Project 1: Quantum Computing Results

N. Raymundi Pinheiro

12 January 2023

1 Dirac Notation

1.1 Pretty Print Binary Numbers

The code was separated for better understanding. All of it is printed on the same line.
Input:

```
def prettyPrintBinary(state):
    # If state is empty:
    if (len(state) == 0):
        return '(  )'

    # Find out how many wires there are:
    numberOfWires = len(state[0][1])


    prettyPrint = '( '
    count = len(state)
    for element in state:
        count -= 1
        if (element[0] != 0.0 and count > 0):
            prettyPrint += str(element[0]) + '  |' + element[1] + '> + '

        if (count == 0):
            prettyPrint += str(element[0]) + '  |' + element[1] + '>'

    prettyPrint += ' )'

    return prettyPrint
```

Output:

```
state = [(np.sqrt(0.1)*1.j, '101'),
         (np.sqrt(0.5),     '000'),
         (-np.sqrt(0.4),    '010')]

prettyPrintBinary(state)
# Outputs: ( 0.31622776601683794j  |101>
#          + 0.7071067811865476  |000>
#          + -0.6324555320336759  |010> )
```

1.2 Pretty Print Integer Numbers

The code was separated for better understanding. All of it is printed on the same line.
Input:

```
def prettyPrintInteger(state):
    # If state is empty:
    if (len(state) == 0):
        print("State length is zero. Nothing to be done here.")
        return '(  )'

    prettyPrint = '( '
    count = len(state)
    for element in state:
        count -= 1
        if (element[0] != 0.0 and count > 0):
            prettyPrint += str(element[0]) + '  |' + \
                    str(binaryToInt(element[1])) + '> + '

        if (count == 0):
            prettyPrint += str(element[0]) + '  |' + \
                    str(binaryToInt(element[1])) + '>'

    prettyPrint += ' )'

    return prettyPrint
```

Output:

```
state = [(np.sqrt(0.1)*1.j, '101'),
         (np.sqrt(0.5),     '000'),
         (-np.sqrt(0.4),    '010')]

prettyPrintInteger(state)
# Outputs: ( 0.31622776601683794j  |5>
#          + 0.7071067811865476  |0>
#          + -0.6324555320336759  |2> )
```

1.3 State to Vector

Input:

```
def stateToVector(state):
    if (len(state) == 0):
        return  []

    vector = []
    numberOfWires = len(state[0][1])

    unordered = []
    for element in state:
        unordered.append((element[0], binaryToInt(element[1])))

    unordered = sorted(unordered, key=lambda x: x[1])

    count = 0
    for element in unordered:
        while (count != element[1]):
            vector.append(0.0)
            count += 1

        vector.append(element[0])
        count += 1

    while (count < (2 ** numberOfWires)):
        vector.append(0.0)
        count += 1

    return vector
```

Output:

```
state = [(np.sqrt(0.1)*1.j, '101'),
         (np.sqrt(0.5),     '000'),
         (-np.sqrt(0.4),    '010')]

stateToVector(state)
# Outputs: [0.7071067811865476, 0.0, -0.6324555320336759,
#           0.0, 0.0, 0.31622776601683794j, 0.0, 0.0]
```

1.4 Vector to State

Input:

```
def vectorToState(vector):
    state = []
    numberOfWires = np.log(len(vector)) / np.log(2)

    count = 0
    for element in vector:
        amplitude = element
        binary = prettyBinary(count, numberOfWires)

        if (amplitude != 0.0):
            pair = (amplitude, binary) # tuple
            state.append(pair)

        count += 1

    return state
```

Output:

```
state = [(np.sqrt(0.1)*1.j, '101'),
         (np.sqrt(0.5),     '000'),
         (-np.sqrt(0.4),    '010')]

vectorToState(stateToVector(state))
# Outputs: [(0.7071067811865476, '000'),
#           (-0.6324555320336759, '010'),
#           (0.31622776601683794j, '101')]
```

2 Quantum Computer Simulator II

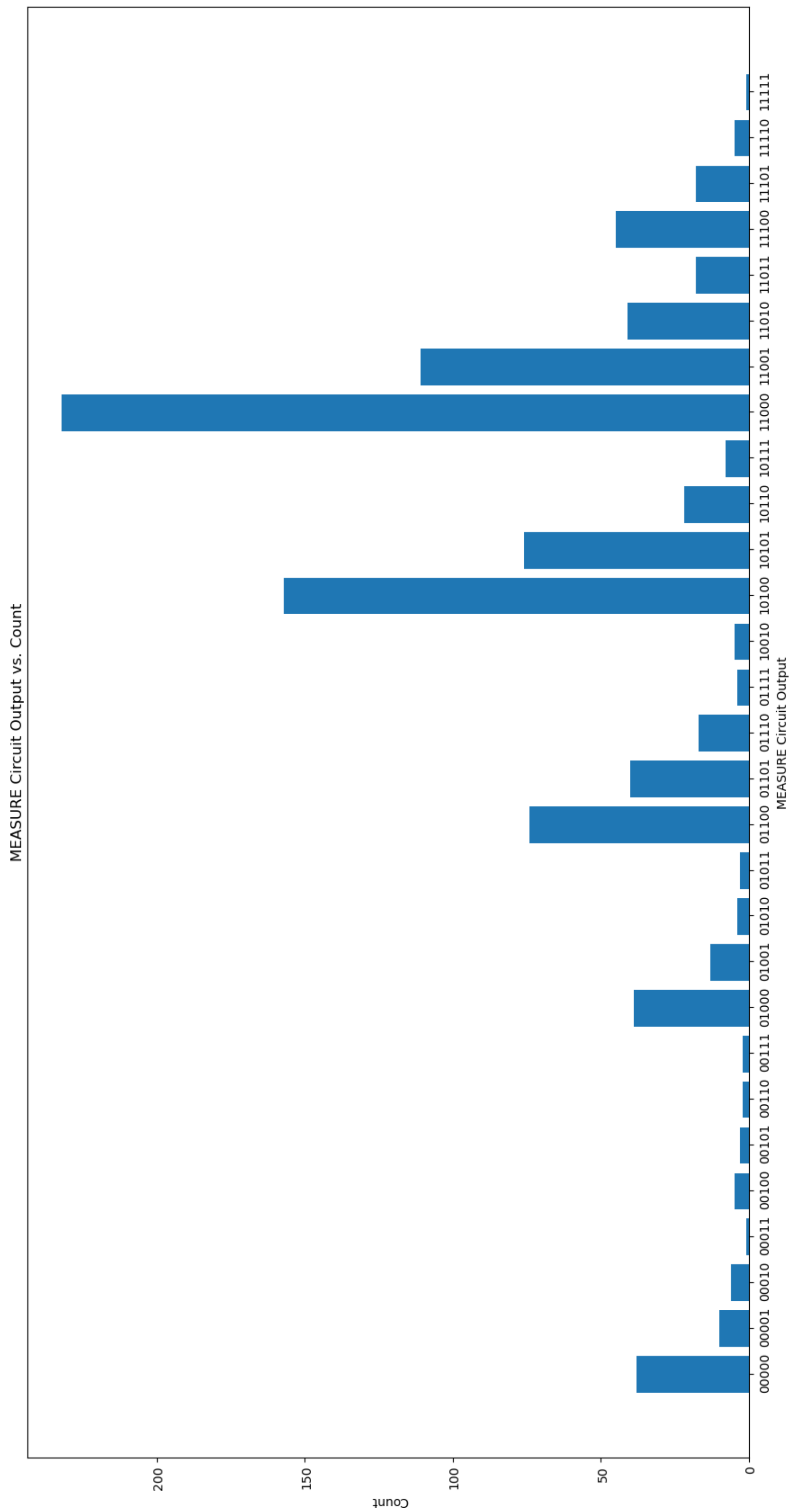
2.1 rand.circuit

Output:

```
[[(0.14565796004368065+0.11348294333323072j), '00000'],
[(-0.00017826157500443196+0.0800545938312926j), '00010'],
[(0.001474690001990138-0.036459032126468324j), '10000'],
[(-0.08205100536822751+0.015513990179495667j), '10010'],
[(0.04309755094683271-0.12184596055817094j), '00001'],
[(0.05035629868818654-0.007282489799443251j), '00011'],
[(-0.012673925486559936+0.005514371048822528j), '10001'],
[(0.01601173967169036+0.022594139337382274j), '10011'],
[(0.07232672559216345-0.022619270303906334j), '00100'],
[(0.053947952411923475+0.008954381028197851j), '00110'],
[(-0.1716561553464805+0.3669221138055029j), '10100'],
[(-0.14693154505266526-0.04361461090635174j), '10110'],
[(-0.034447848175386486-0.04222821092998379j), '00101'],
[(0.00179344586732939-0.026754015134267242j), '00111'],
[(0.27747135781393106+0.04232940722894612j), '10101'],
[(0.00740189352816481+0.10788583514551447j), '10111'],
[(0.13464628848368848-0.12560319656579136j), '01000'],
[(0.010652323173449822+0.07094564243091285j), '01010'],
[(-0.4426796395773186-0.08917051669782344j), '11000'],
[(-0.05478857583103472-0.21473166615430517j), '11010'],
[(-0.10409972085638466-0.06762381241864739j), '01001'],
[(0.031536789559932436-0.035377797720121625j), '01011'],
[(0.03432507377519409+0.3179714507952509j), '11001'],
[(-0.12098548283141346+0.06544840304773131j), '11011'],
[(0.007920474449196607+0.29789851220572544j), '01100'],
[(-0.11306530101030744-0.022843082802562503j), '01110'],
[(-0.18144336972286904-0.07802773562153242j), '11100'],
[(-0.05781065216328528+0.010123356493809629j), '11110'],
[(0.1960812681426865-0.05835344971980591j), '01101'],
[(0.025670653436356015+0.07563944999428006j), '01111'],
[(-0.005850147569994808+0.12703336730376008j), '11101'],
[(-0.01181308123705529+0.02004417137333421j), '11111']]
```

2.2 `measure.circuit`

Output:



2.3 input.circuit

Output:

```
[[(0.14565796004368065+0.11348294333323072j), '00000'],
[(-0.00017826157500443196+0.0800545938312926j), '00010'],
[(0.001474690001990138-0.036459032126468324j), '10000'],
[(-0.08205100536822751+0.015513990179495667j), '10010'],
[(0.04309755094683271-0.12184596055817094j), '00001'],
[(0.05035629868818654-0.007282489799443251j), '00011'],
[(-0.012673925486559936+0.005514371048822528j), '10001'],
[(0.01601173967169036+0.022594139337382274j), '10011'],
[(0.07232672559216345-0.022619270303906334j), '00100'],
[(0.053947952411923475+0.008954381028197851j), '00110'],
[(-0.1716561553464805+0.3669221138055029j), '10100'],
[(-0.14693154505266526-0.04361461090635174j), '10110'],
[(-0.034447848175386486-0.04222821092998379j), '00101'],
[(0.00179344586732939-0.026754015134267242j), '00111'],
[(0.27747135781393106+0.04232940722894612j), '10101'],
[(0.00740189352816481+0.10788583514551447j), '10111'],
[(0.13464628848368848-0.12560319656579136j), '01000'],
[(0.010652323173449822+0.07094564243091285j), '01010'],
[(-0.4426796395773186-0.08917051669782344j), '11000'],
[(-0.05478857583103472-0.21473166615430517j), '11010'],
[(-0.10409972085638466-0.06762381241864739j), '01001'],
[(0.031536789559932436-0.035377797720121625j), '01011'],
[(0.03432507377519409+0.3179714507952509j), '11001'],
[(-0.12098548283141346+0.06544840304773131j), '11011'],
[(0.007920474449196607+0.29789851220572544j), '01100'],
[(-0.11306530101030744-0.022843082802562503j), '01110'],
[(-0.18144336972286904-0.07802773562153242j), '11100'],
[(-0.05781065216328528+0.010123356493809629j), '11110'],
[(0.1960812681426865-0.05835344971980591j), '01101'],
[(0.025670653436356015+0.07563944999428006j), '01111'],
[(-0.005850147569994808+0.12703336730376008j), '11101'],
[(-0.01181308123705529+0.02004417137333421j), '11111']]
```

3 Non-Atomic Gates

3.1 NOT Gate

Input:

```
def notGate(initialState):
    state = [[1, initialState]]

    # NOT gates use H -> P -> H.
    return ignoreZeros(hadamard(0, phase(0, np.pi, hadamard(0, state))))

# This is:
INITSTATE BASIS |1>
H 0
P 0 3.141592653589793
H 0
```

Output:

```
notGate('0')    # Outputs: [[(1-0j), '1']]
notGate('1')    # Outputs: [[(1-0j), '0']]
```

3.2 R_z Gate

Input:

```
def rz(initialState, theta):
    # Rz gates use NOT -> P(-) -> NOT -> P(+).
    state = notGate(initialState)

    state = ignoreZeros(phase(0, - theta / 2, state))

    notGateOutput = notGate(state[0][1])
    state = [[state[0][0], notGateOutput[0][1]]]

    state = ignoreZeros(phase(0, theta / 2, state))

    return state

# This is:
INITSTATE BASIS |1>
H 0
P 0 3.141592653589793
H 0
P 0 -3.141592653589793
H 0
P 0 3.141592653589793
H 0
P 0 3.141592653589793
```

Output:

```
rz('0', np.pi)  # Outputs: [[-1j, '0']]
rz('1', np.pi)  # Outputs: [[1j, '1']]
```