C S 358H: Intro to Quantum Information Science

Sayam Sethi

November 2024

Contents

1 Optimize Grover

Question 1.1

Question. *Suppose Grover's algorithm is used to search a list of size N containing a single marked item. Recall that, after t queries, the probability of having found the marked item is approximately $\sin^2\left(2t/\sqrt{N}\right)$. After approximately how many queries should you halt the algorithm if your goal is to maximize the success probability per unit of time invested? Give a formula in terms of N that works asymptotically for sufficiently large N, and explain how you derived it. Feel free to use any numerical software of your choice for approximating constants in your formula, but you must still explain your formula analytically (your final solution cannot be based only on generalizing a pattern from a few values of t).*

Proof. Since we want to maximize the success probability per unit of time invested, we wish to maximize the function $f(t)$ which is given by:

$$f(t) = \frac{\sin^2\left(2t/\sqrt{N}\right)}{t} \tag{1}$$

Now, if we want to find the maximum of this function, we can take the derivative of $f(t)$ and set it to zero. We have:

$$\begin{aligned}
f'(t) &= \frac{\frac{4t}{\sqrt{N}} \cdot \sin\left(2t/\sqrt{N}\right)\cos\left(2t/\sqrt{N}\right) - \sin^2\left(2t/\sqrt{N}\right)}{t^2} \\
&= \frac{\sin\left(2t/\sqrt{N}\right)}{\sqrt{N}t^2}\left(4t\cos\left(2t/\sqrt{N}\right) - \sqrt{N}\sin\left(2t/\sqrt{N}\right)\right) \\
\implies 0 &= 4t\cos\left(2t/\sqrt{N}\right) - \sqrt{N}\sin\left(2t/\sqrt{N}\right) \\
\implies 2 &= \frac{\tan\left(2t/\sqrt{N}\right)}{2t/\sqrt{N}} \\
\implies 1.165 &\approx \frac{2t}{\sqrt{N}}, \text{ on plotting } \frac{\tan(x)}{x} = 2 \\
\implies t &\approx 0.58\sqrt{N}
\end{aligned} \tag{2}$$

Therefore, from Equation 2, we gain maximum success probability per unit of time invested when we halt the algorithm after $0.58\sqrt{N}$ queries. □

2 The diffusion operator

Recall the $N \times N$ Grover diffusion matrix D, whose diagonal entries are all $\frac{2}{N} - 1$ and whose off-diagonal entries are all $\frac{2}{N}$.

Question 2.1

Question. *Prove that D is indeed a unitary matrix.*

Proof. We can write D as $2|v\rangle\langle v| - \mathbf{I}$, where $|v\rangle = \frac{\sum_{i=0}^{N-1}|i\rangle}{\sqrt{N}}$, as discussed in class. Now we find $D \cdot D^\dagger$:

$$\begin{aligned} D \cdot D^\dagger &= (2|v\rangle\langle v| - \mathbf{I})(2|v\rangle\langle v| - \mathbf{I})^\dagger \\ &= (2|v\rangle\langle v| - \mathbf{I})(2|v\rangle\langle v| - \mathbf{I}) \\ &= 4|v\rangle\langle v||v\rangle\langle v| - 2|v\rangle\langle v| - 2|v\rangle\langle v| + \mathbf{I} \\ &= 4|v\rangle\langle v||v\rangle\langle v| - 4|v\rangle\langle v| + \mathbf{I} \\ &= \mathbf{I} \end{aligned} \tag{3}$$

Hence, proved. □

Question 2.2

Question. *Prove that, up to scalar multiplication, D is actually the only real orthogonal matrix whose diagonal entries are all the same and whose off-diagonal entries are all the same, other than the identity matrix. This provides a big hint about how one might have discovered D if one didn't already know Grover's algorithm.*

Proof. Note that any matrix that has all diagonal entries equal and also off-diagonal entries equal, can be written as the form $M = a|v\rangle\langle v| + b\mathbf{I}$, where $|v\rangle$ is the uniform superposition state as stated in Question 2.1. Now, since we know that the matrix has real entries, we have a, b are real. Additionally, since the matrix has to be orthognal, we have:

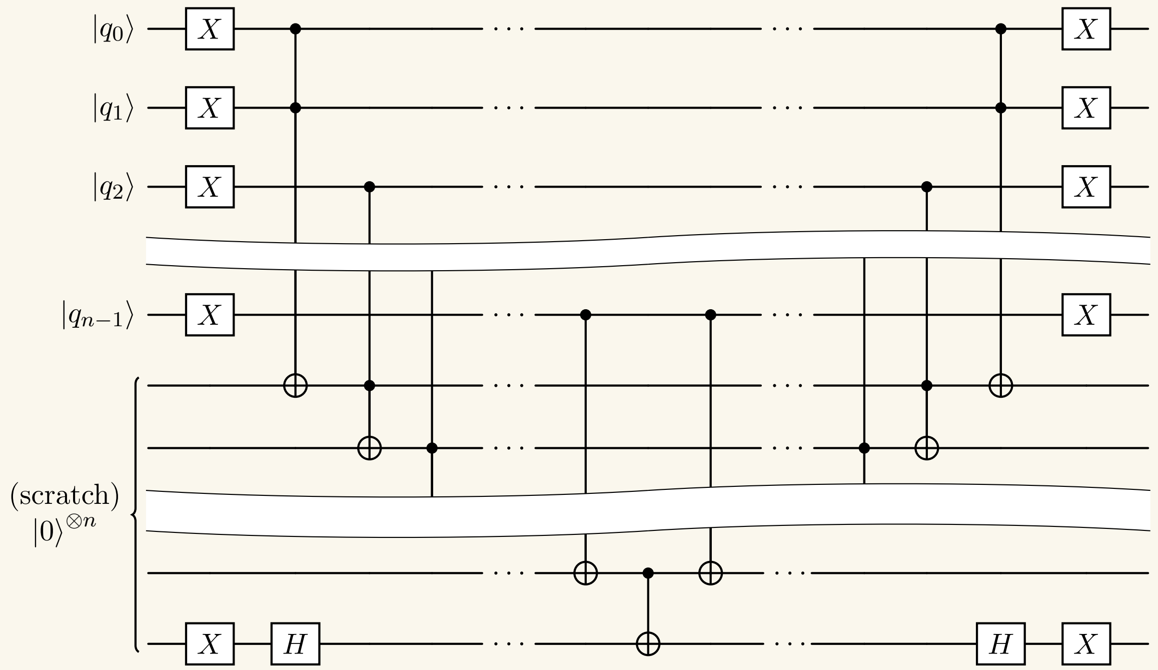
$$\begin{aligned} & M \cdot M^\dagger = (a|v\rangle\langle v| + b\mathbf{I})(a|v\rangle\langle v| + b\mathbf{I}) \\ \implies & \mathbf{I} = a^2|v\rangle\langle v||v\rangle\langle v| + ab|v\rangle\langle v| + ab|v\rangle\langle v| + b^2\mathbf{I} \\ \implies & \mathbf{I} = (a^2 + 2ab)|v\rangle\langle v| + b^2\mathbf{I} \\ \implies & a^2 + 2ab = 0,\ b^2 = 1, \text{(equating the diagonal and off-diagonal elements)} \\ \implies & a = \mp 2, b = \pm 1 \text{ or } a = 0, b = \pm 1 \end{aligned} \tag{4}$$

From Equation 4, we have $M = \mathbf{I}$ or M is equivalent to D up to a scalar multiple of -1. Hence, proved. □

Question 2.3

Question. *Recall $D = H^{\otimes n} A H^{\otimes n}$, where A is the diagonal matrix* $\operatorname{diag}(1, -1, -1, \ldots, -1)$. *Show an actual circuit made of Toffoli gates, Hadamard gates, X gates, and Phase gates for applying A. Your circuit should have $O(n) = O(\log N)$ gates and is allowed to use ancilla* *qubits initialized to any state you like provided that you return them to that state afterwards.*

Solution. We can consider the global phase equivalent version of A as $A \equiv \operatorname{diag}(-1, 1, 1, \ldots, 1)$. Therefore, we need to apply a phase of -1 iff the input state is the $|0\rangle^{\otimes n}$. We can achieve this using the following circuit:



The circuit above computes the AND of all the qubits after applying an X gate. Therefore, the only state that will lead to a $|1\rangle$ state on the penultimate scratch qubit is $|0\rangle^{\otimes n}$. Now, when we apply a CNOT onto the final scratch qubit initialised to the $|-\rangle$ state, we get a phase of -1 on the input state via the phase-kickback trick. Therefore, the circuit above applies the unitary A upto a global phase of -1 using a total of $O(n) = O(\log N)$ gates. Also note that we uncompute the scratch ancillas back to their beginning state in the above circuit. □

3 Grover's Algorithm with Multiple Marked Items

Question 3.1

Question. *Given a list of size N, which is promised to contain K marked items, suppose we want to find any one of the marked items. In class, we saw when we apply Grover's algorithm unmodified to the N-element list, we have a constant probability of observing a marked item if we halt the algorithm and measure after only $O(\sqrt{N/K})$ queries.*
Suppose that we cannot / don't want to apply Grover's algorithm to the entire N-element list, and instead apply Grover's algorithm to lists of size N/K. Show how we can still accomplish our original goal of finding any one of K marked items in a list of size N using $O(\sqrt{N/K})$ queries. Explain why this algorithm succeeds with constant probability (an informal, but still complete, explanation is okay).
Be precise when describing your algorithm.

Solution. We can use the following algorithm to find any one of the K marked items by only performing search on a list of size N/K:

Grover Search on Sublist

1. Sample a random sublist of size N/K from the list of size N.
2. Apply Grover's algorithm to the sublist to find a marked item (assuming a single marked item). This requires $O(\sqrt{N/K})$ queries.
3. Verify if the item found is a marked item using $O(1)$ queries. If it is, return the found item. Otherwise, restart with probability p_{restart}.

Figure 1: Modified Grover's Algorithm to find marked item by only search lists of small size

Note that in the above algorithm, p_{restart} is a constant probability of restart that can be chosen depending on how high we want the probability of success to be.
To prove that Algorithm 1 works, we can assume wlog that each element has a probability K/N of being marked. Therefore, if we sample N/K elements at random, the expected number of marked items will be:

$$\begin{aligned}\mathbb{E}[\text{marked items in sublist}] &= \sum_{i=1}^{N/K} \mathbb{E}[\text{marked item in } i\text{th element}], \text{ using linearity of expectation}\\ &= \frac{N}{K} \cdot \mathbb{E}[\text{marked item in any element}]\\ &= \frac{N}{K} \cdot \frac{K}{N}\\ &= 1\end{aligned} \tag{5}$$

Therefore, we can run Grover's algorithm to find a single marked item from the sampled sublist. Additionally, since Equation 5 is true in expectation, we will need to run the algorithm $O(1)$ times to sample a sublist that has exactly 1 marked item. We achieve this by running the algorithm with a constant probability of restart $p_{restart}$, which will ensure that the algorithm succeeds with very high probability. □

Question 3.2

Question. *Assume Grover's algorithm is optimal for the single marked item case, as proved in class. Prove that it's optimal for the multiple marked item case as well. In other words, let N and K be given. Show that any quantum algorithm that finds a marked item with constant probability, given an N-element unordered list that contains K marked items,* **must** *use $\Omega(\sqrt{N/K})$ queries to the list.*
Hint: *given a hypothetical quantum algorithm that was faster, can you derive a contradiction in the single-marked-item case?*

Proof. Let us assume that there exists an algorithm that finds K marked items in $o(\sqrt{N/K})$ queries. Let that algorithm be $\mathcal{A}$. We will now use this algorithm to construct an algorithm, $\mathcal{B}$, that can find a single marked item in $o(\sqrt{N})$ queries:

Single Marked Item Algorithm $\mathcal{B}$

1. Construct a list of size $N \times K$ by duplicating the input list K times.
2. Shuffle the list randomly.
3. Run $\mathcal{A}$ on the shuffled list and return the output that $\mathcal{A}$ gives.

Figure 2: Algorithm $\mathcal{B}$ to find a single marked item in $o(\sqrt{N})$ queries

Now, since $\mathcal{A}$ runs on a list of size $N \cdot K$ and we have K marked items, it will perform $o(\sqrt{N \cdot K/K}) = o(\sqrt{N})$ queries to the oracle. Additionally, since we shuffle the list, there is no way for $\mathcal{A}$ to behave differently in the case when we input this special list, therefore guaranteeing the query complexity. This is a contradiction to the assumption that Grover's algorithm is optimal for the single marked item case since we cannot have any algorithm that can find the marked item in $o(\sqrt{N})$ queries. Therefore, any algorithm that can find a marked item in a list of size N with K marked items will require $\Omega(\sqrt{N/K})$ queries.
Hence, proved. □

Question 3.3

Question. *Now suppose you want to find not just any one of the K marked items, but you want to find all of them. Show that Grover's algorithm can be used to do that as well with constant success probability using $O(\sqrt{NK}\log(N))$ queries.*

Proof. We propose the following algorithm to find all K marked items using Grover's algorithm:

Find all Marked Items

Repeat K times:

1. Apply Grover's algorithm to find any one of $K-i+1$ marked items. Modify the Grover oracle to "unmark" the i^{th} found items in iteration i.
2. Update the list with the found item, if found. Else repeat step 1.

Figure 3: Algorithm to find all marked items using $O(\sqrt{NK}\log(N))$ queries

Now, to modify the oracle, we can simply use the technique proposed in Question 2.3 to *unmark* the found items. However, note that after we find each item, we will need to perform Grover's with $K-i+1$ items and hence the total number of queries will be slightly different. This can be computed as:

$$\begin{aligned}
\sum_{i=1}^{K}\sqrt{\frac{N}{K-i+1}} &= \sqrt{N}\sum_{i=1}^{K}\frac{1}{\sqrt{K-i+1}} \\
&= \sqrt{N}\sum_{i=1}^{K}\frac{1}{\sqrt{i}} \\
&= \sqrt{N}\left(\frac{1}{\sqrt{1}}+\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{3}}\right)+\left(\frac{1}{\sqrt{4}}+\cdots+\frac{1}{\sqrt{7}}\right)+\right. \\
&\quad \left.\cdots+\left(\frac{1}{\sqrt{2^{\lfloor\log K\rfloor}}}+\cdots\frac{1}{\sqrt{K}}\right)\right) \\
&\leq \sqrt{N}\left(\sum_{i=0}^{\lfloor\log K\rfloor}\frac{2^i}{\sqrt{2^i}}\right) = \sqrt{N}\sum_{i=0}^{\lfloor\log K\rfloor}2^{i/2} \\
&\leq \sqrt{N}\sum_{i=0}^{\lfloor\log K\rfloor}2^{\lfloor\log K\rfloor/2} = \sqrt{N}\cdot\sqrt{K}\log K \\
&\leq \sqrt{NK}\log N = O\left(\sqrt{NK}\log N\right)
\end{aligned} \tag{6}$$

Therefore, from Equation 6 we have that the number of queries required to find all K marked items is $O(\sqrt{NK}\log N)$.
Hence, proved. □

4 The Graph Connectivity Problem

In the Graph Connectivity problem, we're given an n-vertex undirected graph G in adjacency matrix format. In other words, we have an oracle that given any basis state of the form $|i, j, a\rangle$, where $i, j \in \{1, ..., n\}$ and $a \in \{0, 1\}$, maps the basis state to $|i, j, \mathrm{NOT}(a)\rangle$ if G contains an edge between vertices i and j or to $|i, j, a\rangle$ otherwise. The problem is to decide whether G is connected — in other words, whether every vertex is reachable from every other.

Question 4.1

Question. *Prove that any possible classical algorithm for this problem – even a randomized algorithm that succeeds with high probability – must make $\Omega(n^2)$ queries to the adjacency matrix. (An informal proof is okay here — we won't grade as strictly as in HW9.)*
Hint: Find an example of a family of graphs such that, given a graph G from that family, a brute-force search among $\Omega(n^2)$ potential edges is the only way to decide whether G is connected.

Proof. We can show the lower bound on the number of queries required by any classical algorithm by considering the following adversary that adaptively constructs the graph based on the queries made by the algorithm.

Adversary for Query Lower Bound

Maintain a list of connected components, $\mathcal{L}$ and the current adjacency matrix, $\mathcal{A}$.

1. Initialize the list of connected components with each vertex in its own connected component.
2. For each query by the classical algorithm, which will be of the form $e = (v_1, v_2)$, check if e is a bridge between the connected components containing v_1 and v_2. That is, all other pairs of edges e' between the components (except e) have already been queried and have $\mathcal{A}(e') = 0$. If there are pairs of vertices between these components that are yet to be queried, then e is not considered a bridge since we can have any one of those unqueried edges make the graph a connected graph.
3. If e is a bridge as computed in step 2, then return $\mathcal{A}(e) = 1$ to the classical algorithm. Else, return $\mathcal{A}(e) = 0$.
4. Update the adjacency matrix, $\mathcal{A}$ and the list of connected components, $\mathcal{L}$.

Figure 4: Description of an Adaptive Adversary that requires $\Omega(n^2)$ queries

Note that the adversary described in Figure 4 will adaptively construct a tree on the n input vertices and this requires the classical algorithm to make $\Omega(n^2)$ queries. To prove this, notice that if we have two connected components of sizes s_1, s_2, there are $s_1 \cdot s_2$ possible edges between these components. Therefore, any algorithm that needs to check whether the graph is connected or not will need to perform exactly these many queries since the adaptive adversary will only return true for the final query. Additionally, there is no way for the algorithm to be able to determine that the graph is connected before it has made all $s_1 \cdot s_2$ queries. This is irrespective of whether the algorithm makes all such queries consecutively or interleaves queries across different connected components.

The number of connected components only reduce by 1 in each such sequence and therefore, we require $n-1$ such sequence of queries. Therefore, irrespective of the order of merging these connected components, we will require a total of $\Omega(n^2)$ queries. Hence, proved. □

Question 4.2

Question. *Give a quantum algorithm that solves this problem with high probability and that makes only $O(n^{3/2}\log(n))$ queries. Prove it.*
Hint: *You're welcome to use Grover's algorithm as an ingredient in your algorithm, as well as your favorite classical graph search algorithms like BFS/DFS/Dijkstra! Just make sure that the error probability stays bounded.*

Proof. Similar to the technique used by the adaptive adversary in Question 4.1, we will attempt to construct the graph by connecting components via Grover's algorithm. Before we describe the algorithm, we define a unitary, $U_{\mathcal{C}(\mathcal{T})}$, which is a function of the current spanning tree built in the algorithm, denoted by $\mathcal{T}$. $U_{\mathcal{C}(\mathcal{T})}$ will be used in conjuction with the unitary for the adjacency matrix which we call $U_{\mathcal{A}}$.

$$U_{\mathcal{C}(\mathcal{T})}\,|i,j,a\rangle = \begin{cases} |i,j,\bar{a}\rangle\,, & \text{if } \mathsf{components}(\mathcal{T}\cup\{(i,j)\}) < \mathsf{components}(\mathcal{T}) \\ |i,j,a\rangle\,, & \text{otherwise} \end{cases} \tag{7}$$

We describe the algorithm below:

Quantum algorithm for Graph Connectivity

The algorithm maintains the currently constructed spanning tree $\mathcal{T}_i$, after i iterations. The list of connected components is initialized as $\mathcal{L}$ with each vertex in its own connected component.
Repeat $n-1$ times:

1. Run Grover's algorithm to find an edge that satisfies $\mathcal{A}(i,j) = 1 \wedge \mathcal{C}(\mathcal{T}_i) = 1$.
2. If an edge is found, update $\mathcal{T}_{i+1} = \mathcal{T}_i \cup \{(i,j)\}$. Else, terminate the algorithm and output that the graph is not connected.

Return that the graph is connected.

Figure 5: Quantum algorithm that uses Grover's algorithm to determine if the graph is connected in $O(n^{3/2}\log n)$ queries

Note that to perform Grover's on the AND of the two functions, we can simply use three ancilla bits. The first two ancillas store the result of the two unitaries $U_{\mathcal{A}}$ and $U_{\mathcal{C}(\mathcal{T}_i)}$ and the third ancilla stores the result of the AND of the previous two ancillas via a Toffoli gate. Now we show that the algorithm defined in Figure 5 returns whether the graph described by adjacency matrix $\mathcal{A}$ is connected or not with high probability. After the i^{th} iteration, we know that we will have $n-i$ connected components. Therefore, if the graph is connected, in the $(i+1)^{\text{th}}$ iteration, we will have at least one edge $e \in \mathcal{A}$ that satisfies $C(\mathcal{T}_i) = 1$ and Grover's algorithm will find it. This will hold true for each of the $n-1$ iterations. Conversely, if the graph is disconnected, then Grover's algorithm will be unable to find any solution and the search will fail, returning that the graph is disconnected.

We now prove the query complexity. Even though the outer loop is run $O(n)$ times, however, note that the number of valid solutions in each iteration are different. In the case when the graph is disconnected, the algorithm will terminate before $n-1$ iterations. Therefore, the worst case query complexity can be obtained by considering the case when the graph is connected. In the i^{th} iteration, in the case of a connected graph, we have at least $n-i$ valid edges that connect each of the $n-i+1$ components with at least another component. Therefore, the query complexity for the i^{th} iteration will be $O\left(\sqrt{\frac{n^2}{n-i}}\right) = O\left(\frac{n}{\sqrt{n-i}}\right)$. This gives us a total query complexity of:

$$\begin{aligned}\sum_{i=1}^{n-1} O\left(\frac{n}{\sqrt{n-i}}\right) &= O\left(n\sum_{i=1}^{n-1}\frac{1}{\sqrt{n-i}}\right)\\ &= O\left(n\sum_{i=1}^{n-1}\frac{1}{\sqrt{i}}\right)\\ &= O(n)\cdot O\left(\sum_{i=1}^{n-1}\frac{1}{\sqrt{i}}\right)\\ &= O(n^{3/2}\log n),\text{ substituting } K=n, N=n \text{ in Equation 6}\end{aligned} \tag{8}$$

Therefore, from Equation 8, we have shown that the query complexity of the algorithm is $O(n^{3/2}\log n)$.
Hence, proved. □

Question 4.3

Question. *Show that any quantum algorithm for Graph Connectivity must make $\Omega(n)$ queries.*
Hint: Combine your answer from part a with the BBBV Theorem, which shows that Grover's algorithm is optimal, in the sense that $\Omega(\sqrt{M})$ quantum queries are needed to compute the OR of M independent bits

Proof. From BBBV we know that we have a lower bound on the query complexity for computing the OR of M independent bits as $\Omega(\sqrt{M})$. We will now show a reduction from the OR problem to the graph connectivity problem. Consider an OR problem on $n^2/4$ bits and a graph G on n vertices.
Define the vertex set as $\{v_{0i},\ v_{1i} \mid i \in \{0,1,\ldots,n/2-1\}\}$. Define the initial edge set as $E' = \{(v_{ji}, v_{j(i+1)}) \mid j \in \{0,1\}, i \in \{0,1,\ldots,n/2-2\}\}$, i.e., all the vertices of the form v_{0i} are connected with each other and the vertices of the form v_{1i} are connected with each other. However, vertices from v_{0i} are not connected with any vertex from v_{1i}.
Now for each input bit b_k in the OR problem, where $k \in \{0,1,\ldots,n^2/4-1\}$, insert an edge $(v_{0(2k/n)}, v_{1(k \bmod n/2)})$ in the graph (or equivalently in the adjacency matrix) iff $b_k = 1$. In other words, we insert an edge between the two subgraphs made by v_{0i}'s and v_{1i}'s iff any one of the input bits is 1. Define these edges as F.
We now show that this reduction exactly maps the OR problem to a graph connectivity problem. Note that the subgraphs made by v_{bi} for each $b \in \{0,1\}$ are individually connected

with each other. However, to connect the entire graph defined by V, we need to add at least one edge between the two subgraphs. Since all *cross-edges* are mapped from the input bits of the OR problem to a unique pair of vertices in the graph connectivity problem, we will have a 1 in the adjacency matrix for any one of the $n^2/4$ pairings iff atleast one of the input bits is 1. Therefore, if the input to the OR problem is a yes instance, we will also have a yes instance for $G = (V, E = E' \cup F)$. Conversely, if the input to the OR problem is a no instance, we will have that there is no edge connecting the two subgraphs ($F = \phi$) and therefore, we will have a no instance to the graph connectivity problem for $G = (V, E = E' \cup \phi)$ as well.
Now, since we have from BBBV that any quantum algorithm for the OR problem must make $\Omega(\sqrt{n^2/4}) = \Omega(n)$, we also have a lower bound for the quantum query complexity for this specialized graph connectivity problem on n vertices as $\Omega(n)$. This implies that we have a lower bound for an arbitrary graph connectivity problem on n vertices as $\Omega(n)$.
Hence, proved. □

5 The 4-Qubit Code

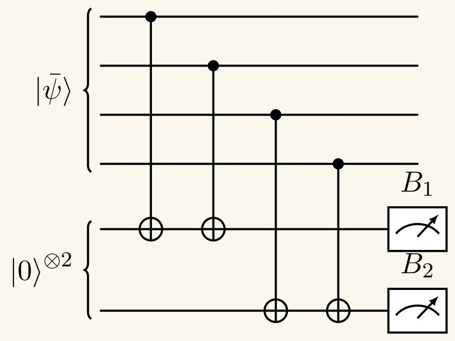
Question 5.1

Question. *Show that the 4-qubit code*

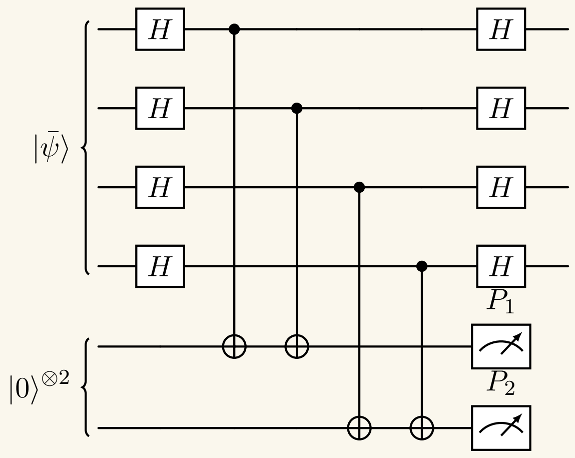
$$|0\rangle \to \frac{(|00\rangle + |11\rangle)^{\otimes 2}}{2},$$
$$|1\rangle \to \frac{(|00\rangle - |11\rangle)^{\otimes 2}}{2}$$

can detect *either a bit flip or a phase flip on any of the 4 qubits.*

Proof. We have the following circuit to detect bit flips:



For phase flips, we have the following circuit:



To see why both circuits work, notice that if either of the first two qubits has a bit flip (since the logical qubits are symmetric in the first two and the last two qubits, we can simply consider the case for the first two qubits), the state we end up in is $\pm(|01\rangle \pm |10\rangle)$. Now the circuit to detect the bit flips computes the parity of the first two qubits and stores it in the measurement result B_1. This result will be 1 iff there was a bit flip, else it will be 0. Therefore, we are able to detect a bit flip. Similarly, we can also detect a bit flip on either of the last two qubits (independently).

Now, for the case of detection of a phase flip, note that the state $|00\rangle + |11\rangle$ is an eigenstate of the $H^{\otimes 2}$ gate with an eigenvalue of $+1$ whereas $H^{\otimes 2}(|00\rangle - |11\rangle) = |01\rangle + |10\rangle$. Therefore, if we measure the parities of the first two qubits and the last two qubits as P_1 and P_2 respectively, and if $P_1 \neq P_2$, then this implies that we have a phase flip error. □

Question 5.2

Question. *Show that the 4-qubit code from part (a) cannot* correct *a bit flip.*

Proof. Notice that the state on the first two qubits that we obtain in the case of a bit flip error is the same irrespective of whether we reached it from an error on the first qubit or the second qubit. Therefore, there is no way for us to determine if the original state had an error on the first or the second qubit. If we apply a bit flip correction on the wrong qubit, we will instead introduce a new error instead of correcting the error. The same argument holds for a bit flip error on either of the last two qubits.
Therefore, we cannot correct a bit flip error using this code. □

Question 5.3

Question. *Show that the 4-qubit code from part (a) cannot correct a phase flip error.*

Proof. Notice that in the case of a phase flip on any of the four qubits, we cannot determine if the error was on the $|\bar{0}\rangle$ state or the $|\bar{1}\rangle$ state since a phase error on both states leads to the same state. Therefore, we will be unable to determine if the phase flip error originated on the first two qubits or the last two qubits.
Therefore, we cannot correct a phase flip error using this code. □

6 Error detecting code

In class, we saw the simplest classical error-correcting code, which encodes one bit into three bits via $\bar{0} = 000$ and $\bar{1} = 111$, and which corrects any single bit-flip error. We then saw the quantum generalization of that code, the Shor 9-bit code, which encodes 1 qubit into 9 qubits via:

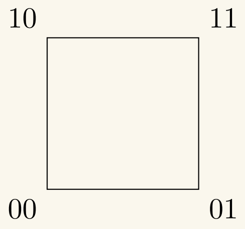
$$|\bar{0}\rangle = \frac{(|000\rangle + |111\rangle)^{\otimes 3}}{2\sqrt{2}}; \quad |\bar{1}\rangle = \frac{(|000\rangle - |111\rangle)^{\otimes 3}}{2\sqrt{2}}$$

and which corrects any single bit-flip or phase-flip error — and therefore, by linearity, any single-qubit error at all.

Question 6.1

Question. *Prove that any classical encoding of a single bit, which corrects an arbitrary single bit-flip error, requires at least 3 bits for the codewords, i.e. there is no such code using 2 bits.*

Proof. Consider the Hamming distance between the codewords that we can have in the classical error detection code constructed using 2 bits. The Hamming distance for any chosen pair of codewords to represent the logical 0 and the logical 1 will be at most 2. Additionally, each of the codewords is a point on the square. Therefore, we can consider the two cases for the Hamming distance between the codewords separately:



Distance is 1: In this case, there exists a bit flip error such that we transform one logical state into the other, which will go undetectable and hence uncorrectable.

Distance is 2: In this case, for an arbitrary bit flip error on either of the two bits from one logical state, there exists some way to reach the same erroneous state via a bit flip on a different bit from the other logical state. Therefore, we will be unable to determine the correct correction operation for the bit flip error, although we will be able to detect arbitrary bit flip errors in this case.

Thus, we cannot correct arbitrary bit flip errors using just 2 bits. □

Question 6.2

Question. *A weaker notion than an error-correcting code is an error-detecting code — one that detects whether an error has occurred, though doesn't necessarily help correct the error. Give a classical error-detecting code which encodes 1 bit into 2 bits and which detects a single bit-flip error.*

Solution. From the observation in Question 6.1, we can define our error detecting code as $\bar{0} = 00, \bar{1} = 11$. For error detection, we simply measure the parity of the two bits and if they differ in parity, we report that a bit flip error has occured, else there has been no error in our state. □

Question 6.3

Question. *Now give a quantum error-detecting code which encodes 1 qubit into 2 qubits and which detects a single phase-flip error.*

Solution. We can define our logical states as $|\bar{0}\rangle = |{+}{+}\rangle, |\bar{1}\rangle = |{-}{-}\rangle$. For error detection, we apply $H^{\otimes 2}$ to our qubit and compute the parity of the bits using an additional ancilla qubit (similar to the parity cmputation in Question 5.1) and then reset the state back by applying another set of $H^{\otimes 2}$. If we get a parity of 1, this implies that we have a phase flip error. If we have a parity of 0, this means that no error occured. This works because a phase flip on either qubit transforms it from the $|+\rangle$ state to the $|-\rangle$ state and vice versa. Therefore, if there is a phase flip error on either qubit, we will have different states on applying the Hadamard gate and measuring the parity. □

Question 6.4

Question. *Give a quantum error-detecting code which encodes 1 qubit into 4 qubits and which detects a single bit-flip error or a single phase-flip error (the code is allowed to fail if both errors occur simultaneously).*

Solution. We give the same code as proposed in Question 5.1 since it detects either a single bit flip or a single phase flip error on any of the input qubits, although it fails to correct any of the errors. We can also obtain the same code from the Shor code by reducing the number of qubits from 3×3 to 2×2 since we just want to detect errors and not correct them, motivated by our answer for Question 6.1 and Question 6.2. We rewrite the code here:

$$|\bar{0}\rangle = \frac{(|00\rangle + |11\rangle)^{\otimes 2}}{2}; |\bar{1}\rangle = \frac{(|00\rangle - |11\rangle)^{\otimes 2}}{2} \tag{9}$$

□

Question 6.5

Question. *Consider the quantum code $|\bar{0}\rangle = |00\rangle$ and $|\bar{1}\rangle = |11\rangle$. Give an example of a qubit encoded using this code and a single-qubit error on the encoded qubit such that this code fails to even detect the error.*

Solution. Consider the state $|\bar{+}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$ and a phase flip error on either qubit. We get the state $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$. This is a valid state and represents the logical $|\bar{-}\rangle$ state and therefore, the phase flip error will go undetected. If the error was detectable, we would not have a valid representation for the logical $|\bar{-}\rangle$ thus being an invalid quantum system, leading to a contradiction. □