Lecture 8 – Rollercoasters

(the "Bernstein-Vazirani algorithm")

Recapping last time: "Bias-Busting Algorithm"

[I call it this also to remind you of the more usual meaning of an admirable program to help you recognize & overcome your unconscious biases]

Input:

Classical code C computing some $F: \{0,1\}^n \to \{0,1\}$

[The input being classical code is very common in quantum algorithms....]

Problem: Is the truth table of F "biased"?
[unequal # of 0's, 1's]

Quantum alg:

$\left\{\begin{array}{l} \bullet \text{ Add\&Diff on } X_1, \ldots, X_n \quad \text{// prep. unif. superpos} \\ \bullet \text{ If } F(X_1, \ldots, X_n) \text{ Then Minus} \quad \text{// sign-compute F} \end{array}\right.$

"Load the ± truth table of F into amplitudes"

- Avg&Diff on $X_1, \ldots, X_n$
- If PrintAll $\neq 00\cdots0$, output "Busted!"

Recall: this gives a probabilistic alg. with "no false positives":

F balanced $\Rightarrow$ never say "Busted!"

F biased $\Rightarrow$ $\Pr[\text{"Busted!"}] > 0$ [nonzero probability]

ex: The nonzero probability might be as small as $\frac{4}{4^n}$.

[This is "unphysically small". Would have to repeat exponentially many times to notice. So this quantum alg. is practically useless. Still, we know (assuming P$\neq$NP variant) no classical probabilistic alg. can have the same behavior.]

[The main quantum algs showing some advantage over classical...]

	Alg.	Interesting Problem?	Speedup
	Bias-Busting	sorta	not really
today:	Roller-coaster-Bravery	no	modest (polynomial)
	SAT in $\sqrt{2}^n$ time (Grover)	yes!	modest
	Simon's Problem	no	exponential
	Factoring	yes!	exponential

<u>Last time</u>: Doing ~~Hadamard Transform~~ Add & Diff all to 000.....0

yields "uniform superposition" → ampl. 1 on all strings

<u>Today</u>: What does it do when starting from, say, 10110?

[[To explain this, let me first tell you about....]]

Rollercoasters:

To ride, you must arrive. [[Can't ride Phantom's Revenge if you don't even go to Kennywood.]]

To ride, you must be brave.

[[So that's who will try to board the coaster. But will it let us ride?]]

For the rollercoaster to run, must be an EVEN # of riders. Otherwise, it displays **X**. [[Else it's imbalanced & dangerous!]]

[[Let's play a game. There's...

Patrick, Quinn, Rhea, Sylvia, Tommy

...I secretly know who's brave, who isn't. You try to find out by picking diff. subsets to arrive each day...]]

eg:

	your arrivers	rollercoaster response
Day 1:	P, Q	**X**
Day 2:	Q, S, T	✓
		......

[[so: both brave or both cowards]]

[After playing a bit, probably you realize that for n people, n days are necessary & sufficient. Sufficiency: ask about each person individually.
Necessity: each response gives only 1 bit of info, need n bits of info.]

	Person 1	2	3	4	5
Available?	0	1	0	1	1
Brave?	1	0	1	1	0
X?	AND each column, XOR results.				

Define X.On.Rollercoaster $(A_1, \dots, A_n, B_1, \dots, B_n)$

[& = AND, ⊕ = XOR] $= (A_1 \& B_1) \oplus (A_2 \& B_2) \oplus \cdots \oplus (A_n \& B_n)$

Def: $XOR_{B_1 \cdots B_n}(A_1, \dots, A_n) =$ ↗

"B"rave people, or "B"itmask

e.g.: $XOR_{10110} : \{0,1\}^5 \to \{0,1\}$ is

$XOR_{10110}(A_1, A_2, A_3, A_4, A_5) = A_1 + A_3 + A_4 \mod 2$

more eg: $XOR_{01000}(A_1, \dots, A_5) = A_2$.

$XOR_{00000}(A_1, \dots, A_5) = 0$.

Fun fact: $XOR_{B \dots}(A \dots) = XOR_{A \dots}(B \dots)$

Question: Is [[a function like]] $XOR_{11010110}(A_1, \ldots, A_n)$ easy to compute classically?

Answer: Certainly.

[[So you can also easily compute it quantumly, (reversibly & trash-freely). But in fact, don't need to go thru whole rigamarole; XOR functions are really easy quantumly]]

"Add $XOR_{1101}(A_1, A_2, A_3, A_4)$ To Ans":

Add A_1 to Ans

Add A_2 to Ans

Add A_4 to Ans // finally adds 1 to Ans iff odd # of $A_1, A_2, A_4 = 1$

[[What about sign-computing? Again, we know a generic way to do it, but it's super-easy for XORs]]

"If $XOR_{1101}(A_1, A_2, A_3, A_4)$ Then Minus":

If A_1 Then Minus

If A_2 Then Minus

If A_4 Then Minus // Finally amplitude is minused iff odd # of $A_1, A_2, A_4 = 1$

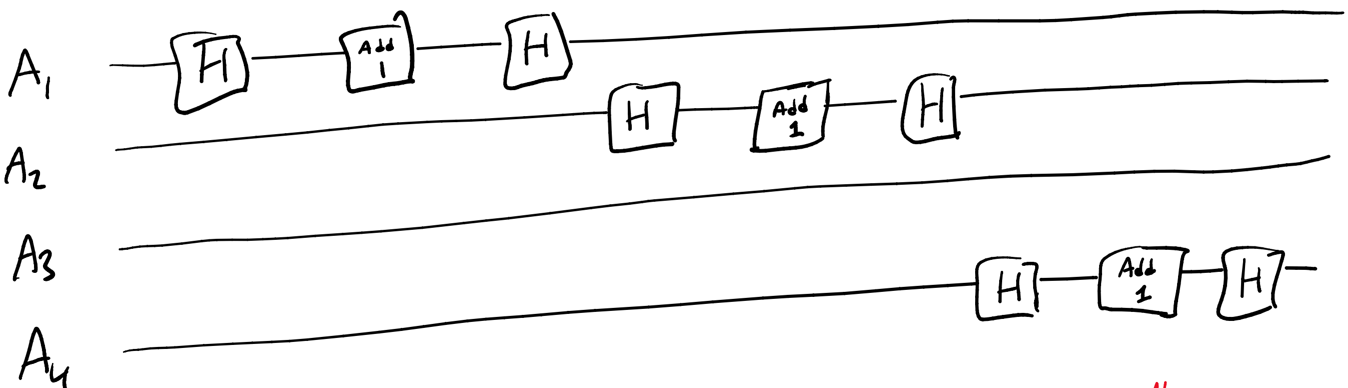
[By the way, recall how to do those...]

"If A Then Minus":

- Had on A
- Add 1 to A // NOT on A
- Had on A

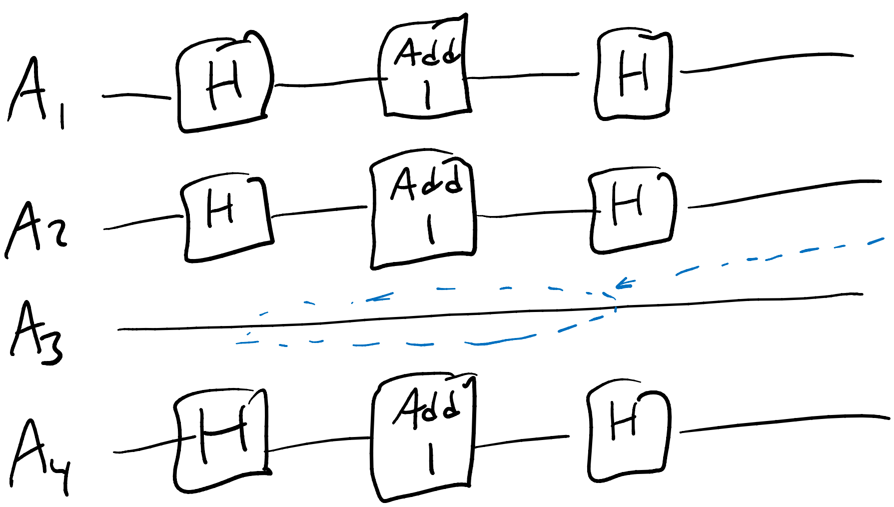
[in Lecture 5, if you had forgotten]

∴ quantum circuit diag. for "If $XOR_{1101}(A_1,\ldots,A_4)$ Then Minus":

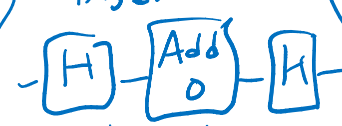


[Just like w/ prob. computing, OK to temporally rearrange ops on different qubits. Think about it.]

equivalently...



Could artificially insert



here!

Conclusion:

"If $XOR_{1101}(A_1,\dots,A_4)$ Then Minus" ⊛

$\equiv$

- Had on $A_1,\dots,A_4$
- Add 1101 to $A_1,\dots,A_4$
- Had on $A_1,\dots,A_4$

(⊛⊛)

$\therefore$

- Had on $A_1,\dots,A_4$
- ⊛

$\equiv$

- Had on $A_1,\dots,A_4$
- (⊛⊛)

[[Hads cancel]]

$\equiv$

- Add 1101 to $A_1,\dots,A_4$
- Had on $A_1,\dots,A_4$

Replace Had with Add & Diff & Start from 0000

- Make $A_1,\dots,A_4$
- Add & Diff on $A_1,\dots,A_4$
- If $XOR_{1101}(A_1,\dots,A_4)$ Then Minus

- Make $A_1,\dots,A_4$
- Add 1101 to $A_1,\dots,A_4$
- Add & Diff on $A_1,\dots,A_4$

[[Make unif superpos, then sign-compute XOR_{1101}...]]

[So we see that doing Add & Diff starting from 1101 is equivalent to "loading the ± truth table of XOR_{1101} into the state!]

THEOREM:

For any string $b \in \{0,1\}^n$, if you initialize vars $B_1, \ldots, B_n$ to b (ampl. 1 on this string), then do Add & Diff on $B_1, B_2, \ldots, B_n$, resulting (unnormalized) state is the ± truth table of XOR_b; that is,

$$\text{Ampl}[a] = \begin{cases} +1 & \text{if } XOR_b(a) = 0 \\ -1 & \text{if } XOR_b(a) = 1 \end{cases}.$$

eg: For $b = 000\cdots0$, we get ampl. +1 on all a.
(I.e., the uniform superposition, as we saw last time.]

Corollary: [By taking the inverse of the code:]

If you start with the ± truth table of XOR_b loaded into state & then do Avg & Disp on all bits, you end up with state "ampl. 1 on b".

[[Let's put this to use....]]

"Bernstein-Vazirani Problem" (circa 1991)

[[Like the Rollercoaster Brave-People-Ident. game played earlier, but imagine game master replaced by classical computer code you have access to.]]

Input: Classical code C computing a function $F: \{0,1\}^n \to \{0,1\}$. [[Imagine $n = 1000$.]]

Promise: $F = XOR_b$ for some string $b \in \{0,1\}^n$.

Note: C need not be computing XOR_b in an "obvious" way; could be highly obfuscated.

Say C is T instructions (say $T = 1$ million AND/OR/NOT gates)

[[So running C on any input $A_1, \ldots, A_n$ no big deal, but code is too inscrutable to deduce b.]]

Q: How can you be sure F is really XOR_b for some b?

A: You can't, really. Just have to accept promise.

Task: deduce b.

[[As we saw, if you really can't figure out the code, and you can just run C, it's like you're playing the Rollercoaster game & you must make n "queries".]]

Classically: need to run C n times:

$\approx n \cdot T$ steps.

Quantum solution: ① Produce quantum code

QC = "If $F(A_1, \ldots, A_n)$ Then Minus"

↳ $\approx 2T$ instructions [[Recall lecs 5,6; $T + \underbrace{T}_{\text{cleaning trash}}$]]

② Make $A_1, \ldots, A_n$, Had on each // make unif superpos
Run QC // now ± 1 truth table of $F = XOR_b$ is loaded
Avg & Disp on $A_1, \ldots, A_n$ // by Corollary, now ampl 1 on b.
Print All // prints solution $b \in \{0,1\}^n$ with 100% prob.!

$\approx 2n + 2T$ instructions.

Speedup! Classical $\approx n \cdot T$

Quantum $\approx n + T$

Quadratic improvement if $T \approx n$.

Alg.	Interesting Problem?	Speedup
✓ Bias-Busting	sorta	not really
✓ Rollercoaster-Bravery	no	modest (polynomial)
SAT in $\sqrt{2}^n$ time (Grover)	yes!	modest
Simon's Problem	no	exponential
Factoring	yes!	exponential

[Honestly, wouldn't be hard to blast thru Grover & Simon now. Factoring a bit more complicated.

But... we are going to slow down and do some more basic theory — linear algebra & geometry — to get things in a broader context!]