参考论文

[1]Daniel Gottesman. “The Heisenberg Representation of Quantum Computers.” *ArXiv:Quant-Ph/9807006*, July 1, 1998. http://arxiv.org/abs/quant-ph/9807006.

[2]Aaronson, Scott, and Daniel Gottesman. “Improved Simulation of Stabilizer Circuits.” *Physical Review A* 70, no. 5 (November 30, 2004): 052328. https://doi.org/10.1103/PhysRevA.70.052328.

1 Clifford线路基础

这一部分是pauli矩阵、clifford线路、稳定子的定义和性质。

1.1 Pauli矩阵

四个单比特Pauli矩阵定义：

$$
\begin{aligned}
I &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
X &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\
Y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \\
Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}
\end{aligned}
$$

关系：

$$
\begin{aligned}
X^2 = Y^2 = Z^2 = I \\
XY = iZ, \quad YX = -iZ \\
YZ = iX, \quad ZY = -iX \\
ZX = iY, \quad XZ = -iY
\end{aligned}
$$

Pauli门是量子门中最基础的门。

多比特Pauli矩阵定义：单比特Pauli矩阵的张量积。例如：$X \otimes I \otimes Z$。

注：张量积的乘法：若ABCD分别为4个2*2的矩阵，则

$$
(A \otimes B)(C \otimes D) = (AC) \otimes (BD)
$$

即，两个张量积矩阵的乘积就是对应位置矩阵乘积的张量积，容易计算。

1.2 Clifford矩阵

定义：

对于任意2^n维的Pauli矩阵P，如果2^n维矩阵Q满足：QPQ^{-1}仍然是Pauli矩阵，则Q是2^n维的Clifford矩阵。

定义的意义见1.5节。

性质：

1. 两个Clifford矩阵的乘积是Clifford矩阵。
2. 任意Clifford矩阵可以由Hadamard门、Phase门、CNOT门以及单位门I的张量积和矩阵乘积表示（论文[1]，没找到证明）。因此，后续只要研究这三种Clifford门的张量积和乘积即可。

Hadamard门（简称H门），Phase门（简称S门），CNOT门：

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$
$$S = \begin{bmatrix}1 & 0\\ 0 & i\end{bmatrix}$$
$$CNOT = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$$

Clifford线路定义：初始态$|0\rangle^{\otimes n}$，仅使用CNOT门，H门，S门，测量门构造的线路。

1.3 Clifford矩阵与Pauli矩阵关系

H门：

$$H^{-1} = H$$
$$HXH = Z$$
$$HYH = -Y$$
$$HZH = X$$

S门：

$$SS = Z$$
$$S^{-1} = SSS$$
$$SXS^{-1} = Y$$
$$SYS^{-1} = -X$$
$$SZS^{-1} = Z$$

CNOT门：（这里简写为C）

$$C^{-1} = C$$
$$C(X \otimes I)C = X \otimes X$$
$$C(X \otimes X)C = X \otimes I$$
$$C(X \otimes Y)C = Y \otimes Z$$
$$C(X \otimes Z)C = -Y \otimes Y$$
$$\cdots \text{略}$$

1.4 稳定子stabilizer

1.4.1 定义

对于矩阵U和向量ψ，如果有$U\psi = \psi$，即ψ是U的特征值为1的特征向量，则称U是ψ的稳定子（stabilizer）。

令$stab(\psi)$表示ψ的稳定子集合：

$$stab(\psi) := \{U : U\psi = \psi\}$$

1.4.2 性质

参考论文[2]

1. $stab(\psi) = stab(c\psi), \forall c \in \mathbb{C}, c \neq 0$。
2. 若$U \in stab(\psi)$，则$U^{-1} \in stab(\psi)$。
3. 若$U, V \in stab(\psi)$，则$UV \in stab(\psi)$。
4. 若ψ与φ线性无关，则$stab(\psi) \neq stab(\varphi)$。

由性质4得，如果不考虑向量的整体系数c，即把ψ和$c\psi$视为相同的，那么稳定子集合$stab(\psi)$就唯一确定了向量ψ。

以下所有的向量都再不考虑它的整体系数，原因见1.6节。

1.4.3 Pauli矩阵作为稳定子

单比特Pauli矩阵分别是哪些向量的稳定子：

$$
\begin{array}{ll}
X:|0\rangle+|1\rangle & -X:|0\rangle-|1\rangle \\
Y:|0\rangle+i|1\rangle & -Y:|0\rangle-i|1\rangle \\
Z:|0\rangle & -Z:|1\rangle \\
I:\text{任意量子态} & -I:\text{没有}
\end{array}
$$

1.4.4 定理

参考论文[2]

对于一个n比特量子态$|\psi\rangle$，以下命题等价：

1. $|\psi\rangle$可以由以下线路构造：初始态$|0\rangle^{\otimes n}$，仅使用CNOT门，H门，S门。
2. $|\psi\rangle$可以由以下线路构造：初始态$|0\rangle^{\otimes n}$，仅使用CNOT门，H门，S门，测量门。
3. $|\psi\rangle$恰好有2^n个Pauli矩阵稳定子（即是Pauli矩阵又是$|\psi\rangle$的稳定子）。
4. $|\psi\rangle$可以由它的Pauli矩阵稳定子唯一确定。

无证明。

由定理得，一个Clifford线路最终的量子态（定理第一条）可以由它的Pauli矩阵稳定子唯一确定（定理第四条）。因此，以下我们不再考虑所有稳定子集合，只考虑Pauli矩阵稳定子集合。

1.4.5 Pauli矩阵稳定子集合的一组“基”

由1.4.2节性质3，稳定子UV可以由稳定子U和V通过乘积表示。类似于线性代数中的“线性组合”和“基”的概念，可以找到一组矩阵作为Pauli矩阵稳定子集合的“基”，通过他们的乘积张成稳定子集合。选用乘法而不是线性组合的原因是Pauli矩阵都是张量积形式，乘法容易计算而且能维持张量积形式。

1.5节给出一个例子，初始态的Pauli矩阵稳定子集合的一组基。

1.5 计算Clifford线路的稳定子

类似归纳法，分初始情况和第k步的情况。

初始态：

n比特Clifford线路的初始态为$|0\rangle^{\otimes n}$，可以写出它的Pauli矩阵稳定子集合的一组基：

$$
\{+Z\otimes I\otimes I\otimes\cdots\otimes I,+I\otimes Z\otimes I\otimes\cdots\otimes I,+I\otimes I\otimes I\otimes\cdots\otimes Z\}
$$

加号是为了维持格式统一，因为Pauli矩阵稳定子的符号只可能是$\{+,+i,-,-i\}$这四种。每个矩阵都是n个单比特Pauli矩阵的张量积，共n个矩阵。这n个矩阵可以唯一确定初始态为$|0\rangle^{\otimes n}$。

左乘k次Clifford门后：

假设此时量子态为ψ，它的Pauli矩阵稳定子的一组基是$\{U_1,U_2,\cdots,U_n\}$，即

$$
U_i\psi=\psi
$$

此时对量子态ψ左乘一个Clifford矩阵P，则新量子态$P\psi$的稳定子为PU_iP^{-1}：

$$
(PU_iP^{-1})(P\psi)=P\psi
$$

且由Clifford矩阵定义，PU_iP^{-1}仍是Pauli矩阵，得到新量子态$P\psi$的Pauli矩阵稳定子的一组基$\{PU_1P^{-1}, PU_2P^{-1}, \cdots, PU_nP^{-1}\}$。

通过这个方法，可以计算任意Clifford线路最终的量子态的Pauli矩阵稳定子的一组基，且这组基唯一确定最终的量子态。

1.6 从Pauli矩阵稳定子集合到量子态

猜想：

Clifford线路的量子态可以写成如下形式：

$$|\psi\rangle = c(r_1|\psi_1\rangle + r_2|\psi_2\rangle + \cdots + r_k|\psi_k\rangle), \quad r_i \in \{1, i, -1, -i\}$$

即，所有非零的量子态的系数的模都相同，且相对相位都是i的倍数。

例子1：线路 `H(0),CNOT(0,1),S(1)` 的结果为

$$1/\sqrt{2}(|00\rangle + i|11\rangle)$$

例子2：线路 `H(0),S(1),H(0)` 的结果为

$$\begin{aligned}&\frac{1}{2}(|0\rangle + |1\rangle) + \frac{i}{2}(|0\rangle - |1\rangle)\\ =&\frac{1}{2}(1+i)|0\rangle + \frac{1}{2}(1-i)|1\rangle\\ =&\frac{1+i}{2}(|0\rangle - i|1\rangle)\end{aligned}$$

还没想到证明。

在猜想成立的情况下，只需要弄清所有的$r_i|\psi_k\rangle$即可，模是标准化系数，整体相位对量子态不重要。

而下面的快速算法chp的程序中刚好有这个功能，可以通过量子态的Pauli矩阵稳定子的一组基计算出所有的$r_i|\psi_k\rangle$。具体算法没看懂，但是有代码，`clifford-simulator/tableau.py` 中的 `Tableau.compute_ket()`。

2 快速算法chp

chp算法用于使用经典计算机模拟量子计算中的Clifford线路，在参考论文[2]中提出。

chp算法的原作者的c语言程序：“CHP: CNOT-Hadamard-Phase”，https://www.scottaaronson.com/chp/

我参考c语言程序写出python程序clifford-simulator，核心算法于c语言相同。

2.1 辅助表格

chp算法使用一个表格作为辅助：（类似于单纯形法的单纯形表）

$$\left(\begin{array}{ccc|ccc|c} x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} & r_n \\ \hline x_{(n+1)1} & \cdots & x_{(n+1)n} & z_{(n+1)1} & \cdots & z_{(n+1)n} & r_{n+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{(2n)1} & \cdots & x_{(2n)n} & z_{(2n)1} & \cdots & z_{(2n)n} & r_{2n} \end{array}\right)$$

表格的含义：

引入“destabilizer（去稳定子）”矩阵集合作为辅助变量，即量子态的Pauli矩阵稳定子集合对于n比特Pauli矩阵集合的补集。如上图所示，表格共需要2n行，(2n+1)列。

表格中，第1-n行表示去稳定子矩阵$R_1, \cdots, R_n$。第n+1到2n行表示稳定子矩阵$R_{n+1}, \cdots, R_{2n}$。

每一行，有$R_k = i^{r_k} P_{k,1} \cdots P_{k,n}$，省略了张量积符号。

每一行中的x_{kj}, z_{kj}两个字符表示一个单比特Pauli矩阵$P_{k,j}$：

$$\begin{gathered} 00 : I \\ 10 : X \\ 11 : Y \\ 01 : Z \end{gathered}$$

最后一位r_k表示矩阵前面的相位i^{r_k}，$r_k \in \{0,1,2,3\}$分别表示了$\{+, +i, -, -i\}$四种系数。

如果不需要求出量子态表达式只需要测量值，可以简化为$r_k \in \{0,1\}$，分别代表$\{+,-\}$两种系数。

例如，初始化一个$|00\rangle$，它的stab为$+ZI, +IZ$。它的表格为

$$\left(\begin{array}{cc|cc|c} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

这里的下半张表格对应1.5节提到的Clifford线路初始态的Pauli矩阵稳定子集合的一组基。之后每作用一个Clifford门，就在表格上进行对应的修改，从而得到更新后的Pauli矩阵稳定子集合的基。

对应python代码：`clifford-simulator/tableau.py`中的`Tableau.__init__()`。

2.2 H门、S门、cnot门

三种基础Clifford门对应的表格运算规则：

CNOT from control a to target b. For all $i \in \{1,\ldots,2n\}$, set $r_i := r_i \oplus x_{ia} z_{ib} (x_{ib} \oplus z_{ia} \oplus 1)$, $x_{ib} := x_{ib} \oplus x_{ia}$, and $z_{ia} := z_{ia} \oplus z_{ib}$.

Hadamard on qubit a. For all $i \in \{1,\ldots,2n\}$, set $r_i := r_i \oplus x_{ia} z_{ia}$ and swap x_{ia} with z_{ia}.

Phase on qubit a. For all $i \in \{1,\ldots,2n\}$, set $r_i := r_i \oplus x_{ia} z_{ia}$ and then set $z_{ia} := z_{ia} \oplus x_{ia}$.

原理就是按照1.4.6节修改了Pauli矩阵稳定子的基，并保存在表格下半部分。表格的上半部分同步修改，我没仔细看上半部分在做什么，但是上半部分在后续的功能中也很重要，具体见参考论文[2]。

对应python代码：`clifford-simulator/tableau.py`中的`Tableau.h(a),Tableau.s(a),Tableau.cx(a,b)`。

2.3 表格基本行变换

表格与矩阵类似，有基本的行变换：行交换和行乘积。

行交换：由于表格每行表示一个矩阵，前n行表示destabilizer门，后n行表示stabilizer门，这些门没有顺序要求，因此前n行可以两两交换，后n行可以两两交换，表格等价。

对应python代码：`clifford-simulator/tableau.py`中的`Tableau._row_swap(a,b)`。

行乘积：由1.4.2节stabilizer性质的第三条，两个stabilizer矩阵乘积还是stabilizer，而且每个stabilizer矩阵是用张量积形式保存的，两个张量积形式矩阵的乘积就是对应位置矩阵乘积的张量积，方便计算和保存。

对应python代码：`clifford-simulator/tableau.py`中的`Tableau._row_mul(a,b)`。

这两个基本行变换会多次用于后续的测量和计算量子态。

2.4 测量

measure伪代码：a为测量的qubit位置。分成随机和确定两种情况。

```
for row in [n:2n]:
    if tableau[row, a] == 1:
        # case 1: 0或1都有可能，随机决定并改变tableau
        return measure_random(a, row)
    # case 2: tableau[:,a]全都是0，测量结果是唯一确定的
    return measure_determinate(a)
```

解释：比如$\psi = |00\rangle + |11\rangle$，先测量0号qubit，则需要随机指定。由于0号和1号qubit有纠缠关系，因此对0号的测量会改变表格，使得之后对1号测量时测量值是确定的，一定和0号相同。

判断的条件`tableau[p,a]==1`，说明第p个stab矩阵的a位置为X或Y，由1.4.3节看出，X或Y矩阵对应的稳定子一定是同时有0和1的。

两种情况的测量算法具体见python代码：`clifford-simulator/tableau.py`中的`Tableau._measure_random(a,p)`和`Tableau._measure_determinate(a)`。

2.5 求量子态的向量表示

在1.6节中已经提到过。只是有代码，但是算法没看懂。

程序可以拆分成如下几个函数：

1. 对表格作高斯消去标准化，python代码：`Tableau.gaussian()`
2. 从标准化后的表格依次计算$r_i|\psi_i\rangle$。
    1. 计算$r_i|\psi_i\rangle$并保存在表格的第2n+1行（专门临时保存数据的位置），python代码：`Tableau.compute_ket()`
    2. 读取表格的第2n+1行并转化成字符串格式输出，python代码：`Tableau._get_basis_state()`

关于表格方法的更多分析，参考论文[2]。

3 python程序clifford-simulator使用方法

chp算法的原作者的c语言程序：“CHP: CNOT-Hadamard-Phase”，https://www.scottaaronson.com/chp/

我参考c语言程序写出python程序clifford-simulator，核心算法于c语言相同。

3.1 功能简介

clifford-simulator的功能如下：

1. 第2节中提到的功能
    1. 表格
    2. h门、s门、cnot门在表格中对应的运算
    3. measure
    4. gaussian表格标准化
    5. 计算最终量子态的ket向量形式
2. 其他功能
    1. 打印表格
    2. 从表格打印稳定子矩阵
    3. 重复测量并统计结果
    4. 从量子态向量形式计算密度矩阵

3.2 输入输出介绍

下面展示如何在main.py文件中编辑clifford线路，如何查看输出结果。

3.2.1 import部分

```
import util
import numpy as np
from tableau import Tableau
```

1. tableau为算法部分。
2. util为一些辅助函数，例如一些print功能

3.2.2 circuit函数：编辑Clifford线路

```
def circuit():
    n_bit = 2
    t = Tableau(n_bit)
    t.h(0)
    t.cx(0, 1)
    t.s(1)
    return t
```

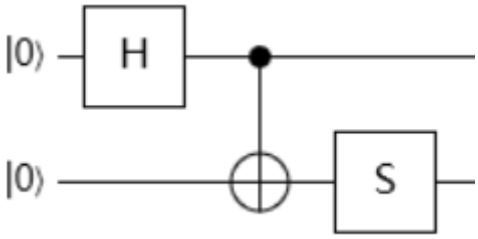
在这个函数中设计需要模拟的clifford线路。

`n_bit = 2` 声明要用到的qubit个数。

`t = Tableau(n_bit)` 声明一个空线路对象并初始化，所有qubit初始为0态。

后面的部分是在空线路中依次添加量子门。可用的基础clifford门为h,cx,s三种，分别是Hadamard门，CNOT门，Phase门。所有clifford门可表示成他们的组合。括号中的为作用的qubit序号。

例如，这个circuit实现的电路图如下：



三行从上到下分别是0号、1号qubit，初始态都是0。这个线路的理论结果是$1/\sqrt{2}*(|00\rangle+i|11\rangle)$。

3.2.3 主函数部分：计算和输出

第一部分：构造线路t

```
t = circuit()
```

第二部分：打印表格和稳定子

```
t.gaussian()
t.show_tableau()
t.show_stab()
```

先将表格标准化，再打印表格，根据表格下半部分打印稳定子集合的一组基。

输出如下：

```
tableau:
0 0 | 1 0 | 0
0 1 | 0 1 | 0
----|-----|--
1 1 | 0 1 | 0
0 0 | 1 1 | 0

stab set:
 +XY
 +ZZ
```

XYZ表示三种Pauli矩阵，I为单位矩阵，省略了张量积符号。例如，`+XY`表示$+1 * X \otimes Y$，是一个4*4的矩阵。

第三部分：计算量子态

```
states = t.compute_ket()
util.print_state_string(states)
```

`states = t.compute_ket()`得到量子态，states是一个list，每一个list是用字符串储存混合态中的一个态。

`util.print_state_string(states)`打印这些states的字符串形式。

输出如下：

```
2 nonzero basis states:
 +|00>
+i|11>
```

输出表明，最终的量子态由这2个非零系数的基态组成。前面的符号可能是$+, +i, -, -i$。

从字符串转化成numpy向量形式

```
vector = util.get_state_vector(states)
print('state vector:')
print(vector)
```

输出如下：

```
state vector:
[[0.70710678+0.j        ]
 [0.        +0.j        ]
 [0.        +0.j        ]
 [0.        +0.70710678j]]
```

将向量形式转化为密度矩阵：

```
density_matrix = np.matmul(vector, vector.T)
print('density matrix:')
print(density_matrix)
```

输出如下：

```
density matrix:
[[ 0.5+0.j   0. +0.j   0. +0.j   0. +0.5j]
 [ 0. +0.j   0. +0.j   0. +0.j   0. +0.j ]
 [ 0. +0.j   0. +0.j   0. +0.j   0. +0.j ]
 [ 0. +0.5j  0. +0.j   0. +0.j  -0.5+0.j ]]
```

第四部分：模拟测量

```
measure_times = 10000
measure_results = util.measure_tableau(t, measure_times)
util.print_measure_results(measure_results, measure_times)
```

测量所有比特，重复测量10000次，输出所有测量结果的比例。输出如下：

```
measure results:
|11>: 0.4948
|00>: 0.5052
```