CMSC 858L: Quantum Complexity

Instructor: Daniel Gottesman

Spring 2023

9 Oracles

9.1 References

The discussion of (classical) oracles and results relative to oracles are in most standard references on classical complexity theory. The Deutsch-Jozsa algorithm and a basic discussion of quantum oracles are both standard quantum textbook material, for instance in Nielsen and Chuang, *Quantum Computation and Quantum Information.*

9.2 Generalities about oracles

An oracle is complexity-speak for a black box that computes a function but you have no idea what is going on inside. Oracles have two primary applications in quantum complexity. One of them is the study of oracles in their own right, through the study of *query complexity*, which is the number of times you need to query the oracle to determine some property of the oracle. Query complexity is a nice thing to study because it is possible to actually prove bounds on query complexity in many cases, letting us determine the actual complexity of many problems. The other application of oracles is to study regular complexity by looking at relationships between regular complexity classes which have access to oracles. We'll discuss both applications of oracles to some degree, but we will start off with query complexity.

First, we need to note that there is a distinction between classical and quantum oracles.

Definition 1. *A* classical oracle $O(x)$ *takes a* query x *and gives an output, often 1 bit. There is a corresponding* quantum oracle O *which acts on a query state* $|\psi\rangle = \sum_x \alpha_x |x\rangle$ *and an ancilla qubit* $|c\rangle$ *and does a controlled-*O *operation:*

$$O|\psi\rangle|c\rangle = \sum_x \alpha_x |x\rangle |c + O(x)\rangle. \tag{1}$$

A classical oracle is a function from n bits to 1 bit (or more), whereas a quantum oracle is a unitary operation on $n+1$ qubits, but one that corresponds to the reversible version of the classical oracle applied to superpositions. Sometimes people also talk about more general versions of quantum oracles, *unitary oracles*, which are general black-box unitary operations not necessarily of the form above, so there is no corresponding classical oracle.

Definition 2. *We can think of a classical oracle and the corresponding quantum oracle as a long bit string, with the ith bit of the string being the value* $O(i)$*. Let* $f(O)$ *be some function of the oracle (thought of as a bit string). The* classical query complexity $D(f)$ *is the minimum number of classical oracle calls needed by a deterministic classical algorithm with access to the oracle* O *to compute* $f(O)$*. The* randomized query complexity $R(f)$ *is the minimum number of classical oracle calls needed by a randomized classical algorithm with access to the oracle* O *to compute* $f(O)$ *with probability at least* $2/3$*. The* quantum query complexity $Q(f)$ *is the minimum number of quantum oracle calls needed by a quantum algorithm with access to the oracle* O *to compute* $f(O)$ *with probability at least* $2/3$*. In some cases, we have a* promise *on the oracle, which means the oracles are drawn from a subset of possible bit strings.*

The query complexities $D(f)$, $R(f)$, and $Q(f)$ in some sense capture the difficulty of computing the function $f(O)$, but note that the query complexity makes *no* restriction on the number of qubits or gates used other than oracle calls. In many cases, there are algorithms known for which the non-oracle resources used are comparable or less than the number of queries needed, but not always. In particular, for the non-abelian hidden subgroup problem, the quantum query complexity is polynomial, but the only known algorithms still use exponentially many gates.

Nevertheless, in many cases, oracle algorithms can be translated into algorithms of similar efficiency when the black box is replaced by an explicitly computable function. Thus, study of query complexity can help give us find algorithms. Lower bounds on query complexity can also give us insight into potential non-black-box algorithms, in that they tell us that if there is an algorithm that does better than the query complexity, it must use some structure in the explicit function that is not present or not accessible in the oracle.

A study of quantum query complexity generally involves 4 pieces: First, an *upper bound on the classical deterministic and/or randomized query complexity*, generally in the form of a classical algorithm to determine the function. Whether we are primarily interested in the deterministic or randomized complexity depends on the problem. Second, a *lower bound on the classical query complexity* showing that this algorithm is the best achievable. Third, an *upper bound on the quantum query complexity*, in the form of a quantum algorithm to compute the function. We are usually (but not always) interested in problems with a quantum speedup, in which case we want the quantum *upper* bound to be lower than the classical *lower* bound. Finally, we also would like a *lower bound on the quantum query complexity*, to show that the quantum algorithm is the best achievable.

In many cases, one or more of these will be trivial. However, if some are unknown, then that means we don't fully understand the problem. For instance, if a classical lower bound is known but the upper bound is missing, that may mean that the classical lower bound should actually be higher; whereas if the classical upper bound is known, but the lower bound is missing, that means there could potentially be a better classical algorithm, which in turn means we can't be sure that the problem does offer a quantum speedup. Not infrequently, though, we can actually prove (with no complexity assumptions) that all four of these bounds have specific values and with the upper and lower bounds matching.

Note that $Q(f) \leq R(f) \leq D(f)$ because any deterministic algorithm can be performed as a "randomized" algorithm that doesn't make use of any random bits, and any randomized algorithm can be simulated on a quantum system using $H|0\rangle = |+\rangle$ states as random coins. If there are N possible query values (i.e., x is a $\log_2 N$ bit number), then $D(f) \leq N$ since we can learn the whole oracle by making N queries and then compute the function f from those values.

9.3 Deutsch-Jozsa Algorithm

The first example of quantum query complexity is always the Deutsch-Jozsa problem. In this problem, we are promised that the oracle $O : \mathbb{Z}_2^n \to \mathbb{Z}_2$ is either *constant* (i.e., $O(x) = O(y)$ for all x, y) or *balanced* ($|\{x \text{ s.t. } O(x) = 0\}| = |\{x \text{ s.t. } O(x) = 1\}|$). Our goal is to determine which with 100% probability.

First we analyze the classical upper and lower bounds. We are going to look first at $D(f)$, the deterministic classical query complexity. Certainly there is an algorithm that uses $2^{n-1} + 1$ queries: Perform queries for that many values of x, say $x = 0, 1, 2, \ldots, 2^{n-1}$. If they are all the same, return "constant;" otherwise return "balanced." If the oracle is balanced, at most 2^{n-1} entries can can have the same value, so the algorithm will return the correct answer, and if it is constant, all entries will indeed be the same and therefore the algorithm is again correct.

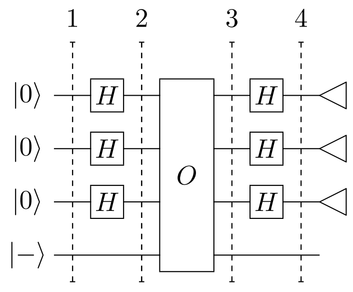
Claim 1. *To get the right answer, a deterministic classical algorithm needs at least* $2^{n-1} + 1$ *queries in the worst case.*

Proof. Suppose we have an algorithm that makes only 2^{n-1} queries. Let the output of the first query x_0 be a. Based on that, the algorithm decides to make a second query x_1, and suppose that the output of that query is also a. There are certainly oracles which have $O(x_0) = O(x_1) = a$, so this is a possibility. The algorithm continues deciding on queries x_i for $i < 2^{n-1}$, and suppose that all of the queries return a.

There exists a constant oracle O_1 for which $O(x) = a$ for all x that is consistent with all of these queries. There is also a balanced oracle O_2 consistent with all of these queries: $O(x_i) = a$ for $i = 0, \ldots, 2^{n-1} - 1$ and $O(y) = a \oplus 1$ for $y \neq x_i$ for any i. This algorithm must pick one of O_1 and O_2 and therefore will fail if presented with the other oracle. $\square$

These two arguments together show that $D(f) = 2^{n-1} + 1$ for the Deutsch-Jozsa problem.

Now let us look at the quantum complexity. First, the upper bound: There is a quantum algorithm to solve this problem with just 1 quantum query:



The last qubit starts in the state $|-\rangle = |0\rangle - |1\rangle$. After the measurement, we do classical post-processing to determine the output: If the measurement result is all 0, output "constant". Otherwise, output "balanced".

To see that this works, let us follow the state of the system at each step:

1. $|00\ldots0\rangle|-\rangle$
2. $(\sum_x |x\rangle)|-\rangle$
3. $\sum_x (-1)^{O(x)}|x\rangle|-\rangle$

At this point, what happens next depends on if it is constant or balanced.

- **Constant:** The state pre-Hadamards is $\pm \sum_x |x\rangle|0\rangle$, so the Hadamards give $\pm|00\ldots0\rangle|0\rangle$.
- **Balanced:** Let $|\psi\rangle = \sum_x |x\rangle$ and $|\phi\rangle = \sum_x (-1)^{O(x)}|x\rangle$. Then $\langle\psi|\phi\rangle = \sum_x (-1)^{O(x)} = 0$ since there are as many $+1$ terms as -1 terms. But $H^{\otimes n}$ is unitary and $H^{\otimes n}|\psi\rangle = |00\ldots0\rangle$. Thus, $H^{\otimes n}|\phi\rangle$ is orthogonal to $|00\ldots0\rangle$, which means the measurement outcome will always have at least one 1 in it.

Therefore, the quantum query complexity $Q(f) \leq 1$. But we know that $Q(f) \neq 0$, since it is not possible to determine if the oracle is constant or balanced without looking at it at all, so we find $Q(f) = 1$.

So this is pretty good: We have a nice separation between $Q(f) = 1$ and $D(f) = 2^{n-1} + 1$. Unfortunately, the separation is not so impressive if we compare to $R(f)$, the classical randomized query complexity: If we choose random queries, if the function is constant, the answer will always be the same, but if the function is balanced, the odds are good we will rapidly get a different answer. So here is a randomized algorithm: Make r queries to random values of x. If all answers are the same, return "constant." Otherwise, return "balanced." When the function is balanced, the probability that each additional query is different from the first one is $1/2$, and all are independent. (We can increase this slightly by making sure we don't repeat queries; then the probability drifts slightly above $1/2$ because we have more unused values that are different to the first query than ones that are the same.) Therefore, with r queries, the chance that we see a different response is $1 - 2^{-(r-1)}$. To get the success probability above $2/3$, we only need $r = 3$ queries. Thus, $R(f) \leq 3$. We can't do it with 1 query (since any response we could get is consistent with either type of function), and with 2 queries, there is still at least a probability $2^{n-1}/(2^n - 1)$ that the two responses are the same even with a balanced function, in which case we cannot distinguish between constant and balanced. Thus, $R(f) = 3$. This means that there is only. a constant separation between $Q(f)$ and $R(f)$.

9.4 Complexity classes relative to an oracle

Nevertheless, this result lets us say something meaningful about complexity classes. We do this by switching to the question of oracle separations, so let us first discuss what this means.

Definition 3. *A language O can be interpreted as an oracle, or really a famiy of oracles, which outputs $O(x) = 1$ if $x \in O$ and $O(x) = 0$ if $x \notin O$. We can define a classical or quantum circuit or Turing machine (deterministic, probabilistic, or non-deterministic) with oracle access to O as one which has the standard capabilities plus the extra capability to make oracle queries to O as a single computational step. A complexity class C* relative to an oracle *O is written C^O and indicates a class defined as usual but using circuits or machines with oracle access to O.*

Thus, P^O consists of problems solved by uniform polynomial size circuits that can query O and get an answer in 1 step, and BQP^O consists of problems solved with bounded error by uniform polynomial size quantum circuits that can query the quantum oracle O and get an answer in 1 step. Essentially we can think of classes relative to oracle O as living in some parallel universe in which there is some fundamental gate that solves O.

For instance, imagine that some extinct alien civilization left behind lots of black boxes with advanced technology that we don't understand, but we do know how to send information x into them and get a single bit output $O(x)$. What can we do with computers that incorporate these alien black boxes? If we are using deterministic classical computers, we get P^O; if quantum computers, we get BQP^O.

The oracle O can in principle be anything. If it is something weak, then working relative to O may not change anything. For instance, if $O \in \mathrm{P}$, then $\mathrm{P}^O = \mathrm{P}$. Why? Because $O \in \mathrm{P}$, there exists an algorithm A with running time $f(|x|)$, $f = O(\mathrm{poly}(|x|))$, that decides O. If we have a circuit of size $g(|x|)$ using O, we can substitute A for O and get the same behavior. The size of the circuit has increased, since the 1-step O is replaced by the $f(|x|)$ steps for A, but the total size of the modified circuit is at most $f(|x|)g(|x|)$, and if $g(|x|) = O(\mathrm{poly}(|x|))$, then so is this product. Therefore, any language decided in polynomial time by a circuit with oracle access to O can also be decided in polynomial time by a circuit with no access to O. This result is usually written as $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$ since it holds for any language in P.

Similarly, $\mathrm{BQP}^{\mathrm{BQP}} = \mathrm{BQP}$ using essentially the same argument. In the case of BQP, we have to be a bit careful because the modified algorithm can potentially fail both because the original oracle algorithm can fail (with probability at most $1/3$) but because each of the algorithms subsituting for O can fail as well. We can solve this problem by amplifying the success probability of all the subroutines through repetition to a sufficiently high value that the overall success probability of the modified algorithm remains at least $2/3$.

We can also have crazy, powerful oracles that can't exist in reality, like an oracle for the halting problem. That's OK; oracles are not necessarily supposed to represent the real world. They just provide us an avenue for studying the relationships between complexity classes. Many complexity-theoretic techniques *relativize*, meaning whatever relationship they prove between complexity classes still holds when both classes are taken relative to some oracle. As a very simple example, we know that $\mathrm{P} \subseteq \mathrm{BQP}$ because any classical circuit can be implemented as a quantum circuit. This same argument shows that $\mathrm{P}^O \subseteq \mathrm{BQP}^O$.

One advantage of thinking about classes relative to oracles is that not infrequently, we can find an oracle O such that we can *prove* a separation between complexity classes relative to O. In other cases, we can find an oracle O' such that we can prove that two complexity classes are equal relative to O'. In fact, it is fairly common that we can do both: prove that the classes are different relative to some O and that they are the same relative to O'. For instance, there exist oracles O and O' such that $\mathrm{P}^O \neq \mathrm{NP}^O$ and $\mathrm{P}^{O'} = \mathrm{NP}^{O'}$. This might make it seem this was all a futile exercise, but we have actually learned something important: If we want to prove a definitive relationship between these two classes, it will need to use a technique that does not relativize. It's just that . . . we don't know too many such techniques. But we do know some, and many of the most famous complexity theory results (including ones we'll discuss in this class) don't relativize.

With those caveats in mind, let's see how the Deutsch-Jozsa algorithm implies an oracle separation between P and BQP. Consider generating a language O (which will be our oracle) using the following procedure: For each value of n, choose a random bit $b(n)$. If $b(n) = 0$, then choose a random constant function $f(x)$ for $|x| = n$; if $b(n) = 1$, choose a random balanced function $f(x)$ for $|x| = n$. Then O includes those x with $|x| = n$ and $f(x) = 1$. Now consider the language B which consists of 1^n for any n such that $b(n) = 1$.

Claim 2. *$B \notin P^O$ but $B \in BQP^O$.*

Proof. Given instance 1^n of B, to decide it, a circuit (classical or quantum) needs to determine if O is constant or balanced on inputs of size n. A call to O does exactly what an oracle query does in the Deutsch-Jozsa problem. A quantum circuit can do it with one call to O and a linear number of standard quantum gates. A classical circuit requires at least $2^{n-1} + 1$ calls to O, and in particular, no polynomial size classical deterministic circuit can decide B. $\square$