CSE 291 / Math 277A - Quantum Complexity Theory November 1, 2022

Lecture 11

Lecturer: Daniel Grier *Scribe: Yiyang Qiu*

1 Warm-up: Element Distinctness

Recall the collision problem and a new promise-free version called "element distinctness":

Collision:
Oracle: $f\colon \{0,1\}^n \to \{0,1\}^n$
Promise: f is either 1-to-1 or 2-to-1
Goal: Decide which

Element Distinctness:
Oracle: $f\colon \{0,1\}^n \to \{0,1\}^n$
Goal: Decide whether f is 1-to-1 or not.

In other words, the element distinctness question can be phrased as "Is every element in the image of f distinct (i.e., f is 1-to-1), or are there are pair of distinct inputs x, y such that $f(x) = f(y)$?" Proving a quantum query lower bound from scratch for element distinctness is quite challenging, but suppose we knew the following lower bound for the collision problem:

Theorem 1 ([Shi02]). *The quantum query complexity of* Collision *is* $\Omega(2^{n/3})$.

Our goal will be to reduce the collision problem to the element distinctness problem, so that a lower bound for element distinctness will imply a lower bound for collision.

Theorem 2. *The quantum query complexity of* Element Distinctness *is* $\Omega(2^{2n/3})$.

Proof. Suppose there is a quantum query algorithm for Element Distinctness with $o(2^{2n/3})$ queries. We claim that this implies a quantum query algorithm for Collision with $o(2^{n/3})$ queries, contradicting the lower bound of Shi (Theorem 1).

Therefore, suppose we have some instance f of Collison. Sample a random subset of the domain $R \subset \{0,1\}^n$ of size $|R| = 2^{n/2}$. By the Birthday Paradox, with high probability, there exists $x, y \in R$ such that $f(x) = f(y)$ if f is 2-to-1. Now run the Element Distinctness algorithm with f restricted to the subset R to determine whether a collision exists, which then implies whether f is 2-to-1 or 1-to-1. The query complexity of the algorithm is then $o((2^{n/2})^{2/3}) = o(2^{n/3})$. We conclude that any quantum query algorithm for Element Distinctness must make $\Omega(2^{\frac{2n}{3}})$ oracle queries. $\square$

2 Pros and Cons of Query Complexity

The quantum query complexity of a function measures the number of queries that are made to the function in order to determine some property of the function. There are pros and cons to this approach:
Pros:

1. We can actually prove rigorous lower bounds using query complexity, e.g. BBBV theorem, Reduction, Polynomial Method, Adversary. Using query complexity method, we can also prove quantum advantage of some quantum algorithms.

2. Query algorithms are the foundations for useful quantum algorithms, e.g. Grover's algorithm, and Simon's Algorithm.

Cons:

1. It is not always reasonable to model functions as a black box. The conclusions implied by query complexity analysis do not necessarily generalize to the non-black box setting.

Next we will attempt to get over this con. In other word, we would try to avoid black box function in our query complexity model. In the pursuit of this goal, we will sacrifice Pro 2. First, we will take a pretty significant detour...

3 Clifford Circuits

First, a simple gate-based definition:

Definition 3. *A* Clifford circuit *is constructed by composing any number of controlled-not* (CNOT)*, Hadmard* (H)*, and Phase* (S) *gates. The set of all unitaries that arise from Clifford circuits is called the* Clifford group*. A* Cifford state *is obtained by applying a Clifford circuit to the all-zeros state.*

To recap, the fundamental Clifford gates are

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

We have seen throughout class that Clifford circuits have been important subroutines in many quantum algorithms. We will see later that Clifford circuits can be used to show some type of quantum advantage. For the moment, however, we begin with a somewhat surprising theorem that shows that Clifford circuits can be simulated in polynomial time on a classical computer.

Theorem 4 (Gottesman–Knill [Got98])**.** *Clifford circuits can be efficiently simulated by a classical computer.*

There are several possible proofs for the Gottesman-Knill theorem, each of which reveals some useful structure within the Clifford group. Furthermore, while each proof results in a polynomial-time simulation algorithm for Clifford circuit simulation, the exact exponent of the polynomial depends on the specific representation that is used.

Let us start with what might be called the "straightforward" method of simulation. Namely, we will keep track of the state vector of the quantum state as we apply each Clifford gate in the circuit. Of course, we can't write out the entire state vector, as this would require exponential space. It turns out, however, that Clifford states have the following special form:

Lemma 5. *[DM03, VDN10] Every n-qubit Clifford state can be written as*

$$|\psi\rangle = \frac{1}{\sqrt{|\mathcal{A}|}} \sum_{x \in \mathcal{A}} (-1)^{q(x)} i^{\ell(x)} |x\rangle$$

where

- *$\mathcal{A}$ is an affine space: $\mathcal{A} = \{My + b \pmod 2 \mid y \in \{0,1\}^r\}$ for $M \in \{0,1\}^{n \times r}$ and $b \in \{0,1\}^n$.*
- *$q(x)$ is a quadratic form: $q(x) = \sum_{i<j} q_{ij} x_i x_j$ with $q_{ij} \in \{0,1\}$.*
- *$\ell(x)$ is a linear form: $\ell(x) = \sum_i \ell_i x_i$ with $\ell_i \in \{0,1,2,3\}$.*

The natural proof of this statement also gives a classical simulation algorithm. That is, starting with the all-zeroes state (which is trivially of the above form), show how it evolves under the application of each one of the fundamental Clifford operations ($CNOT$, H, or S). Since each update to the state takes polynomial time, the entire computation will take polynomial time. By induction, we need to understand the following cases:

- **Apply S on qubit i:**

$$S|\psi\rangle = \frac{1}{\sqrt{|\mathcal{A}|}} \sum_{x \in \mathcal{A}} (-1)^{q(x)} i^{\ell(x)} S|x\rangle = \frac{1}{\sqrt{|\mathcal{A}|}} \sum_{x \in \mathcal{A}} (-1)^{q(x)} i^{\ell(x)} i^{x_i} |x\rangle$$

  In other words, if we let $\ell'(x) = l(x) + x_i$, then we have an updated representation of the state with Affine space $\mathcal{A}$, quadratic form $q(x)$, and linear form $\ell'(x)$.

- **Apply** CNOT **from qubit** i **to qubit** j**:**

$$\text{CNOT}\,|\psi\rangle = \frac{1}{\sqrt{|\mathcal{A}|}} \sum_{x\in\mathcal{A}} (-1)^{q(x)} i^{\ell(x)}\, \text{CNOT}\,|x\rangle = \frac{1}{\sqrt{|\mathcal{A}|}} \sum_{x\in\mathcal{A}} (-1)^{q(x)} i^{\ell(x)}\, |\text{CNOT}\,x\rangle$$

  where the last equation reflects the fact that CNOT can be identified with an $n \times n$ Boolean matrix which XOR's the ith bit into the jth bit of the n-bit vector x. Therefore, we now have

  – *Affine space:* $\mathcal{A}' = \{M'y + b' \mid \forall y \in \{0,1\}^n$ for $M' = \text{CNOT}\,M$ and $b' = \text{CNOT}\,b$.

  – *Quadratic form:* Notice that we can write $q(x) = x^T Q x$ for some upper triangular matrix Q. Therefore, the updated quadratic form can be written as $q'(x) = q(\text{CNOT}\,x) = x^T Q' x$ with $Q' = \text{CNOT}\,Q\,\text{CNOT}$.

  – *Linear form:* Similar to above, we can write $\ell(x) = \ell^T x$ for the vector $\ell = (\ell_1, \ldots, \ell_n)$. Therefore, the updated linear form can be written as $\ell'(x) = \ell(\text{CNOT}\,x) = (\ell')^T x$ where $(\ell')^T = \ell^T\,\text{CNOT}$.

- **Apply** H**:** The proof for applying H is nontrivial, so we leave it for now. See proof in [VDN10].

In conclusion every Clifford state $|\psi\rangle$ can be written of the form in Lemma 5. Furthermore, the inductive proof reveals a simple algorithm to compute $\mathcal{A}, \ell, q$ of a state given the sequence of Clifford gates that construct the state.

4 Heisenberg Picture

In the Heisenberg picture of simulation [Got98], we do not represent the quantum state by its state vector, but rather as a list of "stabilizers" of the state, i.e., unitary matrices for which the original state vector is an eigenvector. It will turn out that this is a particularly useful representation for states generated by a Clifford circuits. Before we do this, however, let us describe a important subgroup of the Clifford group called the Pauli group that will be the basis of this representation.

4.1 Pauli Group

The single-qubit Pauli group is generated by Pauli matrices, which are:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The Pauli matrices also have a bunch of nice properties (I is the 2×2 identity matrix):

- Hermitian: $X = X^\dagger$, $Y = Y^\dagger$, $Z = Z^\dagger$
- Square to the identity: $X^2 = Y^2 = Z^2 = I$
- Traceless: $\text{tr}(X) = \text{tr}(Y) = \text{tr}(Z) = 0$
- Same determinant: $\det(X) = \det(Y) = \det(Z) = -1$
- Anticommutation: $XY = -YX$, $XZ = -ZX$, $YZ = -ZY$
- Cyclic structure: $\begin{matrix} XY = iZ & YZ = iX & ZX = iY \\ YZ = -iX & ZY = -iX & XZ = -iY \end{matrix}$

These Pauli matrices form a group $\mathcal{P}_1$ of order 16 under matrix multiplication, where each element is of the form αP with $\alpha \in \{\pm 1, \pm i\}$ and $P \in \{I, X, Y, Z\}$. More generally, each element of the n-qubit Pauli group $\mathcal{P}_n$ is of the form $\alpha P_1 \otimes P_2 \otimes \cdots \otimes P_n$ where $\alpha \in \{\pm 1, \pm i\}$ and $P_i \in \{I, X, Y, Z\}$.

Although $|\mathcal{P}_n| = 4 \cdot 4^n$, $\mathcal{P}_n$ is generated by just $2n$ elements: Pauli elements with a single Z and Pauli elements with a single X. For example, on 3-qubits, the generators are

Z-elements	X-elements
$Z \otimes I \otimes I$	$X \otimes I \otimes I$
$I \otimes Z \otimes I$	$I \otimes X \otimes I$
$I \otimes I \otimes Z$	$I \otimes I \otimes X$

and using the multiplication properties of the Pauli matrices, one can check that these do indeed generate the entire group.

We will often refer to the Pauli elements without the phase as the n-qubit Pauli matrices. In fact, the n-qubit Pauli matrices are particularly nice because they form a basis for all complex matrices:

Fact 6. *The n-qubit Pauli matrices form a basis for all complex $2^n \times 2^n$ matrices.*

Proof. Treat each $2^n \times 2^n$ matrix A as a vector of length 4^n denoted by $\mathrm{vec}(A)$. Then, we can express inner products between matrices A and B as $\mathrm{vec}(A) \cdot \mathrm{vec}(B) = \mathrm{tr}(AB^\dagger)$.

We now claim that all 4^n Pauli matrices are linearly independent. To see this, let $P = P_1 \otimes \cdots \otimes P_n$ and $Q = Q_1 \otimes \cdots \otimes Q_n$ be two distinct n-qubit Pauli matrices. We have that

$$\mathrm{tr}(PQ) = \mathrm{tr}(P_1Q_1 \otimes \cdots \otimes P_nQ_n) = \mathrm{tr}(P_1Q_1)\cdots \mathrm{tr}(P_nQ_n) = 0$$

since there must exist at least some index i for which $P_i \neq Q_i$. In more detail, notice that when $P_i \neq Q_i$ for $P_i, Q_i \in \{I, X, Y, Z\}$ that $P_iQ_i = \alpha R$ for $\alpha \in \{\pm 1, \pm i\}$ and $R \in \{X, Y, Z\}$. Therefore, $\mathrm{tr}(P_iQ_i) = \alpha\,\mathrm{tr}(R) = 0$ since the Pauli matrices are traceless.

Since we have a space of dimension 4^n and all 4^n Pauli matrices are linearly independent, we must have that the span of the Pauli matrices is the entire space. $\square$

As a special case, we can look at the Pauli decomposition for density matrices of pure states.

Fact 7. *Let $|\psi\rangle$ be an n-qubit pure state. The density matrix $|\psi\rangle\langle\psi| = \sum_{P \in \{I,X,Y,Z\}^{\otimes n}} \alpha_P P$ with $\alpha_P \in \mathbb{R}$ and $\sum_P \alpha_P^2 = 2^{-n}$.*

Proof. By Fact 6, we can write $|\psi\rangle\langle\psi| = \sum_P \alpha_P P$ where $P \in \{I, X, Y, Z\}^{\otimes n}$ and $\alpha_P \in \mathbb{C}$. Since the Pauli matrices are Hermitian, we have

$$|\psi\rangle\langle\psi| = \sum_P \alpha_P P = \sum_P \alpha_P^* P.$$

This implies that $\alpha_P = \alpha_P^*$ since the P are linearly independent, which in turn implies that the α_P coefficients are real. Furthermore, using the purity of $|\psi\rangle$ we have

$$1 = \mathrm{tr}(|\psi\rangle\langle\psi|) = \mathrm{tr}(|\psi\rangle\langle\psi| \cdot |\psi\rangle\langle\psi|) = \sum_{P,Q} \alpha_P\alpha_Q\,\mathrm{tr}(PQ) = \sum_P \alpha_P^2\,\mathrm{tr}(I^{\otimes n}) = 2^n \sum_P \alpha_P^2$$

where we've used that $\mathrm{tr}(PQ) = 0$ for $P \neq Q$, $P^2 = I^{\otimes n}$, and $\mathrm{tr}(I^{\otimes n}) = 2^n$. $\square$

4.2 Pauli matrices and stabilizer groups

Now that we have defined the Pauli group, let's use it to help us represent a quantum state.

Definition 8. *For n-qubit state $|\psi\rangle$, we say unitary U stabilizes $|\psi\rangle$ iff $U|\psi\rangle = |\psi\rangle$. Let the stabilizer group* $\mathrm{Stab}(|\psi\rangle) \subseteq \mathcal{P}_n$ *be the set of all Pauli elements that stabilize $|\psi\rangle$.*

Fact 9. $\mathrm{Stab}(|\psi\rangle)$ *is an Abelian group under matrix multiplication.*

Proof. If Pauli elements P and Q both stabilize $|\psi\rangle$, then so do PQ and $P^\dagger$.

To argue that this group must be Abelian, notice that any two Pauli's P and Q either commute ($PQ = QP$) or anti-commute ($PQ = -QP$). Suppose that P and Q anti-commute. We get the following contradiction:

$$|\psi\rangle = PQ\,|\psi\rangle = -QP\,|\psi\rangle = -\,|\psi\rangle\,.$$

Therefore, P and Q must commute, and the stabilizer group is Abelian. $\square$

Does the stabilizer group constitute a reasonable representation state? By a counting argument, one can see that there exist many quantum states whose stabilizer groups are empty, so this stabilizer representation won't be very good for them. On the other hand, if the stabilizer group is large enough, then it is a unique representation of the state:

Fact 10. *For any stabilizer group of size 2^n, there is only one state (up to global phase) with that stabilizer group.*

Proof. Let $|\psi\rangle$ be an n-qubit state with $|\operatorname{Stab}(|\psi\rangle)| = 2^n$. Let $|\varphi\rangle\langle\varphi|$ be the density matrix of any state stabilized by every element in $\operatorname{Stab}(|\psi\rangle)$. We claim that this density matrix is unique. To see this, first expand $|\varphi\rangle\langle\varphi|$ in the Pauli basis using Fact 7: $|\varphi\rangle\langle\varphi| = \sum_P \alpha_P P$. Now take any $Q \in \operatorname{Stab}(|\psi\rangle)$. We have

$$1 = \operatorname{tr}(|\varphi\rangle\langle\varphi|) = \operatorname{tr}(Q\,|\varphi\rangle\langle\varphi|) = \sum_P \alpha_P \operatorname{tr}(QP) = \alpha_Q \operatorname{tr}(I^{\otimes n}) = \frac{\alpha_Q}{2^n}$$

where we have used (in order) that Q stabilizes $|\varphi\rangle$, that $\operatorname{tr}(QP) = 0$ for $Q \neq P$, and that $Q^2 = I^{\otimes n}$ for any Pauli. In other words, for each of the 2^n stabilizers in $\operatorname{Stab}(|\psi\rangle)$, the corresponding coefficient in the Pauli expansion is 2^{-n}. Notice that this implies all other Pauli coefficients must be zero since by Fact 7 we have

$$2^{-n} = \sum_P \alpha_P^2 = \sum_{Q \in \operatorname{Stab}(|\psi\rangle)} \alpha_Q^2 + \sum_{P \notin \operatorname{Stab}(|\psi\rangle)} \alpha_P^2 = 2^{-n} + \sum_{P \notin \operatorname{Stab}(|\psi\rangle)} \alpha_P^2.$$

Since the α_P coefficients are real, their squares must be non-negative. On the other hand, the above equation implies that $\sum_{P \notin \operatorname{Stab}(|\psi\rangle)} \alpha_P^2 = 0$, so they must all be zero. □

Because of this, let's focus our attention on states that have stabilizer groups of size 2^n. This raises the obvious question: which states have these large stabilizer groups? Well, to start, notice that the all-zeroes state is stabilized by every Pauli matrix which is a tensor product of identity matrices (I) and Pauli-Z matrices. There are 2^n such matrices, so they comprise the entire stabilizer group.

We now have a stabilizer representation of our initial state. A reasonable requirement is that we can determine how the stabilizer group changes when we apply a unitary to the state:

Fact 11. *U stabilizes $|\psi\rangle$ iff $VUV^\dagger$ stabilizes $V|\psi\rangle$.*

Proof. $|\psi\rangle = U|\psi\rangle \iff V|\psi\rangle = VU|\psi\rangle = (VUV^\dagger)V|\psi\rangle$ □

In other words, if we apply a gate to our state, then we can update the stabilizer group representation by conjugating every stabilizer by the gate. In general, we don't have any guarantee on the form of $VUV^\dagger$. That is, even if U is a Pauli matrix, it's conjugation under an arbitrary unitary might not be Pauli.

We are now ready to reveal the key feature of Clifford circuits:

Theorem 12. *The Clifford group is the* normalizer *of the Pauli group. That is, a unitary U is Clifford iff $UPU^\dagger$ is in the Pauli group for all Pauli matrices P.*

For now, we just describe the direction which is important Clifford circuit simulation: if U is Clifford, then $UPU^\dagger$ is in the Pauli group. To make our lives easier, we will simplify down to just a few special cases that we have to check:

- *Only have to check* CNOT*, H, and S:* Since U is Clifford, we can write $U = g_1 \cdots g_m$ as a product of CNOT, H, and S gates. Therefore, if each gate g_i maps Pauli elements to Pauli elements under conjugation, then we have

$$UPU^\dagger = g_1 \cdots g_{m-1}(g_m P g_m^\dagger)g_{m-1}^\dagger \cdots g_1 = g_1 \cdots g_{m-2}(g_{m-1} P' g_{m-1}^\dagger)g_{m-2}^\dagger \cdots g_1 = \ldots$$

  is another Pauli matrix.

- *Only have to check generators of the Pauli group:* Recall that every n-qubit Pauli P can be expressed as the product of $2n$ different generators, which consist of the Pauli elements with a single Z term and a single X term. Therefore, if we specify how a unitary affects each such generator under conjugation, then we can determine its more general behavior. Namely, if $P = P_1 P_2 \cdots P_k$ for Pauli generators P_i, then
$$UPU^\dagger = UP_1P_2\cdots P_kU^\dagger = (UP_1U^\dagger)(UP_2U^\dagger)U\cdots U^\dagger(UP_kU^\dagger)$$
for any unitary U.

To complete the proof, we can simply show how each of CNOT, H, and S affects the Pauli generators:

P	$HPH^\dagger$
X	Z
Z	X

P	$SPS^\dagger$
X	Y
Z	Z

P	$\text{CNOT}\,P\,\text{CNOT}^\dagger$
$X\otimes I$	$X\otimes X$
$I\otimes X$	$I\otimes X$
$Z\otimes I$	$Z\otimes I$
$I\otimes Z$	$Z\otimes Z$

As a direct consequence, we can simulate Clifford circuits by keeping track of the stabilizer group and how it changes under the application of each gate in the circuit.

5 Simulation of Clifford circuits using stabilizer groups

There are two remaining issues to address in order to obtain an efficient classical simulation of Clifford circuits using stabilizer groups. The first is a question of efficiency: if we need to keep track of all 2^n stabilizer elements, then our algorithm would take exponential time. However, once again, we only need to keep track of the *generators* of the stabilizer group:

Fact 13. *Let $|\psi\rangle$ be an n-qubit Clifford state. There exists n Pauli generators $g_1, \ldots g_n \in \mathcal{P}_n$ such that every $P \in \mathrm{Stab}(|\psi\rangle)$ can be expressed as the product of generator elements. Furthermore, the generators are independent in the sense that no generator can be expressed as the product of the other generators.*

Proof. Recall that the stabilizer group of the all-zeroes state are all the Z-type Pauli elements. One can check that this group is generated by the Pauli matrices with a single Z term: $Z\otimes I\otimes\cdots\otimes I$, $I\otimes Z\otimes\cdots\otimes I$, $\ldots$, $I\otimes I\otimes\cdots\otimes Z$. There are n such generators, and it is of minimal size.

Since every Clifford state is of the form $U|0^n\rangle$ for Clifford unitary U, we have that the stabilizer group is generated by $UZ_iU^\dagger$ where Z_i is the Pauli matrix with a single Z in the ith register. ☐

It's worth noting that although every stabilizer group can be represented by n generators $g_1, \ldots, g_n$, this representation is far from unique. In particular, one can check that multiplying the first generator into the second yields a new set of generators $g_1, g_1g_2, \ldots, g_n$. In some cases, this idea will allow us to simplify our set of stabilizer generators using a multiplicative version of Gaussian elimination.

Let's turn our attention to the final issue: how do we deal with measurements? Without loss of generality, let's just focus on a computational basis measurement on the first qubit. There are two cases:

Deterministic Measurement: This occurs when the state is either $|0\rangle\otimes|\psi'\rangle$ or $|1\rangle\otimes|\psi'\rangle$. In other words, measuring the state results in $|0\rangle$ or $|1\rangle$ with probability 1, and it doesn't change the state. Even though we don't need to update our stabilizer group representation, there are still two potential issues: how do we determine if the measurement will be deterministic? and how can we determine the outcome of the measurement?

For the first problem, notice that there cannot be any Pauli stabilizers of the form $X\otimes P$ or $Y\otimes P$ for $P\in\mathcal{P}_{n-1}$ because both stabilizers would flip the first qubit. We claim that if these stabilizers are not present, then the measurement will be deterministic. To see this, notice that a computational basis measurement projects the first qubit onto $|0\rangle$ or $|1\rangle$. We can express this projection as $|0\rangle\langle 0| = (I+Z)/2$ and $|1\rangle\langle 1| = (I-Z)/2$, respectively. Notice, however that if g stabilizes $|\psi\rangle$ and it is of the form $I\otimes P$ or $Z\otimes P$, then it still stabilizes the state after projection:
$$g\left(\frac{I\pm Z}{2}\otimes I\otimes\cdots\otimes I\right)|\psi\rangle = \left(\frac{I\pm Z}{2}\otimes I\otimes\cdots\otimes I\right)g|\psi\rangle = \left(\frac{I\pm Z}{2}\otimes I\otimes\cdots\otimes I\right)|\psi\rangle.$$

Since the stabilizer group has not changed and the stabilizer group is unique (Fact 10), the measurement must have been deterministic.

In summary, we can now easily detect whether or not the measurement will be deterministic by checking if all of the stabilizer generators start with either an I or a Z. To determine the result of the measurement, we must now learn whether or not the state is of the form $|0\rangle \otimes |\psi'\rangle$ or $|1\rangle \otimes |\psi'\rangle$. Notice that the first state is stabilized by $Z \otimes I \otimes \cdots \otimes I$, while the second state is stabilized by $-Z \otimes I \otimes \cdots \otimes I$. We simply need to decide which. We can find it using Gaussian elimination in time $O(n^3)$.

Random Measurement: Since the measurement is not deterministic, there must exist some stabilizer generator that starts with either an X or a Y. Using Lemma 5, one can check that the measurement result is either $|0\rangle$ or $|1\rangle$ with 50% probability. Therefore, it is easy to output the result of the measurement. The difficulty is in updating the stabilizer representation.

Once again, the stabilizers that start with I or Z remain in the stabilizer group because they still stabilize the state after projection. On the other hand, we do need to remove the stabilizers which start with X or Y because they anti-commute with the projection. There is a fairly nice way of doing this: take one of the stabilizer generators that starts with an X or Y and multiply it into all of the remaining stabilizer generators that start with an X or Y. One can check that now all the stabilizer generators start with either an I or Z except for one. Now, remove the remaining stabilizer generator that starts with an X or Y. What's left are $n-1$ stabilizer generators, so we only need to add one more. If the measurement outcome was $|0\rangle$, add the stabilizer generator $Z \otimes I \otimes \cdots \otimes I$ and if it was $|1\rangle$ add the stabilizer generator $-Z \otimes I \otimes \cdots \otimes I$. This completes the measurement protocol. It takes time $O(n^2)$.

References

[DM03] Jeroen Dehaene and Bart De Moor. Clifford group, stabilizer states, and linear and quadratic operations over GF(2). *Physical Review A*, 68(4), oct 2003.

[Got98] Daniel Gottesman. The Heisenberg representation of quantum computers, 1998.

[Shi02] Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 513–519. IEEE, 2002.

[VDN10] Maarten Van Den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation*, 10(3):258–271, 2010.