E-402-STFO PROBLEMS FOR MODULE 4

CREATED BY HENNING ULFARSSON

This module is concerned with serial killers, cheating in card games and recognizing hand-written digits.

You get a perfect score for this module by getting 31 points or more.

1. Serial Killers (21 points)

There are various methods used to catch criminals. In this section we will look at how mathematics, modeling and programming can be used in *geoprofiling* and determining the residence of a serial killer.

Center of gravity. Richard Chase, also known as the Vampire of Sacremento, was a serial killer who murdered five people in January 1978. If a 30×30 grid is overlayed on the region he was active the body-drop sites are

```
richardChaseSites = [(17,3), (3,15), (27,19), (22,21) ,(18,25)]
```

He himself lived at `richardChaseResidence = (17,19)`. If we calculate the *center of gravity* of the drop sites by taking the average of the x-coordinates and the y-coordinates we get a reasonable estimate of his residence, see Figure 1.

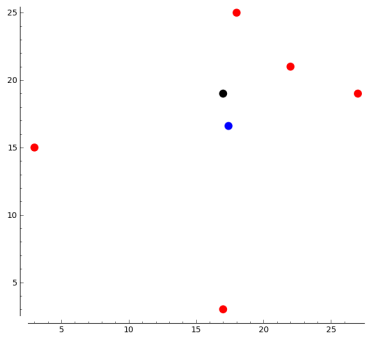

Figure 1. The drop sites are red, the Chase residence is black and the center of gravity is blue

The following code was used to produce the figure.

```
pts1 = points(sites, size = 100, color = "red")

if residence != -1:
    pts2 = point(residence, size = 100, color = "black")
else:
    pts2 = Graphics()

pts3 = points(m4p1(sites), size = 100, color = "blue")
show(pts1+pts2+pts3, aspect_ratio = 1)
```

Date: Updated October 4, 2013.

In the first problem your task is to implement a function to calculate the center of gravity and use it to guess the residence of some serial killers.

Problem 1 (5 points)**.** Write a function `m4p1(L)` that given a list of coordinates `L`, outputs the center of gravity.

```
Input: m4p1(richardChaseSites)
Output: (83/5,87/5)
```

Rossmo's equation. Now we want to do a bit better: Most serial killers exhibit the following two behaviors:

(1) Do not commit crimes too close to home/base of operation
(2) Will not travel farther than necessary to find victims

These axioms have a large amount of criminology research behind them (see e.g., Turner, 1969, "Delinquency and distance" and LeBeau, 1987, "Patterns of stranger and serial rape offending").

This can be used to create a model that extrapolates the killers residence from the locations of his crimes. The first such model was developed by Rossmo and considers the area under consideration as a grid of $n \times m$ cells and calculates the likelyhood of the criminal's residence lying within each cell. This is given by

$$P(x) = \sum_{\text{crime locations c}} \frac{\phi}{d(x,c)^f} + \frac{(1-\phi)B^{g-f}}{(2B - d(x,c))^g},$$

where $\phi = 1$ if $d(x,c) > B$, and 0 otherwise; x is an arbitrary cell in the grid; $d(x,c)$ is the distance from a cell to a crime location, with some fixed metric d; B is the radius of the buffer zone; and f, g are empirically tuned parameters.

We start by implementing some metric functions (to measure distances).

Problem 2 (2 points)**.** Write a function `m4p2(p,q)` that calculates the *Euclidean distance* between the input points. The inputs `p`, `q` should be tuples of integers. The output is a floating point number.

```
Input: p, q = (1,1), (3,2)
Run: m4p2(p,q)
Output: 2.2360679775
type(_): float
```

Problem 3 (2 points)**.** Write a function `m4p3(p,q)` that calculates the *Manhattan distance* between the input points. The inputs `p`, `q` should be tuples of integers. The output is a floating point number.

```
Input: p, q = (1,1), (3,2)
Run: m4p3(p,q)
Output: 3.0
type(_): float
```

Problem 4 (10 points)**.** Write a function

```
m4p4(sites, gridsize, B, f = 1/2, g = 1, d = m4p3)
```

that implements Rossmos's equation. Note that we give default values for the inputs `f`, `g` and `d`. The input `sites` should be a list of sites of interest, `gridsize` is a tuple giving the size of the grid and `B` is an integer giving the size of the buffer zone.

```
Input: sites = [(0,0),(2,2)]
       gridsize = (3,3)
       B = 1
Run: m4p4(sites, gridsize, B)
Output: [[1.0, 1.5773502691896257, 1.414213562373095],
         [1.5773502691896257, 1.414213562373095, 1.5773502691896257],
         [1.414213562373095, 1.5773502691896257, 1.0]]
```

Now we want to pick out the cell in the grid with the highest value and use that to guess the location of the serial killer.

Problem 5 (2 points)**.** Write a function

```
m4p5(A)
```

that finds a cell (or cells) in the matrix `A` with the highest Rossmo value.

```
Input: sites = [(0,0),(2,2)]
       gridsize = (3,3)
       B = 1
Run: m4p5(m4p4(sites, gridsize, B))
Output: set([(0, 1), (1, 2), (1, 0), (2, 1)])
```

We can use this to better guess the residence of Richard Chase, see Figure 2.

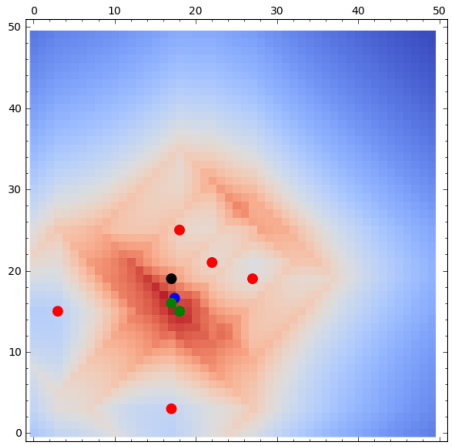

FIGURE 2. The drop sites are red, the Chase residence is black, the center of gravity is blue and the points with the highest Rossmo value is green

Here is the code that was used to produce Figure 2.

```
sites     = richardChaseSites
residence = richardChaseResidence

B = 10

A = m4p4(sites, (50,50), B)

pts1 = points(sites, size = 100, color = "red")
```

```
if residence != -1:
    pts2 = point(residence, size = 100, color = "black")
else:
    pts2 = Graphics()

pts3 = point(m4p1(sites), size = 100, color = "blue")
pts4 = points(m4p5(A), size = 100, color = "green")

plot1 = matrix_plot(matrix(A).transpose(), cmap = 'coolwarm',
xmin = 0, xmax = 50, ymin = 0, ymax = 50)

show(pts1+pts2+pts3+pts4+plot1)
```

Here are more datasets you can play with

(1) Albert deSalvo, the Boston strangler

```
deSalvoSites = [(48,10), (8,13), (11,15), (8,17), (7,18),
(9,18), (4,19), (8,19), (9,20), (20,10), (11,20), (23,29),
(28,33)]
deSalvoResidence = (18,19)
```

(2) Jack the Ripper, real name unknown

```
jackSites = [(1, 5), (69, 1), (15, 47), (84, 50), (28, 61)]
jackResidence = -1
```

Google map `http://goo.gl/maps/kygAc`. The point $(55, 20)$ is in Altab Ali Park.

(3) The Zodiac killer, real name unknown

```
zodiacSites = [(0,0), (26,31), (21,34), (18,82)]
zodiacResidence = -1
```

Google map `http://goo.gl/maps/prsa7`. The origin of the coordinate system is site where Paul Stine was shot by the Zodiac killer.

2. Card games (17 points)

The goal in this section is to explore *optimal stackings* of a deck of cards. Here we assume that we are playing a two player game and we are player one. The deck is optimally stacked if it does not matter how the deck is cut before dealing out the cards, we will always have a winning hand.

First consider a very simple version of poker, so-called Kuhn poker. Here the deck only has three cards: a king (K) a queen (Q) and a jack (J). If the deck starts as KQJ (read from the top) then we have three choices for a cut: at 0 (nothing happens), at 1 (deck becomes QJK) or at 2 (deck becomes JKQ). Assume we cut at 1 so the deck is now QJK. The dealer deals the top card to us (Q) and the second card to the second player (J). We win because a queen beats a jack. If the deck had been cut at 2 we would have lost (jack to a king). So this is not an optimally stacked deck.

In the first problem you explore whether optimal stackings exist for Kuhn poker.

Problem 6 (2 points)**.** Write a function `m4p6()` that outputs the optimal stackings of Kuhn poker. If the stacking KQJ we explored above was the only optimal stacking in Kuhn poker we should output `['KQJ']`. Note that if `JKQ` is in your output then

it is redundant to list its cuts, `KQJ` and `QJK`, in the output as well. So we adopt the convention to list the cut of an optimal deck that starts with `J`.

Now we complicate things a bit. K poker is another two-player game where there are nine cards: the 2,3,4,5,6,7,8,9,10 of hearts. Each player is dealt two cards, one at a time. Then the dealer places four community cards in the center of the table for either player to use. The best five-card poker hand wins, so here there are only two hands: a high card and a straight. It is a good exercise to check that the deck 23456789(10) is an optimal stacking for player two.

Problem 7 (15 points)**.** Write a function `m4p7()` that outputs the optimal stackings of K poker. Again, to avoid redundancy only list the cut of an optimal deck that starts with `2`.

3. Taxicabs and Cabtaxis (10 points)

Ramanujan famously told Hardy that 1729 is the smallest number expressible as a sum of two cubes in two different ways. Thus we define the $Taxicab(2) = 1729$. Then $Taxicab(n)$ is the smallest number expressible as a sum of two cubes in n different ways.

Problem 8 (5 points)**.** Write a function `m4p8(n)` that outputs the `n`-th Taxicab number.

```
Input: m4p8(2)
Output: 1729
```

Only the first six of these numbers are known.

Similarly, $Cabtaxi(n)$ the smallest number that can be formed by the sum and/or difference of two cubes in n different ways. For example $Cabtaxi(2) = 91$ since $91 = 3^3 + 4^3 = 6^3 - 5^3$.

Problem 9 (5 points)**.** Write a function `m4p9(n)` that outputs the `n`-th Cabtaxi number.

```
Input: m4p9(2)
Output: 91
```

Only the first ten of these numbers are known.

School of Computer Science, Reykjavik University, Menntavegi 1, 101 Reykjavík, Iceland

E-mail address: henningu@ru.is