Efficient generation of Clifford circuits

周书予

2000013060@stu.pku.edu.cn

2021 年 12 月 11 日

1 引言

Clifford 线路是由 Hadamard 门 H、相位门 S 以及 controlled-NOT 门 CNOT 组成的量子线路，这三个门被称为 Clifford 门，而由 Clifford 线路实现的算符被称作 Clifford 算符[1]。可以验证所有 Clifford 算符构成一个乘法群，我们把这个群称作 Clifford group，记作 $\mathcal{C}_n$，下标 n 表示作用在 n 个量子比特上。

这次作业的主要工作是研究 Clifford group 的内部结构。section 2 将会证明任意作用在 n 个量子比特上的 Clifford 算符都可以被 $O(n^2)$ 个 Clifford 门组合实现，这意味着任意 Clifford 算符都具有实现高效性；在 section 3 中我们引入了 Canonical form 这一记号，并证明它与 Clifford 算符之间存在一一对应；这是卓有意义的，因为双射的存在将允许我们对 Clifford 线路进行随机采样——我们只需要随机采样 Canonical form 就可以了！这部分工作会在 section 4 中被介绍。

2 第一部分

在展开证明前我们先引入一些记号。记 $\mathcal{P}_n$ 表示 n 个量子比特上的 Pauli group，也就是 $\{I, X, Y, Z\}^{\otimes n}$，乘上 $\pm 1, \pm i$ 作为 global phase。乘上 global phase 的目的是让群对乘法封闭。可以验证 Pauli group $\mathcal{P}_n$ 是 $\mathcal{C}_n$ 的一个子群。

定义 $\mathcal{P}_n$ 的正规化子 (normalizer) $\mathcal{N}(\mathcal{P}_n)$ 为所有满足 $\forall g \in \mathcal{P}, UgU^\dagger \in \mathcal{P}_n$ 的幺正算符 U 构成的集合。通过验证 $\mathcal{N}(\mathcal{P}_n)$ 对乘法以及求逆封闭，我们可以知道 $\mathcal{N}(\mathcal{P}_n)$ 构成 n 阶幺正群的一个子群。

为了完成证明，我们需要证明如下两个引理：

引理 1 *任意 Clifford 算符都是 $\mathcal{N}(\mathcal{P}_n)$ 中的元素。*

引理 2 *任意 $\mathcal{N}(\mathcal{P}_n)$ 中的算符 U 都可以被 $O(n^2)$ 个 Clifford 门实现，至多差一个 global phase。*

根据以上两个引理可以立即得到，任意 Clifford 算符都可以被 $O(n^2)$ 个 Clifford 门实现，至多差一个 global phase。实际上"至多差一个 global phase"的修辞是可以去掉的，因为可以验证 Clifford group 里中包含的 global phase 有且仅有 $\{e^{\pi i k/4} : k \in [0, 7]\}$ 这 8 种（其中 $(SH)^3 = e^{\pi i/4} I$ 可以作为生成元），因此 Clifford group 中任意 global phase 均可以通过 $O(1)$ 个 Clifford 门实现。

值得一提的是，部分文章 [1] 对 Clifford 算符的定义方式就是 $\mathcal{N}(\mathcal{P}_n)/U(1)$，即 $\mathcal{P}_n$ 的正规化子忽略掉 global phase。两种定义在 global phase 的处理上是有所不同的，这会引起一些描述上的偏差，比如说下文中将在忽略 global phase 的基础上叙述 $|\mathcal{C}_n| = 2^{n^2+2n} \prod_{i=1}^{n} (4^i - 1)$，而如果考虑 global phase 的话，其结果还需要乘 8。

2.1 引理 1 的证明

由于任何 Clifford 算符都可以写成 Hadamard 门、相位门以及 CNOT 门乘积的形式，我们只需要证明这三个 Clifford 门属于 $\mathcal{N}(\mathcal{P}_n)$ 就行了，具体地我们要对于 $U \in \{H, S, \mathrm{CNOT}\}$，证明 $\forall g \in \mathcal{P}_n, UgU^\dagger \in \mathcal{P}_n$。

这项工作其实非常简单，只需要列举所有情况（见表 1）验证就可以了。表格里只展示了 $UXU^\dagger$ 和 $UZU^\dagger$ 的结果，因为 $UYU^\dagger = i(UXU^\dagger)(UZU^\dagger)$ 可以被前两者唯一确定。

2.2 引理 2 的证明

这部分的证明就比较麻烦了。总体的思路是对量子比特数 n 作归纳（这是一种极其不直观的方法），分为三步：

- 第一步：证明任意 $U \in \mathcal{N}(\mathcal{P}_1)$ 都可以由 $O(1)$ 个 Hadamard 门与相位门实现；
- 第二步：假设任意 $U' \in \mathcal{N}(\mathcal{P}_n)$ 都可以由 $O(n^2)$ 个 Clifford 门实现，对于任何满足 $UZ_1U^\dagger = X_1 \otimes g$ 以及 $UX_1U^\dagger = Z_1 \otimes g'$ 的 $U \in \mathcal{N}(\mathcal{P}_{n+1})$，证明它都可以由 $O(n^2)$ 个 Clifford 门实现；

[1]还存在一种别的对于 Clifford 算符的定义方式，将在稍后提及。

- 第三步：把第二步的结果推广，证明不做任何限制的 $U \in \mathcal{N}(\mathcal{P}_{n+1})$ 都可以由 $O(n^2)$ 个 Clifford 门实现。

上面的三条叙述都还需要加上“至多差一个 global phase”。

由于这部分的证明有点冗长，并且与接下来将要讨论的内容关系不大，因此就全部搬到附录里了，详见 section B。

3 第二部分

以下的内容都是这篇文章 [2] 中介绍的。在引入所谓 Canonical form 之前，我们先研究一下 Clifford group $\mathcal{C}_n$ 的子群结构。为了行文方便，在接下来的所有叙述中，我们均忽略 global phase[2]。下表中列出了 $\mathcal{C}_n$ 的一些子群以及对应的生成元集合：

记号	名称	生成元集合
$\mathcal{C}_n$	Clifford group	H, S, CNOT
$\mathcal{F}_n$	Hadamard-free group	$X, S, \mathrm{CNOT}, \mathrm{CZ}$
$\mathcal{B}_n$	Borel group	$X, S, \mathrm{CNOT}^{\downarrow}, \mathrm{CZ}$
$\mathcal{S}_n$	Symmetric group	SWAP
$\mathcal{P}_n$	Pauli group	X, Z

其中 CNOT 和 CZ 分别表示 Controlled-X 门与 Controlled-Z 门。

注意到 $\mathcal{C}_n$ 的生成元集合其实也可以写成 $\{X, Z, H, S, \mathrm{CNOT}, \mathrm{CZ}\}$，这是因为 $Z = S^2, X = HZH, \mathrm{CZ} = H_2 \cdot \mathrm{CNOT} \cdot H_2$。这样写就能明显地看出表中的所有群都是 $\mathcal{C}_n$ 的子群。

考虑这样一件事情：Hadamard 门是（在 computational basis 下）唯一会产生叠加态的 Clifford 门，这说明 Hadamard-free group $\mathcal{F}_n$ 中的任何算符都不能产生任何叠加，即只能把基矢映成基矢，因而可以被表示成如下的形式：

$$F|x\rangle = i^{x^{\mathbf{T}}\Gamma x} O|\Delta x\rangle \tag{1}$$

其中 $x \in \mathbb{F}_2^n, \Gamma, \Delta \in \mathbb{F}_2^{n\times n}, O \in \mathcal{P}_n$，所有矩阵乘法均在 $\mathbb{F}_2$ 下进行。Δ 必须是可逆的，它表示了 n 个量子比特之间可能存在的纠缠，而 Γ 的唯一作用则是确定相位[3]，它必须是对称的，这是因为 $i^{x^{\mathbf{T}}\Gamma x} = \prod_{i=1}^{n} i^{x_i\Gamma_{ii}} \prod_{1\le i<j\le n} i^{x_i x_j(\Gamma_{ij}+\Gamma_{ji})}$，一旦 $\Gamma_{ij} \neq \Gamma_{ji}$ 就将导致 controlled-S 门在 $\mathcal{F}_n$ 中的出现，产生矛盾。

$\mathcal{B}_n$ 生成元集合中的 $\mathrm{CNOT}^{\downarrow}$ 记号表示控制比特比目标比特标号小的 CNOT 门，这使得 eq. (1) 式中的 Δ 变成了下三角，且对角元全是 1。这样的简化也使得我们可以直接地写出算符 F 的量子线路表示[4]：

$$F = O\prod_{i=1}^{n} S_i^{\Gamma_{i,i}} \prod_{1\le i<j\le n} \mathrm{CZ}_{i,j}^{\Gamma_{i,j}} \prod_{1\le i<j\le n} \mathrm{CNOT}_{i,j}^{\Delta_{j,i}} \tag{2}$$

接下来，我们会用 $F(O, \Gamma, \Delta)$ 来表示 eq. (2) 中的 F。

定义 1 *我们称算符 $F(O, \Gamma, \Delta)$ 的 Pauli 部分是平凡的，如果 $O = I$。*

顺带一提的是 $|\mathcal{B}_n| = 2^{n^2+2n}$，这是因为 O, Γ, Δ 分别有 $4^n, 2^{\frac{n^2+n}{2}}, 2^{\frac{n^2-n}{2}}$ 种取法，注意忽略了 global phase。

我们引入本文最为核心的一个定理：

定理 1 (Canonical Form) *任意 $U \in \mathcal{C}_n$ 都可以被**唯一地**写成*

$$U = F(I, \Gamma, \Delta) \cdot \left(\prod_{i=1}^{n} H_i^{h_i}\right) \sigma \cdot F(O', \Gamma', \Delta') \tag{3}$$

其中 $h \in \{0,1\}^n, \sigma \in \mathcal{S}_n$ 是 n 阶排列，$F(I, \Gamma, \Delta), F(O', \Gamma', \Delta') \in \mathcal{B}_n$，其中 Γ, Δ 需要满足条件：对于 $\forall 1 \le i, j \le n$:

C1 *若 $h_i = 0, h_j = 0$，则 $\Gamma_{i,j} = 0$;*

C2 *若 $h_i = 1, h_j = 0, \sigma(i) > \sigma(j)$，则 $\Gamma_{i,j} = 0$;*

C3 *若 $h_i = 0, h_j = 0, \sigma(i) > \sigma(j)$，则 $\Delta_{i,j} = 0$;*

C4 *若 $h_i = 1, h_j = 1, \sigma(i) < \sigma(j)$，则 $\Delta_{i,j} = 0$;*

C5 *若 $h_i = 1, h_j = 0$，则 $\Delta_{i,j} = 0$;*

[2] 严谨地讲，是始终只考虑 $\mathcal{C}_n$ 及其子群在商掉 $U(1)$ 后得到的商群。

[3] 这个相位是关于 x 的函数，因此并不是 global phase。

[4] 需要注意的是这里的 Γ 并不严格对应 eq. (1) 式中的 Γ，因为 eq. (1) 中相位被表示成了关于 x 的双线性函数，而在这里，对应给 Γ 的输入等效地变成了 Δx。当然我们可以换一个记号（令 $\Gamma' = \Delta\Gamma\Delta^{\mathrm{T}}$ 仍是对称的）来实现这样的转化，原论文并没有显式地指出这一点。

这个定理初看是十分迷惑的，因为这些不知所云的古怪限制通过我们难以理解的方式，确立了 Canonical form 的表示唯一性。通过 Bruhat decomposition theorem [3] 以及下面两条引理，我们将得到对于 Canonical form 更深入的理解。

定理 2 (Bruhat decomposition) *Clifford group* $\mathcal{C}_n$ 可以写成如下**不交**并的形式

$$\mathcal{C}_n = \bigsqcup_{h\in\{0,1\}^n} \bigsqcup_{\sigma\in\mathcal{S}_n} \mathcal{B}_n \left(\prod_{i=1}^n H_i^{h_i}\right) \sigma \mathcal{B}_n \tag{4}$$

定义 2 对于 $h \in \{0,1\}^n, \sigma \in \mathcal{S}_n$，定义 $\mathcal{B}_n$ 的子群

$$\mathcal{B}_n(h,\sigma) = \{F \in \mathcal{B}_n : W^{-1}FW \in \mathcal{B}_n\} \tag{5}$$

（可以很容易地验证这是一个子群）其中 $W = \left(\prod_{i=1}^n H_i^{h_i}\right)\sigma$，以下会始终沿用这个记号。

引理 3 记 $\overline{h} = h \oplus 1^n$ 表示 h 的按位取反，那么任意算符 $F(O,\Gamma,\Delta) \in \mathcal{B}_n$ 是 $\mathcal{B}_n(\overline{h},\sigma)$ 中的元素，当且仅当 Γ,Δ 对于 h,σ 满足 ***C1-C5***，此外还存在关系

$$\mathcal{B}_n(\overline{h},\sigma) \cap \mathcal{B}_n(h,\sigma) = \mathcal{P}_n \tag{6}$$

引理 4 任意算符 $F \in \mathcal{B}_n$ 都可以被唯一写成 $F = F_L F_R$，其中 $F_R \in \mathcal{B}_n(h,\sigma)$，$F_L \in \mathcal{B}_n(\overline{h},\sigma)$ 且 *Pauli* 部分是平凡的。

可以意识流地构想一下上述结论是如何导出 Canonical form 并证明唯一性的：首先，定理 2 指出可以对 $\mathcal{C}_n$ 中任意元素作形如 $\mathcal{B}_n W \mathcal{B}_n$ 的分解，且 W 是唯一确定的；引理 3 指出了一项重要结论——***C1-C5*** 把 Canonical form 中的前一个 F 限制在了 $\mathcal{B}_n$ 的一个子群 $\mathcal{B}_n(\overline{h},\sigma)$ 中，引理 4 则说明了 Canonical form 的存在性，而唯一性证明则是由 F_L 的平凡 Pauli 部分以及 eq. (6) 引出的。

接下来我们通过上述结论形式化地证明一下定理 1。由定理 2 知任意 $U \in \mathcal{C}_n$ 可以被写成 $U = LWR$，其中 $L,R \in \mathcal{B}_n$，且 W 是唯一确定的。根据引理 4，L 可以被进一步分解为 $L = BC$，其中 $C \in \mathcal{B}_n(h,\sigma), B \in \mathcal{B}_n(\overline{h},\sigma)$ 且 Pauli 部分平凡，因此有

$$U = LWR = BCWR = BWW^{-1}CWR = BWC'R \tag{7}$$

根据 $\mathcal{B}_n(h,\sigma)$ 的定义，有 $C' \triangleq W^{-1}CW \in \mathcal{B}_n$，因而 $C'R \in \mathcal{B}_n$，这证明了 Canonical form 的存在性。至于唯一性，设

$$F_1 W F_1' = F_2 W F_2' \tag{8}$$

满足 $F_1', F_2' \in \mathcal{B}_n, F_1, F_2 \in \mathcal{B}_n(\overline{h},\sigma)$ 且 Pauli 部分平凡，通过对 eq. (8) 的一些变形

$$W^{-1}(F_2^{-1}F_1)W = F_2'(F_1')^{-1} \in \mathcal{B}_n \tag{9}$$

根据 $\mathcal{B}_n(h,\sigma)$ 的定义我们可以得到 $F_2^{-1}F_1 \in B_n(h,\sigma)$，故 $F_2^{-1}F_1 \in \mathcal{B}_n(\overline{h},\sigma) \cap \mathcal{B}_n(h,\sigma) = \mathcal{P}_n$，但两者均只有平凡的 Pauli 部分，这说明了 $F_2^{-1}F_1 = I$，从而 $F_1 = F_2, F_1' = F_2'$，唯一性得证。 ■

接下来我们将重点讨论两个重要引理的证明。当然和之前一样，只展示核心思路，具体细节会被放在附录中。

3.1 引理 3 的证明

证明“当且仅当”的工作就是证明两个方向的蕴含关系。先考虑“⇐”即充分性，也即证明只要 Γ,Δ 对于 h,σ 满足 ***C1-C5***，就可以使 $F(O,\Gamma,\Delta) \in \mathcal{B}(\overline{h},\sigma)$。由于 F 总可以被拆解成若干单 Clifford 门的乘积的形式，因此我们只需要验证 $\mathcal{B}_n$ 生成元集合中的每一个元素——$X, S, \mathrm{CZ}, \mathrm{CNOT}^{\downarrow}$——满足条件即可。列举出所有情况即可。

然后考虑“⇒”即必要性。记 $\mathcal{B}_n'(h,\sigma)$ 表示对于 $\overline{h},\sigma$ 满足 ***C1-C5*** 的算符集合[5]，充分性的证明工作说明了 $\mathcal{B}_n'(h,\sigma) \subseteq \mathcal{B}_n(h,\sigma)$，我们现在希望能得到 $\mathcal{B}_n'(h,\sigma) = \mathcal{B}_n(h,\sigma)$。采取的方法是对集合元素计数，我们有如下的集合大小关系

$$|\mathcal{C}_n| \le \sum_{h\in\{0,1\}^n} \sum_{\sigma\in\mathcal{S}_n} \frac{|\mathcal{B}_n|^2}{|\mathcal{B}_n(h,\sigma)|} \le \sum_{h\in\{0,1\}^n} \sum_{\sigma\in\mathcal{S}_n} \frac{|\mathcal{B}_n|^2}{|\mathcal{B}_n'(h,\sigma)|} \tag{10}$$

想要证明 $|\mathcal{B}_n'(h,\sigma)| = |\mathcal{B}_n(h,\sigma)|$，可以考虑直接证明右侧和式的值就等于 $|\mathcal{C}_n|$。根据 $\mathcal{B}_n'(h,\sigma)$ 的定义，我们知道

$$|\mathcal{B}_n'(h,\sigma)| = |\mathcal{B}_n| 2^{-I_n(h,\sigma)} \tag{11}$$

[5]这里还没有指出是 $\mathcal{B}_n'(h,\sigma)$ 一个群。

其中 $I_n(h,\sigma)$ 表示 $\overline{h},\sigma$ 通过 ***C1-C5*** 给到 Γ,Δ 的限制数目。可以验证

$$I_n(h,\sigma) = \frac{n(n-1)}{2} + |h| + \sum_{1\le i<j\le n:\sigma(i)<\sigma(j)} (-1)^{h_i+1} \tag{12}$$

通过 eq. (12) 我们将会证明 $|\mathcal{C}_n| = \sum_{h\in\{0,1\}^n}\sum_{\sigma\in\mathcal{S}_n}\frac{|\mathcal{B}_n|^2}{|\mathcal{B}'_n(h,\sigma)|}$，从而得到引理 3 的必要性作为结论。此外还需要证明 eq. (6)：可以验证 $\mathcal{B}'_n(h,\sigma)\cap\mathcal{B}'_n(\overline{h},\sigma)=\mathcal{P}_n$——因为同时满足对 (h,σ) 与 $(\overline{h},\sigma)$ 满足 ***C1-C5*** 会直接导致 $\Gamma=\Delta=0^{n\times n}$——那么结合前面的论述就可以得到 eq. (6) 了。本部分的具体论证细节将在 section C 中给出。

3.2 引理 4 的证明

记 $F_1, F_2, \cdots, F_m$ 为 $\mathcal{B}_n(\overline{h},\sigma)$ 中所有 Pauli 部分平凡的元素排成一列，显然有 $m = |\mathcal{B}_n(\overline{h},\sigma)/\mathcal{P}_n| = 4^{-n}|\mathcal{B}_n(\overline{h},\sigma)|$。我们声称所有左陪集 $F_j\mathcal{B}_n(h,\sigma)$ 是两两不交的，这指出了分解的唯一性。而要证明存在性，可以考虑对集合元素计数，具体地，需要验证

$$|\mathcal{B}_n(h,\sigma)|\cdot|\mathcal{B}_n(\overline{h},\sigma)| = |\mathcal{P}_n|\cdot|\mathcal{B}_n| = 4^n|\mathcal{B}_n| \tag{13}$$

注意到 $|\mathcal{B}_n| = 2^{n^2+2n}$ 以及 $|\mathcal{B}_n(h,\sigma)| = |\mathcal{B}_n|2^{-I_n(h,\sigma)}$，而又可以验证 $I_n(h,\sigma)+I_n(\overline{h},\sigma)=n^2$，代入后能直接发现 eq. (13) 成立，从而完成证明。细节可见 section D。

4 第三部分

现在我们可以通过对 Canonical form 作随机采样来实现对 Clifford group 的随机采样了。采样的过程分为两部分，注意到 $\mathcal{C}_n$ 是 $\mathcal{B}_nW\mathcal{B}_n$ 的不交并，我们可以先按照一定的概率分布采样 W，也即采样 $h\in\{0,1\}^n,\sigma\in\mathcal{S}_n$。具体地，这个概率分布为

$$P_n(h,\sigma) = \frac{|\mathcal{B}_nW\mathcal{B}_n|}{|\mathcal{C}_n|} = \frac{|\mathcal{B}_n|^2}{|\mathcal{B}_n(h,\sigma)|\cdot|\mathcal{C}_n|} = \frac{2^{I_n(h,\sigma)}}{\sum_{h,\sigma}2^{I_n(h,\sigma)}} \tag{14}$$

eq. (14) 的形式与一种被称为 Mallows distribution [4] 的概率分布类似，这里 [2] 的作者将其称为 quantum Mallows distribution。可以证明这个采样过程可以被如下的算法实现

Algorithm 1 根据 quantum Mallows distribution $P_n(h,\sigma)$ 来生成 h,σ

$A \leftarrow [1...n]$
for $i=1$ to n **do**
  $m \leftarrow |A|$
  Sample $h_i\in\{0,1\}$ and $k\in[1...m]$ from the probability distribution

$$p(h_i,k) = \frac{2^{m-1+h_i+(m-k)(-1)^{1+h_i}}}{4^m-1}$$

  Let j be the k-th largest element of A
  $\sigma(i)\leftarrow j$
  $A \leftarrow A\setminus\{j\}$
end for
return (h,σ)

引理 5 上述算法可以正确地按照 $P_n(h,\sigma)$ 的分布来采样 h,σ，同时运行时间为 $\tilde{O}(n)$[6]。

证明用的是归纳，与引理 3 中证明 $|\mathcal{C}_n| = \sum_{h,\sigma}\frac{|\mathcal{B}_n|^2}{|\mathcal{B}'_n(h,\sigma)|}$ 的方法类似，详细可参考 section E。

在得到 W 之后，下一步是对左右两侧的 $\mathcal{B}_n$ 进行采样。这个过程只需要按照 Canonical form eq. (3) 对 $\Gamma,\Delta,O',\Gamma',\Delta'$ 进行采样即可，因此是简单的。代码展示在 section F 中。

值得注意的是，即使去掉 Canonical form eq. (3) 中对 Γ,Δ 的限制（***C1-C5***），即把算符 F 的限制从 $\mathcal{B}_n/\mathcal{B}_n(h,\sigma)\cong\mathcal{B}_n(\overline{h},\sigma)/\mathcal{P}_n$ 扩展到 $\mathcal{B}_n/\mathcal{P}_n$，得到的采样结果也仍是均匀分布的。这有赖于群论中的 Lagrange 定理 [5]，它指出了子群的每一个陪集的大小都是相同的。Qiskit 中所实现的 random_clifford 方法就采用了这种策略，这使得程序变得更加简短，实现效率上也有所提升，代价则是消耗了更多的随机比特。

[6]渐进复杂度记号 $\tilde{O}$(tilde big-O) 表示忽略对数因子，即 $\tilde{O}(n)=O(n\log^k n)$。

参考文献

[1] Robert Koenig and John A. Smolin. How to efficiently select an arbitrary clifford group element. *Journal of Mathematical Physics*, 55(12):122202, 2014.

[2] Sergey Bravyi and Dmitri Maslov. Hadamard-free circuits expose the structure of the clifford group. *IEEE Transactions on Information Theory*, 67(7):4546–4563, Jul 2021.

[3] Dmitri Maslov and Martin Roetteler. Shorter stabilizer circuits via bruhat decomposition and quantum circuit transformations. *IEEE Transactions on Information Theory*, 64(7):4729–4738, Jul 2018.

[4] Tyler Lu and Craig Boutilier. Learning mallows models with pairwise preferences. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 145–152, Madison, WI, USA, 2011. Omnipress.

[5] Wikipedia contributors. Lagrange's theorem (group theory) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Lagrange%27s_theorem_(group_theory)&oldid=1039199376, 2021. [Online; accessed 11-December-2021].

附录

A 引理 1 的详细证明

U	g	$UgU^\dagger$
controlled-NOT	X_1	X_1X_2
	X_2	X_2
	Z_1	Z_1
	Z_2	Z_1Z_2
Hadamard H	X	Z
	Z	X
phase S	X	Y
	Z	Z

表 1: 验证 *Clifford* 门都是 $\mathcal{P}_n$ 的正规化子

B 引理 2 的详细证明

B.1 第一步

先假设正规化子 U 满足 $UZU^\dagger = Z$，那么 $UZ = ZU$ 由此说明 U, Z 对易，从而 U 只能有非零对角元

$$U = e^{i\theta}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

因为 U 是正规化子，我们也有

$$UXU^\dagger = \begin{pmatrix} 0 & e^{i\phi} \\ e^{-i\phi} & 0 \end{pmatrix} \in \{\pm X, \pm iX, \pm Y, \pm iY\}$$

那么 $e^{i\phi} \in \{\pm 1, \pm i\}$，这说明 U 可以由相位门 S 实现，至多差一个 global phase。

现在考虑更一般的情况。对于任意的 $U \in \mathcal{N}(\mathcal{P}_1)$，必然存在一个 $g \in \mathcal{P}_1 \setminus \{\pm I, \pm iI\}$ 满足 $UgU^\dagger = Z$。注意到 $HXH^\dagger = Z, (HS)Y(HS)^\dagger = -Z$，因此总可以找到一个由 Hadamard 门与相位门组成的 V，满足 $VgV^\dagger \in \{\pm Z, \pm iZ\}$。把 U 写为 $U = U_1V$，那么 $Z = UgU^\dagger = U_1VgV^\dagger U_1^\dagger \in \{\pm U_1ZU_1^\dagger, \pm iU_1ZU_1^\dagger\}$，根据之前的论述，$U_1$ 可以由相位门实现，至多差一个 global phase，因此 $U = U_1V$ 可以由 $O(1)$ 个 Hadamard 门和相位门实现，至多差一个 global phase。

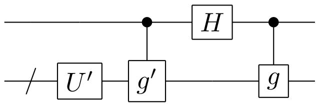

图 1: 证明中构造的量子线路

B.2 第二步

考虑图 1 中的量子线路，其中 U' 定义为满足 $U'|\psi\rangle = \sqrt{2}\langle 0|U(|0\rangle \otimes |\psi\rangle)$ 的算符。这里假设对于某个 $g, g' \in \mathcal{P}_n$，有

$$\begin{aligned} UZ_1U^\dagger &= X_1 \otimes g \\ UX_1U^\dagger &= Z_1 \otimes g' \end{aligned} \tag{15}$$

注意到两个等式的左侧都是 Hermitian 的，我们有 $g = g^\dagger, g' = g'^\dagger$。

我们证明这个量子线路实现的算符——姑且称之为 $\tilde{U}$——与 U 是等价的，且可以由 $O(n^2)$ 个 Clifford 门组合实现。首先，我们可以写出 eq. (15) 的一些变式

$$U = (X_1 \otimes g)UZ_1 \tag{16}$$

$$= (Z_1 \otimes g')UX_1 \tag{17}$$

$$= (Z_1X_1 \otimes g'g)U(Z_1X_1) \tag{18}$$

将其代入 $U'|\psi\rangle = \sqrt{2}\langle 0|U(|0\rangle \otimes |\psi\rangle)$ 后可以得到

$$U'|\psi\rangle = \sqrt{2}\langle 0|U(|0\rangle \otimes |\psi\rangle) \tag{19}$$

$$= \sqrt{2}\langle 1|gU(|0\rangle \otimes |\psi\rangle) \tag{20}$$

$$= \sqrt{2}\langle 0|g'U(|1\rangle \otimes |\psi\rangle) \tag{21}$$

$$= -\sqrt{2}\langle 1|g'gU(|1\rangle \otimes |\psi\rangle) \tag{22}$$

为了说明 $\tilde{U} = U$，我们其实只需要说明对于任意的 $|\alpha\rangle, |\beta\rangle \in \{|0\rangle, |1\rangle\}$，都满足 $\langle\alpha|\tilde{U}(|\beta\rangle \otimes |\psi\rangle) = \langle\alpha|U(|\beta\rangle \otimes |\psi\rangle)$，于是

$$\begin{aligned} \langle 0|\tilde{U}(|0\rangle \otimes |\psi\rangle) &= \langle 0|(\frac{1}{\sqrt{2}}|0\rangle \otimes U'|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes gU'|\psi\rangle) \\ &= \frac{1}{\sqrt{2}}U'|\psi\rangle \\ &= \frac{1}{\sqrt{2}} \cdot \sqrt{2}\langle 0|U(|0\rangle \otimes |\psi\rangle) \\ &= \langle 0|U(|0\rangle \otimes |\psi\rangle) \end{aligned} \tag{23}$$

$$\begin{aligned} \langle 1|\tilde{U}(|0\rangle \otimes |\psi\rangle) &= \langle 1|(\frac{1}{\sqrt{2}}|0\rangle \otimes U'|\psi\rangle + \frac{1}{\sqrt{2}}|1\rangle \otimes gU'|\psi\rangle) \\ &= \frac{1}{\sqrt{2}}gU'|\psi\rangle \\ &= \frac{1}{\sqrt{2}}g \cdot \sqrt{2}\langle 1|gU(|0\rangle \otimes |\psi\rangle) \\ &= \langle 1|U(|0\rangle \otimes |\psi\rangle) \end{aligned} \tag{24}$$

$$\begin{aligned} \langle 0|\tilde{U}(|1\rangle \otimes |\psi\rangle) &= \langle 0|(\frac{1}{\sqrt{2}}|0\rangle \otimes g'U'|\psi\rangle - \frac{1}{\sqrt{2}}|1\rangle \otimes gg'U'|\psi\rangle) \\ &= \frac{1}{\sqrt{2}}g'U'|\psi\rangle \\ &= \frac{1}{\sqrt{2}}g' \cdot \sqrt{2}\langle 0|g'U(|1\rangle \otimes |\psi\rangle) \\ &= \langle 0|U(|1\rangle \otimes |\psi\rangle) \end{aligned} \tag{25}$$

$$\begin{aligned} \langle 1|\tilde{U}(|1\rangle \otimes |\psi\rangle) &= \langle 1|(\frac{1}{\sqrt{2}}|0\rangle \otimes g'U'|\psi\rangle - \frac{1}{\sqrt{2}}|1\rangle \otimes gg'U'|\psi\rangle) \\ &= -\frac{1}{\sqrt{2}}gg'U'|\psi\rangle \\ &= -\frac{1}{\sqrt{2}}gg' \cdot -(\sqrt{2}\langle 1|g'gU(|1\rangle \otimes |\psi\rangle)) \\ &= \langle 1|U(|1\rangle \otimes |\psi\rangle) \end{aligned} \tag{26}$$

σ	σ'	选取的 V
X	Y	HSH
X	Z	I
Y	X	SH
Y	Z	S
Z	X	H
Z	Y	HS

表 2: 验证不同的 *Pauli* 矩阵 σ,σ' 总可以被 H,S 的乘积共轭变换到 X,Z

由此我们证明了 $\tilde{U}=U$。注意到以上线路除了 U' 外，只包含一个 controlled-g' 门、一个 Hadamard 门和一个 controlled-g 门，而任意的 controlled-$\{X,Y,Z\}$ 门都是可以由 $O(1)$ 个 Clifford 门实现的，因此整个线路除了 U' 外，可以通过 $O(n)+1+O(n)=O(n)$ 个 Clifford 门来实现。

结合归纳假设，我们证明了满足 eq. (15) 的 U 可以被 $O(n^2)+O(n)=O(n^2)$ 个 Clifford 门组合实现。

B.3 第三步

我们希望把第二步中的结果推广。考虑任意的 $U\in\mathcal{N}(\mathcal{P}_{n+1})$，记 $G=UZ_1U^\dagger, G'=UX_1U^\dagger$，那么显然有 $G,G'\in\mathcal{P}_{n+1}$。注意到 G 与 G' 必须反对易——因为 Z 与 X 就是反对易的——那么必然存在某个 $j\in[1,n+1]$ 使得 G 与 G' 在第 j 个量子比特上作用了 $\{X,Y,Z\}$ 中不同的门，分别记为 σ 与 σ'。额外添加第一个量子比特与第 j 个量子比特之间的交换门[7]，可以得到

$$\begin{aligned}(\mathrm{SWAP}_{1,j}U)Z_1(\mathrm{SWAP}_{1,j}U)^\dagger &= \mathrm{SWAP}_{1,j}\cdot G\cdot \mathrm{SWAP}_{1,j}^\dagger=\sigma\otimes g\\ (\mathrm{SWAP}_{1,j}U)X_1(\mathrm{SWAP}_{1,j}U)^\dagger &= \mathrm{SWAP}_{1,j}\cdot G'\cdot \mathrm{SWAP}_{1,j}^\dagger=\sigma'\otimes g'\end{aligned}\tag{27}$$

其中 $g,g'\in\mathcal{P}_n$。值得一提的是，对于 Pauli 矩阵 σ,σ'，总会存在由 Hadamard 门与相位门组成的 V，满足 $V\sigma V^\dagger=X, V\sigma' V^\dagger=Z$，至多差一个 global phase。这个结论可以通过表 2 来验证

因此进一步地，我们可以得到

$$\begin{aligned}(V_1\mathrm{SWAP}_{1,j}U)Z_1(V_1\mathrm{SWAP}_{1,j}U)^\dagger &= V_1\mathrm{SWAP}_{1,j}\cdot G\cdot \mathrm{SWAP}_{1,j}^\dagger V_1^\dagger=V_1\sigma V_1^\dagger\otimes g=X\otimes g\\ (V_1\mathrm{SWAP}_{1,j}U)X_1(V_1\mathrm{SWAP}_{1,j}U)^\dagger &= V_1\mathrm{SWAP}_{1,j}\cdot G'\cdot \mathrm{SWAP}_{1,j}^\dagger V_1^\dagger=V_1\sigma' V_1^\dagger\otimes g'=Z\otimes g'\end{aligned}\tag{28}$$

其中 V 的下标表示 V 作用在第一个量子比特上。根据第二步中的结论，我们知道 $V_1\mathrm{SWAP}_{1,j}U$ 可以由 $O(n^2)$ 个 Clifford 门实现，注意到 $(V_1\mathrm{SWAP}_{1,j})^\dagger$ 仅包含 $O(1)$ 个 Clifford 门，这说明了任意的 $U\in\mathcal{N}(\mathcal{P}_{n+1})$ 都可以由 $O(n^2)$ 个 Clifford 门实现，至多差一个 global phase。

C 引理 3 的详细证明

C.1 充分性证明

为了方便起见，我们用 $\overline{h}=h\oplus 1^n$ 代替 h 来改写一下定理 1 中的 ***C1-C5***:

B1 若 $h_i=1,h_j=1$，则 $\Gamma_{i,j}=0$;

B2 若 $h_i=0,h_j=1,\sigma(i)>\sigma(j)$，则 $\Gamma_{i,j}=0$;

B3 若 $h_i=1,h_j=1,\sigma(i)>\sigma(j)$，则 $\Delta_{i,j}=0$;

B4 若 $h_i=0,h_j=0,\sigma(i)<\sigma(j)$，则 $\Delta_{i,j}=0$;

B5 若 $h_i=0,h_j=1$，则 $\Delta_{i,j}=0$。

现在，我们证明对于满足上述条件的 Γ,Δ，都有 $F(O,\Gamma,\Delta)\in\mathcal{B}(h,\sigma)$ 即 $W^{-1}FW\in\mathcal{B}_n$。先验证对于 $\mathcal{B}_n$ 的生成元集 $\{X,S,\mathrm{CNOT}^\downarrow,\mathrm{CZ}\}$，上述叙述成立:

1. $F=X_i$。可以验证此时 $W^{-1}FW$ 一定是 Pauli 算符（注意到 $HX_iH=Z_i$），因此是 $\mathcal{B}_n$ 中的元素。

[7]交换门 SWAP 可以由被三个 CNOT 门实现，因此是 Clifford 算符，自然也是 $\mathcal{P}_{n+1}$ 的正规化子。

2. $F=S_i$。此时 $W^{-1}FW=\begin{cases}S_{\sigma(i)}, & h_i=0\\ H_{\sigma(i)}S_{\sigma(i)}H_{\sigma(i)}, & h_i=1\end{cases}$，注意到 S_i 的存在表明 $\Gamma_{i,i}=1$，从而根据 ***B1*** 有 $h_i=0$，此时 $W^{-1}FW=S_{\sigma(i)}\in\mathcal{B}(h,\sigma)$。

3. $F=\mathrm{CNOT}_{i,j}$，其中 $i<j$，此时有 $\Delta_{j,i}=1$。根据 h_i,h_j 的不同取值可以得到如下表格

h_i	h_j	$W^{-1}\mathrm{CNOT}_{i,j}W$
0	0	$\mathrm{CNOT}_{\sigma(i),\sigma(j)}$
0	1	$\mathrm{CZ}_{\sigma(i),\sigma(j)}$
1	0	$(H\otimes I\cdot\mathrm{CNOT}\cdot H\otimes I)_{\sigma(i),\sigma(j)}$
1	1	$\mathrm{CNOT}_{\sigma(j),\sigma(i)}$

观察到 ***B1-B5*** 在 $\Delta_{j,i}=1$ 时否定了如下几种情况：$h_i=h_j=1$ 且 $\sigma(j)>\sigma(i)$；$h_i=h_j=0$ 且 $\sigma(i)>\sigma(j)$；$h_i=1$ 且 $h_j=0$。可以发现除了这些情况外，$W^{-1}FW\in\mathcal{B}(h,\sigma)$ 总是成立的。

4. $F=\mathrm{CZ}_{i,j}$，此时有 $\Gamma_{i,j}=\Gamma_{j,i}=1$ 根据 h_i,h_j 的不同取值可以得到如下表格

h_i	h_j	$W^{-1}\mathrm{CNOT}_{i,j}W$
0	0	$\mathrm{CZ}_{\sigma(i),\sigma(j)}$
0	1	$\mathrm{CNOT}_{\sigma(i),\sigma(j)}$
1	0	$\mathrm{CNOT}_{\sigma(j),\sigma(i)}$
1	1	$(H\otimes I\cdot\mathrm{CNOT}\cdot H\otimes I)_{\sigma(i),\sigma(j)}$

观察到 ***B1-B5*** 在 $\Gamma_{i,j}=\Gamma_{j,i}=1$ 时否定了如下几种情况：$h_i=h_j=1$；$h_i=0,h_j=1$ 且 $\sigma(i)>\sigma(j)$；$h_i=1,h_j=0$ 且 $\sigma(j)>\sigma(i)$。可以发现除了这些情况外，$W^{-1}FW\in\mathcal{B}(h,\sigma)$ 总是成立的。

对于任意满足上述条件的 $F(O,\Gamma,\Delta)$，由于其一定可以分解成 $\{X,S,\mathrm{CNOT}^{\downarrow},\mathrm{CZ}\}$ 乘积的形式，从而也可以验证 $W^{-1}FW\in\mathcal{B}_n$，这证明了引理的充分性。

C.2 必要性证明

我们先验证 eq. (12) 的结论：$I_n(h,\sigma)$ 是 $\overline{h},\sigma$ 通过 ***C1-C5*** 给 Γ,Δ 的限制数目，也就等于 h,σ 通过 ***B1-B5*** 给 Γ,Δ 的限制数目。为了方便我们沿用后者，并且可以显式地把限制数目写下来

	给 Γ,Δ 的限制数目	记号
B1	$\lvert h\rvert+\sum_{i>j}h_ih_j$	$\lvert h\rvert$ + ①
B2	$\sum_{i>j,\sigma(i)>\sigma(j)}\overline{h}_ih_j+\sum_{i>j,\sigma(i)<\sigma(j)}h_i\overline{h}_j$	② + ③
B3	$\sum_{i>j,\sigma(i)>\sigma(j)}h_ih_j$	④
B4	$\sum_{i>j,\sigma(i)<\sigma(j)}\overline{h}_i\overline{h}_j$	⑤
B5	$\sum_{i>j}\overline{h}_ih_j$	⑥

简单验证一下上面的六项加起来就能够得到 eq. (12)：

$$\begin{aligned}I_n(h,\sigma)&=\lvert h\rvert+①+②+③+④+⑤+⑥\\&=\lvert h\rvert+(①+⑥)+(②+④)+(③+⑤)\\&=\lvert h\rvert+\sum_{i>j}h_j+\sum_{i>j,\sigma(i)>\sigma(j)}h_j+\sum_{i>j,\sigma(i)<\sigma(j)}\overline{h}_j\\&=\lvert h\rvert+\sum_{i>j}h_j+\sum_{i>j}\overline{h}_j+\sum_{i>j,\sigma(i)>\sigma(j)}h_j-\overline{h}_j\\&=\frac{n(n-1)}{2}+\lvert h\rvert+\sum_{1\le i<j\le n:\sigma(i)<\sigma(j)}(-1)^{1+h_i}\end{aligned}\tag{29}$$

根据 [1]，我们知道 $\lvert\mathcal{C}_n\rvert=2^{n^2+2n}\prod_{i=1}^{n}(4^i-1)$，回忆 $\lvert\mathcal{B}_n\rvert=2^{n^2+2n}$，为了完成证明，我们只需要验证如下等式成立

$$F(n)\triangleq\sum_{h\in\{0,1\}^n}\sum_{\sigma\in\mathcal{S}_n}2^{I_n(h,\sigma)}=\prod_{i=1}^{n}(4^i-1)\tag{30}$$

这里的证明手段是归纳。$n=1$ 时，σ 一定是单位置换 e，$I_1(0,e)=0,I_1(1,e)=1$ 而 $F(1)=3=2^{I_1(0,e)}+2^{I_1(1,e)}$，故成立。

假设结论 eq. (30) 对于 n 成立。对于任意的 $h \in \{0,1\}^{n+1}$ 以及 $\sigma \in \mathcal{S}_{n+1}$，将 h 写成 $h = (h_1, h')$ 的形式，其中 $h' \in \{0,1\}^n$；取 $\sigma' \in \mathcal{S}_n$ 满足 $\sigma'(j) = \begin{cases} \sigma(j+1), & \sigma(j+1) < \sigma(1) \\ \sigma(j+1) - 1, & \sigma(j+1) > \sigma(1) \end{cases}$（相当于移除 σ 的第一位后得到新的 n 阶排列），可以通过验证得到

$$I_{n+1}(h,\sigma) = I_n(h',\sigma') + n + h_1 + (n+1-\sigma(1))(-1)^{1+h_1} \tag{31}$$

注意到映射 $(h,\sigma) \to (h_1, h', \sigma(1), \sigma')$ 是一一对应的，因此

$$\begin{aligned} F(n+1) &= F(n) \sum_{h_1 \in \{0,1\}} \sum_{\sigma(1)=1}^{n+1} 2^{n+h_1+(n+1-\sigma(1))(-1)^{1+h_1}} \\ &= F(n) \cdot 2^n \sum_{m=1}^{n+1} (2^{-n-1+m} + 2^{n-m+2}) \\ &= F(n) \cdot 2^n \sum_{m=-n}^{n+1} 2^m \\ &= F(n) \cdot 2^n (2^{n+2} - 2^{-n}) \\ &= (4^{n+1} - 1) F(n) \end{aligned} \tag{32}$$

从而证明了 eq. (30) 对于任何 n 都成立，同时也完成了引理的必要性证明。

D 引理 4 的详细证明

只需要补充说明所有左陪集 $F_j\mathcal{B}_n(h,\sigma)$ 是两两不交的即可。假设对于 $F_i \neq F_j$ 有 $F_i\mathcal{B}_n(h,\sigma) \cap F_j\mathcal{B}_n(h,\sigma) \neq \varnothing$，这就说明 $F_j^{-1}F_i \in \mathcal{B}_n(h,\sigma)$，然而 $F_i, F_j \in \mathcal{B}(\overline{h},\sigma)$ 故 $F_j^{-1}F_i \in \mathcal{B}(\overline{h},\sigma)$，根据引理 3 可知 $F_j^{-1}F_i \in \mathcal{P}_n$。注意到定义指出 F_i, F_j 的 Pauli 部分都是平凡的，这导致了 $F_j^{-1}F_i = I$ 即 $F_i = F_j$，产生矛盾。故左陪集两两不交。

E 引理 5 的详细证明

首先考察该算法的正确性，即验证其确实按照 $P_n(h,\sigma)$ 的概率分布来生成了 h 和 σ。考虑对 n 归纳：$n = 1$ 时，有 $P_1(0,e) = p(0,1) = \frac{1}{3}, P_1(1,e) = p(1,1) = \frac{2}{3}$，因此该算法按照 $P_1(h,\sigma)$ 的概率分布生成了 h 和 σ；$n > 1$ 时，考虑在 section C 中使用过的分解，即对于 h,σ 构造 h',σ'，由于 $I_n(h,\sigma) = I_{n-1}(h',\sigma') + (n-1) + h_1 + (n-\sigma(1))(-1)^{1+h_i}$，故一轮循环实质上按照正确的概率生成了 h_1 和 $\sigma(1)$，注意到 $(h,\sigma) \to (h_1, h', \sigma(1), \sigma')$ 的一一对应关系，只需要在接下来的循环中生成 h', σ' 即可。结合归纳假设，算法的正确性得证。

关于算法的复杂度，我们证明单轮循环（即单次对 $h_1, \sigma(1)$ 的采样）的复杂度为 $\tilde{O}(1)$。考虑按照一定顺序写下 $\{0,1\} \times [1,m]$ 中的所有有序对

$$P \triangleq [p(1,1), p(1,2), \cdots, p(1,m), p(0,m), p(0,m-1), \cdots, p(0,1)] \tag{33}$$

可以验证 P 作为一个数组（也可以视作概率分布）满足

$$P_a = \frac{2^{2m-1-a}}{2^{2m}-1}, a \in [0, 2m-1] \tag{34}$$

因此想要实现对概率分布 P 的随机采样，只需要令

$$a = 2m - \lceil \log_2(r(2^{2m}-1)+1) \rceil \tag{35}$$

其中 $r \in [0,1]$ 是一个均匀随机分布的实数。因此，一轮循环需要消耗 $O(\log n)$ 个随机比特，同时运行时间为 $\tilde{O}(1)$。

F 代码实现

代码实现主要参考自Qiskit，修改了对 eq. (3) 中算符 F 的采样方式，即严格按照 ***C1-C5*** 的限制对 Γ, Δ 进行采样。

```
import numpy as np
from numpy.random import default_rng
from qiskit.quantum_info import Clifford, StabilizerTable, random_clifford

```

```python
def my_random_clifford(num_qubits, seed=None):
    """
    Return a random Clifford operator, minimizing the number of random bits used.
    Args:
        num_qubits (int): the number of qubits for the Clifford
        seed (int or np.random.Generator): Optional. Set a fixed seed or generator for RNG.
    Returns:
        Clifford: a random Clifford operator.
    """
    if seed is None:
        rng = np.random.default_rng()
    elif isinstance(seed, np.random.Generator):
        rng = seed
    else:
        rng = default_rng(seed)

    had, perm = _sample_qmallows(num_qubits, rng)
    gamma1, delta1, gamma2, delta2 = _generate_tril(had, perm, rng)

    # For large num_qubits numpy.inv function called below can
    # return invalid output leading to a non-symplectic Clifford
    # being generated. This can be prevented by manually forcing
    # block inversion of the matrix.
    block_inverse_threshold = 50

    # Compute stabilizer table
    zero = np.zeros((num_qubits, num_qubits), dtype=np.int8)
    prod1 = np.matmul(gamma1, delta1) % 2
    prod2 = np.matmul(gamma2, delta2) % 2
    inv1 = _inverse_tril(delta1, block_inverse_threshold).transpose()
    inv2 = _inverse_tril(delta2, block_inverse_threshold).transpose()
    table1 = np.block([[delta1, zero], [prod1, inv1]])
    table2 = np.block([[delta2, zero], [prod2, inv2]])

    # Apply qubit permutation
    table = table2[np.concatenate([perm, num_qubits + perm])]

    # Apply layer of Hadamards
    inds = had * np.arange(1, num_qubits + 1)
    inds = inds[inds > 0] - 1
    lhs_inds = np.concatenate([inds, inds + num_qubits])
    rhs_inds = np.concatenate([inds + num_qubits, inds])
    table[lhs_inds, :] = table[rhs_inds, :]

    # Apply table
    table = np.mod(np.matmul(table1, table), 2).astype(bool)

    # Generate random phases
    phase = rng.integers(2, size=2 * num_qubits).astype(bool)
    return Clifford(StabilizerTable(table, phase))

```

```python
def _generate_tril(had, perm, rng):
    """
    Return four 01-matrices gamma1, delta1, gamma2, delta2, minimizing the number of random bits used.
    Args:
        had: the Hadamard layer
        perm: the permutation layer
        both of them put some restrictions on gamma1 and delta1
        rng: random number generator
    Returns:
        four 01-matrices gamma1, delta1, gamma2, delta2 for Clifford sampling.
    """

    num_qubits = had.size

    gamma1 = np.diag(rng.integers(2, size=num_qubits, dtype=np.int8))
    gamma2 = np.diag(rng.integers(2, size=num_qubits, dtype=np.int8))
    delta1 = np.eye(num_qubits, dtype=np.int8)
    delta2 = delta1.copy()

    for i in range(num_qubits):
        for j in range(i):
            gamma2[i, j] = gamma2[j, i] = rng.integers(2, dtype=np.int8)
            delta2[i, j] = rng.integers(2, dtype=np.int8)

            # restrictions for gamma1
            if had[i] == 1 and had[j] == 1:
                gamma1[i, j] = gamma1[j, i] = rng.integers(2, dtype=np.int8)
            if had[i] == 1 and had[j] == 0 and perm[i] < perm[j]:
                gamma1[i, j] = gamma1[j, i] = rng.integers(2, dtype=np.int8)
            if had[i] == 0 and had[j] == 1 and perm[i] > perm[j]:
                gamma1[i, j] = gamma1[j, i] = rng.integers(2, dtype=np.int8)

            # restrictions for delta1
            if had[i] == 0 and had[j] == 1:
                delta1[i, j] = rng.integers(2, dtype=np.int8)
            if had[i] == 1 and had[j] == 1 and perm[i] > perm[j]:
                delta1[i, j] = rng.integers(2, dtype=np.int8)
            if had[i] == 0 and had[j] == 0 and perm[i] < perm[j]:
                delta1[i, j] = rng.integers(2, dtype=np.int8)

    return gamma1, delta1, gamma2, delta2

def _sample_qmallows(n, rng=None):
    """Sample from the quantum Mallows distribution."""

    if rng is None:
        rng = np.random.default_rng()

    # Hadmard layer
    had = np.zeros(n, dtype=bool)

```

```
    # Permutation layer
    perm = np.zeros(n, dtype=int)

    inds = list(range(n))
    for i in range(n):
        m = n - i
        eps = 4 ** (-m)
        r = rng.uniform(0, 1)
        index = -int(np.ceil(np.log2(r + (1 - r) * eps)))
        had[i] = index < m
        if index < m:
            k = index
        else:
            k = 2 * m - index - 1
        perm[i] = inds[k]
        del inds[k]
    return had, perm

def _inverse_tril(mat, block_inverse_threshold):
    """Invert a lower-triangular matrix with unit diagonal."""
    # Optimized inversion function for low dimensions
    dim = mat.shape[0]

    if dim <= 2:
        return mat

    if dim <= 5:
        inv = mat.copy()
        inv[2, 0] = mat[2, 0] ^ (mat[1, 0] & mat[2, 1])
        if dim > 3:
            inv[3, 1] = mat[3, 1] ^ (mat[2, 1] & mat[3, 2])
            inv[3, 0] = mat[3, 0] ^ (mat[3, 2] & mat[2, 0]) ^ (mat[1, 0] & inv[3, 1])
        if dim > 4:
            inv[4, 2] = (mat[4, 2] ^ (mat[3, 2] & mat[4, 3])) & 1
            inv[4, 1] = mat[4, 1] ^ (mat[4, 3] & mat[3, 1]) ^ (mat[2, 1] & inv[4, 2])
            inv[4, 0] = (
                mat[4, 0]
                ^ (mat[1, 0] & inv[4, 1])
                ^ (mat[2, 0] & inv[4, 2])
                ^ (mat[3, 0] & mat[4, 3])
            )
        return inv % 2

    # For higher dimensions we use Numpy's inverse function
    # however this function tends to fail and result in a non-symplectic
    # final matrix if n is too large.
    if dim <= block_inverse_threshold:
        return np.linalg.inv(mat).astype(np.int8) % 2

    # For very large matrices  we divide the matrix into 4 blocks of
    # roughly equal size and use the analytic formula for the inverse
```

```
    # of a block lower-triangular matrix:
    # inv([[A, 0],[C, D]]) = [[inv(A), 0], [inv(D).C.inv(A), inv(D)]]
    # call the inverse function recursively to compute inv(A) and invD

    dim1 = dim // 2
    mat_a = _inverse_tril(mat[0:dim1, 0:dim1], block_inverse_threshold)
    mat_d = _inverse_tril(mat[dim1:dim, dim1:dim], block_inverse_threshold)
    mat_c = np.matmul(np.matmul(mat_d, mat[dim1:dim, 0:dim1]), mat_a)
    inv = np.block([[mat_a, np.zeros((dim1, dim - dim1), dtype=int)], [mat_c, mat_d]])
    return inv % 2

if __name == "__main__":
    circuit = random_clifford(4)
    print(circuit)
    circuit2 = my_random_clifford(4)
    print(circuit2)
```