Koushik Sahu
118CS0597
Soft Computing Lab – IV
31st January 2022

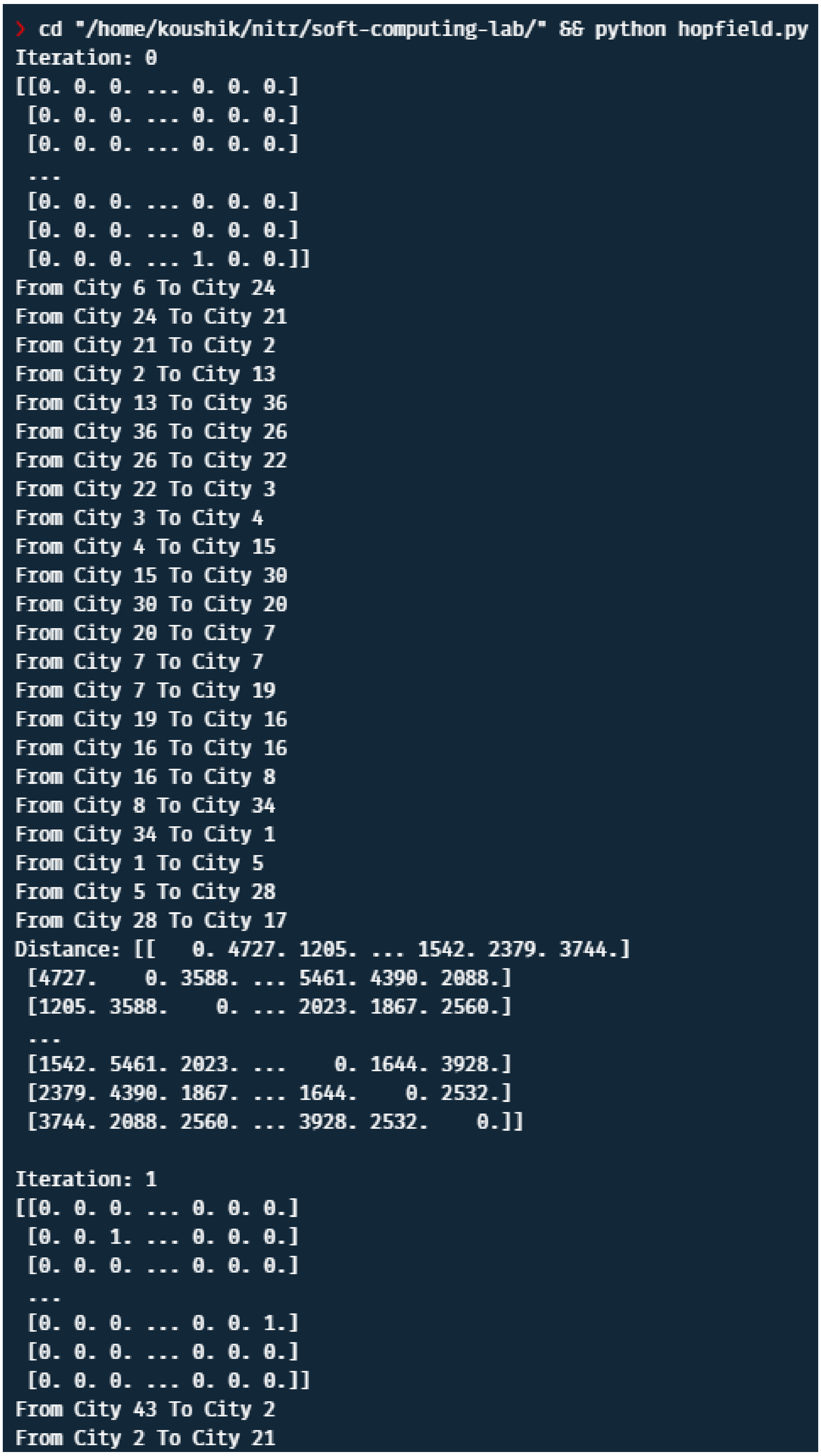
Code:

```
from random import randint
from random import uniform
import numpy as np


"""
        Author: Koushik Sahu
        Created: 07 February 2022 Mon 13:31:27
"""


def kronecker_delta(x, y):
    if x == y:
        return 1
    return 0


class Hopfield:
    def __init__(self, cities, d, alpha):
        self.cities = cities
        self.neurons = cities**2
        self.d = d
        self.alpha = alpha

        self.w = np.zeros([self.neurons, self.neurons])

        self.x_to_index = {}
        self.index_to_x = {}

        cnt = 0
        for i in range(cities):
            for j in range(cities):
                self.index_to_x[cnt] = (i, j)
                self.x_to_index[(i, j)] = cnt
                cnt += 1

    def train(self, A, B, C, D):
        n = self.cities
        for i in range(n):
            for a in range(n):
                for j in range(n):
                    for b in range(n):
                        wc = -C
                        wa = -A*(1-kronecker_delta(a, b))*kronecker_delta(i, j)
                        wb = -B*(1-kronecker_delta(i, j))*kronecker_delta(a, b)
                wd = -D*self.d[i][j]*(1-kronecker_delta(i, j))*(kronecker_delta(a-1, b) +
kronecker_delta(a+1, b))

                self.w[self.x_to_index[(i, a)]][self.x_to_index[(j, b)]] = wa + wb + wc + wd

    def predict(self, A, B, C, D, max_iterations):
        self.train(A, B, C, D)

        u = np.zeros([self.neurons, 1])
        for i in range(self.cities):
            for j in range(self.cities):
                u[i][0] = uniform(0, 0.03)

        prev_error = self.calc_error(u, A, B, C, D)
        repeated = 0
        max_repeat = 15
        for iteration in range(max_iterations):
            u = self.update(u, C)
            error = self.calc_error(u, A, B, C, D)
            if error == prev_error:
                repeated += 1
            else:
                repeated = 0

            if repeated > max_repeat:
                break
            prev_error = error
        ret = np.zeros([self.cities, self.cities])
        for i in range(self.cities):
            for j in range(self.cities):
                ret[i][j] = u[self.x_to_index[(i, j)]][0]

        return ret

    def update(self, u, C):
        n = self.cities
        for iteration in range(5*n**2):
            i = randint(0, n-1)
            x = randint(0, n-1)
            u[self.x_to_index[(i, x)]][0] = self.f(np.dot(u.transpose(), self.w[:, self.x_to_index[(i, x)]]) +
C*(n+1))
        return u

    def f(self, x):
        return 0.5*(1+np.tanh(self.alpha*x))

    def calc_error(self, u, A, B, C, D):
        tmpA = 0
        n = self.cities
        for i in range(n):
            for a in range(n):
```

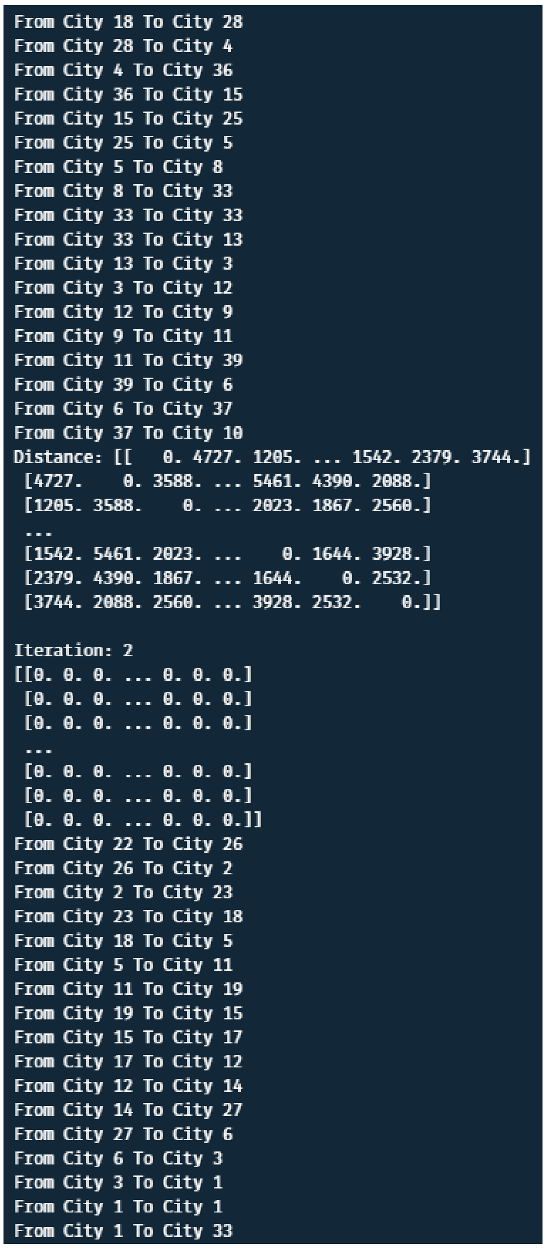
```
            for b in range(n):
               if a != b:
                  tmpA += u[self.x_to_index[(i, a)]][0] * u[self.x_to_index[(i, b)]][0]
      tmpA *= (A/2.0)

      tmpB = 0
      for i in range(n):
         for j in range(n):
            for a in range(n):
               if i != j:
                  tmpB += u[self.x_to_index[(i, a)]][0] * u[self.x_to_index[(j, a)]][0]
      tmpB *= (B/2.0)

      tmpC = 0
      for i in range(n):
         for a in range(n):
            tmpC += u[self.x_to_index[(i, a)]][0]
      tmpC = (tmpC - n)**2
      tmpC *= (C/2.0)

      tmpD = 0
      for i in range(n):
         for j in range(n):
            for a in range(n):
               if 0 < a < n-1:
                  tmpD += self.d[i][j]*u[self.x_to_index[(i, a)]][0]*(u[self.x_to_index[(j, a+1)]][0] +
u[self.x_to_index[(j, a-1)]][0])
               elif a > 0:
                  tmpD += self.d[i][j]*u[self.x_to_index[(i, a)]][0]*(u[self.x_to_index[(j, a-1)]][0] +
u[self.x_to_index[(j, 0)]][0])
               elif a < n-1:
                  tmpD += self.d[i][j]*u[self.x_to_index[(i, a)]][0]*(u[self.x_to_index[(j, a+1)]][0] +
u[self.x_to_index[(j, n-1)]][0])
      tmpD *= (D/2.0)

      return tmpA + tmpB + tmpC + tmpD


def main():
       n = 48

       dist = list()

       with open('att48_d.txt', 'rb') as f:
              dist_str = f.read().decode('utf-8')
              dist_str = dist_str.strip()
              dist_str_lists = dist_str.split('\n')

              for dist_str_list in dist_str_lists:
                     dist_list = dist_str_list.split()
                     dist.append([int(x) for x in dist_list])
```

```
    dist = np.array(dist, dtype=np.float32)

    cities = list()

    with open('att48_xy.txt', 'rb') as f:
        coord_str = f.read().decode('utf-8')
        coord_str = coord_str.strip()
        coord_str_lists = coord_str.split('\n')

        for coord_str_list in coord_str_lists:
            coord_list = coord_str_list.split()
            cities.append([int(x) for x in coord_list])

    cities = np.array(cities, dtype=np.float32)

    summation = 0
    mini = 1000
    maxi = -1
    for iteration in range(3):
        print("Iteration:", iteration)
        hf = Hopfield(n, dist, 50.0)
        v = hf.predict(A=100.0, B=100.0, C=90.0, D=110.0, max_iterations=10)
        print(v)

        d = 0
        prev_row = -1
        for col in range(v.shape[1]):
            for row in range(v.shape[0]):
                if v[row][col] == 1:
                    if prev_row != -1:
                        d += dist[prev_row][row]
                        print("From City {} To City {}".format(prev_row +
1, row + 1))
                    prev_row = row
                    break
        summation += d
        mini = min(mini, d)
        maxi = max(maxi, d)
        print("Distance:", dist, "\n")

    print("\nMin: {}\nMax: {}\nAverage: {}".format(mini, maxi, summation / 1000.0))


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

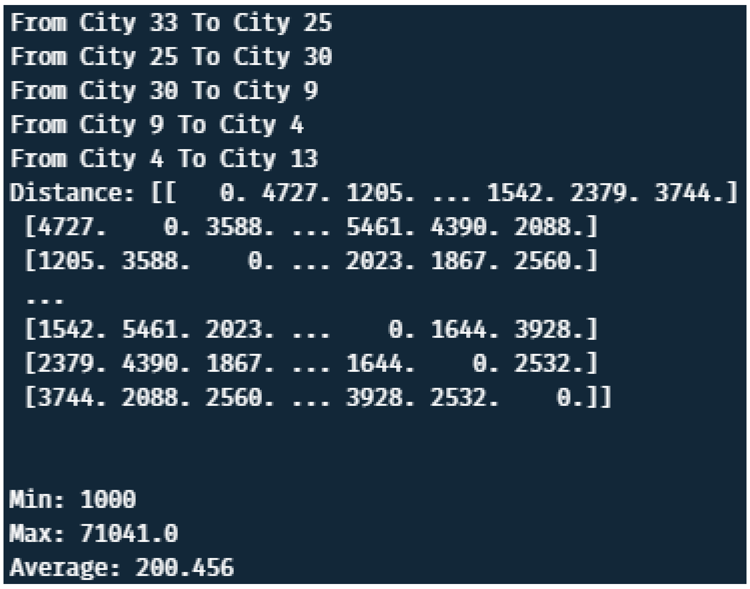
Terminal Screenshots:



```
❯ cd "/home/koushik/nitr/soft-computing-lab/" && python hopfield.py
Iteration: 0
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]]
From City 6 To City 24
From City 24 To City 21
From City 21 To City 2
From City 2 To City 13
From City 13 To City 36
From City 36 To City 26
From City 26 To City 22
From City 22 To City 3
From City 3 To City 4
From City 4 To City 15
From City 15 To City 30
From City 30 To City 20
From City 20 To City 7
From City 7 To City 7
From City 7 To City 19
From City 19 To City 16
From City 16 To City 16
From City 16 To City 8
From City 8 To City 34
From City 34 To City 1
From City 1 To City 5
From City 5 To City 28
From City 28 To City 17
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]

Iteration: 1
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
From City 43 To City 2
From City 2 To City 21
```



```
From City 18 To City 28
From City 28 To City 4
From City 4 To City 36
From City 36 To City 15
From City 15 To City 25
From City 25 To City 5
From City 5 To City 8
From City 8 To City 33
From City 33 To City 33
From City 33 To City 13
From City 13 To City 3
From City 3 To City 12
From City 12 To City 9
From City 9 To City 11
From City 11 To City 39
From City 39 To City 6
From City 6 To City 37
From City 37 To City 10
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]

Iteration: 2
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
From City 22 To City 26
From City 26 To City 2
From City 2 To City 23
From City 23 To City 18
From City 18 To City 5
From City 5 To City 11
From City 11 To City 19
From City 19 To City 15
From City 15 To City 17
From City 17 To City 12
From City 12 To City 14
From City 14 To City 27
From City 27 To City 6
From City 6 To City 3
From City 3 To City 1
From City 1 To City 1
From City 1 To City 33
```



```
From City 33 To City 25
From City 25 To City 30
From City 30 To City 9
From City 9 To City 4
From City 4 To City 13
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]


Min: 1000
Max: 71041.0
Average: 200.456
```

Terminal Output:

```
❯ cd "/home/koushik/nitr/soft-computing-lab/" && python hopfield.py
Iteration: 0
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 1. 0. 0.]]
From City 6 To City 24
From City 24 To City 21
From City 21 To City 2
From City 2 To City 13
From City 13 To City 36
From City 36 To City 26
From City 26 To City 22
From City 22 To City 3
From City 3 To City 4
From City 4 To City 15
From City 15 To City 30
From City 30 To City 20
From City 20 To City 7
From City 7 To City 7
From City 7 To City 19
From City 19 To City 16
From City 16 To City 16
From City 16 To City 8
From City 8 To City 34
From City 34 To City 1
From City 1 To City 5
From City 5 To City 28
From City 28 To City 17
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]

Iteration: 1
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
From City 43 To City 2
From City 2 To City 21
From City 21 To City 1
From City 1 To City 18
From City 18 To City 28
From City 28 To City 4
From City 4 To City 36
From City 36 To City 15
From City 15 To City 25
From City 25 To City 5
From City 5 To City 8
From City 8 To City 33
From City 33 To City 33
From City 33 To City 13
From City 13 To City 3
From City 3 To City 12
From City 12 To City 9
From City 9 To City 11
From City 11 To City 39
From City 39 To City 6
From City 6 To City 37
From City 37 To City 10
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]

Iteration: 2
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
From City 22 To City 26
From City 26 To City 2
From City 2 To City 23
From City 23 To City 18
From City 18 To City 5
From City 5 To City 11
From City 11 To City 19
From City 19 To City 15
From City 15 To City 17
From City 17 To City 12
From City 12 To City 14
From City 14 To City 27
From City 27 To City 6
From City 6 To City 3
From City 3 To City 1
From City 1 To City 1
From City 1 To City 33
From City 33 To City 25
From City 25 To City 30
From City 30 To City 9
From City 9 To City 4
From City 4 To City 13
Distance: [[   0. 4727. 1205. ... 1542. 2379. 3744.]
 [4727.    0. 3588. ... 5461. 4390. 2088.]
 [1205. 3588.    0. ... 2023. 1867. 2560.]
 ...
 [1542. 5461. 2023. ...    0. 1644. 3928.]
 [2379. 4390. 1867. ... 1644.    0. 2532.]
 [3744. 2088. 2560. ... 3928. 2532.    0.]]


Min: 1000
Max: 71041.0
Average: 200.456
```