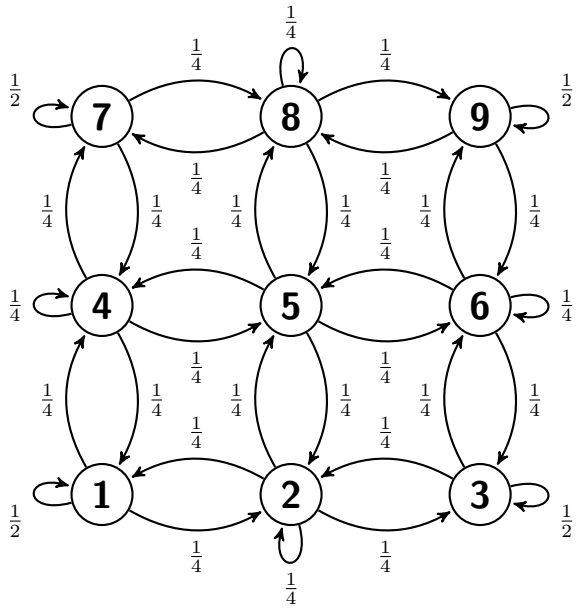
Natural Computation - Assignment 1

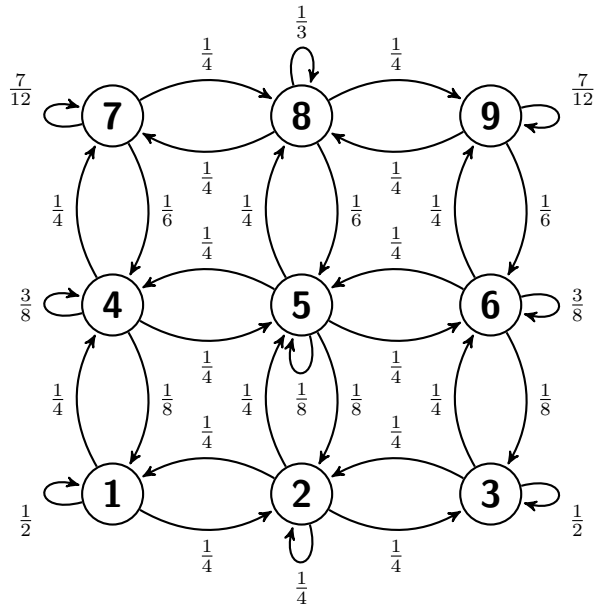
Stuart Stobie (ss835)

October 2015

1 Task 1



2 Task 2



3 Task 3

Cell 1	Cell 3	Cell 9
$0.246 \pm 0.004 (3sf)$	$0.0783 \pm 0.002 (3sf)$	0.0 ± 0.0

4 Task 4

Cell 1	Cell 3	Cell 9
$0.0574 \pm 0.008 (3sf)$	$0.0545 \pm 0.008 (3sf)$	$0.169 \pm 0.0118 (3sf)$

5 Task 5

```
# imports
from random import randint
from random import uniform
from statistics import mean
from statistics import stdev
# setup grid: grid[x][y] => grid[0][0]=1, grid[0][1]=4, grid[0][2]=7...
grid = [[y*3+x+1 for y in range(3)] for x in range(3)]
# probabilities to be in corresponding section of grid in task 2.
prob = [[(y+1)/18 for y in range(3)] for x in range(3)]

# returns object to store probabilities in.
def prob_obj():
    return { 1: [], 3: [], 9: [] }

# find the probabity of acceptance.
def acceptance(x, y, new_x, new_y):
    # cases off of the grid return 0
    if new_x < 0 or new_x > 2:
        return 0
    if new_y < 0 or new_y > 2:
        return 0
    # metropolis algorithm
    probability = prob[new_x][new_y] / prob[x][y]
    if probability >= 1:
        return 1
    return probability

# decide where to attempt to transition
def move(x, y):
    new_x = x
    new_y = y
    dir = randint(0,3)
    if dir == 0:
        new_y += 1 # North
    if dir == 1:
        new_x += 1 # East
    if dir == 2:
        new_y -= 1 # South
    if dir == 3:
        new_x -= 1 # West
    accept = acceptance(x, y, new_x, new_y)
    if accept == 0:
        return [x,y] # Stay
    if accept == 1:
        return [new_x,new_y] # Move
    if uniform(0.0, 1.0) <= accept:
        return [new_x,new_y]
    else:
        return [x,y]

def calc_prob(n, positions):
    return positions.count(n) / len(positions)

def mean_and_std_dev(numbers):
    return [mean(numbers), stdev(numbers)]
```

```
def pretty_print(probabilities):
    for i in range(3):
        n = 3**i # looks at numbers 1, 3 & 9
        print('Probability of ending in ' + str(n) + ': ' +
            str(probabilities[n][0]) + '    ' + str(probabilities[n][1]))

def task3():
    runs = 100 # repeats in order to find mean and standard deviation
    steps = 3 # time steps to perform
    repeats = 10000
    probabilities = prob_obj()
    for r in range(runs):
        end_positions = []
        for repeat in range(repeats):
            pos = [0,0] # start at 1
            for s in range(steps):
                pos = move(pos[0], pos[1])
            end_positions.append(grid[pos[0]][pos[1]])
        for i in range(3):
            n = 3**i # looks at numbers 1, 3 & 9
            probabilities[n].append(calc_prob(n, end_positions))
    for i in range(3):
        n = 3**i
        probabilities[n] = mean_and_std_dev(probabilities[n])
    print('Task 3')
    pretty_print(probabilities)

def task4():
    runs = 100 # repeats in order to find mean and standard deviation
    steps = 1000000 # time steps to perform
    rec = 1000 # number of time steps to take between recording position
    probabilities = prob_obj()
    for r in range(runs):
        rec_positions = []
        pos = [0,0] # start at 1
        for s in range(steps):
            pos = move(pos[0], pos[1])
            if s > 0 and (s+1)%rec == 0:
                rec_positions.append(grid[pos[0]][pos[1]])
        for i in range(3):
            n = 3**i # looks at numbers 1, 3 & 9
            probabilities[n].append(calc_prob(n, rec_positions))
    for i in range(3):
        n = 3**i
        probabilities[n] = mean_and_std_dev(probabilities[n])
    print('Task 4')
    pretty_print(probabilities)

task3()
task4()
```