Appendix-2

Directory:

Code:

Main.py:

```
import tkinter as tk
import subprocess
from GA import GA
from CA import CA
import os
class main:
    @staticmethod
    def start():
        window =tk.Tk()

        window.title('Airplane program')
        window.geometry('400x300')
        window.grid()


button=tk.Button(window,text="exit",command=window.quit,width=20,heig
ht=2)
        label1=tk.Label(window,text="The Airplane App
```

```
thing",width=50,height=2)
        button0=tk.Button(window,text="Passenger
Distribution",command=GA.genetic,width=20,height=2)
        button00=tk.Button(window,text="Boarding
Method",command=CA.cell,width=20,height=2)


        label1.grid(row=1,column=0)
        button0.grid(row=3,column=0)
        button00.grid(row=4,column=0)
        button.grid(row=6,column=0)

        window.mainloop()
main.start()
```

CA.py:

```
import tkinter as tk
import tkinter.font as tkFont
import numpy as np
import os
from cellular import *
from moviepy.editor import *
import cv2
from os import startfile
d=0
e=0
f=0
class CA:
    @staticmethod
    def function(d,e,f):
            a,b,c=0.0,0.0,0.0
            a=float(d)/192 #computed to probability
            b=float(e)/192
            c=float(f)/192

            if (a<1 and b<1 and c<1 and a+b+c>=0):
                data = np.array([our(a,b,c).returnTime(),
backfront(a,b,c).returnTime(), reversepy(a,b,c).returnTime(),
outsidein(a,b,c).returnTime()]) #calls all boarding methods
                index = 0
                print (data)
                for i in range(1, 4): #comparison of speed
```

```
                    if data[i] < data[index]:
                        index = i
                if i==1: #compute the fastest method
                    arr = os.listdir('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\our')
                    print (arr)
                    a=[]
                    for i in arr:
                        a.append('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\our\\'+i)
                if i==2:
                    arr = os.listdir('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\outsidein')
                    print (arr)
                    a=[]
                    for i in arr:
                        a.append('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\outsidein\\'+i)

                if i==3:
                    arr = os.listdir('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\reversepy')
                    print (arr)
                    a=[]
                    for i in arr:
                        a.append('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\reversepy\\'+i)
                if i==0:
                    arr = os.listdir('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\backfront')
                    print (arr)
                    a=[]
                    for i in arr:
                        a.append('C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\backfront\\'+i)
                a=sorted(a)
                print (a)
                clip = ImageSequenceClip(a, fps=3)
                clip.write_videofile("C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\video.mp4", fps=5)


    @staticmethod
    def video():
```

```
        os.startfile("C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\video.mp4")

    class IntEntry(tk.Entry):
        def get(self):
            val = super().get()
            return int(val)
    @staticmethod
    def cell():
        window = tk.Tk()
        window.title('Boarding Assessment')
        window.geometry('350x300')
        window.grid()
        font1 = tkFont.Font(family="Helvetica", weight="bold")
        title = tk.Label(window, text='Boarding Method \nGenerator',
font=font1)
        label1 = tk.Label(window, text='Passengers with luggage')
        label2 = tk.Label(window, text='passengers with purse')
        label3 = tk.Label(window, text='Passengers with
disabilities')
        label4 = tk.Label(window, text='Entry must be >=0 and <192')


        entry1 = CA.IntEntry(window)
        entry2 = CA.IntEntry(window)
        entry3 = CA.IntEntry(window)
        def sub():
            label4.grid(row=6, column=0)
            if entry1.get()<192 and entry2.get()<192 and
entry3.get()<192 and (entry1.get()+entry2.get()+entry3.get())>=0 :
                CA.function(entry1.get(),entry2.get(),entry3.get())
            else:
                label4.grid(row=6, column=0)


        button = tk.Button(window, command=sub,text="Generate Order")
        button1 = tk.Button(window, command=CA.video, text="Play
Video")
        label1.grid(row=1, column=0)
        label2.grid(row=2, column=0)
        label3.grid(row=3, column=0)


        entry1.grid(row=1, column=1)
```

```
        entry2.grid(row=2,column=1)
        entry3.grid(row=3,column=1)
        button.grid(row=4,column=0)
        button1.grid(row=5,column=0)
        window.mainloop()
```

GA.py:

```
import local_simple_database
import tkinter as tk
import tkinter.font as tkFont
import json
from dataStructures import Genetic


class GA():
    def genetic():

        window = tk.Tk()
        window.title('Distribution Algorithm')
        window.geometry('350x200')
        window.grid()
        font1 = tkFont.Font(family="Helvetica", weight="bold")
        label0 = tk.Label(window, text='Passenger Distribution
\nGenerator',font=font1)
        label1=tk.Label(window,text='Number of Passengers')
        label2 = tk.Label(window, text='Passenger must be less than
192')
        label4 = tk.Label(window, text='all input must be >=0 and
<192')
        entry1=tk.Entry(window)
        def a ():
            if((int(entry1.get()))<192):
                thing=Genetic(int(entry1.get()),"0").generate()
            else:
                label2.grid(row=3,column=0)



        button1=tk.Button(window,text='Generate',command=a)
```

```
        label0.grid(row=0, column=0)
        label1.grid(row=1,column=0)
        entry1.grid(row=1,column=1)
        button1.grid(row=2,column=0)
        label4.grid(row=4, column=0)



        window.mainloop()
```

Cellular.py

```
import numpy as np, pandas as pd, random, matplotlib.pyplot as plt,
seaborn as sns
from IPython.display import clear_output
from PyProbs import Probability as pr

class passenger(object):
    def __init__(self,disabled,purse,luggage):
        self.aisle = 0
        self.seat = 0
        self.speed = 99 - 50*pr.Prob(disabled)
        self.purse = pr.Prob(purse)
        self.carry = pr.Prob(luggage)
        self.purse = 0
        self.carry = 0
        self.timer = 2
        self.first = 1
        self.action = 0
        self.board = 0
        self.wait = 0
        self.walk = 0
        self.bag = 0
        self.stop_counter = 0
        self.prev_stop = 0



def initialize_array(narrow):
    for i in range(32):
        narrow[i][7] = 12
    return narrow

def initialize_timer(p_list):
    for i in range(len(p_list)):
        p_list[i].timer +=
int(np.random.weibull(2)*(3*p_list[i].purse + 5*p_list[i].carry))
```

```
    return p_list

def run_narrow(p_list, narrow,s):
    plt.close("all")
    a = 0
    #fig, ax = plt.subplots()
    #plt.ion()
    #plt.show()
    sit = 0
    time = 0
    seated = []
    viewer = zeros = [[0] * 7 for _ in range(32)]
    while sit < 192:
        time += 1
        for i in range(32):
            change = 0
            if narrow[i][3] != 0 and narrow[i][3].action == 0:
                narrow[i][3].action = 1
                if narrow[i][3].aisle != i:
                    if narrow[i+1][3] == 0:
                        narrow[i][3].prev_stop = 0
                        if random.randrange(100) <
narrow[i][3].speed:
                            narrow[i+1][3] = narrow[i][3]
                            narrow[i][3] = 0
                            change = 1
                    else:
                        narrow[i][3].wait += 1
                        if narrow[i][3].prev_stop == 0:
                            narrow[i][3].prev_stop = 1
                            narrow[i][3].stop_counter += 1
                else:
                    if narrow[i][3].first:
                        narrow[i][3].first = 0
                        if narrow[i][7] < 0:
                            narrow[i][3].timer += 5
                        else:
                            narrow[i][3].timer += int(3.07*10**(-
5))*np.e**(12-narrow[i][7])
                        narrow[i][7] -= narrow[i][3].purse +
narrow[i][3].carry
                        if narrow[i][3].seat == 0:
                            narrow[i][3].timer +=
int(np.random.weibull(2))*(5*narrow[i][1] + 3*narrow[i][2])
```

```
                        elif narrow[i][3].seat == 1:
                            narrow[i][3].timer +=
int(np.random.weibull(2))*(3*narrow[i][2])
                        elif narrow[i][3].seat == 5:
                            narrow[i][3].timer +=
int(np.random.weibull(2))*(3*narrow[i][4])
                        elif narrow[i][3].seat == 6:
                            narrow[i][3].timer +=
int(np.random.weibull(2))*(5*narrow[i][5] + 3*narrow[i][4])
                        if narrow[i][3].speed == 49:
                            narrow[i][3].timer *= 2
                        narrow[i][3].bag = narrow[i][3].timer
                    if narrow[i][3].timer != 0:
                        narrow[i][3].timer -= 1
                    else:
                        change = 1
                        if narrow[i][3].seat >= 3:
                            narrow[i][narrow[i][3].seat + 1] = 1
                            sit += 1
                        else:
                            narrow[i][narrow[i][3].seat] = 1
                            sit += 1
                        seated.append(narrow[i][3])
                        narrow[i][3] = 0
            if i == 0:
                if narrow[i][3] == 0 and len(p_list) != 0:
                    narrow[i][3] = p_list.pop()
                    narrow[i][3].action = 1
                    change = 1

        for i in range(32):
            for j in range(7):
                if narrow[i][j] != 0:
                    if j != 3:
                        viewer[i][j] = 10
                    else:
                        if narrow[i][j].speed == 49:
                            viewer[i][j] = 3
                        else:
                            viewer[i][j] = 10
                elif j == 3:
                    viewer[i][j] = 0
                else:
                    viewer[i][j] = 5
```

```
        plt.close()

        #plt.draw()
        if time % 6 == 0 or sit > 191:
            plt.show()
            a += 1
            plt.close()
            fig, ax = plt.subplots()
            plt.imshow(viewer, cmap='summer', vmin=0, vmax=10)
            plt.savefig('C:\\Users\\Eric
Chang\\PycharmProjects\IA\\'+s+'\\' + str(a/100) + '.png')


        for i in range(32):
            if narrow[i][3] != 0:
                narrow[i][3].action = 0
                narrow[i][3].board += 1
    board_time = []
    wait_time = []
    bag_time = []
    stop_counter = []
    for i in range(len(seated)):
        board_time.append(seated[i].board)
        wait_time.append(seated[i].wait)
        bag_time.append(seated[i].bag)
        stop_counter.append(seated[i].stop_counter)
    return time, np.mean(board_time), np.mean(wait_time),
np.mean(bag_time), np.mean(stop_counter)
class our():
    def __init__(self,disabled,purse,luggage):
        time = []
        board_avg = []
        wait_avg = []
        bag_avg = []
        stop_avg = []
        for i in range(1):
            p_list =
self.create_passenger_list(disabled,purse,luggage)
            p_list = initialize_timer(p_list)
            narrow = [[0] * 8 for _ in range(32)]
            narrow = initialize_array(narrow)
            a, b, c, d, e = run_narrow(p_list, narrow, 'our')
            time.append(a)
            board_avg.append(b)
```

```
            wait_avg.append(c)
            bag_avg.append(d)
            stop_avg.append(e)
        self.t=np.mean(time)
        np_time = np.array(time)
        np_board_avg = np.array(board_avg)
        np_wait_avg = np.array(wait_avg)
        np_bag_avg = np.array(bag_avg)
        np_stop_avg = np.array(stop_avg)
        #np.save('dis2_time.npy', np_time)
        #np.save('dis2_board_avg.npy', np_board_avg)
        #np.save('dis2_wait_avg.npy', np_wait_avg)
        #np.save('dis2_bag_avg.npy', np_bag_avg)
        #np.save('dis2_stop_avg.npy', np_stop_avg)

    def create_passenger_list(self, disabled, purse, luggage):
        p_list = []
        for i in range(32 * 6):
            p_list.append(passenger(disabled, purse, luggage))
            p_list[i].aisle = i // 6
            p_list[i].seat = i % 6
        # p_list.reverse()
        # random.shuffle(p_list)
        temp = p_list[:64]
        random.shuffle(temp)
        p_list[:64] = temp
        temp = p_list[64:128]
        random.shuffle(temp)
        p_list[64:128] = temp
        temp = p_list[128:]
        random.shuffle(temp)
        p_list[128:] = temp
        # random.shuffle(p_list[64:128])
        # random.shuffle(p_list[128:])
        return p_list
    def returnTime(self):
        return self.t
class outsidein():
    def __init__(self,disabled,purse,luggage):
        time = []
        board_avg = []
        wait_avg = []
        bag_avg = []
        stop_avg = []
```

```
        for i in range(1):
            p_list =
self.create_passenger_list(disabled,purse,luggage)
            p_list = initialize_timer(p_list)
            narrow = [[0] * 8 for _ in range(32)]
            narrow = initialize_array(narrow)
            a, b, c, d, e = run_narrow(p_list, narrow, 'outsidein')
            time.append(a)
            board_avg.append(b)
            wait_avg.append(c)
            bag_avg.append(d)
            stop_avg.append(e)
        self.t=np.mean(time)
        np_time = np.array(time)
        np_board_avg = np.array(board_avg)
        np_wait_avg = np.array(wait_avg)
        np_bag_avg = np.array(bag_avg)
        np_stop_avg = np.array(stop_avg)
        #np.save('dis2_time.npy', np_time)
        #np.save('dis2_board_avg.npy', np_board_avg)
        #np.save('dis2_wait_avg.npy', np_wait_avg)
        #np.save('dis2_bag_avg.npy', np_bag_avg)
        #np.save('dis2_stop_avg.npy', np_stop_avg)

    def create_passenger_list(self, disabled, purse, luggage):
        p_list = []
        for i in range(6):
            for j in range(32):
                p_list.append(passenger(disabled, purse, luggage))
                p_list[i * 32 + j].aisle = j
                p_list[i * 32 + j].seat = i
        p_list_2 = []
        p_list_2.extend(p_list[:32])
        p_list_2.extend(p_list[-32:])
        p_list_3 = []
        p_list_3.extend(p_list[32:64])
        p_list_3.extend(p_list[-64:-32])
        p_list_4 = []
        p_list_4.extend(p_list[64:96])
        p_list_4.extend(p_list[-96:-64])
        random.shuffle(p_list_2)
        random.shuffle(p_list_3)
        random.shuffle(p_list_4)
        p_list[:64] = p_list_2
```

```
        p_list[64:128] = p_list_3
        p_list[128:] = p_list_4
        p_list.reverse()
        return p_list
    def returnTime(self):
        return self.t
class reversepy():
    def __init__(self,disabled,purse,luggage):
        time = []
        board_avg = []
        wait_avg = []
        bag_avg = []
        stop_avg = []
        for i in range(1):
            p_list =
self.create_passenger_list(disabled,purse,luggage)
            p_list = initialize_timer(p_list)
            narrow = [[0] * 8 for _ in range(32)]
            narrow = initialize_array(narrow)
            a, b, c, d, e = run_narrow(p_list, narrow, 'reversepy')
            time.append(a)
            board_avg.append(b)
            wait_avg.append(c)
            bag_avg.append(d)
            stop_avg.append(e)
        self.t=np.mean(time)
        np_time = np.array(time)
        np_board_avg = np.array(board_avg)
        np_wait_avg = np.array(wait_avg)
        np_bag_avg = np.array(bag_avg)
        np_stop_avg = np.array(stop_avg)
        #np.save('dis2_time.npy', np_time)
        #np.save('dis2_board_avg.npy', np_board_avg)
        #np.save('dis2_wait_avg.npy', np_wait_avg)
        #np.save('dis2_bag_avg.npy', np_bag_avg)
        #np.save('dis2_stop_avg.npy', np_stop_avg)

    def create_passenger_list(self,disabled,purse,luggage):
        p_list = []
        for i in range(4):
            for j in range(48):
                p_list.append(passenger(disabled,purse,luggage))
                if i == 0:
                    if j % 2 == 0:
```

```
                    p_list[48 * i + j].seat = 0
                else:
                    p_list[48 * i + j].seat = 5
                p_list[48 * i + j].aisle = 31 - j // 2
            if i == 1:
                if j < 16:
                    if j % 2 == 0:
                        p_list[48 * i + j].seat = 0
                    else:
                        p_list[48 * i + j].seat = 5
                    p_list[48 * i + j].aisle = 31 - (j // 2 + 24)
                else:
                    if j % 2 == 0:
                        p_list[48 * i + j].seat = 1
                    else:
                        p_list[48 * i + j].seat = 4
                    p_list[48 * i + j].aisle = 31 - (j // 2 - 8)
            if i == 2:
                if j < 32:
                    if j % 2 == 0:
                        p_list[48 * i + j].seat = 1
                    else:
                        p_list[48 * i + j].seat = 4
                    p_list[48 * i + j].aisle = 31 - (j // 2 + 16)
                else:
                    if j % 2 == 0:
                        p_list[48 * i + j].seat = 2
                    else:
                        p_list[48 * i + j].seat = 3
                    p_list[48 * i + j].aisle = 31 - (j // 2 - 16)
            if i == 3:
                if j % 2 == 0:
                    p_list[48 * i + j].seat = 2
                else:
                    p_list[48 * i + j].seat = 3
                p_list[48 * i + j].aisle = 31 - (j // 2 + 8)
        p_list_1 = p_list[:48]
        p_list_2 = p_list[48:96]
        p_list_3 = p_list[96:144]
        p_list_4 = p_list[144:]
        random.shuffle(p_list_1)
        random.shuffle(p_list_2)
        random.shuffle(p_list_3)
        random.shuffle(p_list_4)
```

```
        p_list[:48] = p_list_1
        p_list[48:96] = p_list_2
        p_list[96:144] = p_list_3
        p_list[144:] = p_list_4
        p_list.reverse()
        return p_list
    def returnTime(self):
        return self.t
class backfront():
    def __init__(self,disabled,purse,luggage):
        time = []
        board_avg = []
        wait_avg = []
        bag_avg = []
        stop_avg = []
        for i in range(1):
            p_list =
self.create_passenger_list(disabled,purse,luggage)
            p_list = initialize_timer(p_list)
            narrow = [[0] * 8 for _ in range(32)]
            narrow = initialize_array(narrow)
            a, b, c, d, e = run_narrow(p_list, narrow, 'backfront')
            time.append(a)
            board_avg.append(b)
            wait_avg.append(c)
            bag_avg.append(d)
            stop_avg.append(e)
        self.t=np.mean(time)
        np_time = np.array(time)
        np_board_avg = np.array(board_avg)
        np_wait_avg = np.array(wait_avg)
        np_bag_avg = np.array(bag_avg)
        np_stop_avg = np.array(stop_avg)
        #np.save('dis2_time.npy', np_time)
        #np.save('dis2_board_avg.npy', np_board_avg)
        #np.save('dis2_wait_avg.npy', np_wait_avg)
        #np.save('dis2_bag_avg.npy', np_bag_avg)
        #np.save('dis2_stop_avg.npy', np_stop_avg)

    def create_passenger_list(self, disabled, purse, luggage):
        p_list = []
        for i in range(32 * 6):
            p_list.append(passenger(disabled, purse, luggage))
            p_list[i].aisle = i // 6
```

```
            p_list[i].seat = i % 6
        # p_list.reverse()
        # random.shuffle(p_list)
        temp = p_list[:64]
        random.shuffle(temp)
        p_list[:64] = temp
        temp = p_list[64:128]
        random.shuffle(temp)
        p_list[64:128] = temp
        temp = p_list[128:]
        random.shuffle(temp)
        p_list[128:] = temp
        # random.shuffle(p_list[64:128])
        # random.shuffle(p_list[128:])
        return p_list
        return p_list
    def returnTime(self):
        return self.t
```

datastructures.py:

```
import matplotlib.pyplot as plt
from py4j.java_gateway import JavaGateway, GatewayParameters
import jpype
import os
import numpy as np
import json
class Genetic():
    data=0
    passengers=0
    aircraft=0
    def __init__(self, num, type):
        self.passengers = num
        self.airCraft = type
        jarpath = os.path.join(os.path.abspath('.'), r'C:\Users\Eric
Chang\PycharmProjects\IA\IA.jar')
        dependency = os.path.join(os.path.abspath('.'),
'C:\\Users\\Eric Chang\\PycharmProjects\\IA\\dependencies')

        jvmPath = 'C:\Program Files\Java\jdk-
18.0.2.1\\bin\server\jvm.dll'

        jpype.startJVM(jvmPath, "-ea", "-Djava.class.path=%s" %
jarpath)
```

```
        JClass = jpype.JClass('geneticRef.narrow')
        thing = JClass(num, type)
        jpype.shutdownJVM()
        data=np.zeros((32,7))
        f = open("C:\\Users\\Eric
Chang\\PycharmProjects\\IA\\problemname.txt", "r")
        for i in range (0,32):
            arr=f.readline().split(',')
            for j in range(0,7):
                data[i][j]=int(arr[j])
        self.data=data
        f.readline()
        a=f.readline()
        a=float(a.replace("\n",""))
        #print (a)
        f.close()



    def generate(self):
        plt.plot()
        plt.imshow(self.data, cmap='summer', interpolation='nearest')
        plt.show()
        return self.data
```

IA.jar

```
package geneticRef;
import py4j.GatewayServer;
import java.io.*;
import java.util.*;
public class narrow {
    private static ArrayList <Double> HighScore=new ArrayList <Double> ();
    private static String type;
    private static int passengers;
    private static int [][] narrow={
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
```

```
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,}
        };

    static ArrayList <Double> top;

    public narrow (int n,String types)throws IOException {

        PrintWriter pw = new PrintWriter("C:\\Users\\Eric Chang\\PycharmProjects\\IA\\problemname.txt");
        String str="";
        passengers=n;
        type=types;
        int[][] arr=narrow.clone();
        ArrayList <set> a=Genetics(arr,passengers);
        int count=0;
        for(set s:a) {
            arr[s.x][s.y]=2;
            count++;
        }
        for(int i=0;i<arr.length;i++) {
```

```
                pw.print("");
                for(int j=0;j<arr[0].length;j++) {
                        pw.print(arr[i][j]+",");
                }
                pw.println("");
        }

        pw.println("Highest:");

        pw.println(HighScore.get(HighScore.size()-1));
        pw.close();
    }
    public static int count(int [][]a) {
        int count=0;
        for(int i=0;i<a.length;i++) {
            for(int j=0;j<a[i].length;j++) {
                if (a[i][j]==1) {
                    count++;
                }
            }
        }
        return count;
    }

    public static double score(ArrayList <set> a) {
        double score=0;
        for(int i=0;i<a.size();i++) {
            for(int j=i;j<a.size();j++) {
                if(Math.sqrt(Math.pow((double)(a.get(i).x-
a.get(j).x),2)+Math.pow((double)(a.get(i).y-a.get(j).y),2))>0){
                    if(Math.sqrt(0.75*Math.pow((double)(a.get(i).x-
a.get(j).x),2)+0.75*Math.pow((double)(a.get(i).y-a.get(j).y),2))<4) {
                        score+=-
18.19*Math.log(Math.sqrt(0.75*Math.pow((double)(a.get(i).x-
a.get(j).x),2)+0.75*Math.pow((double)(a.get(i).y-a.get(j).y),2)))+43;}

                    }
                }
            }
        return score;
        }


    public static ArrayList<set> Genetics(int [][] a, int count) {
```

```
HighScore=new ArrayList <Double> ();
int generations=500;
int num=count(a);
//initalize
ArrayList <ArrayList<set>> thing=new ArrayList <ArrayList <set>> ();
ArrayList <Double> score=new ArrayList <Double> ();

top=new ArrayList<Double>();
for(int i=0;i<10;i++) {
	thing.add(randomize(count,a));
	score.add(score(thing.get(i)));

}
for(int gen=0;gen<generations;gen++) {

	int first=0;
	int second=0;

	for(int i=0;i<score.size();i++) {
		if(score.get(i)<score.get(first)) {
			first=i;

		}
	}
	for(int i=0;i<score.size();i++) {
		if(score.get(i)<score.get(second)&&i!=first) {
			second=i;

		}
	}

	for(int i=0;i<thing.size();i++) {
		if (i!=first&&i!=second) {
			thing.set(i, offspring(thing.get(first),thing.get(second)));
			score.set(i,score(thing.get(i)));

		}
	}
	double max=score.get(0);
	for(double d:score) {
		if(d<max) {
			max=d;
		}
	}
```

```
            HighScore.add(max);
        }
        int max=0;
        for(int i=0;i<score.size();i++) {
            if(score.get(i)<score.get(max)) {
                max=i;
            }
        }
        return thing.get(max);
    }
    public static ArrayList <set >offspring(ArrayList <set>one, ArrayList<set> two) {
        int rand1=(int)(Math.random()*(one.size()/2));
        int rand2=(int)(Math.random()*(one.size()/2)+one.size()/2);
        int num=(int)(Math.random()*2);
        ArrayList<ArrayList<set>> union=new ArrayList<ArrayList<set>>();
        union.add(one);
        union.add(two);
        ArrayList<set> New=new ArrayList<set>();
        if(num==1) {
            for(int i=0;i<one.size();i++) {
                if(i>rand1&&i<rand2) {
                    New.add(union.get(1).get(i));
                }
                else {
                    New.add(union.get(0).get(i));
                }
            }
        }
        else{
            for(int i=0;i<one.size();i++) {
                if(i>rand1&&i<rand2) {
                    New.add(union.get(0).get(i));
                }
                else {
                    New.add(union.get(1).get(i));
                }
            }
        }
        double mutate=Math.random();
        if(mutate<=0.5) {
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
```

```
                    continue;
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
                    continue;
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
                    continue;
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
        }
        return New;
}
public static ArrayList <set> randomize(int a,int [][] ba) {
        ArrayList <set> array=new ArrayList <set> ();

            array=new ArrayList <set> ();
            ArrayList <set> exists=new ArrayList <set> ();
            while (array.size()<a) {
                int x=(int) (Math.random()*ba.length);
                int y=(int)(Math.random()*ba[0].length);
                boolean b=true;
```

```
                    for(set s:exists) {
                        if(s.x==x&&s.y==y) {
                            b=false;
                        }
                    }
                    if(ba[x][y]==0) {
                        b=false;
                    }
                    if(b==true) {
                        set s=new set (x,y);
                        array.add(s);
                        exists.add(s);
                    }

            }

            return array;
        }
}
```

Historical Versions:

FlyingWing.java:

```
package geneticRef;
import py4j.GatewayServer;
import java.io.*;
import java.util.*;
public class narrow {
    private static ArrayList <Double> HighScore=new ArrayList <Double> ();
    private static String type;
    private static int passengers;
    private static int [][] narrow={
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
            {1,1,1,0,1,1,1,},
```

```
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,},
        {1,1,1,0,1,1,1,}
        };

  static ArrayList <Double> top;

  public narrow (int n,String types)throws IOException {

      PrintWriter pw = new PrintWriter("C:\\Users\\Eric Chang\\PycharmProjects\\IA\\problemname.txt");
      String str="";
      passengers=n;
      type=types;
      int[][] arr=narrow.clone();
      ArrayList <set> a=Genetics(arr,passengers);
      int count=0;
      for(set s:a) {
          arr[s.x][s.y]=2;
          count++;
      }
      for(int i=0;i<arr.length;i++) {
          pw.print("");
```

```
                for(int j=0;j<arr[0].length;j++) {
                        pw.print(arr[i][j]+",");
                }
                pw.println("");
        }

        pw.println("Highest:");

        pw.println(HighScore.get(HighScore.size()-1));
        pw.close();
    }
    public static int count(int [][]a) {
        int count=0;
        for(int i=0;i<a.length;i++) {
            for(int j=0;j<a[i].length;j++) {
                if (a[i][j]==1) {
                    count++;
                }
            }
        }
        return count;
    }

    public static double score(ArrayList <set> a) {
        double score=0;
        for(int i=0;i<a.size();i++) {
            for(int j=i;j<a.size();j++) {
                if(Math.sqrt(Math.pow((double)(a.get(i).x-
a.get(j).x),2)+Math.pow((double)(a.get(i).y-a.get(j).y),2))>0){
                    if(Math.sqrt(0.75*Math.pow((double)(a.get(i).x-
a.get(j).x),2)+0.75*Math.pow((double)(a.get(i).y-a.get(j).y),2))<4) {
                        score+=-
18.19*Math.log(Math.sqrt(0.75*Math.pow((double)(a.get(i).x-
a.get(j).x),2)+0.75*Math.pow((double)(a.get(i).y-a.get(j).y),2)))+43;}

                    }
                }
            }
        return score;
        }


    public static ArrayList<set> Genetics(int [][] a, int count) {
        HighScore=new ArrayList <Double> ();
```

```
int generations=500;
int num=count(a);
//initalize
ArrayList <ArrayList<set>> thing=new ArrayList <ArrayList <set>> ();
ArrayList <Double> score=new ArrayList <Double> ();

top=new ArrayList<Double>();
for(int i=0;i<10;i++) {
	thing.add(randomize(count,a));
	score.add(score(thing.get(i)));

}
for(int gen=0;gen<generations;gen++) {

	int first=0;
	int second=0;

	for(int i=0;i<score.size();i++) {
		if(score.get(i)<score.get(first)) {
			first=i;

		}
	}
	for(int i=0;i<score.size();i++) {
		if(score.get(i)<score.get(second)&&i!=first) {
			second=i;

		}
	}

	for(int i=0;i<thing.size();i++) {
		if (i!=first&&i!=second) {
			thing.set(i, offspring(thing.get(first),thing.get(second)));
			score.set(i,score(thing.get(i)));

		}
	}
	double max=score.get(0);
	for(double d:score) {
		if(d<max) {
			max=d;
		}
	}
	HighScore.add(max);
```

```
        }
        int max=0;
        for(int i=0;i<score.size();i++) {
            if(score.get(i)<score.get(max)) {
                max=i;
            }
        }
        return thing.get(max);
    }
    public static ArrayList <set >offspring(ArrayList <set>one, ArrayList<set> two) {
        int rand1=(int)(Math.random()*(one.size()/2));
        int rand2=(int)(Math.random()*(one.size()/2)+one.size()/2);
        int num=(int)(Math.random()*2);
        ArrayList<ArrayList<set>> union=new ArrayList<ArrayList<set>>();
        union.add(one);
        union.add(two);
        ArrayList<set> New=new ArrayList<set>();
        if(num==1) {
            for(int i=0;i<one.size();i++) {
                if(i>rand1&&i<rand2) {
                    New.add(union.get(1).get(i));
                }
                else {
                    New.add(union.get(0).get(i));
                }
            }
        }
        else{
            for(int i=0;i<one.size();i++) {
                if(i>rand1&&i<rand2) {
                    New.add(union.get(0).get(i));
                }
                else {
                    New.add(union.get(1).get(i));
                }
            }
        }
        double mutate=Math.random();
        if(mutate<=0.5) {
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
                    continue;
```

```
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
                    continue;
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
            while(true) {
                int x=(int) (Math.random()*narrow.length);
                int y=(int)(Math.random()*narrow[0].length);
                if (narrow[x][y]==0) {
                    continue;
                }
                else{
                int rand3=(int)Math.random()*one.size();
                New.set(rand3, new set (x,y));
                break;
                }
            }
        }
        return New;
}
public static ArrayList <set> randomize(int a,int [][] ba) {
        ArrayList <set> array=new ArrayList <set> ();

            array=new ArrayList <set> ();
            ArrayList <set> exists=new ArrayList <set> ();
            while (array.size()<a) {
                int x=(int) (Math.random()*ba.length);
                int y=(int)(Math.random()*ba[0].length);
                boolean b=true;
                for(set s:exists) {
```

```
                    if(s.x==x&&s.y==y) {
                        b=false;
                    }
                }
                if(ba[x][y]==0) {
                    b=false;
                }
                if(b==true) {
                    set s=new set (x,y);
                    array.add(s);
                    exists.add(s);
                }

            }

            return array;
        }
}
```

Client.py

```python
import socket
import tkinter
import tkinter.messagebox
import threading
import json
import tkinter.filedialog
from tkinter.scrolledtext import ScrolledText

IP = ''
PORT = ''
user = ''
listbox1 = ''
show = 1
users = []
chat = '------Group chat-------'

root0 = tkinter.Tk()
root0.geometry("300x150")
root0.title('Login Page')
root0.resizable(0,0)
one = tkinter.Label(root0,width=300,height=150,bg="LightBlue")
one.pack()

IP0 = tkinter.StringVar()
IP0.set('')
USER = tkinter.StringVar()
USER.set('')

labelUSER = tkinter.Label(root0,text='Username',bg="LightBlue")
labelUSER.place(x=20,y=70,width=100,height=40)

entryUSER = tkinter.Entry(root0, width=60, textvariable=USER)
entryUSER.place(x=120,y=75,width=100,height=30)

def Login(*args):
    global IP, PORT, user
    IP, PORT = str('127.0.0.1:9999').split(':')
    user = entryUSER.get()
    if not user:
        tkinter.messagebox.showwarning('warning', message='Username blank!')
    else:
        root0.destroy()

loginButton = tkinter.Button(root0, text ="login", command = Login,bg="Yellow")
loginButton.place(x=135,y=110,width=40,height=25)
root0.bind('<Return>', Login)

root0.mainloop()


global s
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((IP, int(PORT)))
if user:
    s.send(user.encode())
else:
```

```
listbox1 = tkinter.Listbox(root1)
listbox1.place(x=510, y=0, width=130, height=320)


def send(*args):
    message = entryIuput.get() + '~' + user + '~' + chat
    s.send(message.encode())
    INPUT.set('')

sendButton = tkinter.Button(root1, text ="Send",anchor = 'n',command = send,font=('Helvetica', 15),bg = 'white')
sendButton.place(x=585,y=320,width=55,height=300)
root1.bind('<Return>', send)


def receive():
    global uses
    while True:
        data = s.recv(1024)
        data = data.decode()
        print(data)
        try:
            uses = json.loads(data)
            listbox1.delete(0, tkinter.END)
            listbox1.insert(tkinter.END, "Online")
            listbox1.insert(tkinter.END, "------Group chat-------")
            for x in range(len(uses)):
                listbox1.insert(tkinter.END, uses[x])
            users.append('------Group chat-------')
        except:
            data = data.split('~')
            message = data[0]
            userName = data[1]
            chatwith = data[2]
            message = '\n' + message
            if chatwith == '------Group chat-------':
                if userName == user:
                    listbox.insert(tkinter.END, message)
                else:
                    listbox.insert(tkinter.END, message)
            elif userName == user or chatwith == user:
                if userName == user:
                    listbox.tag_config('tag2', foreground='red')
                    listbox.insert(tkinter.END, message, 'tag2')
                else:
                    listbox.tag_config('tag3', foreground='green')
                    listbox.insert(tkinter.END, message,'tag3')

            listbox.see(tkinter.END)
r = threading.Thread(target=receive)

r.start()

root1.mainloop()
```

Serve.py

```python
import socket
import threading
import queue
import json
import os
import os.path
import sys


IP = '127.0.0.1'
PORT = 9999
messages = queue.Queue()
users = []
lock = threading.Lock()

def onlines():
    online = []
    for i in range(len(users)):
        online.append(users[i][0])
    return online

class ChatServer(threading.Thread):
    global users, que, lock

    def __init__(self):
        threading.Thread.__init__(self)
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        os.chdir(sys.path[0])

    def receive(self, conn, addr):
        user = conn.recv(1024)
        user = user.decode()
        if user == 'user does not exist':
            user = addr[0] + ':' + str(addr[1])
        tag = 1
        temp = user
        for i in range(len(users)):
            if users[i][0] == user:
                tag = tag + 1
                user = temp + str(tag)
        users.append((user, conn))
        USERS = onlines()
        self.Load(USERS,addr)

        try:
            while True:
                message = conn.recv(1024)
                message = message.decode()
                message = user + ':' + message
                self.Load(message,addr)
            conn.close()
        except:
            j = 0
            for man in users:
                if man[0] == user:
                    users.pop(j)
```

```
            USERS = onlines()
            self.Load(USERS,addr)
            conn.close()


    def Load(self, data, addr):
        lock.acquire()
        try:
            messages.put((addr, data))
        finally:
            lock.release()


    def sendData(self):
        while  True:
            if not messages.empty():
                message = messages.get()
                if isinstance(message[1], str):
                    for i in range(len(users)):
                        data = ' ' + message[1]
                        users[i][1].send(data.encode())
                        print(data)
                        print('\n')

                if isinstance(message[1], list):
                    data = json.dumps(message[1])
                    for i in range(len(users)):
                        try:
                            users[i][1].send(data.encode())
                        except:
                            pass

    def run(self):
        self.s.bind((IP,PORT))
        self.s.listen(5)
        q = threading.Thread(target=self.sendData)
        q.start()
        while True:
            conn, addr = self.s.accept()
            t = threading.Thread(target=self.receive, args=(conn, addr))
            t.start()
        self.s.close()
if __name__ == '__main__':
    cserver = ChatServer()
cserver.start()
```