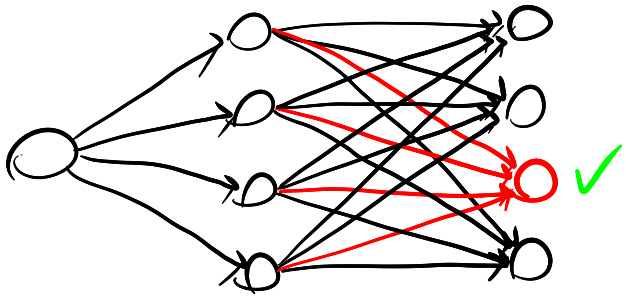
CMPT 476 Lecture 16

Quantum algorithms!!!



Last class we saw that we can (with enough ancillas and the Toffoli gate which can be constructed from a universal gate set) implement any classical function $f:\{0,1\}^n \longrightarrow \{0,1\}^m$ reversibly as

Number of Toffoli gates is polynomial

$$U_f:|x\rangle|y\rangle \longmapsto |x\rangle|y \oplus f(x)\rangle$$

You can apply U on all the superposition of the n bits.
By doing this, you can apply the function on all the possible inputs.

What happens if instead of $|x\rangle$, we give U_f a superposition of every classical n-bit state

$$\frac{1}{\sqrt{2}^n}\sum_{x\in\{0,1\}^n}|x\rangle$$

By linearity,

$$U_f\left(\frac{1}{\sqrt{2}^n}\sum_x |x\rangle|0\rangle\right) = \frac{1}{\sqrt{2}^n}\sum_x |x\rangle|f(x)\rangle$$

So in effect we compute f 2^n times! This is often called quantum parallelism. Unfortunately, if we measure the state we only get one value of $f(x)$, so parallelism alone is not enough to do something unclassical. Today we begin our study of quantum algorithms, which use parallelism and interference together to outperform classical algorithms.

Quantum parallelism can do the computation on all the possible inputs.

Potential algorithm for SAT

Let phi be a propositional formula in n variables.
We will need to evaluate all possible solutions with 2^n configurations of the variables.

We know we can evaluate (phi x) in poly(|phi|) time.

1. prepare all possible input states
2. Apply a unitary to all the states above
3. We have the evaluations of phi over all the bit strings. The problem is that if we measure, we only obtain one possible state

Text

Quantum algorithm is not doing computation on all inputs.
Quantum algorithms use superposition and interference to learn a global property

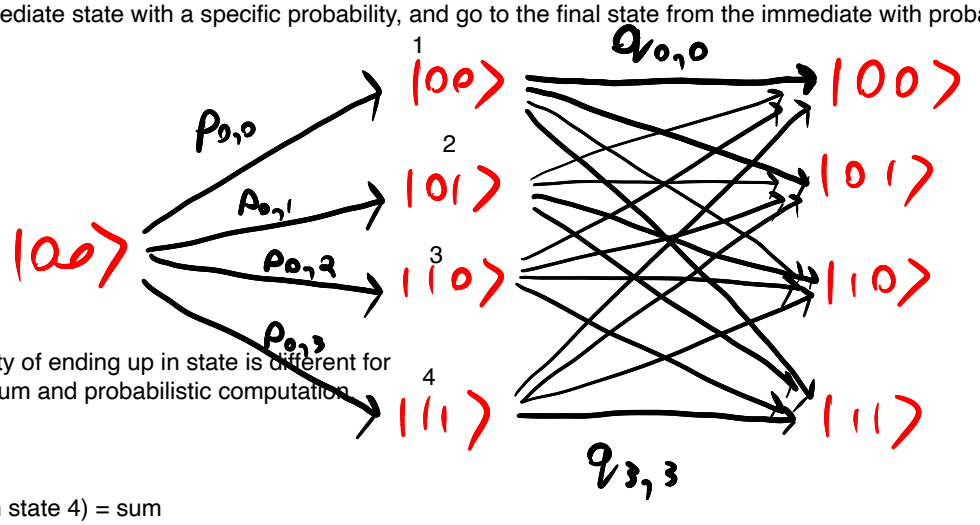
What are global properties?
Are there local properties?

(Probabilistic vs. quantum algorithms)

As quantum algorithms are closely related to probabilistic algorithms, we'll start our discussion by seeing how they differ from classical probabilistic algorithms through the use of interference.

Consider a classical probabilistic algorithm which starts in the $|00\rangle$ state, then transitions to one of the 4 states $|00\rangle, |10\rangle, |01\rangle, |11\rangle$ with some particular probability, and does this a few times. We can describe the algorithm as the probabilistic transition system below:

We transition to an immediate state with a specific probability, and go to the final state from the immediate with probability



Probability of ending up in state is different for quantum and probabilistic computation

Pr(end up in state 4) = sum

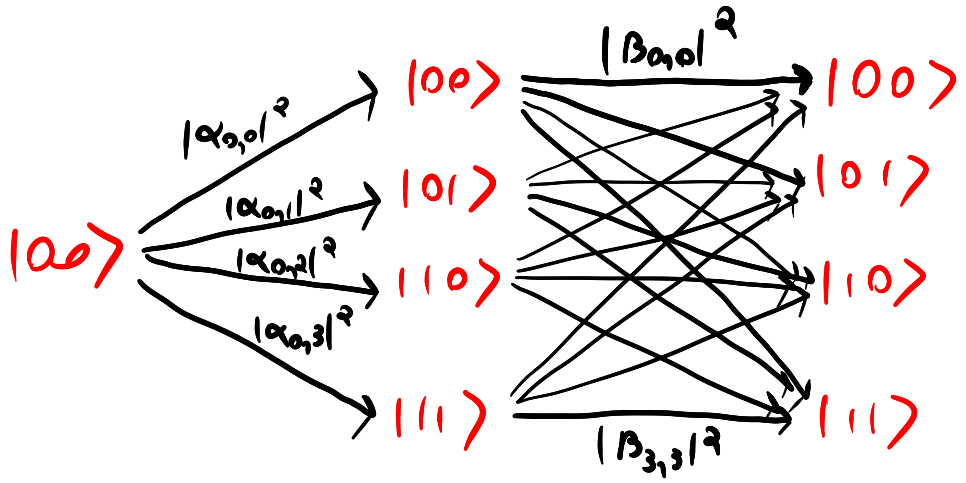
Remember, quantum probabilities can be negative.

The probability we end up in state $|00\rangle$ for instance is the sum of the probabilities of each path to $|00\rangle$. In particular,

$$pr(00) = \sum_j p_{0,j} q_{j,0}$$

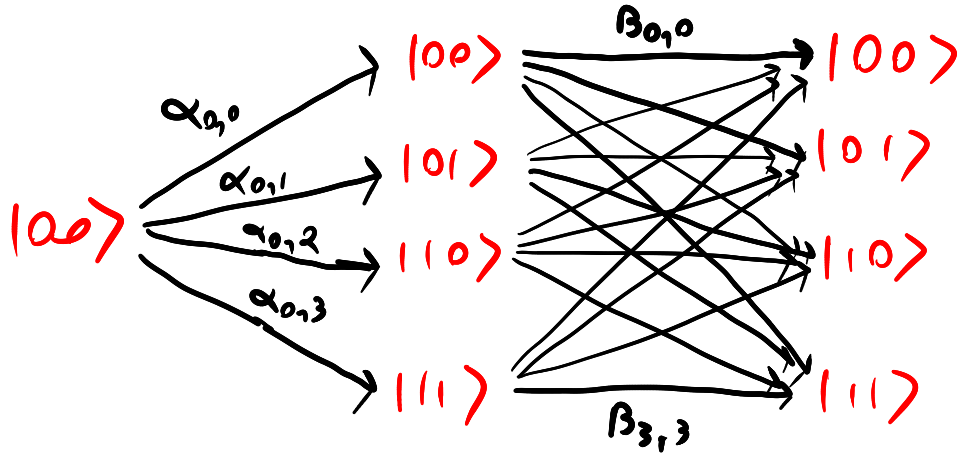
Note that $p_{i,j} q_{j,k} \geq 0$, hence each path adds to the probability of getting a particular result.

If we instead had a quantum process with probability amplitudes $\alpha_{i,j}$ & $\beta_{j,k}$ and we measured at each step, we would have the following transition probabilities



and so $pr(00) = \Sigma_j |\alpha_{0,j}|^2 |\beta_{j,0}|^2$ where $|\alpha_{0,j}|^2 |\beta_{j,0}|^2 \geq 0$

If instead we only measure at the end, then the probability amplitudes add instead, and since the can be negative, some of the paths leading to the same state may cancel out or interfere.

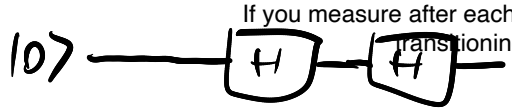


Here, $pr(00) = |\Sigma_j \alpha_{0,j} \beta_{j,0}|^2$ where $\alpha_{0,j} \beta_{j,0}$ can be negative.

(A concrete example of interference)

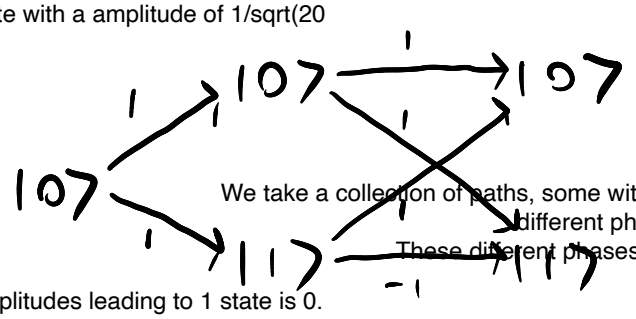
Consider the circuit

If you measure after each time slice, it is probabilitstic
If you measure after each time slice, then all the probabilities of transitioning become non-negative



We can draw the (quantum) transitions as

transition to the 0 state with a amplitude of 1/sqrt(20



Note: each transition is scaled by $1/\sqrt{2}$

We take a collection of paths, some with the same phase and some with different phases.
These different phases may cancel out.

sum of amplitudes leading to 1 state is 0.
p(0) =

If we were to measure at the end, we get

Scaled by a factor of 1/2

$$pr(0) = \left|\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}}\right|^2 = 1$$

$$pr(1) = \left|\frac{1}{\sqrt{2}}\cdot\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}\cdot\frac{-1}{\sqrt{2}}\right|^2 = 0$$

That is, the two paths $|0\rangle \to |0\rangle \to |1\rangle$ and $|0\rangle \to |1\rangle \to |1\rangle$ have opposite phase and hence cancel out!

Global properties: a property pertaining to all of the inputs.

This process of writing the amplitude of a transition as the sum over all paths is known as the sum-over-paths or sum-over-histories technique and is closely related to Feynman's path integral formulation of quantum mechanics. Much of my own research relates to formalizing the sum-over-paths technique to do symbolic manipulations of quantum algorithms. Ask me about it if you're interested!

(Ingredients for quantum algorithms)

To outperform a classical algorithm, 3 ingredients are needed:

1. A large superposition
2. Interference
3. A highly entangled state at some point

Why do we need all 3? Well, if we drop any requirement, we could classically simulate the algorithm efficiently:

1. Only poly(n) states in superposition
   → only poly(n) amplitudes to keep track of
2. No interference
   → probabilistic simulation suffices
3. No entanglement
   → $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$
   poly(n) amplitudes to track

So without all 3, we can simulate the algorithm on a classical computer. With all 3 however, magical things can happen...

(Black-box model and query complexity)

In quantum algorithms (particularly early ones) we often consider problems in the **black-box model**.

Black-box problem

input: a **black-box** or **oracle** computing a function f

goal: determine some **property** of f by making as **few calls to f as possible**

Compute a property of f by using few queries

Example: polynomial interpolation

input: a function $f: \mathbb{R} \to \mathbb{R}$ promised to be equal to a degree d polynomial

Promise of f: f can be represented as a degree d polynomial in the reals

goal: find a polynomial $g(x) = a_0 + a_1 x + \cdots + a_d x^d$ such that $g(x) = f(x)$ $\forall x \in \mathbb{R}$

How many **queries** of f are needed?

$\longrightarrow d+1$

(Query complexity)

The query complexity of a problem is the minimum number of queries needed to solve it in the black-box model.

Query complexity allows us to show that quantum computing can produce an advantage.
It is not a good complexity model.

(Deutsch's problem)

We need to evaluate f(0) and f(1) to evaluate f(0) XOR f(1)

input: a function $f: \{0,1\} \longrightarrow \{0,1\}$

goal: determine the **parity** $f(0) \oplus f(1)$ (i.e. whether f is constant)

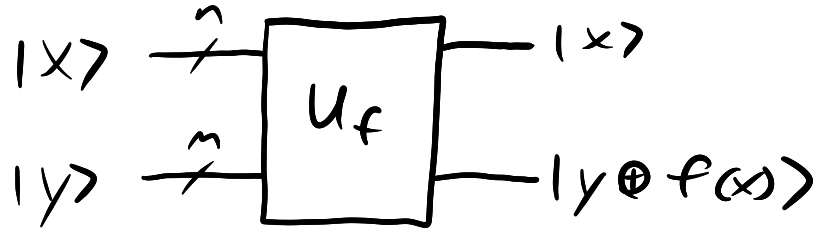
The classical query complexity of Deutsch's problem is 2:

- If we learn $f(0)$, then $f(0) \oplus f(1)$ can be either 0 or 1, so either guess can be wrong
- Ditto for $f(1)$.

In the mid 1980's, David Deutsch showed that this problem could be solved by a single quantum query. We first need to be clear about what a quantum query is however.

(Quantum black-box model)

Take a quantum oracle (i.e. black-box) for $f: \{0,1\}^n \to \{0,1\}^m$ as $U_f: |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$

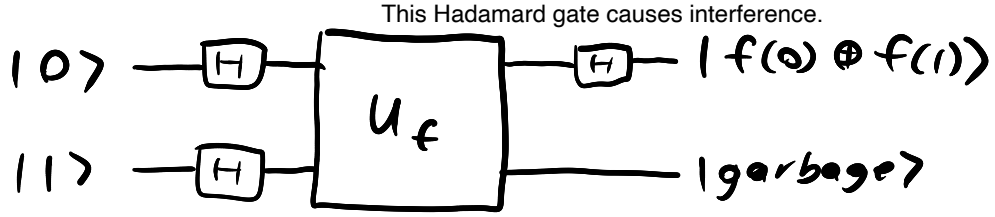


For below, normalization factor does not matter.
After (tensor-prod H H), we have |+>|-> = |0->+ |1->

(Deutsch's algorithm)

A quantum circuit computing $f(0) \oplus f(1)$ with one quantum query to f is

This Hadamard gate causes interference.



Why the algorithm is important:
It provides the first example of a quantum speedup in the query complexity model

Why does this work? Well, the set-up before we apply the query U_f is

$$(H \otimes H)|0\rangle|1\rangle = |+\rangle|-\rangle$$

In this case it will be convenient to view $|+\rangle$ as a superposition of all classical bits, so that

$$|+\rangle|-\rangle = \left(\frac{1}{\sqrt{2}} \sum_x |x\rangle\right) \otimes |-\rangle$$

We can assocaite

Now, what happens when we query f with the second qubit in the state $|-\rangle$?

$$U_f|x\rangle|-\rangle = U_f\left(\frac{1}{\sqrt{2}}|x\rangle|0\rangle - \frac{1}{\sqrt{2}}|x\rangle|1\rangle\right)$$

$$= \frac{1}{\sqrt{2}}|x\rangle|f(x)\rangle - \frac{1}{\sqrt{2}}|x\rangle|1\oplus f(x)\rangle$$

$$= \begin{cases} \frac{1}{\sqrt{2}}|x\rangle|0\rangle - \frac{1}{\sqrt{2}}|x\rangle|1\rangle = |x\rangle|-\rangle \text{ if } f(x)=0 \\ \frac{1}{\sqrt{2}}|x\rangle|1\rangle - \frac{1}{\sqrt{2}}|x\rangle|0\rangle = |x\rangle(-|-\rangle) \text{ if } f(x)=1 \end{cases}$$

We end up with the same state, but with differing phases.

$$= (-1)^{f(x)}|x\rangle|-\rangle$$

Phase kickback

This is known as **phase kickback**: since $|-\rangle$ is an **eigenvector** of the transformation

$$|y\rangle \longmapsto |y\oplus f(x)\rangle,$$

it picks up a phase of ± 1, which we **kick back** to the first qubit.

We want to evaluate f in the phase instead of the state

Instead of doing the computation of f(x), we can do it in the phase

Now, since we queried f on a superposition of $|x\rangle$'s, we get the phase $(-1)^{f(x)}$ for each x:

$$U_f\left(\frac{1}{\sqrt{2}}\Sigma_x|x\rangle\otimes|-\rangle\right) = \left(\frac{1}{\sqrt{2}}\Sigma_x(-1)^{f(x)}|x\rangle\right)\otimes|-\rangle$$

$$= \frac{1}{\sqrt{2}}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)\otimes|-\rangle$$

$$|\psi\rangle$$

This means we have the same phase.

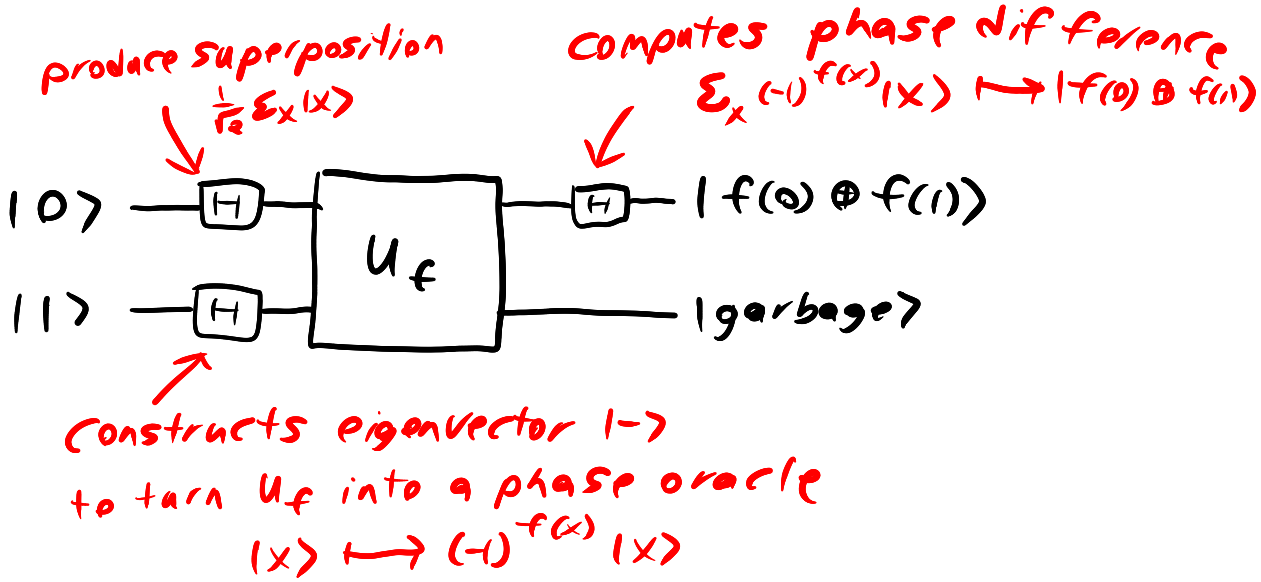
If $f(0)=f(1)$, then $|\psi\rangle = \pm|+\rangle$, and if $f(0)\neq f(1)$, the $|0\rangle$ & $|1\rangle$ states have a **relative phase difference** of (-1), so $|\psi\rangle = \pm|-\rangle$. Hence,

$$(H\otimes I)|\psi\rangle\otimes|-\rangle = \begin{cases} \pm|0\rangle|-\rangle \text{ if } f(0)=f(1) \\ \pm|1\rangle|-\rangle \text{ if } f(0)\neq f(1) \end{cases}$$

Just a global phase difference of -1

$$= \pm|f(0)\oplus f(1)\rangle|-\rangle$$

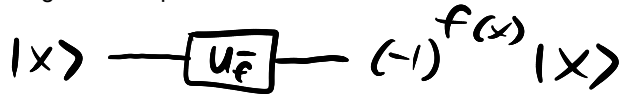
To summarize,



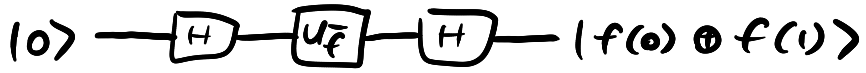
(Phase oracles)

In one sense, the second qubit is not really necessary: if in our black-box model, we could query f *in the phase*, i.e $U_{\bar{f}} : |x\rangle \mapsto (-1)^{f(x)}|x\rangle$

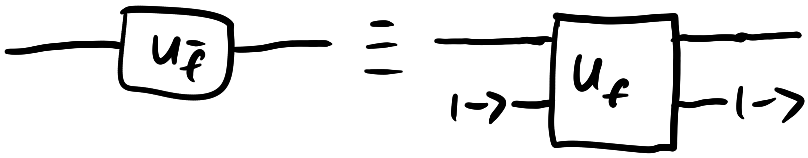
If we have f in the state, then we can get f in the phase



we could write Deutsch's algorithm as



The second qubit exists solely to implement the phase oracle $U_{\bar{f}}$ from the state oracle U_f via phase kickback. In general,



Note however that this only works for single-output functions $f: \{0,1\}^n \to \{0,1\}$.

After we do H, we get |+>, measuring, we get 0 or 1 with the same probability

Hadamard gate sends 0 to either 0 or 1

Amplitude of the zero state is

constant, then the probability for the 1 path cancels out

The amplitude for the zero path sums together

balanced, then the probability for the 0 path cancels out

the amplitude for the one path has a probability