Lab3: Shor and Grover Algorithm

3220106039 李瀚轩

一、Shor 算法

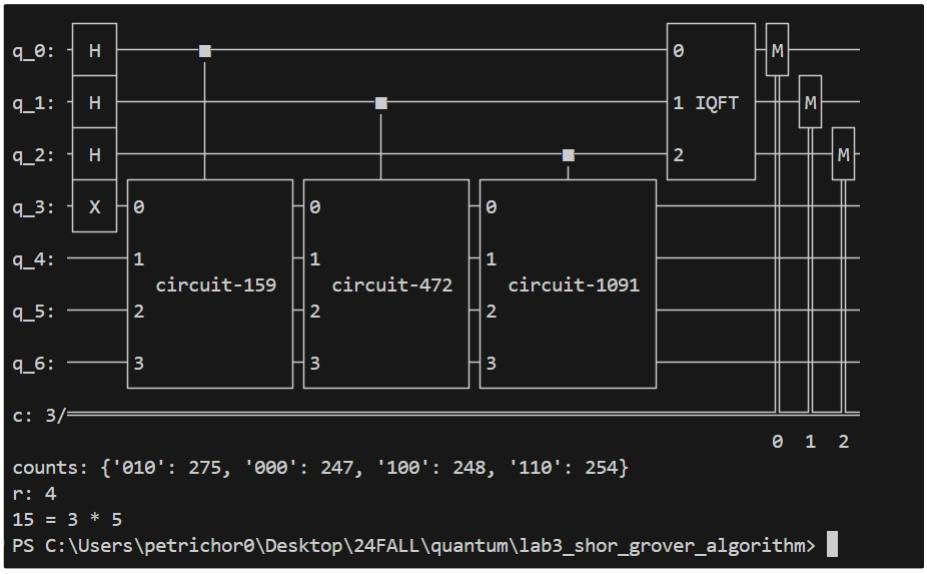
Shor 算法是一种量子算法，用于分解大整数 N，其核心思想是利用量子计算对周期性问题进行求解，从而找到 N 的因子。Shor 算法的主要步骤如下：

- 随机选择一个数 a，使得 a 和 N 互质。
- 通过量子相位估计算法计算 a 的阶 r，使得 $a^r \equiv 1 \mod N$。
- 利用阶 r 来计算 N 的因子。

对于我们需要实现的 `mod_circuit` 函数，目标是构建一个量子电路，对输入的量子比特执行模乘操作。该函数中，我们使用了一个矩阵来表示模乘操作。矩阵的构建方法如下：创建一个大小为 $2^{n_v} \times 2^{n_v}$ 的矩阵，其中 n_v 是值寄存器的量子比特数。填充矩阵，使其实现模 N 运算，具体通过 $(a \times i) \mod N$ 来计算。对于大于 N 的元素，保持单位矩阵的形式，以确保量子电路的正确性。代码如下：

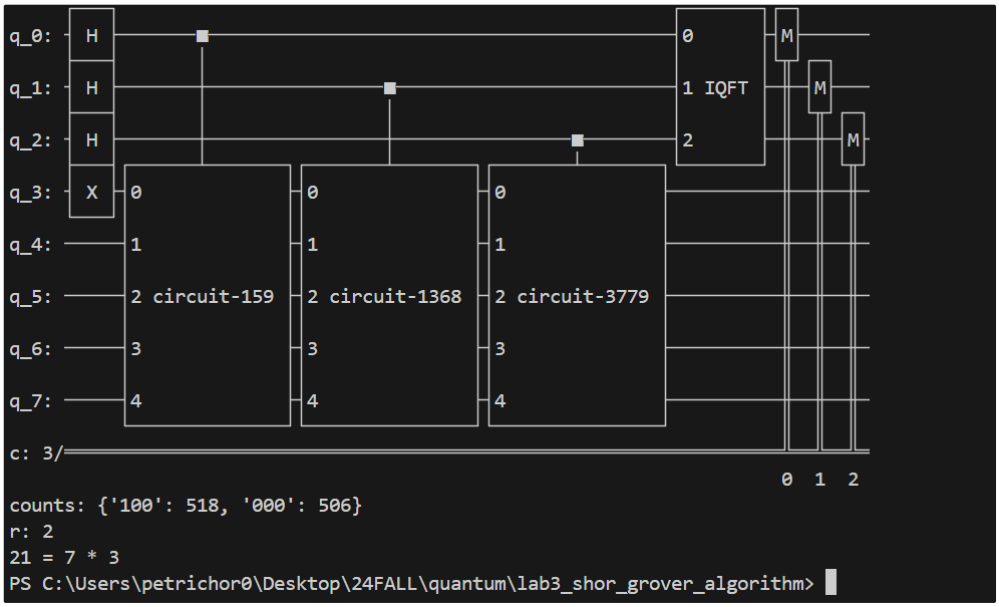
```
def mod_circuit(a, N, n_v):
    matrix = np.zeros((2 ** n_v, 2 ** n_v), dtype=int)
    #Todo
    for i in range(N):
        matrix[(a * i) % N][i] = 1
    for i in range(N, 2 ** n_v):
        matrix[i][i] = 1
    return UnitaryGate(matrix)
```

运行补全的代码，取 $a = 7$，分解整数 $N = 15$， 我们可以得到以下输出：



可以发现测量结果为 0,0.25,0.5,0.75 四个概率相等的相位，并且分解的结果也符合预期。

接下来我们分解 $N = 21$，修改参数 $a = 8, n_v = 5$，得到代码的结果如下：



可以发现结果符合预期。

二、Grover 算法

2.1 Oracle 目标态分析

`oracle()` 函数定义了一个四量子比特的量子电路，并且使用了 `cz(0, 3)` 操作。体来说，`qc.cz(0, 3)` 会对量子比特 0 和量子比特 3 之间施加控制-Z 门。

2.2 基于 `qiskit` 实现 Grover 算法

Grover 算法的基本步骤如下：

1. **初始化**：将所有量子比特初始化为均匀叠加态。
2. **Oracle**：通过 Oracle 对目标态进行标记。
3. **扩展算符**：扩展算符（Diffusion Operator）对状态进行幅度放大。
4. **迭代**：重复执行 Oracle 和扩展算符，以增加目标态的概率幅度。

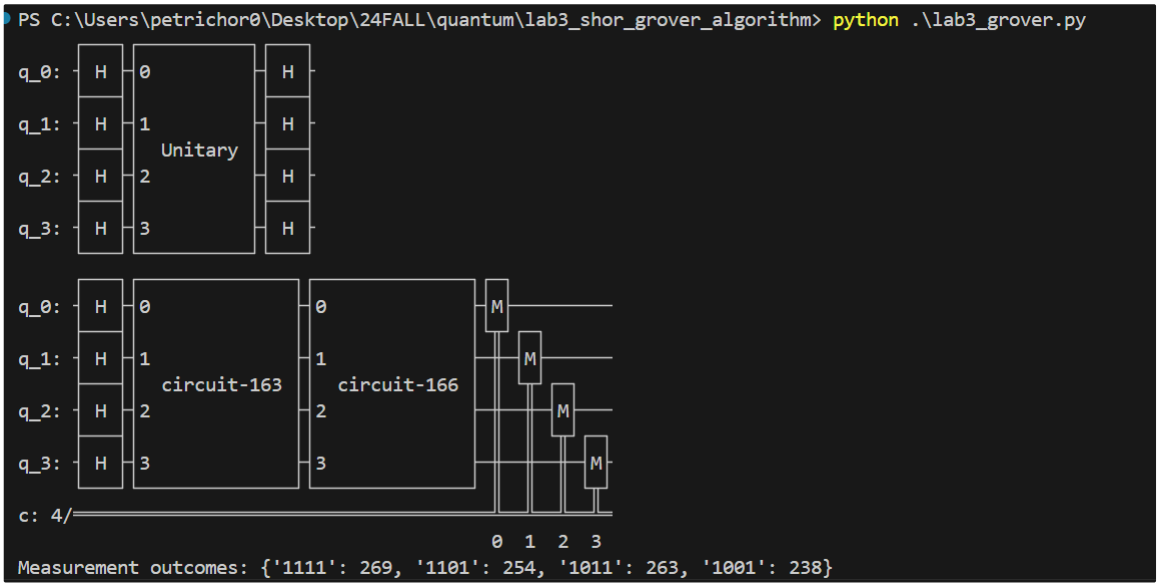
代码如下：

```
import numpy as np
from qiskit import QuantumCircuit, transpile
from qiskit_aer import Aer
from qiskit.circuit.library import UnitaryGate

def shift_operator(n):
    matrix = np.eye(2 ** n, dtype=int)
    matrix[0][0] = -1
    return UnitaryGate(matrix)

def diffusion_operator(n):
    qc = QuantumCircuit(n)
    qc.h(range(n))
    qc.append(shift_operator(n), range(n))
    qc.h(range(n))
    print(qc.draw())
    return qc
```

```python
def oracle():
    qc = QuantumCircuit(4)
    qc.cz(0, 3)
    return qc
# def oracle(n=4):
#     matrix = np.eye(2 ** n, dtype=int)
#     matrix[0][0] = -1
#     return UnitaryGate(matrix)

def grover_circuit(n=4, iterations=1):
    qc = QuantumCircuit(n, n)
    qc.h(range(n))

    for _ in range(iterations):
        qc.append(oracle(), range(n))
        qc.append(diffusion_operator(n), range(n))

    qc.measure(range(n), range(n))
    print(qc.draw())
    return qc

def run_grover():
    qc = grover_circuit(n=4, iterations=1)
    simulator = Aer.get_backend('qasm_simulator')
    transpiled_qc = transpile(qc, simulator)
    result = simulator.run(transpiled_qc, shots=1024).result()
    counts = result.get_counts()
    print("Measurement outcomes:", counts)

run_grover()
```

Grover 算法的核心是幅度的放大过程，本次实验总共有 16 个态，初态可以表示为 $\frac{\sqrt{3}}{2}|\alpha\rangle + \frac{1}{2}|\beta\rangle$. 初始夹角为 30 度，因此每次迭代旋转 60 度，迭代一次即可得到目标态，因此迭代次数为 1.

运行代码得到结果：



然后我们将目标态改成 $|0000\rangle$，我们首先得修改 oracle：

```python
def oracle(n=4):
    matrix = np.eye(2 ** n, dtype=int)
    matrix[0][0] = -1
    return UnitaryGate(matrix)
```

这时候，初始的夹角约为 14.5 度，每次迭代会向目标态旋转 29 度。

我们依次选择迭代次数为 1，2，3，观察结果。我们的预期是 0000 的概率越来越大：

```
Measurement outcomes: {'0000': 499, '1001': 36, '1111': 37, '0111': 29, '1011': 34, '1101': 38, '0100': 44, '0011': 39, '1010': 29,
0110': 34, '1100': 37, '0101': 41, '0010': 32, '1000': 31, '0001': 29, '1110': 35}
PS C:\Users\petrichor0\Desktop\24FALL\quantum\lab3_shor_grover_algorithm>
```

```
Measurement outcomes: {'0000': 916, '1101': 9, '0001': 10, '1001': 13, '0011': 9, '1111': 5, '0110': 3, '1011': 10, '1010': 8, '0101'
: 9, '0010': 4, '0100': 7, '1100': 4, '0111': 7, '1110': 6, '1000': 4}
PS C:\Users\petrichor0\Desktop\24FALL\quantum\lab3_shor_grover_algorithm>
```

```
Measurement outcomes: {'1000': 3, '0000': 977, '1001': 7, '1010': 2, '1110': 4, '0100': 2, '1100': 2, '1011': 4, '0110': 4, '0101': 5
, '1111': 4, '0001': 3, '0010': 4, '1101': 1, '0011': 1, '0111': 1}
PS C:\Users\petrichor0\Desktop\24FALL\quantum\lab3_shor_grover_algorithm>
```

可以看见符合预期。但是当迭代次数为 5 的时候，可以看见概率减小，因为我们已经旋转过去了，离目标态会越来越远：

```
Measurement outcomes: {'1000': 52, '0000': 142, '0010': 57, '0100': 67, '1101': 68, '1110': 68, '1111': 57, '1100': 59, '0011': 58, '
0001': 67, '0111': 46, '1010': 53, '0101': 76, '0110': 61, '1011': 49, '1001': 44}
PS C:\Users\petrichor0\Desktop\24FALL\quantum\lab3_shor_grover_algorithm>
```

结果均符合我们的预期。