**VPM's B.N.Bandodkar College of Science (Autonomous) Thane.**

**INDEX**

| Sr.No | Aim | Date | Teacher's Sign |
|---|---|---|---|
| **1** | Write a program to implement prim's algorithm using C/C++/Python/Java/C# language. | | |
| **2** | Write a program to implement linear search using C/C++/Python/Java/C# language. | | |
| **3** | Write a program to implement Divide and Concur algorithm using C/C++/Python/Java/C# language. | | |
| **4** | Write a program to implement DFS algorithm using C/C++/Python/Java/C# language. | | |
| **5** | Write a program to implement BFS algorithm using C/C++/Python/Java/C# language. | | |
| **6** | Write a program to implement Dijkstra's shortest path algorithm using C/C++/Python/Java/C# language. | | |
| **7** | Implementation of convolutional neural network to predict numbers | | |
| **8** | Implementing regularization to avoid overfitting in binary classification. | | |

**PRACTICAL NO 1**

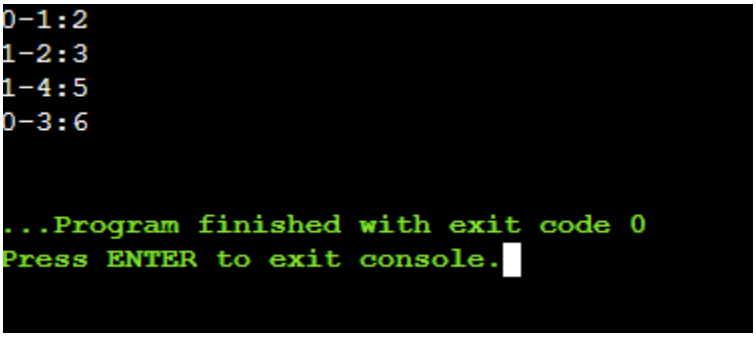
**AIM:-** Write a program to implement prim's algorithm using Python language.

**CODE:-**

```
def prim_mst(graph):
    V = len(graph)
    selected = [False] *
    no_edge = 0
    selected[0] = True
    while (no_edge < V - 1):
        minimum = float('inf')
        x = 0
        y = 0
        for m in range(V):
            if selected[m]:
                for n in range(V):
                    if ((not selected[n]) and graph[m][n]):
                        if minimum > graph[m][n]:
                            minimum = graph[m][n]
                            x = m
                            y = n
        print(str(x) + "-" + str(y) + ":" + str(graph[x][y]))
        selected[y] = True
        no_edge += 1
```

```
# Example graph represented as an adjacency matrix

graph = [[0, 2, 0, 6, 0],

        [2, 0, 3, 8, 5],

        [0, 3, 0, 0, 7],

        [6, 8, 0, 0, 9],

        [0, 5, 7, 9, 0]]

prim_mst(graph)
```

**OUTPUT:-**



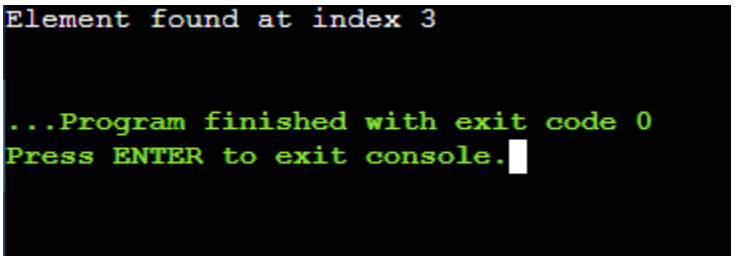
**PRACTICAL NO 2**

**AIM:-** Write a program to implement linear search using Python language.

**CODE:-**

```
def linear_search(arr, x):
    for i in range(len(arr)):
        if arr[i] == x:
            return i
    return -1
# Example usage:
arr = [34, 51, 1, 32, 11]
x = 32
result = linear_search(arr, x)
if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found in the array")
```

**OUTPUT:-**



**PRACTICAL NO 3**

**AIM:-** Write a program to implement Divide and Concur algorithm using Python language.

**CODE:-**

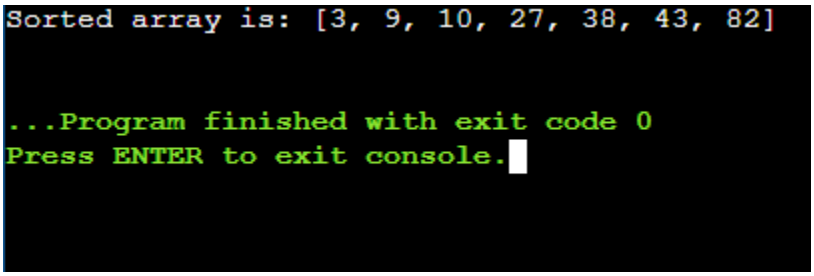
```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
    mid = len(arr) // 2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])
    return merge(left, right)
def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result.extend(left[i:])
    result.extend(right[j:])
    return result
```

# Example usage:

arr = [38, 27, 43, 3, 9, 82, 10]

print("Sorted array is:", merge_sort(arr))

**OUTPUT:-**



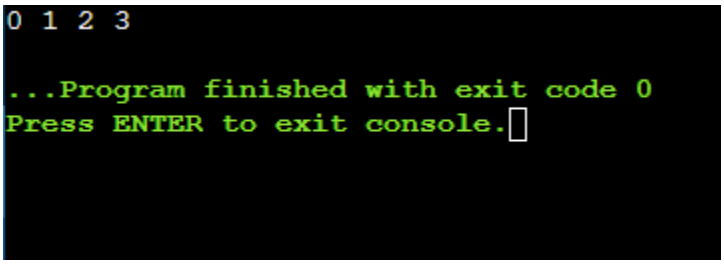
**PRACTICAL NO 4**

**AIM:-** Write a program to implement DFS algorithm using Python language.

**CODE:-**

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start, end=' ')
    for next in graph[start]:
        if next not in visited:
            dfs(graph, next, visited)
    return visited
graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}
dfs(graph, 0)
```

**OUTPUT:-**



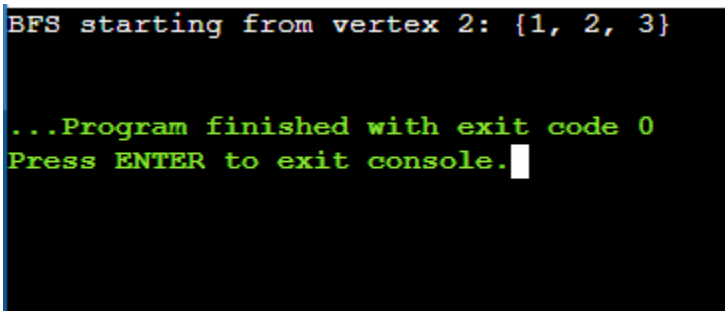
**PRACTICAL NO 5**

**AIM:-** Write a program to implement BFS algorithm using Python language.

**CODE:-**

```
from collections import deque

def bfs(graph, start):
    visited, queue = set(), deque([start])
    while queue:
        vertex = queue.popleft()
        if vertex not in visited:
            visited.add(vertex)
            queue.extend(set(graph[vertex]) - visited)
    return visited

graph = {0: [1, 2], 1: [2], 2: [3], 3: [1, 2]}

print("BFS starting from vertex 2:", bfs(graph, 2))
```

**OUTPUT:-**



**PRACTICAL NO 6**

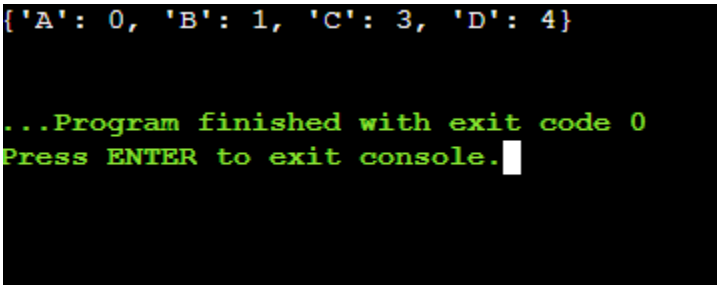
**AIM:-** Write a program to implement Dijkstra's shortest path algorithm using Python language

**CODE:-**

```
import heapq

def dijkstra(graph, start):
    distances = {vertex: float('infinity') for vertex in graph}
    distances[start] = 0
    priority_queue = [(0, start)]


    while priority_queue:
        current_distance, current_vertex = heapq.heappop(priority_queue)
        if current_distance > distances[current_vertex]:
            continue
        for neighbor, weight in graph[current_vertex].items():
            distance = current_distance + weight
            if distance < distances[neighbor]:
                distances[neighbor] = distance
                heapq.heappush(priority_queue, (distance, neighbor))
    return distances
# Example usage:
graph = {
    'A': {'B': 1, 'C': 4},
    'B': {'A': 1, 'C': 2, 'D': 5},
```

```
    'C': {'A': 4, 'B': 2, 'D': 1},
    'D': {'B': 5, 'C': 1}
}
start_node = 'A'
shortest_paths = dijkstra(graph, start_node)
print(shortest_paths)
```

**OUTPUT:-**



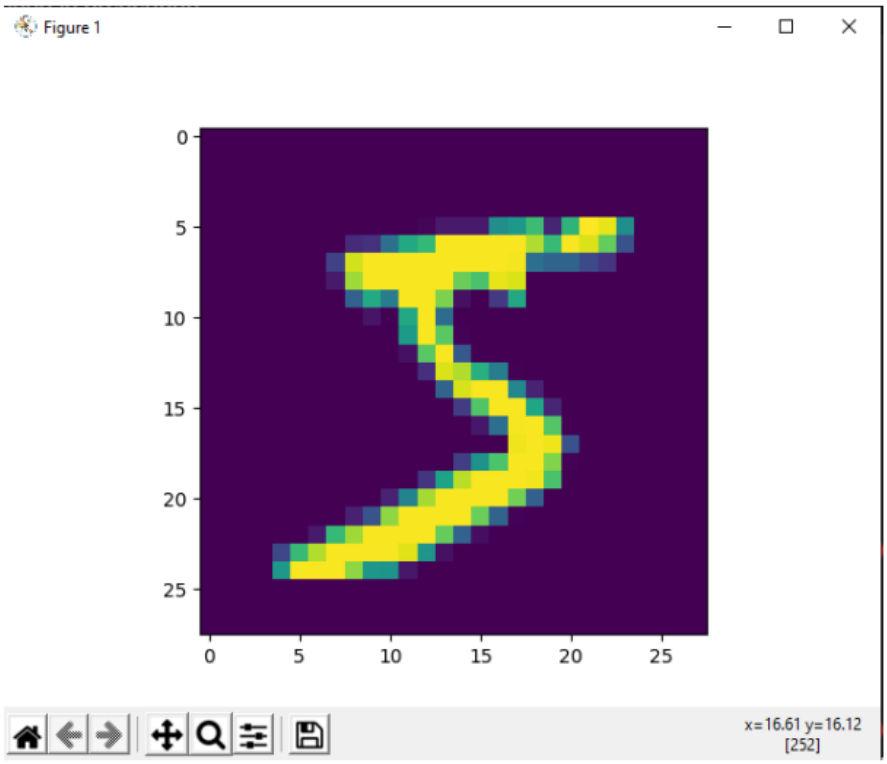
**PRACTICAL NO 7**

**AIM:-** Implementation of convolutional neural network to predict numbers

**CODE:-**

```
from number images
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense,Conv2D,Flatten
import matplotlib.pyplot as plt

#download mnist data and split into train and test sets
(X_train,Y_train),(X_test,Y_test)=mnist.load_data()

#plot the first image in the dataset
plt.imshow(X_train[0])
plt.show()
print(X_train[0].shape)
X_train=X_train.reshape(60000,28,28,1)
X_test=X_test.reshape(10000,28,28,1)
Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
Y_train[0]
print(Y_train[0])
model=Sequential()

#add model layers
#learn image features

model.add(Conv2D(64,kernel_size=3,activation='relu',input_shape=(28,28,1)))
model.add(Conv2D(32,kernel_size=3,activation='relu'))
model.add(Flatten())
model.add(Dense(10,activation='softmax'))
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])

#train
model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=3)
print(model.predict(X_test[:4]))

#actual results for 1st 4 images in the test set
print(Y_test[:4])
```

**OUTPUT:-**




(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]

```
(venv) PS D:\keras> python pract6.py
(28, 28)
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
Epoch 1/3
1875/1875 [==============================] - 235s 124ms/step - loss: 0.9714 - accuracy: 0.9111
- val_loss: 0.1084 - val_accuracy: 0.9661
Epoch 2/3
1875/1875 [==============================] - 242s 129ms/step - loss: 0.0663 - accuracy: 0.9789
- val_loss: 0.0787 - val_accuracy: 0.9758
Epoch 3/3
1875/1875 [==============================] - 241s 128ms/step - loss: 0.0458 - accuracy: 0.9854
- val_loss: 0.0904 - val_accuracy: 0.9751
[[8.5066381e-09 1.9058415e-15 1.5103029e-09 6.2544638e-07 4.8599115e-14
  3.8009873e-13 8.0967405e-13 9.9999940e-01 2.3813423e-10 1.8504194e-09]
 [4.6695381e-10 4.9075446e-09 1.0000000e+00 1.4425230e-12 5.5351397e-15
  1.4244286e-16 4.9031729e-10 2.1196991e-15 8.1773255e-13 2.7225001e-19]
 [1.4877173e-06 9.9855584e-01 1.0760028e-04 1.4199993e-07 1.0726219e-03
  6.1853432e-05 5.0982948e-05 6.4035441e-05 8.5100648e-05 3.5164564e-07]
 [9.9999988e-01 7.7231385e-13 9.2269055e-08 2.9055267e-10 1.8901826e-10
  2.9204628e-09 8.1175129e-09 4.1387605e-12 6.0085120e-10 1.4425010e-08]]
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
(venv) PS D:\keras>
```

**PRACTICAL NO 8**

**AIM:-** Implementing regularization to avoid overfitting in binary classification.

**CODE:-**

```
from matplotlib import pyplot
from sklearn.datasets import make_moons
from keras.models import Sequential
from keras.layers import Dense
X,Y=make_moons(n_samples=100,noise=0.2,random_state=1)
n_train=30
trainX,testX=X[:n_train,:],X[n_train:]
trainY,testY=Y[:n_train],Y[n_train:]

#print(trainX)
#print(trainY)
#print(testX)
#print(testY)

model=Sequential()
model.add(Dense(500,input_dim=2,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
history=model.fit(trainX,trainY,validation_data=(testX,testY),epochs=4000)
pyplot.plot(history.history['accuracy'],label='train')
pyplot.plot(history.history['val_accuracy'],label='test')
pyplot.legend()
pyplot.show()
```

**OUTOUT:-**