**Exp No. 8**

**Rishab Mandal**

**Batch: C23**

**Code:**

**Exp8FrequencyReuse.py :**

#!/usr/bin/python

from math import \*

# import everything from Tkinter module

from tkinter import \*

# Base class for Hexagon shape

class Hexagon(object):

def \_\_init\_\_(self, parent, x, y, length, color, tags):

self.parent = parent

self.x = x

self.y = y

self.length = length

self.color = color

self.size = None

self.tags = tags

self.draw\_hex()

# draw one hexagon

def draw\_hex(self):

start\_x = self.x

start\_y = self.y

angle = 60

coords = []

for i in range(6):

end\_x = start\_x + self.length \* cos(radians(angle \* i))

end\_y = start\_y + self.length \* sin(radians(angle \* i))

coords.append([start\_x, start\_y])

start\_x = end\_x

start\_y = end\_y

self.parent.create\_polygon(coords[0][0],

coords[0][1],

coords[1][0],

coords[1][1],

coords[2][0],

coords[2][1],

coords[3][0],

coords[3][1],

coords[4][0],

coords[4][1],

coords[5][0],

coords[5][1],

fill=self.color,

outline="black",

tags=self.tags)

# class holds frequency reuse logic and related methods

class FrequencyReuse(Tk):

CANVAS\_WIDTH = 800

CANVAS\_HEIGHT = 650

TOP\_LEFT = (20, 20)

BOTTOM\_LEFT = (790, 560)

TOP\_RIGHT = (780, 20)

BOTTOM\_RIGHT = (780, 560)

def \_\_init\_\_(self, cluster\_size, columns=16, rows=10, edge\_len=30):

Tk.\_\_init\_\_(self)

self.textbox = None

self.curr\_angle = 330

self.first\_click = True

self.reset = False

self.edge\_len = edge\_len

self.cluster\_size = cluster\_size

self.reuse\_list = []

self.all\_selected = False

self.curr\_count = 0

self.hexagons = []

self.co\_cell\_endp = []

self.reuse\_xy = []

self.canvas = Canvas(self,

width=self.CANVAS\_WIDTH,

height=self.CANVAS\_HEIGHT,

bg="#4dd0e1")

self.canvas.bind("<Button-1>", self.call\_back)

self.canvas.focus\_set()

self.canvas.bind('<Shift-R>', self.resets)

self.canvas.pack()

self.title("Frequency reuse and co-channel selection")

self.create\_grid(16, 10)

self.create\_textbox()

self.cluster\_reuse\_calc()

# show lines joining all co-channel cells

def show\_lines(self):

# center(x,y) of first hexagon

approx\_center = self.co\_cell\_endp[0]

self.line\_ids = []

for k in range(1, len(self.co\_cell\_endp)):

end\_xx = (self.co\_cell\_endp[k])[0]

end\_yy = (self.co\_cell\_endp[k])[1]

# move i^th steps

l\_id = self.canvas.create\_line(approx\_center[0], approx\_center[1],

end\_xx, end\_yy)

if j == 0:

self.line\_ids.append(l\_id)

dist = 0

elif i >= j and j != 0:

self.line\_ids.append(l\_id)

dist = j

# rotate counter-clockwise and move j^th step

l\_id = self.canvas.create\_line(

end\_xx, end\_yy, end\_xx + self.center\_dist \* dist \*

cos(radians(self.curr\_angle - 60)),

end\_yy + self.center\_dist \* dist \*

sin(radians(self.curr\_angle - 60)))

self.line\_ids.append(l\_id)

self.curr\_angle -= 60

def create\_textbox(self):

txt = Text(self.canvas,

width=80,

height=1,

font=("Helvatica", 12),

padx=10,

pady=10)

txt.tag\_configure("center", justify="center")

txt.insert("1.0", "Select a Hexagon")

txt.tag\_add("center", "1.0", "end")

self.canvas.create\_window((0, 600), anchor='w', window=txt)

txt.config(state=DISABLED)

self.textbox = txt

def resets(self, event):

if event.char == 'R':

self.reset\_grid()

# clear hexagonal grid for new i/p

def reset\_grid(self, button\_reset=False):

self.first\_click = True

self.curr\_angle = 330

self.curr\_count = 0

self.co\_cell\_endp = []

self.reuse\_list = []

for i in self.hexagons:

self.canvas.itemconfigure(i.tags, fill=i.color)

try:

self.line\_ids

except AttributeError:

pass

else:

for i in self.line\_ids:

self.canvas.after(0, self.canvas.delete, i)

self.line\_ids = []

if button\_reset:

self.write\_text("Select a Hexagon")

# create a grid of Hexagons

def create\_grid(self, cols, rows):

size = self.edge\_len

for c in range(cols):

if c % 2 == 0:

offset = 0

else:

offset = size \* sqrt(3) / 2

for r in range(rows):

x = c \* (self.edge\_len \* 1.5) + 50

y = (r \* (self.edge\_len \* sqrt(3))) + offset + 15

hx = Hexagon(self.canvas, x, y, self.edge\_len, "#fafafa",

"{},{}".format(r, c))

self.hexagons.append(hx)

# calculate reuse distance, center distance and radius of the hexagon

def cluster\_reuse\_calc(self):

self.hex\_radius = sqrt(3) / 2 \* self.edge\_len

self.center\_dist = sqrt(3) \* self.hex\_radius

self.reuse\_dist = self.hex\_radius \* sqrt(3 \* self.cluster\_size)

def write\_text(self, text):

self.textbox.config(state=NORMAL)

self.textbox.delete('1.0', END)

self.textbox.insert('1.0', text, "center")

self.textbox.config(state=DISABLED)

#check if the co-channels are within visible canvas

def is\_within\_bound(self, coords):

if self.TOP\_LEFT[0] < coords[0] < self.BOTTOM\_RIGHT[0] \

and self.TOP\_RIGHT[1] < coords[1] < self.BOTTOM\_RIGHT[1]:

return True

return False

#gets called when user selects a hexagon

#This function applies frequency reuse logic in order to

#figure out the positions of the co-channels

def call\_back(self, evt):

selected\_hex\_id = self.canvas.find\_closest(evt.x, evt.y)[0]

hexagon = self.hexagons[int(selected\_hex\_id - 1)]

s\_x, s\_y = hexagon.x, hexagon.y

approx\_center = (s\_x + 15, s\_y + 25)

if self.first\_click:

self.first\_click = False

self.write\_text(

"""Now, select another hexagon such

that it should be a co-cell of

the original hexagon."""

)

self.co\_cell\_endp.append(approx\_center)

self.canvas.itemconfigure(hexagon.tags, fill="green")

for \_ in range(6):

end\_xx = approx\_center[0] + self.center\_dist \* i \* cos(

radians(self.curr\_angle))

end\_yy = approx\_center[1] + self.center\_dist \* i \* sin(

radians(self.curr\_angle))

reuse\_x = end\_xx + (self.center\_dist \* j) \* cos(

radians(self.curr\_angle - 60))

reuse\_y = end\_yy + (self.center\_dist \* j) \* sin(

radians(self.curr\_angle - 60))

if not self.is\_within\_bound((reuse\_x, reuse\_y)):

self.write\_text(

"""co-cells are exceeding canvas boundary.

Select cell in the center"""

)

self.reset\_grid()

break

if j == 0:

self.reuse\_list.append(

self.canvas.find\_closest(end\_xx, end\_yy)[0])

elif i >= j and j != 0:

self.reuse\_list.append(

self.canvas.find\_closest(reuse\_x, reuse\_y)[0])

self.co\_cell\_endp.append((end\_xx, end\_yy))

self.curr\_angle -= 60

else:

curr = self.canvas.find\_closest(s\_x, s\_y)[0]

if curr in self.reuse\_list:

self.canvas.itemconfigure(hexagon.tags, fill="green")

self.write\_text("Correct! Cell {} is a co-cell.".format(

hexagon.tags))

if self.curr\_count == len(self.reuse\_list) - 1:

self.write\_text("Great! Press Shift-R to restart")

self.show\_lines()

self.curr\_count += 1

else:

self.write\_text("Incorrect! Cell {} is not a co-cell.".format(

hexagon.tags))

self.canvas.itemconfigure(hexagon.tags, fill="red")

if \_\_name\_\_ == '\_\_main\_\_':

print(

"""Enter i & j values. common (i,j) values are:

(1,0), (1,1), (2,0), (2,1), (3,0), (2,2)"""

)

i = int(input("Enter i: "))

j = int(input("Enter j: "))

if i == 0 and j == 0:

raise ValueError("i & j both cannot be zero")

elif j > i:

raise ValueError("value of j cannot be greater than i")

else:

N = (i\*\*2 + i \* j + j\*\*2)

print("N is {}".format(N))

freqreuse = FrequencyReuse(cluster\_size=N)

freqreuse.mainloop()

**Output:**

(base) PS C:\Users\Rishab\OneDrive\Desktop\MCC Exp Documents> python Exp8code.py

Enter i & j values. common (i,j) values are:

(1,0), (1,1), (2,0), (2,1), (3,0), (2,2)

Enter i: 2

Enter j: 1

N is 7

