Lab1: Quantum Circuit Simulation

3220106039 李瀚轩

一、`qubit-simulator` 源码分析

1.1 基本原理

`qubit-simulator` 的核心原理是通过状态向量（state vector）来表示量子系统的状态，并使用矩阵运算来实现量子门的作用。状态向量的每个元素表示量子系统的某个特定状态的概率振幅。

1.2 主要代码结构

主要是 `smulator.py` 中的 `QubitSimulator` 类，表示量子比特的模拟器，支持多种量子门的应用。

一些比较重要的变量有：

- `num_qubits`：模拟器中的量子比特数量。
- `state_vector`：用于存储量子比特的状态向量，初始化为 $|0\rangle$ 态。
- `circuit`：用于记录已应用的量子门和其对应的量子比特。

除此之外，在 `Gates.py` 中封装了量子计算中的基本操作，使用 NumPy 数组来表示各种量子门的矩阵形式。类中包括了 Hadamard 门、π/8 门和 Pauli-X (NOT) 门，以及几个静态方法用于创建通用门、受控门和逆门。并且给出了验证量子门的方法，确保量子门是否是有效的单位矩阵，确保 $UU^+ = I$。

1.3 运行流程

- 初始化
  - 在初始化时，`QubitSimulator` 会接收一个参数 `num_qubits`，代表量子比特的数量。此时，状态向量会初始化为一个复数数组，其中第一个元素为 1，表示量子系统处于 |0〉状态，其余元素为 0。
  - ```
    self.num_qubits = num_qubits
    self.state_vector = np.zeros(2**num_qubits, dtype=complex)
    self.state_vector[0] = 1
    ```
- 量子门的应用
  - 量子门的应用通过 `_apply_gate` 方法实现。该方法首先验证目标量子比特的索引，然后生成量子门的操作矩阵，最后通过矩阵乘法更新状态向量。
  - 其中门的定义在 `gates.py` 中。
- 测量
  - 测量功能通过 `measure` 和 `run` 方法实现。`measure` 方法计算量子状态的概率，并根据概率进行测量，返回结果。`run` 方法用于执行多次测量并返回结果的统计信息。
- 可视化
  - 通过 `plot_wavefunction` 方法，可以可视化当前状态向量的幅度和相位，绘制出相应的相位圆图。
- 状态重置
    ```

- `reset` 方法允许用户将模拟器重置到初始状态，即将状态向量和电路记录清空。

二、模拟量子电路

根据 `QubitSimulator` 类以及 GHZ 的结构很容易能够模拟该电路，代码如下：

```
from qubit_simulator import QubitSimulator
import matplotlib.pyplot as plt

simulator = QubitSimulator(num_qubits=5)

simulator.h(target_qubit=0)
simulator.cx(control_qubit=0, target_qubit=1)
simulator.cx(control_qubit=1, target_qubit=2)
simulator.cx(control_qubit=2, target_qubit=3)
simulator.cx(control_qubit=3, target_qubit=4)

print(simulator.run(shots=1000))
print(simulator)
```

可以看到模拟出来的结果符合预期：

```
PS C:\Users\petrichor0\Desktop\24FALL\quantum\lab1_quantum_circuit_simulation> python lab1.py
{'00000': 500, '11111': 500}
--------------------
| H | @ |   |   |   |
|   | X | @ |   |   |
|   |   | X | @ |   |
|   |   |   | X | @ |
|   |   |   |   | X |
--------------------
```

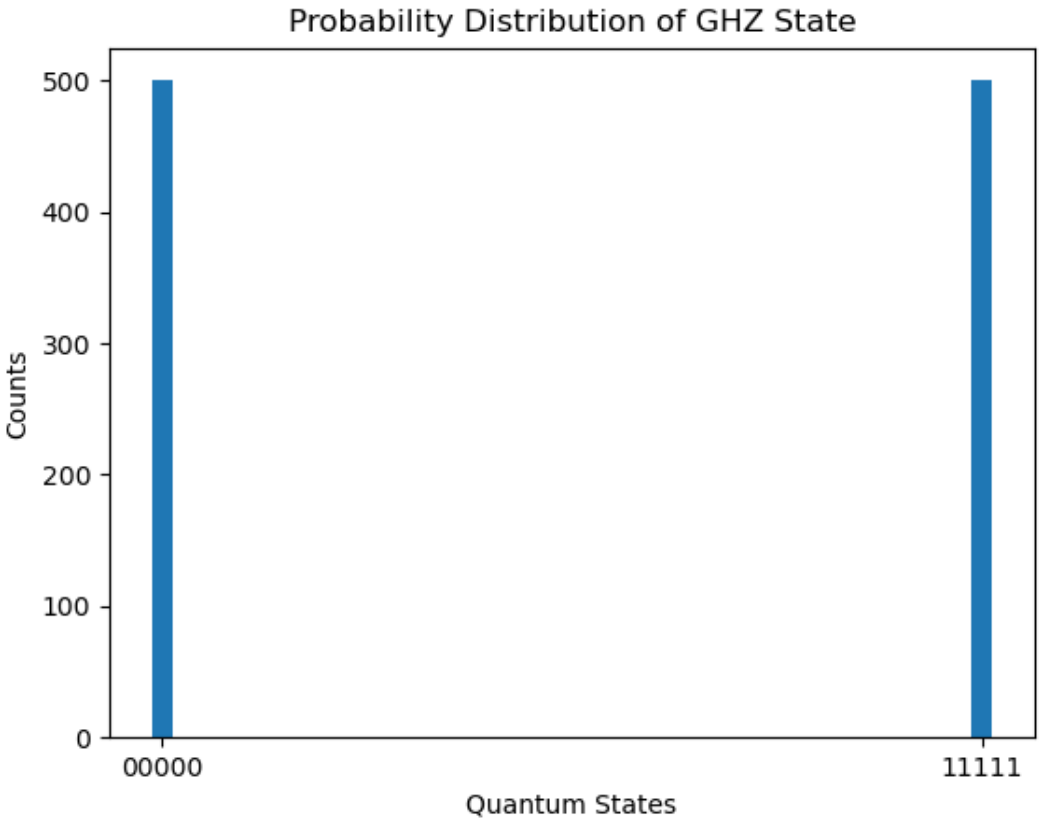
可以使用 `matplotlib` 来绘制出的频率分布直方图，代码如下：

```
results = simulator.run(shots=1000)

states = list(results.keys())
counts = list(results.values())

state_labels = [int(state, 2) for state in states]
plt.bar(state_labels, counts)
plt.xticks(state_labels, states)
plt.xlabel('Quantum States')
plt.ylabel('Counts')
plt.title('Probability Distribution of GHZ State')
plt.show()
```

频率分布直方图如下图所示：



结果符合理论的分析，符合预期。

三、量子电路分析

将实验文档给的代码进行修改，添加一个 `for` 循环来模拟不同 `n_qubits` 的情形，主体代码如下：

```
n_qubits_list = []
run_time_list = []

# Change this value up to 16
for n_qubits in range(2, 16):
    simulator = QubitSimulator(n_qubits)

    start_time = time.time()
    apply_circuit(simulator, n_qubits)
    elapsed_time = time.time() - start_time

    print(f"Number of qubits: {n_qubits}, Time taken: {elapsed_time:.6f}
seconds")

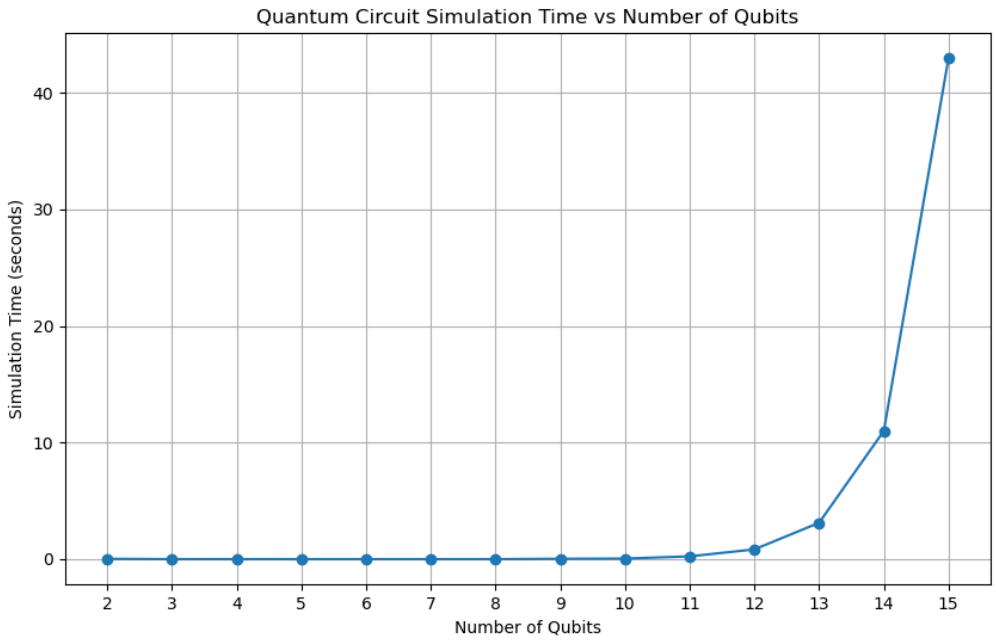
    n_qubits_list.append(n_qubits)
    run_time_list.append(elapsed_time)
```

记录运行时间如下：

```
Number of qubits: 2, Time taken: 0.029480 seconds
Number of qubits: 3, Time taken: 0.000000 seconds
Number of qubits: 4, Time taken: 0.002457 seconds
Number of qubits: 5, Time taken: 0.000000 seconds
```

```
Number of qubits: 6, Time taken: 0.000000 seconds
Number of qubits: 7, Time taken: 0.000000 seconds
Number of qubits: 8, Time taken: 0.000000 seconds
Number of qubits: 9, Time taken: 0.033254 seconds
Number of qubits: 10, Time taken: 0.050832 seconds
Number of qubits: 11, Time taken: 0.242236 seconds
Number of qubits: 12, Time taken: 0.840940 seconds
Number of qubits: 13, Time taken: 3.121972 seconds
Number of qubits: 14, Time taken: 10.997584 seconds
Number of qubits: 15, Time taken: 42.991418 seconds
```

绘制量子电路模拟运行时间与量子电路比特数的关系图：



- 复杂度分析
  - 对于 `n_qubits` 的每个可能取值，状态向量的大小为 2^n。
  - Hadmard 门的应用：我们首先对最后一个量子比特应用 Hadmard 门，复杂度为 $O(2^n)$
  - 控制 U 门的应用：应用 $n-1$ 个，每个门的复杂度 $O(2^n)$
  - 因此量子电路的复杂度为 $O(n \cdot 2^n)$