IQUHack

February 2, 2025

[38]:
```
import QuantumRingsLib
from QuantumRingsLib import QuantumRegister, AncillaRegister,␣
 ↪ClassicalRegister, QuantumCircuit
from QuantumRingsLib import QuantumRingsProvider
from QuantumRingsLib import job_monitor
from QuantumRingsLib import JobStatus
from matplotlib import pyplot as plt
import numpy as np
```

[2]:
```
provider = QuantumRingsProvider(
    token='rings-200.8TW9mufBz1NptaCeUjJp9QEQyE7YyhKI',
    name='davidrm3@g.ucla.edu'
)
backend = provider.get_backend("scarlet_quantum_rings")
shots = 1024
```

[48]:
```
def plot_histogram (counts, title=""):
    """
    Plots the histogram of the counts

    Args:

        counts (dict):
            The dictionary containing the counts of states

        titles (str):
            A title for the graph.

    Returns:
        None

    """
    fig, ax = plt.subplots(figsize =(10, 7))
    plt.xlabel("States")
    plt.ylabel("Counts")
    mylist = [key for key, val in counts.items() for _ in range(val)]
```

```
    unique, inverse = np.unique(mylist, return_inverse=True)
    bin_counts = np.bincount(inverse)

    plt.bar(unique, bin_counts)

    maxFreq = max(counts.values())
    plt.ylim(ymax=np.ceil(maxFreq / 10) * 10 if maxFreq % 10 else maxFreq + 10)
    # Show plot
    plt.title(title)
    plt.show()
    return
```

[354]:
```
def swap(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array):
                Carry function positions
    Returns:
        None
    """

    qc.cx( b[n[0]], b[n[1]])
    qc.cx( b[n[1]], b[n[0]])
    qc.cx( b[n[0]], b[n[1]])

    return
```

[420]:
```
def cswap(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array):
                Carry function positions
    Returns:
        None
    """

    qc.ccx( b[n[0]], b[n[1]], b[n[2]])
    qc.ccx( b[n[0]], b[n[2]], b[n[1]])
    qc.ccx( b[n[0]], b[n[1]], b[n[2]])
```

```
    return
```

[235]:
```
def carry(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array):
                Carry function positions
    Returns:
        None
    """

    qc.ccx(b[n[1]], b[n[2]], b[n[3]])
    qc.cx( b[n[1]], b[n[2]])
    qc.ccx(b[n[0]], b[n[2]], b[n[3]])
    return

def inv_carry(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int):
                Starting position of carry function
    Returns:
        None
    """



    qc.ccx(b[n[0]], b[n[2]], b[n[3]])
    qc.cx(b[n[1]], b[n[2]])
    qc.ccx(b[n[1]], b[n[2]], b[n[3]])

    return
```

[236]:
```
def summer(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
```

```
                The target register
        n (int array):
                Starting position of carry function
    Returns:
        None
    """

    qc.cx(b[n[1]], b[n[2]])
    qc.cx(b[n[0]], b[n[2]])
    return

def inv_summer(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array):
                Starting position of carry function
    Returns:
        None
    """

    qc.cx(b[n[0]], b[n[2]])
    qc.cx(b[n[1]], b[n[2]])

    return
```

[237]:
```
def adder(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """



    carry(qc, b, [n[i] for i in [17,0,8,18]])
    carry(qc, b, [n[i] for i in [18,1,9,19]])
    carry(qc, b, [n[i] for i in [19,2,10,20]])
    carry(qc, b, [n[i] for i in [20,3,11,21]])
```

```
    carry(qc, b, [n[i] for i in [21,4,12,22]])
    carry(qc, b, [n[i] for i in [22,5,13,23]])
    carry(qc, b, [n[i] for i in [23,6,14,24]])
    carry(qc, b, [n[i] for i in [24,7,15,16]])


    qc.cx(b[n[7]], b[n[15]])

    summer(qc, b, [n[i] for i in [24,7,15]])
    inv_carry(qc, b, [n[i] for i in [23,6,14,24]])
    summer(qc, b, [n[i] for i in [23,6,14]])
    inv_carry(qc, b, [n[i] for i in [22,5,13,23]])
    summer(qc, b, [n[i] for i in [22,5,13]])
    inv_carry(qc, b, [n[i] for i in [21,4,12,22]])
    summer(qc, b, [n[i] for i in [21,4,12]])
    inv_carry(qc, b, [n[i] for i in [20,3,11,21]])
    summer(qc, b, [n[i] for i in [20,3,11]])
    inv_carry(qc, b, [n[i] for i in [19,2,10,20]])
    summer(qc, b, [n[i] for i in [19,2,10]])
    inv_carry(qc, b, [n[i] for i in [18,1,9,19]])
    summer(qc, b, [n[i] for i in [18,1,9]])
    inv_carry(qc, b, [n[i] for i in [17,0,8,18]])
    summer(qc, b, [n[i] for i in [17,0,8]])
```

[238]:
```
def inv_adder(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    inv_summer(qc, b, [n[i] for i in [17,0,8]])
    carry(qc, b, [n[i] for i in [17,0,8,18]])
    inv_summer(qc, b, [n[i] for i in [18,1,9]])
    carry(qc, b, [n[i] for i in [18,1,9,19]])
    inv_summer(qc, b, [n[i] for i in [19,2,10]])
    carry(qc, b, [n[i] for i in [19,2,10,20]])
    inv_summer(qc, b, [n[i] for i in [20,3,11]])
```

```
    carry(qc, b, [n[i] for i in [20,3,11,21]])
    inv_summer(qc, b, [n[i] for i in [21,4,12]])
    carry(qc, b, [n[i] for i in [21,4,12,22]])
    inv_summer(qc, b, [n[i] for i in [22,5,13]])
    carry(qc, b, [n[i] for i in [22,5,13,23]])
    inv_summer(qc, b, [n[i] for i in [23,6,14]])
    carry(qc, b, [n[i] for i in [23,6,14,24]])
    inv_summer(qc, b, [n[i] for i in [24,7,15]])

    qc.cx(b[n[7]], b[n[15]])

    inv_carry(qc, b, [n[i] for i in [24,7,15,16]])
    inv_carry(qc, b, [n[i] for i in [23,6,14,24]])
    inv_carry(qc, b, [n[i] for i in [22,5,13,23]])
    inv_carry(qc, b, [n[i] for i in [21,4,12,22]])
    inv_carry(qc, b, [n[i] for i in [20,3,11,21]])
    inv_carry(qc, b, [n[i] for i in [19,2,10,20]])
    inv_carry(qc, b, [n[i] for i in [18,1,9,19]])
    inv_carry(qc, b, [n[i] for i in [17,0,8,18]])
```

[247]:
```
def adder_mod(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """
```

```
    adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    swap(qc, b, [n[i] for i in [0,25]])
    swap(qc, b, [n[i] for i in [1,26]])
    swap(qc, b, [n[i] for i in [2,27]])
    swap(qc, b, [n[i] for i in [3,28]])
    swap(qc, b, [n[i] for i in [4,29]])
    swap(qc, b, [n[i] for i in [5,30]])
    swap(qc, b, [n[i] for i in [6,31]])
    swap(qc, b, [n[i] for i in [7,32]])
    inv_adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    qc.x(b[n[16]])
    qc.cx(b[n[16]], b[n[33]])
    qc.x(b[n[16]])
    # Here is where N would change
    qc.cx(b[n[33]], b[n[7]])
    qc.cx(b[n[33]], b[n[3]])
    qc.cx(b[n[33]], b[n[2]])
    qc.cx(b[n[33]], b[n[1]])
    qc.cx(b[n[33]], b[n[0]])
    # END
    adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    # Here is where N would change
    qc.cx(b[n[33]], b[n[7]])
    qc.cx(b[n[33]], b[n[3]])
    qc.cx(b[n[33]], b[n[2]])
    qc.cx(b[n[33]], b[n[1]])
    qc.cx(b[n[33]], b[n[0]])
    # END
    swap(qc, b, [n[i] for i in [0,25]])
    swap(qc, b, [n[i] for i in [1,26]])
    swap(qc, b, [n[i] for i in [2,27]])
    swap(qc, b, [n[i] for i in [3,28]])
    swap(qc, b, [n[i] for i in [4,29]])
    swap(qc, b, [n[i] for i in [5,30]])
    swap(qc, b, [n[i] for i in [6,31]])
    swap(qc, b, [n[i] for i in [7,32]])
    inv_adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    qc.cx(b[n[16]], b[n[33]])
    adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])

    return
```

[251]:
```
def inv_adder_mod(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    # Reversed sequence
    inv_adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    qc.cx(b[n[16]], b[n[33]])
    adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    swap(qc, b, [n[i] for i in [7,32]])
    swap(qc, b, [n[i] for i in [6,31]])
    swap(qc, b, [n[i] for i in [5,30]])
    swap(qc, b, [n[i] for i in [4,29]])
    swap(qc, b, [n[i] for i in [3,28]])
    swap(qc, b, [n[i] for i in [2,27]])
    swap(qc, b, [n[i] for i in [1,26]])
    swap(qc, b, [n[i] for i in [0,25]])
    # Here is where N would change
    qc.cx(b[n[33]], b[n[0]])
    qc.cx(b[n[33]], b[n[1]])
    qc.cx(b[n[33]], b[n[2]])
    qc.cx(b[n[33]], b[n[3]])
    qc.cx(b[n[33]], b[n[7]])
    # END
    inv_adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    # Here is where N would change
    qc.cx(b[n[33]], b[n[0]])
    qc.cx(b[n[33]], b[n[1]])
    qc.cx(b[n[33]], b[n[2]])
    qc.cx(b[n[33]], b[n[3]])
    qc.cx(b[n[33]], b[n[7]])
    # END
    qc.x(b[n[16]])
    qc.cx(b[n[16]], b[n[33]])
```

```
    qc.x(b[n[16]])
    adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    swap(qc, b, [n[i] for i in [7,32]])
    swap(qc, b, [n[i] for i in [6,31]])
    swap(qc, b, [n[i] for i in [5,30]])
    swap(qc, b, [n[i] for i in [4,29]])
    swap(qc, b, [n[i] for i in [3,28]])
    swap(qc, b, [n[i] for i in [2,27]])
    swap(qc, b, [n[i] for i in [1,26]])
    swap(qc, b, [n[i] for i in [0,25]])
    inv_adder(qc, b, [n[i] for i in␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]])
    return
```

[337]:
```
def mod_exp_y0(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    #2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[13]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[13]])

    #2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
```

```
    qc.ccx(b[n[0]], b[n[2]], b[n[14]])

    #2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[15]])

    #2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])

    #2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[10]])
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[10]])
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])

    #2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])

    #2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])
```

```
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])

    #2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
    qc.ccx(b[n[0]], b[n[8]], b[n[14]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
    qc.ccx(b[n[0]], b[n[8]], b[n[14]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])


    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.x(b[n[0]])



    return
```

[338]:
```
def mod_exp_y1(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
```

```
    Returns:
        None
    """

    #2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])

    #2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[13]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[13]])

    #2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])

    #2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[14]])
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[14]])
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])

    #2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])

    #2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[10]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
```

```
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[10]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])

    #2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])

    #2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[12]])
    qc.ccx(b[n[0]], b[n[8]], b[n[14]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[12]])
    qc.ccx(b[n[0]], b[n[8]], b[n[14]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])


    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.x(b[n[0]])



    return
```

[382]:
```
def mod_exp_y2(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    #2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])

    #2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])

    #2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])

    #2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])

    #2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])

    #2^5
```

```
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])

    #2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[15]])

    #2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[16]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[16]])

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.x(b[n[0]])



    return
```

[383]:
```
def mod_exp_y3(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """
```

```
    #2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])

    #2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])

    #2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])

    #2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])

    #2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])

    #2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])

    #2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[15]])
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[15]])

    #2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[16]])
```

```
    adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[16]])

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.x(b[n[0]])



    return
```

[407]:
```
def inv_mod_exp_y0(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    # Reversed sequence

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.x(b[n[0]])
```

```
    # Reverse 2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[11]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[11]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])

    # Reverse 2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[16]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[16]])

    # Reverse 2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[9]])
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[16]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[9]])
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[16]])

    # Reverse 2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[10]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    qc.ccx(b[n[0]], b[n[5]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[10]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
    qc.ccx(b[n[0]], b[n[5]], b[n[13]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    qc.ccx(b[n[0]], b[n[5]], b[n[15]])
```

```
    # Reverse 2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[11]])
    qc.ccx(b[n[0]], b[n[4]], b[n[14]])
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[11]])
    qc.ccx(b[n[0]], b[n[4]], b[n[14]])
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])

    # Reverse 2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[9]])
    qc.ccx(b[n[0]], b[n[3]], b[n[10]])
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[12]])
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[9]])
    qc.ccx(b[n[0]], b[n[3]], b[n[10]])
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[12]])
    qc.ccx(b[n[0]], b[n[3]], b[n[13]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])

    # Reverse 2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[11]])
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[13]])
    qc.ccx(b[n[0]], b[n[2]], b[n[14]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[11]])
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[13]])
    qc.ccx(b[n[0]], b[n[2]], b[n[14]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])
```

```
    # Reverse 2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[15]])
    qc.ccx(b[n[0]], b[n[1]], b[n[14]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[15]])
    qc.ccx(b[n[0]], b[n[1]], b[n[14]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])



    return
```

[385]:
```
def inv_mod_exp_y1(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """

    # Reversed sequence

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.x(b[n[0]])

    # Reverse 2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[11]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
```

```
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[11]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
    qc.ccx(b[n[0]], b[n[8]], b[n[15]])

    # Reverse 2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[9]])
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[12]])
    qc.ccx(b[n[0]], b[n[7]], b[n[13]])

    # Reverse 2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[10]])
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[13]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[10]])
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[13]])
    qc.ccx(b[n[0]], b[n[6]], b[n[14]])

    # Reverse 2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[11]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    qc.ccx(b[n[0]], b[n[5]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[11]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
    qc.ccx(b[n[0]], b[n[5]], b[n[14]])
    qc.ccx(b[n[0]], b[n[5]], b[n[15]])

    # Reverse 2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
```

```
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[9]])
    qc.ccx(b[n[0]], b[n[4]], b[n[12]])
    qc.ccx(b[n[0]], b[n[4]], b[n[15]])

    # Reverse 2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[9]])
    qc.ccx(b[n[0]], b[n[3]], b[n[10]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[9]])
    qc.ccx(b[n[0]], b[n[3]], b[n[10]])

    # Reverse 2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[11]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[10]])
    qc.ccx(b[n[0]], b[n[2]], b[n[11]])

    # Reverse 2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[11]])
    qc.ccx(b[n[0]], b[n[1]], b[n[12]])

    return
```

[386]:
```
def inv_mod_exp_y2(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
    Returns:
        None
    """
```

```
    # Reversed sequence

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.x(b[n[0]])

    # Reverse 2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])

    # Reverse 2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])

    # Reverse 2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[12]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[12]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])

    # Reverse 2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
```

```
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])

    # Reverse 2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])

    # Reverse 2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])

    # Reverse 2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])

    # Reverse 2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])

    return
```

[427]:
```
def inv_mod_exp_y3(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
                The target register
        n (int array of positions):
                Starting position of carry function
```

```
    Returns:
        None
    """

    # Reversed sequence

    qc.x(b[n[0]])
    qc.ccx(b[n[0]], b[n[8]], b[n[24]])
    qc.ccx(b[n[0]], b[n[7]], b[n[23]])
    qc.ccx(b[n[0]], b[n[6]], b[n[22]])
    qc.ccx(b[n[0]], b[n[5]], b[n[21]])
    qc.ccx(b[n[0]], b[n[4]], b[n[20]])
    qc.ccx(b[n[0]], b[n[3]], b[n[19]])
    qc.ccx(b[n[0]], b[n[2]], b[n[18]])
    qc.ccx(b[n[0]], b[n[1]], b[n[17]])
    qc.x(b[n[0]])

    # Reverse 2^7
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[8]], b[n[9]])
    qc.ccx(b[n[0]], b[n[8]], b[n[10]])
    qc.ccx(b[n[0]], b[n[8]], b[n[13]])

    # Reverse 2^6
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[7]], b[n[10]])
    qc.ccx(b[n[0]], b[n[7]], b[n[11]])
    qc.ccx(b[n[0]], b[n[7]], b[n[14]])

    # Reverse 2^5
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[12]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[6]], b[n[11]])
    qc.ccx(b[n[0]], b[n[6]], b[n[12]])
    qc.ccx(b[n[0]], b[n[6]], b[n[15]])
```

```
    # Reverse 2^4
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[5]], b[n[9]])
    qc.ccx(b[n[0]], b[n[5]], b[n[12]])

    # Reverse 2^3
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[4]], b[n[10]])
    qc.ccx(b[n[0]], b[n[4]], b[n[13]])

    # Reverse 2^2
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[3]], b[n[11]])
    qc.ccx(b[n[0]], b[n[3]], b[n[14]])

    # Reverse 2^1
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[2]], b[n[12]])
    qc.ccx(b[n[0]], b[n[2]], b[n[15]])

    # Reverse 2^0
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])
    inv_adder_mod(qc, b, [n[i] for i in␣
↪[9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,3
    qc.ccx(b[n[0]], b[n[1]], b[n[9]])


    return
```

[435]:
```
def modular_exponentiation(qc, b, n):
    """
    Args:
        qc (QuantumCircuit):
                The quantum circuit
        b (QuantumRegister):
```

```
            The target register
        n (int array of positions):
            Starting position of carry function
    Returns:
        None
    """

    mod_exp_y0(qc, b, [n[i] for i in␣
↪[0,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3
    cswap(qc, b, [n[i] for i in [0,4,20]])
    cswap(qc, b, [n[i] for i in [0,5,21]])
    cswap(qc, b, [n[i] for i in [0,6,22]])
    cswap(qc, b, [n[i] for i in [0,7,23]])
    cswap(qc, b, [n[i] for i in [0,8,24]])
    cswap(qc, b, [n[i] for i in [0,9,25]])
    cswap(qc, b, [n[i] for i in [0,10,26]])
    cswap(qc, b, [n[i] for i in [0,11,27]])
    inv_mod_exp_y0(qc, b, [n[i] for i in␣
↪[0,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3

    mod_exp_y1(qc, b, [n[i] for i in␣
↪[1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3
    cswap(qc, b, [n[i] for i in [1,4,20]])
    cswap(qc, b, [n[i] for i in [1,5,21]])
    cswap(qc, b, [n[i] for i in [1,6,22]])
    cswap(qc, b, [n[i] for i in [1,7,23]])
    cswap(qc, b, [n[i] for i in [1,8,24]])
    cswap(qc, b, [n[i] for i in [1,9,25]])
    cswap(qc, b, [n[i] for i in [1,10,26]])
    cswap(qc, b, [n[i] for i in [1,11,27]])
    inv_mod_exp_y1(qc, b, [n[i] for i in␣
↪[1,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3

    mod_exp_y2(qc, b, [n[i] for i in␣
↪[2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3
    cswap(qc, b, [n[i] for i in [2,4,20]])
    cswap(qc, b, [n[i] for i in [2,5,21]])
    cswap(qc, b, [n[i] for i in [2,6,22]])
    cswap(qc, b, [n[i] for i in [2,7,23]])
    cswap(qc, b, [n[i] for i in [2,8,24]])
    cswap(qc, b, [n[i] for i in [2,9,25]])
    cswap(qc, b, [n[i] for i in [2,10,26]])
    cswap(qc, b, [n[i] for i in [2,11,27]])
    inv_mod_exp_y2(qc, b, [n[i] for i in␣
↪[2,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3
```

```
    mod_exp_y3(qc, b, [n[i] for i in␣
↪[3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3
    cswap(qc, b, [n[i] for i in [3,4,20]])
    cswap(qc, b, [n[i] for i in [3,5,21]])
    cswap(qc, b, [n[i] for i in [3,6,22]])
    cswap(qc, b, [n[i] for i in [3,7,23]])
    cswap(qc, b, [n[i] for i in [3,8,24]])
    cswap(qc, b, [n[i] for i in [3,9,25]])
    cswap(qc, b, [n[i] for i in [3,10,26]])
    cswap(qc, b, [n[i] for i in [3,11,27]])
    inv_mod_exp_y3(qc, b, [n[i] for i in␣
↪[3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,3

    return
```

[445]:
```
numberofqubits = 46
shots = 1024

q = QuantumRegister(numberofqubits , 'q')
c = ClassicalRegister(8 , 'c')
qc = QuantumCircuit(q, c)

qc.x(q[37])
qc.x(q[38])
qc.x(q[39])
qc.x(q[40])
qc.x(q[44])

qc.h(q[0])
qc.h(q[1])
qc.h(q[2])
qc.h(q[3])

qc.x(q[4])




modular_exponentiation(qc, q,␣
↪[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,3




qc.measure(q[4], c[0])
qc.measure(q[5], c[1])
qc.measure(q[6], c[2])
```

```
qc.measure(q[7], c[3])
qc.measure(q[8], c[4])
qc.measure(q[9], c[5])
qc.measure(q[10], c[6])
qc.measure(q[11], c[7])


# Execute the circuit
job = backend.run(qc, shots = shots)
job_monitor(job)
result = job.result()
counts = result.get_counts()

#clean up
del q, c, qc
del result
del job

#visualize
plot_histogram(counts)
```

```
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
```

```
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Running
Job Done.
Ending Job Monitor
```