Binary Strings

Are humans good at generating random numbers? In this activity we will compare the list of run-lengths of binary strings generated by people and computers.

Run-length encoding is a simple method used to compress data in such a way that no information is lost (lossless). The method works by substituting a string of consecutive identical characters (a run) by the character and the length of the run. The list of run-lengths is the list of lengths without the matching characters.

Example: For message: `aaabbbbbcc`

The encoded message is `a3b5c2` and the list of run-lengths is given by `[3,5,2]`.

Goals:

1. Write a function `binString(n)` that, given a positive integer n, returns a randomly generated binary string of length n.
2. Write a function `rle(s)` that, given a binary string s, returns an `np.array` containing the run-lengths of s.
3. Write a function `binStringHist(s)` that, given a binary string s, produces a histogram of the run-lengths of s.
4. Apply the function `binStringHist(s)` to a randomly generated binary string of length 100 that you generate by hand.
5. Apply the function `binStringHist(s)` to a randomly generated binary string of length 100 generated using `binString(n)`.
6. Compare the results.
7. Upload your strings to the following Google document: `https://docs.google.com/document/d/1dtBQ9O9PKjndnv0y_Trujg0t31Jv_VP0BEC8MzG_5nA/edit?usp=sharing`

A solution:

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def binString(n):
    binS = np.random.randint(0,2,100)
    s = ''.join(map(str,binS))
    return s

def rle(s):
    current = s[0]
    count = 0
    runLength = []
    for char in s:
        if char == current:
            count += 1
        else:
            runLength.append(count)
            current = char
            count = 1
    return np.array(runLength)

def binStringHist(s):
    rl = rle(s)
    plt.hist(rl)
    plt.show()
```