# Threshold sample counts

From `gather_values`, we can group the results of the random samples. Now, we want to threshold those values based upon their frequency and value. Given `threshold=2`, we want to keep only those bitstrings with the two highest frequency counts of the `1` value. For example, as before we have,

```
x=['10', '11', '01', '00', '10', '00', '00', '11', '10', '00', '00', '01', '01', '11', '10', '00', '11', '10', '11', '11']
```

according to our last result, we had

```
{'10': [1, 1, 1, 1, 1],
'11': [1, 1, 1, 1, 1, 1],
 '01': [1, 1, 1],
'00': [0, 0, 0, 0, 0, 0]}
```

But because the `threshold=2`, we only want to keep the bitstrings with the two most frequent `1`s and set all of the rest to `0`. In this case, this is `10` and `11` with the following output:

```
{'10': 1,
  '11': 1,
  '01': 0,
 '00': 0}
```

Note that `01` corresponding value was set to `0` because it did not have the top two most frequent values of `1`. If there is a tie, then use the smallest value the tied bitstrings to pick the winner. Here is a more detailed example:

```
seq= ['1111', '0110', '1001', '0011', '0111', '0100', '0111', '1100', '0011', '0010', '0010', '1010', '1010', '1100',
'0110', '0101', '0110', '1111', '1001', '0110', '0010', '1101', '0101', '0010', '0100', '0010', '0000', '0000', '0011',
'0110', '0101', '1010', '1011', '1101', '1100', '0111', '1110', '0100', '0110', '1101', '0001', '1110', '0010', '0001',
'1010', '1010', '0011', '1000', '0010', '0000', '1010', '1101', '1111', '1000', '1000', '0010', '1010', '0101', '0101',
'1101', '0110', '1001', '1100', '1100', '1000', '1010', '0011', '0101', '0101', '0011', '0001', '1010', '0011', '0011',
'1101', '1010', '0101', '0011', '1011', '0101', '0000', '1111', '1001', '0101', '1100', '0011', '1111', '1101', '0001',
'1111', '1110', '1111', '0001', '0010', '0110', '0100', '0101', '1100', '1110', '1001']
```

With corresponding output from `threshold_values`,

```
>>> threshold_values(seq,3)
 {'0000': 0, '0001': 0, '0010': 0, '0011': 1, '0100': 0, '0101': 1, '0110': 0, '0111': 0, '1000': 0, '1001': 0, '1010': 1,
'1011': 0, '1100': 0, '1101': 0, '1110': 0, '1111': 0}
```

Your function signature is `threshold_values(seq,threshold=1)`. Please keep the default values as given in the function signature. Please put your Python code in a Python script file and upload it. Please retain your submitted source files! Remember to use all the best practices we discussed in class. You can use any module in the Python standard library, but third-party modules (e.g., Numpy, Pandas) are restricted to those **explicitly** mentioned in the problem description.

## Tips:

- After you have submitted your file, do **not** use the browser back or reload buttons to navigate or open the page in multiple browser tabs, as this may cause your `attempts` to decrease unexpectedly. It may take up to thirty seconds for your code to be processed, so please be **patient**.
- If you find yourself back at the main page without any feedback or change in your `attempts` then it means that your code timed out or crashed in some unexpected way.
- Ensure that your development environment does not presume the existence of certain packages for the autograder. The autograder does not have anything other than the standard library and those third-party libraries **explicitly** named in the problem description.
- Do not leave extraneous statements in your code like test cases, print statements, or anything else besides what is needed to evaluate your submission because the the autograder will spend its limited time executing those lines, which may result in unexpected crashes or timeouts.

浏览... 0404.py Upload Python source code file

Correct! Back to assignments. functional points = 7 /7 and validation points = 3/3