Lecture 9 — Grover's Algorithm: SAT in $\sqrt{2^n}$ time

[Show the 3B1B sliding blocks video.]

[Today we have one more cool quantum alg., & then we'll go back to basics and talk linear algebra & physics for a while.]

Lou Grover's Alg (ca. 1996):

Gives a "square-root speedup" for "mean estimation" tasks.

Particularly: [Today] Solves SAT in $\approx \sqrt{2^n}$ time.

Recall: $\boxed{\text{(Circuit-)SAT}}$ [The canonical NP-complete problem]

Given "code" C (say, AND/OR/NOT circuit, T gates)

Computing a function $\{0,1\}^n \to \{0,1\}$,

find input $x \in \{0,1\}^n$ s.t. output is 1.

[Or correctly assert no such x exists.]

example interesting C:

- $C(x_1,\dots,x_{1024})$:

  Let y = # whose binary rep. is $x_1 \cdots x_{512}$

  Let z = " " " " " $x_{513} \cdots x_{1024}$

  Let $w = y \cdot z$.

  If $y \neq 1$ & $z \neq 1$ & $w = \underbrace{13506[\cdots 300 \text{ digits} \cdots]7563}_{\text{"RSA-1024"}}$

  then return 1

  else return 0 // now a "satisfying" x is factors of RSA-1024!

- $C(x_1,\dots,x_{32})$:

  hash := SHA256(SHA256($\underbrace{92084\cdots\cdots0122}_{\text{some "block header"}} + x$))

  if hash $< 2^{192}$

  then return 1 else return 0 // basically, C is mining bitcoin

- $C(x_1,\dots,x_{100,000,000})$: // $\approx$ 10MB input

  if Lean proof-checker of "P≠NP because x" checks out, return 1 else return 0

  // satisfying x is proof of P ≠ NP of ≤ 10MB

- $C(x_1,\dots,x_{1,000,000,000})$:

  Use x as parameters in neural net, check if it gets classification error < .1%.

  If so, return 1. //auto-train any learning alg...

"Modest" case: $n = 1000$, $T =$ few million.

[[Unfortunately, no alg. other than brute force known!]]

Brute force time: $2^n \cdot T \approx 2^n$ [[frankly, the T is negligible]]

[[unphysically large]]

[[$n = 100$ is maybe the cutoff for physical conceivableness]]

Aliens: "Solve this SAT problem or we blow up Earth."

$n = 100 \to$ maaaybe humanity can do it

$n > 100 \to$ better attack aliens.

Grover: With quantum computers, can solve SAT in time $\approx T \cdot \sqrt{2^n} \approx \sqrt{2^n} = 2^{n/2}$.

$n = 100 \to$ 2^{50} steps is a few mins on PS5

$n = 200 \to$ Maaaybe doable

$n > 200 \to$ attack aliens.

Today: Just "Unique-SAT": you're promised exactly one input $x^* \in \{0,1\}^n$ makes C output 1: find it. [[Intuitively the hardest case anyway. Holds if C is for Factoring. On HW we'll see how to reduce general SAT to Unique-SAT.]]

<u>Notation</u>: $N = 2^n$. [[Brute force is $\approx N$, we want $\sqrt{N}$.]]

E.g.: $n=3$, $N=8$.

Given [[highly "obfuscated"]] AND/OR/NOT circuit C with T gates, computing some $F: \{0,1\}^3 \to \{0,1\}$. Secretly, $F =$ "Is 101?",

So $F(\underbrace{101}_{\text{"}x^*\text{"}}) = 1$, $F(x) = 0$ for $x \neq 101$.

[[Obviously could find $x^* = 101$ here with $\approx 8T$ steps by trying all $N=8$ inputs. But try to imagine $N \ggg T$.]]

[[Sooorta like the Bias-Busting prob: we're interested in the 0's and 1's of F's truth table. So let's start with that algorithm...]]

Make X, Y, Z. // State is now 000,

[[usually called A,B,C, but let's mix it up]]

aka "1 ampl. on 000
0 ampl. on 001
0 ampl. on 010
⋮
0 ampl. on 111."

$$\begin{array}{c|c}
000 & 1 \\
001 & 0 \\
010 & 0 \\
011 & 0 \\
100 & 0 \\
101 & 0 \\
110 & 0 \\
111 & 0
\end{array}$$

[[we always write state vecs as columns]] ⟵ "state vector"

[But to save vertical space in this lecture, I am forced to write col. vectors horizontally!]

$$\text{"}\underset{000}{a},\ \underset{001}{b},\ \underset{010}{c},\ \underset{\cdots}{\ldots}\text{"} \quad \text{means} \quad \begin{array}{c} 000 \\ 001 \\ 010 \\ \vdots \end{array}\begin{bmatrix} a \\ b \\ c \\ \vdots \end{bmatrix}$$

[Next, we "prepare the uniform superposition":]

Add & Diff All:

Add & Diff on X

Add & Diff on Y

Add & Diff on Z

// state now (1,1,1,1,1,1,1,1)

Unnormalized: $(\text{amp}(\ ))^2$ adds to $N=8$.

[So far, $n=3$ instructions]

Composite operation on 3 qubits

$\equiv$ 8×8 matrix, $\begin{array}{c} \text{input} \\ 000\ 001\ \cdots\ 111 \end{array}$ output $\begin{array}{c} 000 \\ 001 \\ \vdots \\ 111 \end{array}\begin{bmatrix} \text{Add Diff All} \end{bmatrix}$.

$$\begin{bmatrix} \\ \\ & \text{Add Diff All} & \\ \\ \\ \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

"If $F(X,Y,Z)$ Then Minus"

// we can implement this by quantumizing the circuit C, $\approx 2T$ quantum instructions

// State is now F's "$\pm$ truth table":

$$\text{"}\vec{f}\text{"} := \begin{pmatrix} \underset{000}{+1}, & \underset{001}{+1}, & \underset{010}{+1}, & \underset{011}{+1}, & \underset{100}{+1}, & \underset{101}{-1}, & \underset{110}{+1}, & \underset{111}{+1} \end{pmatrix}.$$

$\uparrow$ x^*

[Notice we got ampl. -1 on the "secret" string x^*, even though we don't know how C works. Unfortunately, "Print All" here would reveal nothing. Continuing the Bias-Busting algorithm...]

$$\boxed{\begin{array}{l} \text{Avg \& Disp All:} \\ \quad \text{Avg \& Disp on } X \\ \quad \text{Avg \& Disp on } Y \\ \quad \text{Avg \& Disp on } Z \end{array}}$$

a composite operation, represented by an 8×8 matrix

$$\begin{bmatrix} \text{Avg \& Disp All} \end{bmatrix} = \begin{bmatrix} \text{Add \& Diff All} \end{bmatrix}^{-1}$$

(inverse)

// state is now ....well....

$$\text{"}\vec{g}\text{"} := \begin{pmatrix} \underset{000}{\mu}, & \underset{001}{?}, & ?, & ?, & ?, & ?, & ?, & \underset{111}{?} \end{pmatrix}$$

$\mu = \text{avg}\{f\text{'s values}\}$

$= \dfrac{7 \cdot (+1) + 1 \cdot (-1)}{8}$

$= 6/8 = .75$

Summary:

$$\begin{bmatrix} \text{Avg Disp All} \end{bmatrix} \begin{bmatrix} \vec{f} \end{bmatrix} = \begin{bmatrix} \vec{g} \end{bmatrix} = \begin{bmatrix} \mu = .75 \\ ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$

$$\begin{bmatrix} \text{Add Diff All} \end{bmatrix} \begin{bmatrix} \vec{g} \end{bmatrix} = \begin{bmatrix} \vec{f} \end{bmatrix}$$

[The other ? entries of $\vec{g}$ are some kinds of "weird displacements" of $\vec{f}$'s 8 values. Let's actually introduce the natural "displacement" vals for $\vec{f}$.]

$$\vec{f} = (\ +1,\ \ +1,\ \ \ldots\ \ +1,\ \ -1,\ \ +1,\ \ +1\ )$$

$$= (.75 + .25,\ .75 + .25,\ \cdots,\ .75 + .25,\ .75 - 1.75,\ .75 + .25,\ .75 + .25)$$

$$= \underset{\substack{\parallel \\ .75}}{\mu}(1,1,1,1,1,1,1,1) + \underbrace{(.25, .25, \ldots, .25, -1.75, .25, .25)}_{\text{"}\vec{\Delta}\text{"}}$$

[and on the other hand...]

$$\vec{g} = \overset{\substack{.75 \\ \parallel}}{\mu}(1,0,0,0,0,0,0,0) + \underbrace{(0, \sim, \sim, \sim, \sim, \sim, \sim, \sim)}_{\text{"}\vec{?}\text{"}}$$

[didn't calculate]

[time for a tiny bit of linear algebra]

$$\therefore\ \vec{f} = \mu(1,1,1,\ldots,1) + \vec{\Delta} \quad ①$$

$$= \left[\text{Add\&Diff All}\right]\Big(\mu(1,0,0,\ldots,0) + \underbrace{\vec{?}}_{\vec{g}}\Big) \quad ②$$

$$= \mu \cdot \underbrace{\left[\text{Add Diff All}\right]\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}}_{(1,1,1,\ldots,1)} + \left[\text{Add Diff All}\right]\cdot\left[\vec{?}\right] \quad ③$$

[this is the "preparing the uniform superposition" trick]

So ① = ③ looks like "$P + Q = P + R$"

$P = \mu(1,1,\ldots,1)$

$$\therefore\ Q = R \Rightarrow -Q = -R$$

$$\Rightarrow P - Q = P - R$$

$\Rightarrow$ can change $+$ to $-$ in ①, ②, ③

$$\therefore\ \vec{f}' := \mu(1,1,\ldots,1) - \vec{\Delta}$$

$$= \left[\text{Add\&Diff All}\right]\Big(\underbrace{\mu(1,0,0,\ldots,0) - \vec{?}}_{\text{"}\vec{g}'\text{"}}\Big)$$

$$\vec{g} = \begin{bmatrix}.75\\\alpha\\\beta\\\gamma\\\vdots\\\omega\end{bmatrix} = \underbrace{.75}_{\mu}\begin{bmatrix}1\\0\\0\\\vdots\\0\end{bmatrix} + \underbrace{\begin{bmatrix}0\\\alpha\\\beta\\\gamma\\\vdots\\\omega\end{bmatrix}}_{\vec{?}} \Rightarrow \vec{g}' = \begin{bmatrix}.75\\-\alpha\\-\beta\\-\gamma\\\vdots\\-\omega\end{bmatrix}$$

Summary:

$$\vec{g} = (.75, \alpha, \beta, \ldots, \omega)$$

[$\alpha, \beta, \ldots, \omega$ are #'s we didn't calculate]

$$\vec{g}' = (.75, -\alpha, -\beta, \ldots, -\omega)$$

we can produce!

$$\vec{f} = (1, 1, \ldots, 1, \underset{\uparrow x^*}{-1}, 1, 1)$$

$$= (.75+.25, \ldots, .75+.25, .75-1.75, .75+.25, .75+.25)$$

$$\vec{f}' = (.75-.25, \ldots, .75-.25, .75+1.75, .75-.25, .75-.25)$$

very interesting!

$$= (.5, \ldots, .5, 2.5, .5, .5)$$

f's values got "reflected across their mean"

$$[\vec{f}'] = [\text{Add \& Diff All}]\,[\vec{g}']$$

[How? We're negating every ampl. except the one on $00\cdots0$.]

"If OR($\cdots$) Then Minus"!

Thm: Grover's "<u>Reflection Across The Mean</u>" oper:

- Aug & Disp on $X_1, \ldots, X_n$ // n instrs.
- If OR($X_1, \ldots, X_n$) Then Minus // $\approx 2n$ instrs.
- Add & Diff on $X_1, \ldots, X_n$ // n instrs.

$\approx 4n$ instructions: Effect is $(\mu+\Delta_{000}, \mu+\Delta_{001}, \ldots, \mu+\Delta_{111})$

(where μ = avg. ampl. value) $\mapsto (\mu-\Delta_{000}, \mu-\Delta_{001}, \ldots, \mu-\Delta_{111})$

[[Back to our $n=3$ example...]]

Unif. superpos: $(1, 1, 1, \ldots\ldots, 1)$

If F Then Minus: $(1, 1, \ldots, 1, -1, 1, 1)$

$x^* = 101$ position

"Refl. Across Mean": $(.5, .5, \ldots\ldots, .5, 2.5, .5, .5)$

.75

[[Cool! Got a "big" ampl. on $x^* = 101$. If we "Print All" now, what are probabilities?]]

$$7 \times (.5)^2 + 1 \times (2.5)^2 = 7 \times .25 + 6.25 = 8 \checkmark$$

$$\text{Prob}\left(\text{print } x^* = 101\right) = \frac{6.25}{8} = .78125. \quad 78\% \text{ success!}$$

[[Well, that was $n=3$, $N=8$. General case?]]

Unique-SAT-Solve (C computing $F: \{0,1\}^n \to \{0,1\}$, $F(x^*) = 1$ for unique "secret" x^*)

- Initialize $X_1, \ldots, X_n$
- Add & Diff All

} prep unif superpos (n instrs)

- If $F(X_1, \ldots, X_n)$ Then Minus ($\approx 2T$ instrs.)
- Refl. Across Mean ($\approx 4n$ instrs.)

} "Combo"

[[Let's examine amplitudes]]

After "If F Then Minus":

state is $(1, 1, 1, \ldots, 1, -1, 1, \ldots\ldots)$

$\quad\uparrow x^*$ position. $\quad \Sigma(\text{Ampl})^2 = N = 2^n$

Mean $\mu = \dfrac{(N-1)(+1) + (-1)}{N} = \dfrac{N-2}{N} = 1 - \underbrace{\dfrac{2}{N}}_{\text{(unphysically tiny)}} \approx 1^{\text{ish}}$

Write state as μ + displacements:

technically, $2/N$

$\left(1^{\text{ish}} + 0^{\text{ish}},\ 1^{\text{ish}} + 0^{\text{ish}},\ \ldots,\ 1^{\text{ish}} + 0^{\text{ish}},\ \underbrace{1 - 2}_{=-1},\ 1^{\text{ish}} + 0^{\text{ish}}, \ldots, 1^{\text{ish}} + 0^{\text{ish}}\right)$

technically, $\mu = 1 - 2/N$

- Refl. Across Mean

$\rightarrow \left(1^{\text{ish}} - 0^{\text{ish}},\ 1 - 0,\ \ldots,\ 1 - 0,\ 1 + 2,\ 1 + 0, \ldots, 1 + 0\right)$

$= \left(1, 1, \ldots, 1, 3, 1, \ldots, 1\right)^{\text{ish}}$

[[We got 2.5 here in N=8 case as we've twice neglected $\frac{2}{N}$ here]]

Print All now? $\quad \text{Prob}[x^*] \approx \dfrac{3^2}{N} = \dfrac{9}{N}$

[[I mean, 9x better than random guessing, but still unphysically small...]]

Final idea: Repeat the "combo" k times...

time $k=2$:

$$(1, \ldots, 1, 3, 1, \ldots, 1) \xrightarrow{\text{If F Then Minus}}$$

$$(1, \ldots, 1, -3, 1, \ldots, 1)$$

(mean $\mu \approx 1$ ish)

$$= (1+0, \ldots, 1+0, 1-4, 1+0, \ldots, 1+0) \xrightarrow{\text{Refl. Across Mean}}$$

$$(1-0, \ldots, 1-0, 1+4, 1, \ldots, 1)$$

$$= (1, \ldots, 1, 5, 1, \ldots, 1) \text{ ish}$$

[Cool! "Print All" now gives x^* with prob. $\approx \frac{25}{N}$.]

$k=3$:

$$(1, \ldots, 1, -5, 1, \ldots, 1) \quad \text{If F Minus}$$

mean $\mu \approx 1$ still (actually $1 - 6/N = 1 - 2k/N$)

$$= (1+0, \ldots, 1-6, 1+0, \ldots) \xrightarrow{\text{Refl. Acr. Mean}}$$

$$(1-0, \ldots, 1+6, 1-0, \ldots)$$

$$= (1, \ldots, 7, 1, \ldots)$$

etc.

Conclusion: So long as mean is ≈ 1 ish, ampl. on x^* goes up $+2$ ish each step!

After k repetitions:

- $k \times O(n+T)$ instructions [the $O(n+T)$ will be negligible, so think $\approx k$ time]

- $\text{Ampl}[x^*] \approx 2k+1$,

  total squared ampl. N.

[Well.... this can't be true for $k \gg \frac{\sqrt{N}}{2}$.

The "ish" breaks down once $k \approx (\text{small const.})\sqrt{N}$.

Doing analysis very carefully is slightly tedious....

Let's for now just say....]

$\approx 2k+1$ holds provided $k \leq .01\sqrt{N}$, say.

Conclusion: $k = .01\sqrt{N}$ reps $\Rightarrow$ $\tilde{O}(\sqrt{N}) = \tilde{O}(\sqrt{2^n})$ time

- $\Pr[\text{print out } x^*] \approx (.02\sqrt{N})^2 / N$

  $= .0004 = 1/2500.$

[That's great!!] Double-check any output x by plugging into C. [Only $\approx T$ more steps]

Repeat everything ~~2500~~ 10,000 times, high prob. of eventually getting x^*!

// If you grind on analysis, optimal value of k, leading to $\Pr[x^*] \approx 100\%$ is $k = \ldots \sqrt{N} \cdot \frac{\pi}{4}$!

Yes, pi shows up!

Like in the 3B1B sliding blocks problem, "a circle is involved" somewhere.

So course will now switch gears to geometry, rotations, linear algebra...

PS: not only do they both secretly have circles....

... secretly the sliding blocks problem is identical to Grover's algorithm!! See H.W... //