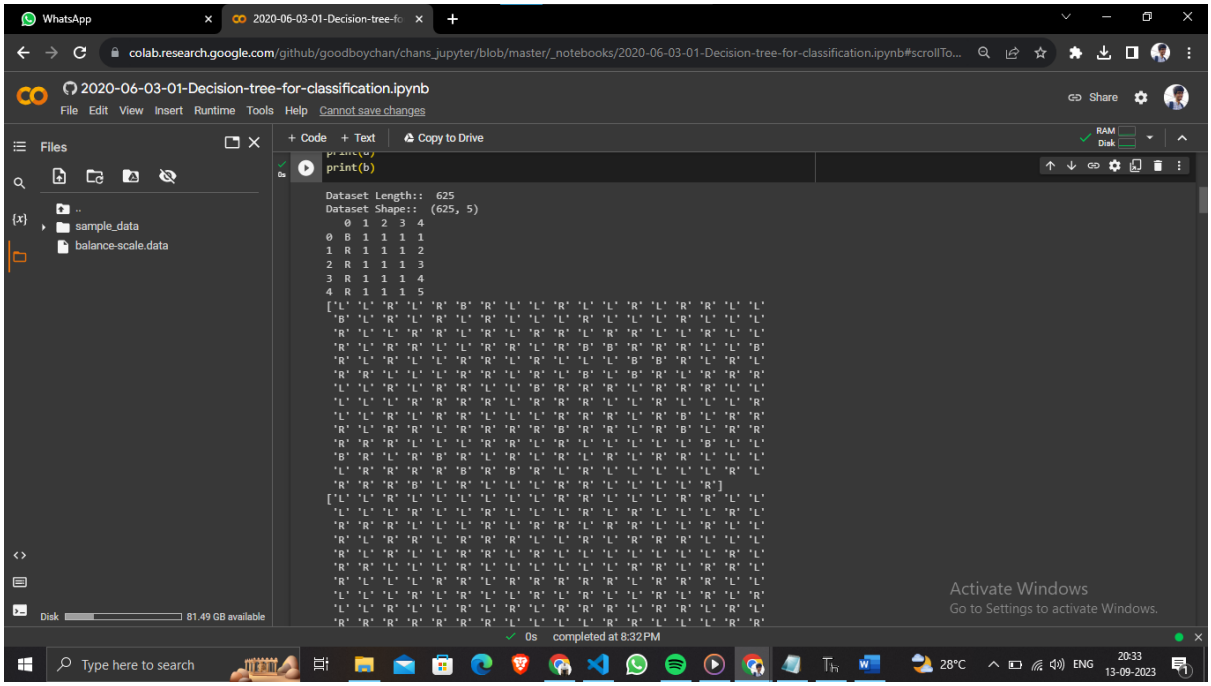
Practical-8

Implement decision tree learning algorithm.

Code snippet:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
balance_data=pd.read_csv("balance-scale.data",sep=",",header=None)
print("Dataset Length:: ",len(balance_data))
print("Dataset Shape:: ",balance_data.shape)
print(balance_data.head())
X=balance_data.values[:, 1:5]#predicator
Y=balance_data.values[:,0]#target attr (B,R,L)
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=0.4,random_state=
100)
clf_entropy=DecisionTreeClassifier(criterion="entropy",random_state=100,max_de
pth=3,min_samples_leaf=5)
clf_entropy.fit(X_train,y_train)
clf_gini=DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=3
,min_samples_leaf=5)
clf_gini.fit(X_train,y_train)
print(y_test)
y_pred_en=clf_entropy.predict(X_test)
y_pred_gini=clf_gini.predict(X_test)
print(y_pred_en)
print(y_pred_gini)
a=accuracy_score(y_pred_en,y_test)*100
b=accuracy_score(y_pred_gini,y_test)*100
print(a)
print(b)
```

Output:


WhatsApp
2020-06-03-01-Decision-tree-fo
colab.research.google.com/github/goodboychan/chans_jupyter/blob/master/_notebooks/2020-06-03-01-Decision-tree-for-classification.ipynb#scrollTo...
2020-06-03-01-Decision-tree-for-classification.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
Files
sample_data
balance-scale.data
+ Code + Text Copy to Drive
RAM
Disk
print(b)
Dataset Length:: 625
Dataset Shape:: (625, 5)
0 1 2 3 4
0 B 1 1 1 1
1 R 1 1 1 2
2 R 1 1 1 3
3 R 1 1 1 4
4 R 1 1 1 5
Disk 81.49 GB available
Activate Windows
Go to Settings to activate Windows.
0s completed at 8:32PM
Type here to search
28°C
ENG
20:33
13-09-2023

Practical-9

To Implement Support Vector Machine Algorithm in Python.

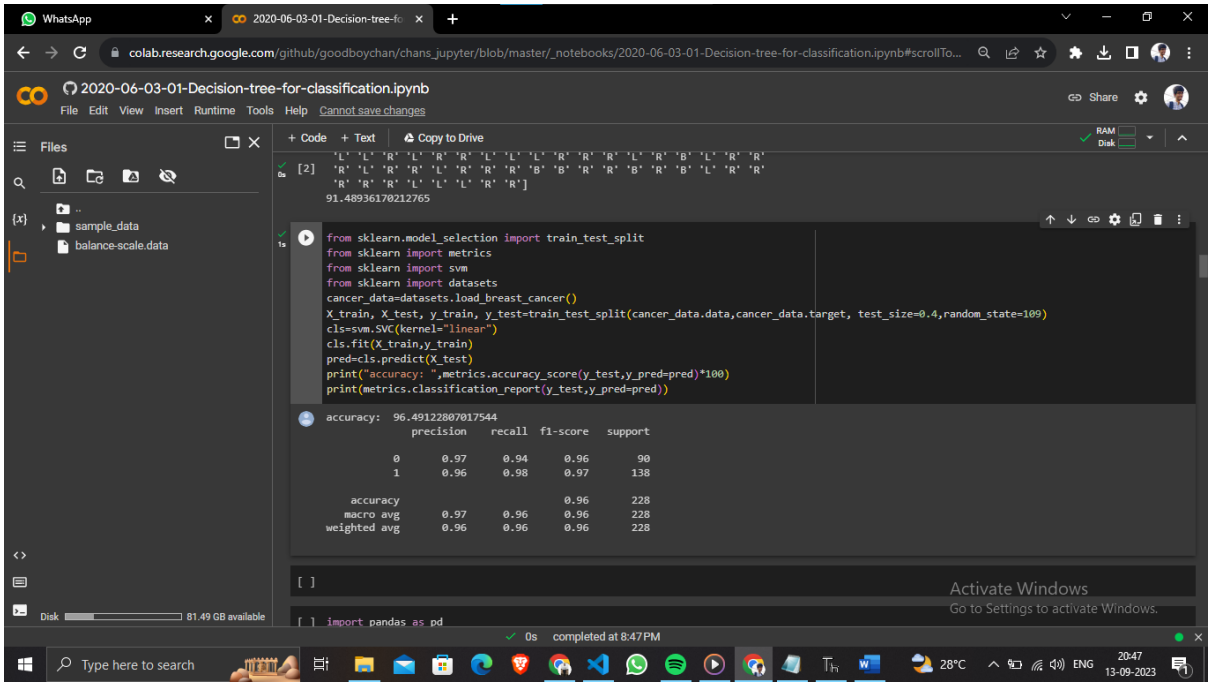
Code snippet:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn import datasets
balance_data=pd.read_csv('balance-scale.data',sep=",",header=None)
print("Dataset Length:: ",len(balance_data))
print("Dataset Shape:: ",balance_data.shape)
print(balance_data.head())
X=balance_data.values[:,1:5]
Y=balance_data.values[:,0]
X_train, X_test, y_train,
y_test=train_test_split(X,Y,test_size=0.3,random_state=100)
svclassifier=SVC(kernel="linear")
svclassifier.fit(X_train,y_train)
print(y_test)
y_pred=svclassifier.predict(X_test)
print(y_pred)
print(accuracy_score(y_pred,y_test)*100)
```

Or

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import svm
from sklearn import datasets
cancer_data=datasets.load_breast_cancer()
X_train, X_test, y_train,
y_test=train_test_split(cancer_data.data,cancer_data.target,
test_size=0.4,random_state=109)
cls=svm.SVC(kernel="linear")
cls.fit(X_train,y_train)
pred=cls.predict(X_test)
print("accuracy: ",metrics.accuracy_score(y_test,y_pred=pred)*100)
print(metrics.classification_report(y_test,y_pred=pred))
```

Output:


2020-06-03-01-Decision-tree-for-classification.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
Files
sample_data
balance-scale.data
+ Code + Text Copy to Drive
91.48936170212765
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn import svm
from sklearn import datasets
cancer_data=datasets.load_breast_cancer()
X_train, X_test, y_train, y_test=train_test_split(cancer_data.data,cancer_data.target, test_size=0.4,random_state=109)
cls=svm.SVC(kernel="linear")
cls.fit(X_train,y_train)
pred=cls.predict(X_test)
print("accuracy: ",metrics.accuracy_score(y_test,y_pred=pred)*100)
print(metrics.classification_report(y_test,y_pred=pred))
accuracy: 96.49122807017544
precision recall f1-score support
0 0.97 0.94 0.96 90
1 0.96 0.98 0.97 138
accuracy 0.96 228
macro avg 0.97 0.96 0.96 228
weighted avg 0.96 0.96 0.96 228
import pandas as pd
Disk 81.49 GB available
0s completed at 8:47 PM
Activate Windows
Go to Settings to activate Windows.
Type here to search
28°C
ENG
20:47
13-09-2023

Practical-10

Implement Adaboost ensemble learning algorithm.

Code snippet:

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import AdaBoostClassifier
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"
names= ['preg','plas','pres', 'skin', 'test','mass', 'pedi','age', 'class']
dataframe = pandas.read_csv(url, names=names)
array=dataframe.values
print("Dataset Length:: ", len(dataframe))
print("Dataset Shape:: ", dataframe.shape)
print(dataframe.head())
X = array[:,0:8]
Y = array[:,8]
seed = 7
num_trees=30
kfold = model_selection.KFold(n_splits=10)
model = AdaBoostClassifier(n_estimators=num_trees, random_state=seed)
results = model_selection.cross_val_score(model, X, Y, cv=kfold)
print(results.mean()*100)
```

Or

```
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-
indians-diabetes.data.csv"
names= ['preg','plas','pres', 'skin', 'test','mass', 'pedi','age', 'class']
dataframe = pandas.read_csv(url, names=names)
array=dataframe.values
X = array[:,0:8]
Y = array[:,8]
seed = 7
kfold = model_selection.KFold(n_splits=10)
# create the sub models
estimators = []
model1=LogisticRegression (solver='lbfgs', max_iter=1000)
estimators.append(('logistic', model1))
model2=DecisionTreeClassifier()
estimators.append(('cart', model2))
```

```
model3 = SVC()
estimators.append(('svm', model3))
# create the ensemble model
ensemble=VotingClassifier (estimators)
results = model_selection.cross_val_score (ensemble, X, Y, cv=kfold)
print(results.mean()*100)
```

Output: