In [1]:

```
%matplotlib inline
from berVaz import *
```

In [2]:

```
#To simulate it on the qasm_simulator, use sim(n), Example:

test = sim('1001')
test
```

Out[2]:

```
(      ┌───┐      ░            ░ ┌───┐ ░ ┌─┐         
 q_0: ┤ H ├──────░───■────────░─┤ H ├─░─┤M├─────────
      ├───┤      ░   │        ░ ├───┤ ░ └╥┘┌─┐      
 q_1: ┤ H ├──────░───┼────────░─┤ H ├─░──╫─┤M├──────
      ├───┤      ░   │        ░ ├───┤ ░  ║ └╥┘┌─┐   
 q_2: ┤ H ├──────░───┼────────░─┤ H ├─░──╫──╫─┤M├───
      ├───┤      ░   │        ░ ├───┤ ░  ║  ║ └╥┘┌─┐
 q_3: ┤ H ├──────░───┼────■───░─┤ H ├─░──╫──╫──╫─┤M├
      ├───┤┌───┐ ░ ┌─┴─┐┌─┴─┐ ░ └───┘ ░  ║  ║  ║ └╥┘
 q_4: ┤ X ├┤ H ├─░─┤ X ├┤ X ├─░───────░──╫──╫──╫──╫─
      └───┘└───┘ ░ └───┘└───┘ ░       ░  ║  ║  ║  ║ 
 c: 4/═══════════════════════════════════╩══╩══╩══╩═
                                         0  1  2  3 ,

 {'1001': 1})
```

In [3]:

```
#or for output alone,

test[1]
```

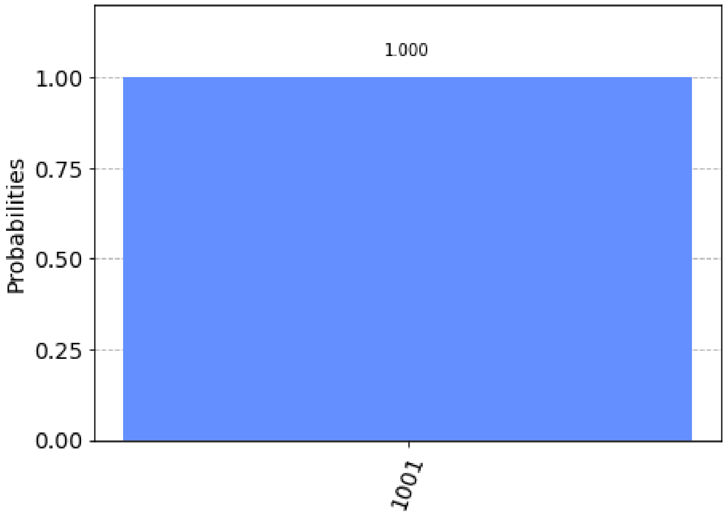
Out[3]:

```
{'1001': 1}
```

In [4]:

```
plot_histogram(test[1])
```

Out[4]:



In [10]:

```
#Now if you want to run this on a  real quantum computer, you can use the run(n, device) function.
If you dont specify
```

```
#the device, it will runn on ibmq_16_melbourne by default
#Note that you have to have the IBM token and everything set up before this because this module ca
n't affect any of
#that
test2 = run('1011')
```

```
ibmqfactory.load_account:WARNING:2020-09-11 05:19:46,302: Credentials are already in use. The exis
ting account in the session will be replaced.
```

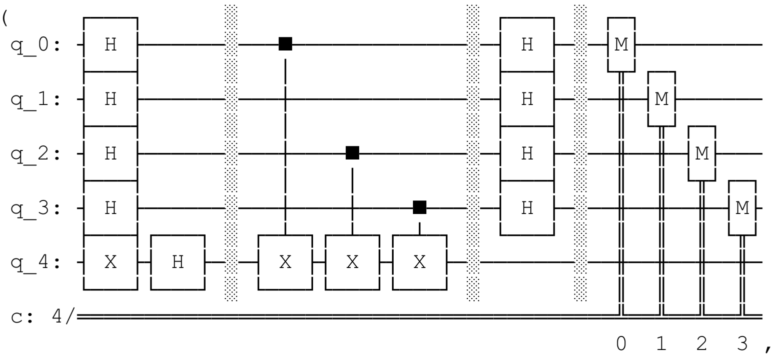
```
Job Status: job has successfully run
```

In [11]:

```
#This prints out the circuit in text and the result
test2
```

Out[11]:

(

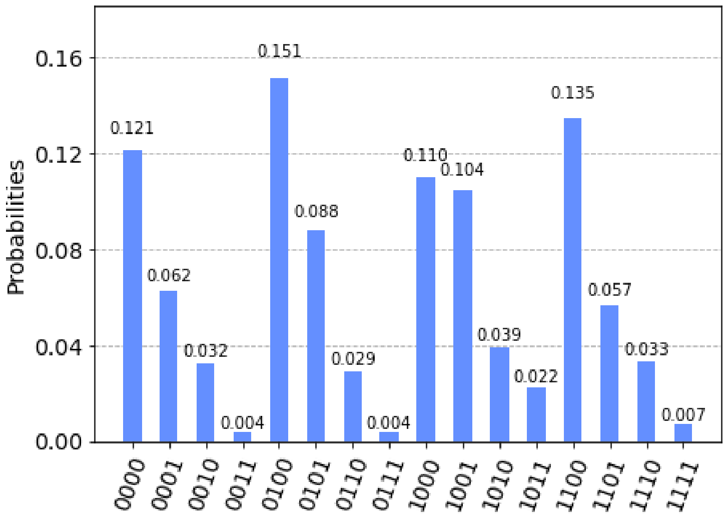


```
 ,
 <qiskit.result.result.Result at 0x129721730>,
 <qiskit.circuit.quantumcircuit.QuantumCircuit at 0x12961f550>)
```

In [13]:

```
#This is what you use to plot the result... It is very underwhelming. Loads of error-correction re
search required lol
plot_histogram(test2[1].get_counts(test2[2]))
```

Out[13]:



In [ ]: