Lecture 10

When designing quantum query algorithms, we'd like to do the following two types of operations

1. operations appearing in randomized query algorithms
2. intermediate measurements: make a measurement after some quantum operations (i.e., unitaries), later operations depend on the outcome of those measurements, and so on.

Let's see that they're already captured by our definition.

Fact 1. Quantum query algorithms can efficiently simulate randomized query algorithms. In particular $Q(f) \leq R(f)$ for any f. Reference: Section 2.3.3 of [de Wolf thesis].

Proof sketch. We will see how a quantum query algorithm can simulate a DDT first by way of an example: consider the obvious depth-2 DDT T that computes $(\neg x_1 \wedge x_2) \vee (x_1 \wedge x_3)$ with 1 labelling the root.

We will use the following

Lemma 2. *Suppose $g\colon [a] \to [b]$, then there exists a unitary U_g (in fact permutation matrix) acting on the space $\mathbb{C}^a \otimes \mathbb{C}^b = \mathbb{C}^{ab}$ ($U_g \in \mathbb{C}^{ab\times ab}$) such that*

$$U_g |i\rangle |1\rangle = |i\rangle |g(i)\rangle \tag{67}$$

for all $i \in [a]$.

Proof. Define U_g by

$$U_g |i\rangle |j\rangle = |i\rangle |(j + g(i) - 1) \mod b\rangle \tag{68}$$

Eq. (68) completely defines U_g, check that the definition implies U_g is unitary – in fact, a permutation matrix. □

Let $I\colon \{0,1\} \to \{2,3\}$ be defined by $I(0) = 2$ and $I(1) = 3$. I maps the bit value of x_1 to the index that is queried next. Let $h\colon \{0,1\} \times \{1,2,3\} \times \{0,1\} \to \{0,1\}$ be defined by

$$h(0,2,0) = 0,\ h(0,2,1) = 1,\ h(1,3,0) = 0,\ h(1,3,1) = 1. \tag{69}$$

We have defined h such that $h(a, I, b)$ is defined to be the value that T outputs if $x_1 = a$, I is the index of the variable queried next, and $x_I = b$.[7]

Register dimensions $\mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^3 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$:

$$\begin{aligned}
&\underbrace{|1\rangle\,|0\rangle}_{\text{query registers}}\ \underbrace{|1\rangle\,|0\rangle\,|0\rangle}_{\text{workspace registers}} \\
&\overset{O_x}{\mapsto} |1\rangle\,|x_1\rangle\,|1\rangle\,|0\rangle\,|0\rangle \\
&\overset{U_I}{\mapsto} |0\rangle\,|x_1\rangle\,|I(x_1)\rangle\,|0\rangle\,|0\rangle && \text{notation follows fact (*)} \\
&\overset{O_x}{\mapsto} |0\rangle\,|x_1\rangle\,|I(x_1)\rangle\,|x_{I(x_1)}\rangle\,|0\rangle \\
&\overset{U_h}{\mapsto} |0\rangle\,|x_1\rangle\,|I(x_1)\rangle\,|x_{I(x_1)}\rangle\,|h(x_1, I(x_1), x_{I(x_1)})\rangle && \text{notation follows fact (*)} \\
&= |0\rangle\,|x_1\rangle\,|I(x_1)\rangle\,|x_{I(x_1)}\rangle\,|T(x)\rangle && \text{definition of } h
\end{aligned}$$

where the $\overset{A}{\mapsto}$ notation means application of matrix A (suitably tensored with identity matrices), and the last line uses the definition of h. Then measuring using $\{\Pi_0 := \mathbb{1}_{36} \otimes |0\rangle\langle 0|, \Pi_1 := \mathbb{1}_{36} \otimes |1\rangle\langle 1|\}$ gives outcome $T(x)$ (with probability 1).

What about RDTs? Recall an RDT $\mathcal{T}$ is a distribution $(p_i, T_i)_{i=1}^K$ over DDTs. We have seen how T_i can be simulated by a quantum query algorithm $\mathcal{A}_i$ for each i. Suppose $\mathcal{A}_i$ is specified by unitaries $\{U^i_j\}_{j=0,\dots,d}$. Then the RDT can be simulated by a quantum query algorithm $\mathcal{A}$ that starts with the state

$$|\psi_0\rangle := |1\rangle \otimes \sum_{i=1}^K \sqrt{p_i}\, |i\rangle\,. \tag{70}$$

Then for $j \in \{0, 1, \dots, d\}$, U_j of $\mathcal{A}$ is defined to be

$$U_j := \sum_{i=1}^K U^i_j \otimes |i\rangle\langle i|. \tag{71}$$

□

[7] There was a question in class about why h wasn't defined on domain $\{0,1\} \times \{2,3\} \times \{0,1\}$ instead. My answer in class about unitaries needing to not depend on the input is *not* the correct answer. The correct answer was Rain's answer during class: to make the example more clearly generalizable. For *this particular* DDT, we could have alternatively used the domain $\{0,1\} \times \{2,3\} \times \{0,1\}$. But if a different DDT queries, e.g., variables 1 or 2 at the second step, then we would need to replace $\{2,3\}$ by $\{1,2\}$, etc.. We can avoid this change of domain when simulating each new DDT by using $\{1,2,3\}$.

The measurement of $\mathcal{A}$ is still $\{\Pi_0 \coloneqq |0\rangle\langle 0|, \Pi_1 \coloneqq |1\rangle\langle 1|\}$ (tensored with identities so that the Π_bs only act non-trivially on the single register that contains $\{T_i(x) \mid i \in \{1, \ldots, K\}\}$.

The state of the quantum query algorithm before measurement is of the form

$$\sum_{i=1}^{K} \sqrt{p_i}\, |\psi_i\rangle\, |T_i(x)\rangle\, |i\rangle\,, \tag{72}$$

where $|\psi_i\rangle$ represent some "junk" state (in the DDT example above where $K = 1$, it's $|0\rangle\, |x_1\rangle\, |I(x_1)\rangle\, |x_{I(x)}\rangle$).)

Performing the measurement gives 0 with probability

$$\left\| \sum_{i=1}^{K} \sqrt{p_i}\, |\psi_i\rangle\, (|0\rangle\langle 0| \cdot |T_i(x)\rangle)\, |i\rangle \right\|^2 = \sum_{i=1}^{K} p_i\, \mathbb{1}[T_i(x) = 0] = \Pr[\mathcal{T}(x) = 0], \tag{73}$$

which shows that the quantum query algorithm simulates the output of the RDT $\mathcal{T}$ (Supposing the codomain of f is $\Gamma = \{0, 1\}$.). The first equality is an exercise.