Lecture 4 –
The Nature of Quantum Operations

This lecture:

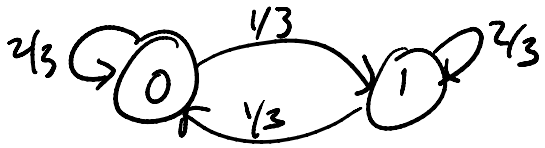
- amplitude trees can be "compressed"
- quantum operations can be represented by bipartite dags ("directed acyclic graphs"), or matrices
- reverses & inverses of operations
- what are the physically "allowed" quantum ops?

Remark: "Make A, B, C, ..." ← Almost always just at start
& "Print All" ← " " " " end
are "special" operations.

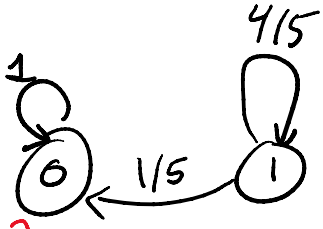
Today: the other, "usual" (computing) operations:
Add 1 To A, Add A or B To C,
Hat A, etc...

[We'll start with simplifications/new perspectives on amplitude (& probability) trees, particularly "merging" nodes to get "dags". For clarity, we'll start by talking about prob. comp. again.]

Recall: "Noise" probabilistic operation:

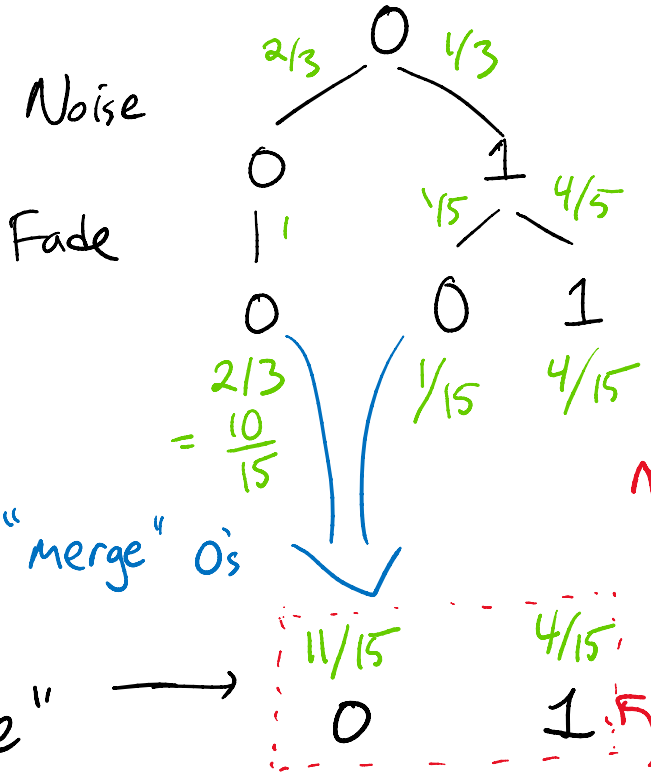


Here's another random one: "Fade":



[0 stays 0, 1 flips to 0 w/ prob. 1/5.]

Analyze Fade(Noise(0)):



[Question: is "merging" OK? Like, if another instruction comes next, do we need to remember 3 leaves or can we continue calcs from these 2? Answer: merging is fine! See hmwk.]

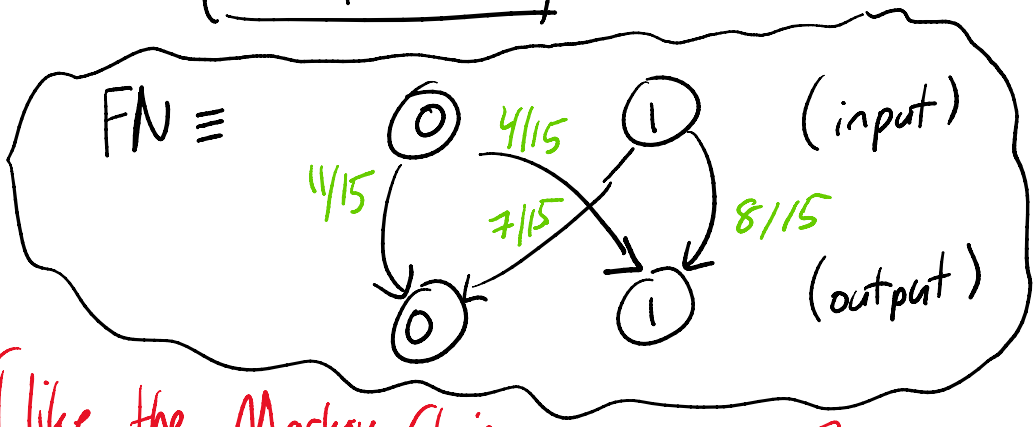
Exercise: Fade(Noise(1)) → probabilistic state

7/15 on 0 — 8/15 on 1

Summary: Define

```
FN(A):
    Noise A
    Fade A.
```

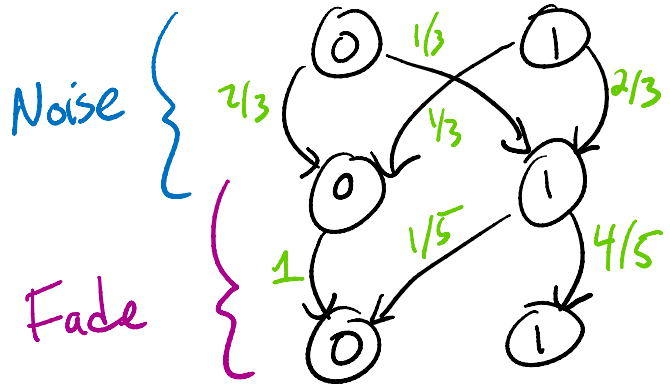
Then FN ≡



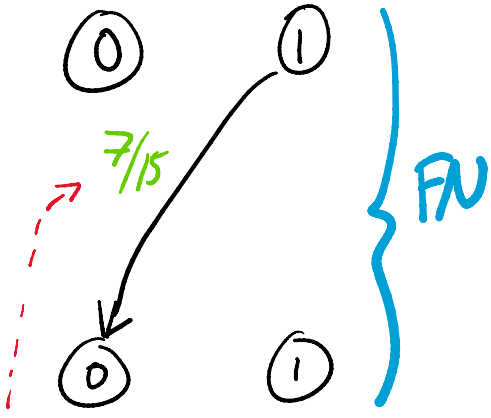
"bipartite probability dag"

[[like the Markov Chain, but input/output is separated]]

Can compute it by....



⇒



[[We "glued" the dags for Noise & Fade together...]]

Each # gotten by the "sum path-products" recipe.

[[Find all composite ①→① paths on left, multiply probs along paths, add results.]]

Recipe: to get the $FN[1\to 0]$ label $\frac{7}{15}$, go over all $1\to\cdots\to 0$ paths on left, multiply, and add:

$$FN(1\to 0) = N[1\to 0]\cdot F[0\to 0] + N[1\to 1]\cdot F[1\to 0]$$
$$1/3 \cdot 1 + 2/3 \cdot 1/5$$

[I don't mean to alarm you, but we just rediscovered.

Matrix Multiplication!]

$$\underset{\text{output}}{\begin{matrix}0\\1\end{matrix}}\overset{\text{input}}{\begin{matrix}0 & 1\end{matrix}}\begin{bmatrix} FN(0\to 0) & FN(1\to 0) \\ & 1\cdot 1/3 + 1/5\cdot 2/3 = 7/15 \\ FN(0\to 1) & FN(1\to 1) \end{bmatrix} = \underset{\text{output}}{\begin{matrix}0\\1\end{matrix}}\overset{\text{input}}{\begin{matrix}0 & 1\end{matrix}}\begin{bmatrix} F[0\to 0] & F[1\to 0] \\ 1 & 1/5 \\ F[0\to 1] & F[1\to 1] \\ 0 & 4/5 \end{bmatrix} \cdot \underset{\text{output}}{\begin{matrix}0\\1\end{matrix}}\overset{\text{input}}{\begin{matrix}0 & 1\end{matrix}}\begin{bmatrix} N[0\to 0] & N[1\to 0] \\ 2/3 & 1/3 \\ N[0\to 1] & N[1\to 1] \\ 1/3 & 2/3 \end{bmatrix}$$

FN $\quad$ F $\quad$ N

[Each bipartite dag for a 1-bit probability transformation has 4 #'s. Arrange them into a 2x2 matrix as above & then composition ≡ matrix multiplication.]

[Sorry that $F\cdot N$ means "first N, then F". Weird ordering is mathematicians' fault.]

Summary

[this is for probabilistic comp., but will be identical for q. comp.]

Every probabilistic operation (1+ "instructions") operating on 1 (or 2, 3, 4...) bits can be encapsulated by:

1. "bipartite prob. dag" with 2 (or 4, 8, 16, ...) inputs & outputs [& arrows out of each input form a "probabilistic state"]
2. a 2x2 (or 4x4, 8x8, 16x16...) matrix, with columns (inputs) & rows (outputs) labeled by binary strings of length 1 (or 2, 3, 4...) [each column is a "prob. state"]

Compositing two operations:

1. glue dags together, sum path-products recipe
2. matrix multiplication

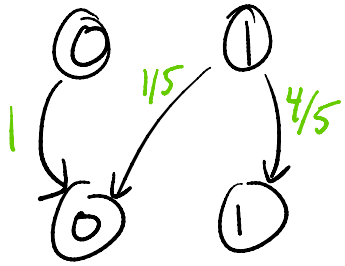
[By the way, mathematicians had barely heard of matrices before quantum mechanics was discovered in the 1920's. Matrices developed b/c of Q.M. Indeed, Q.M. originally called "Matrix Mechanics".]

[[These two equivalent representations tell you immediately what prob. state you get to if you start deterministically at 0 or 1. But what if you do an operation when you're already in a probabilistic state?...]]

Say you're in prob. state: $\frac{1}{4}$ on 0, $\frac{3}{4}$ on 1.

prob. state ≡ column vector

Then you do Fade.

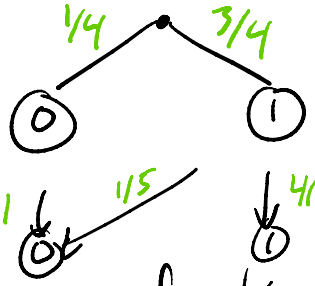


[[def^n of matrix-vector mult.]]

$$\begin{matrix} & 0 & 1 \\ 0 & 1 & 1/5 \\ 1 & 0 & 4/5 \end{matrix} \cdot \begin{matrix} 0 \\ 1 \end{matrix} \begin{bmatrix} 1/4 \\ 3/4 \end{bmatrix}$$

$$= \frac{1}{4}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{3}{4}\begin{bmatrix} 1/5 \\ 4/5 \end{bmatrix}$$

$$= \begin{bmatrix} 8/20 \\ 12/20 \end{bmatrix} \leftarrow \text{new state.}$$

Can also depict state like this: 

, glue together with bip. dag & use sum-of-path-products rule.

EVERYTHING WE'VE DONE SO FAR HOLDS EQUALLY FOR QUANTUM COMPUTING

with one difference:

"prob. state": nonnegative probs that add to 1

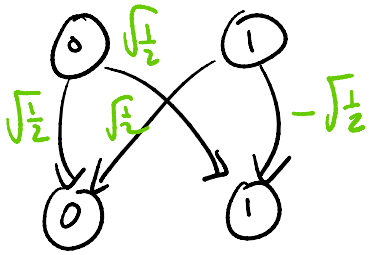
"quantum state": ± amplitudes whose squares add to 1

[in fact, amplitudes can be complex numbers, but we'll get into that later]

Prob. state —(prob. operations)→ prob. state ✓: reasonably obvious

Quantum state —(quantum operations)→ quantum state ✓: not obvious

"Hat": (real name: Hadamard)



$\equiv$

out \ in	0	1
0	$\sqrt{\frac{1}{2}}$	$\sqrt{\frac{1}{2}}$
1	$\sqrt{\frac{1}{2}}$	$-\sqrt{\frac{1}{2}}$

← Hadamard matrix, hence the name

quantum states ✓

[squares add to 1]

So it's obvious that if you do Hat to 0 or 1 you get a valid quantum state, but what if you are already in some "superposition" (general state) & do Hat: why is the result a quantum state?]

[Relies on a crucial property all quantum operations possess related to....]

Reversibility:

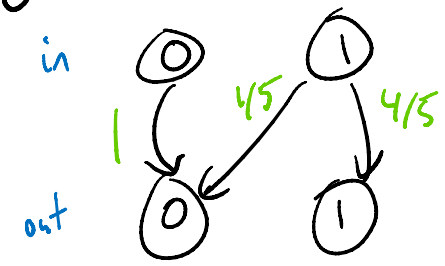
Back to probabilistic computing:

Reversing an operation:

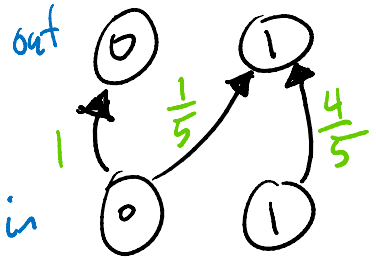
- bip. dag view: reverse all arrows (& input/output)
- matrix view: swap rows/columns. "Transpose"

WARNING: Result might not be a valid prob. oper.

e.g.: Fade



(Fade)$^{\text{Reverse}}$



✗ not valid

$$F: \begin{matrix} \\ \text{out } 0 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} 0 & \text{in} \ 1 \end{matrix} \\ \begin{bmatrix} 1 & 1/5 \\ 0 & 4/5 \end{bmatrix} \end{matrix}$$

$$F^{\top}: \begin{matrix} \\ \text{out } 0 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} 0 & \text{in} \ 1 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 1/5 & 4/5 \end{bmatrix} \end{matrix}$$

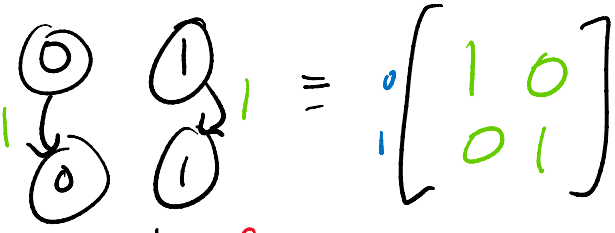
"transpose, the official linear algebra symbol"

OTOH: $\text{Noise} = \begin{bmatrix} 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix}$, so $\text{Noise}^{\top} = \text{Noise}$ ✓ OK

[[Intuition: Watch a video of a Markov chain for a while. Then play it in reverse. Does it still look like a Markov chain? If so, Reverse is a valid prob. operation.]]

Inverse := "undo".

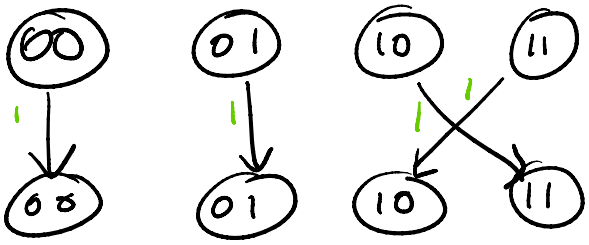
$(\text{Noise})^{-1}$ [[inverse notation]] means an instruction that exactly undoes Noise. That is,

Noise then $(\text{Noise})^{-1} \equiv$ 

There is no such instruction! [[You can't unflip a coin!]]

[[There's also no $(\text{Fade})^{-1}$! If it exists, it's the matrix inverse, which is a little hard to compute. In fact, in Probabilistic Computing, inverses almost never exist. Exception: deterministic instructions that are permutations.]]

"Add A to B:"

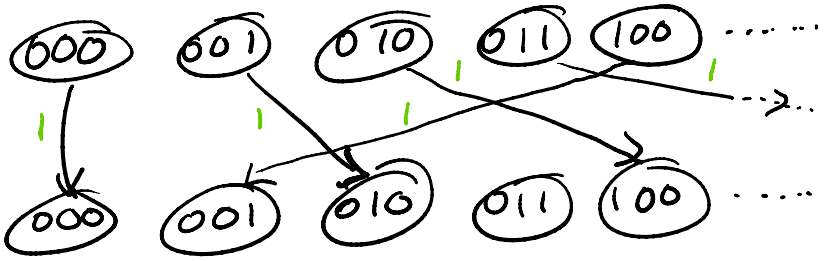


Reverse = Inverse = "Add A to B" !

Same with all other deterministic Scratch instructions

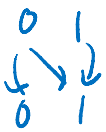
[Not always self-inverse. E.g....]

"Left-Shift (A,B,C):"



$$(\text{Left-Shift})^{T} = (\text{Left-Shift})^{-1} = \text{Right-Shift}$$

(reverse: T; inverse: -1)

Say you do F:  then N:  ⇒ 

[composite op.]

What's its reverse?

It's 

, which is clearly "N^{reverse}, then F^{reverse}".

Matrix Form: $(NF)^{T} = F^{T}N^{T}$.

Similarly: If, hypothetically, N^{-1}, F^{-1} existed, then $(NF)^{-1} = F^{-1}N^{-1}$.

Back to Quantum

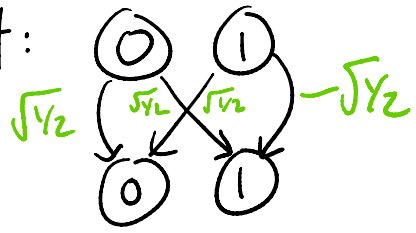
Reverse: still just reverse arrows.

[[And, if there are complex #'s, switch i & $-i$ (complex conjugate).]]

Symbol: $(\text{Code})^{\dagger}$ ← "dagger" [[Looks like T, but dagger reminds you to switch i & $-i$.]]

Inverse: still an instruction that undoes: $(\text{Code})^{-1} \cdot (\text{Code}) =$ do nothing

E.g.: Hat:



$\cdot\ (\text{Hat})^{\dagger} = \text{Hat}$ [[Visibly, from reversing arrows]]

As we saw: $\text{Hat}(\text{Hat}(0)) = 0$

$\text{Hat}(\text{Hat}(1)) = 1$

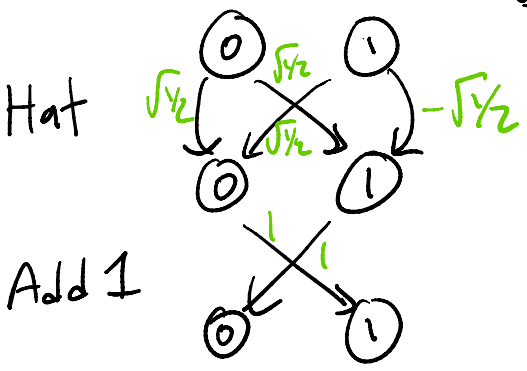
So Hat · Hat does nothing: $(\text{Hat})^{-1} = \text{Hat}$.

[[Like the deterministic permutations...]]

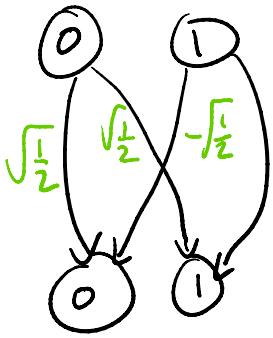
Hat reverse = Hat inverse. (= Hat)

Eg 2: def

Rot(A):
Hat on A
Add 1 to A



composite via sum of path products



$\equiv \begin{bmatrix} \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix}$

[[Any graphics/lin.alg. enthusiasts see why I called it "Rot"]]

$Rot^{reverse}$:

$(\text{Add 1 to A})^{rev} = \text{Add 1 to A}$

$(\text{Hat on A})^{rev} = \text{Hat on A}$.

$Rot^{\dagger} = \begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix}$

What's $(Rot)^{-1}$? To undo Rot, Add 1, then Hat.

So $(Rot)^{-1} = (Rot)^{\dagger}$!

check $\begin{bmatrix} \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \\ -\sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix} \cdot \begin{bmatrix} \sqrt{\frac{1}{2}} & -\sqrt{\frac{1}{2}} \\ \sqrt{\frac{1}{2}} & \sqrt{\frac{1}{2}} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ ✓

Fact: Every quantum operation (1+ instructions) we've seen so far has an inverse, and it's the reverse.

LAW: [of Nature / Physics / Q.M.]

> Any operation U with $U^{-1} = U^\dagger$ is a physically allowable quantum operation, & vice versa.

[Related to time-reversibility of physical laws.]

[Examples: Left & Right Shift are allowable/implementable, as is $U = \begin{bmatrix} .8 & -.6 \\ .6 & .8 \end{bmatrix}$]

Terminology: Linear algebra lovers call such matrices "Unitary".

[Why? Because they preserve unit-length vectors.]

Theorem: A matrix [or bipartite diag.] is "unitary" $\Rightarrow$ it maps quantum states [unit-length column vectors] to quantum states.

[$\Leftarrow$ direction also true, but more of mathematical interest, so we won't prove it here]

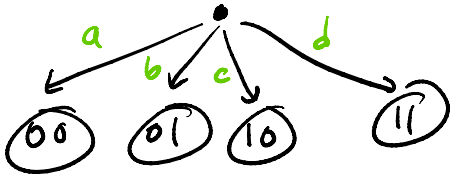
Proof: [[Illustrated in the case of 2-bit operations.]]

Say $\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$ is a quantum state, so

Ampl[00] → a
Ampl[01] → b
Ampl[10] → c
Ampl[11] → d

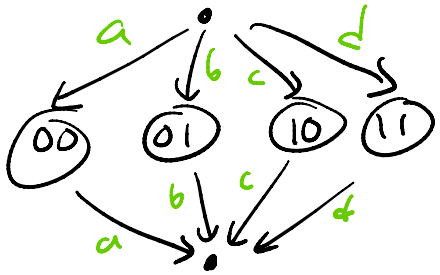
$$a^2 + b^2 + c^2 + d^2 = 1.$$

Recall we drew it like



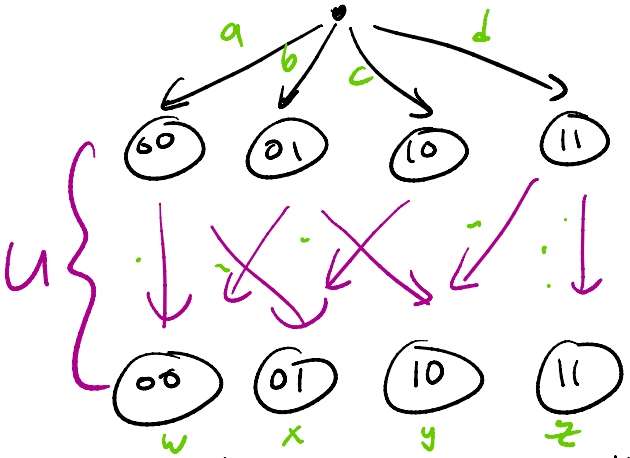
[[How to get at $a^2+b^2+c^2+d^2$?]]

Make its reverse, glue together, do sum of path-products



Sum of path-products = $a^2+b^2+c^2+d^2 = 1$.

Say from this state we do quantum op U.

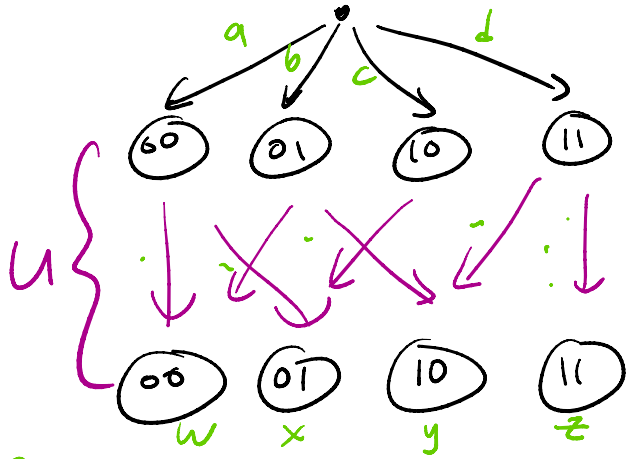


[[Matrix viewpoint:

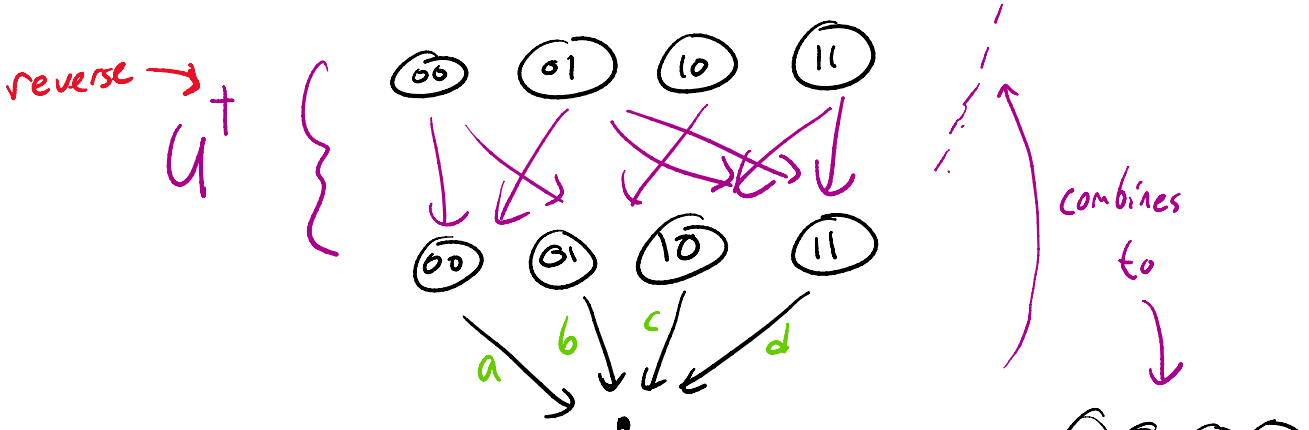
$$\begin{bmatrix} \cdots \\ \cdots U \cdots \\ \cdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

]]

Sum of path-products gives 4 new #s. A quantum state??



To get $w^2+x^2+y^2+z^2$ (sum of final ampls^2), again reverse whole pic, glue, do sum of path prods...



But $U^\dagger \cdot U = U^{-1} \cdot U =$ do nothing!

So $w^2+x^2+y^2+z^2 = a^2+b^2+c^2+d^2 = 1$ ✓

Summary: I only gave you Hat because theoretically it suffices to do quantum comp. But the set of all allowable instructions in QM is all "unitaries". Still, as in programming the interest is in efficiently coding up complicated operations from simple instructions....