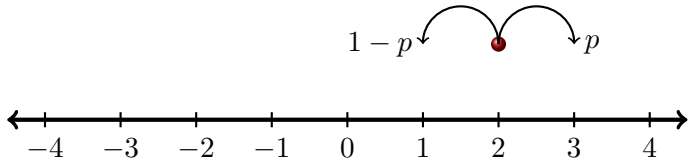
1 Background

In this session we will complete our study of the 1D random walk and move on to applications in higher dimensions. In particular we will prepare our way to study the properties of Brownian motion in 2D.

2 Problems

1. Modify the function `rwalkd1d` that you created last session, to account for the fact that the used coin is biased




```
def rwalk1d(N, p):
```

2. Run n random walks in one dimension and calculate the quantities

$$\langle x \rangle = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma = \left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \langle x \rangle)^2\right)^{1/2}$$

   Use $n = 100$, $N = 1000$ and $q = 0.5, 0.7, 0.2$, what do you observe?

3. Create an array h where the ith position is the number of times that the "drunk" in a 1D random walk ends up i meters away from the center (warning: the final position of the walker can be negative, whereas the position in the array cannot be smaller than 0)

4. Plot h as a function of i. Explain

3 Solutions

1.

```
# returns the result of the random walk when the result of flipping the coin
# is heads with probability p and tails with probability 1-p
def rwalk1d(N, p):

    s = 0
    for i in range(N):

        r = np.random.random()
        if r < p: # heads
            s = s + 1
        else:     # tails
            s = s - 1

    return s
```

2.

```
m = 0
s = 0

n = 100
N = 1000
p = 0.3

for i in range(n):

    x = rwalk1d(N, p)

    m = m + x
    s = s + x**2


m = m / float(n)
s = s / float(n)
s = np.sqrt(s - m**2)

print m, s

# The predicted values are
print N * (2 * p - 1), np.sqrt(4 * N * p * (1 - p))
```

3.

```
N = 1000
n = 1000
p = 0.7

x = np.arange(2 * N + 1) - N
h = np.zeros(2 * N + 1)
for i in range(n):
    xfinal = rwalk1d(N, p)
    h[xfinal + N] += 1
```

4.
```
plt.plot(x, h)
plt.show()
```