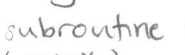
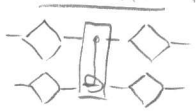
Lecture 8 2/4/10 Fault-tolerant ancilla preparation:

- necessary to start the computation
- key subroutine in Steane-style () and Knill-style () error correction
- often dominant in overhead of a scheme



for a random/typical n-qubit CSS code, transversal gates & measts cost n; preparing state costs $\sim \frac{n^2}{4}$ ($\frac{n}{2}$ X stabilizers, each of wt. $\sim \frac{n}{2}$).

Method 1 · Shor-style prep.

apply EC to an arbitrary (eg. random) state

- circular, except for Shor-style EC
- moves into the codespace, but w/ unknown codeword

→ ok if think of $|\bar{0}\rangle$ as an $[[n,0,d]]$ QECC (encodes 0 qubits, ie. 1-dim.)

ie. extract syndrome of each of the $\frac{n-1}{2}$ X and $\frac{n+1}{2}$ Z stabilizers (works for non-CSS too) using appropriate cat states

disadvantage: works poorly for large n

Method 2 · Steane-style prep (for CSS codes)

- prepare ✓ ← [Steane 0202036]
- verify ✓ ← not ✓

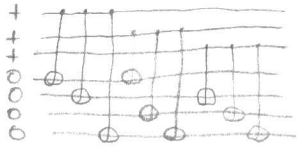
① Preparation

- Gauss ran-eliminate X (or Z) stabilizers into the form $(I \; A)$

eg.
$$\begin{matrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{matrix} \rightsquigarrow \begin{matrix} 1\,0\,0\,1\,1\,0\,1 \\ 0\,1\,0\,1\,0\,1\,1 \\ 0\,0\,1\,1\,0\,1\,1 \end{matrix}$$
(A)

- prepare $|+\rangle^{\otimes \text{num X stabs}} \otimes |0\rangle^{\otimes n - m}$
- apply CNOTs from controls to targets to create X stabs.

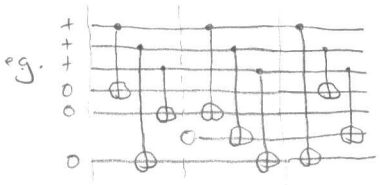
eg.



state necessarily has correct Z stabs. ie is $|\bar{0}\rangle$

Optimized preparation

- don't prepare qubit until needed
- parallelize gates into as few rounds as possible →

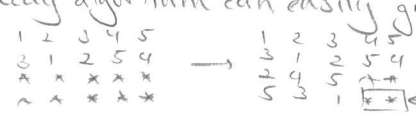
e.g. 

$\longleftrightarrow$

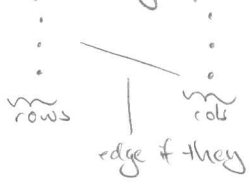
1	2	0	3
3	0	2	1
0	1	3	2

Lemma: Consider a matrix a subset of whose positions have been marked *.
Let n = max. # of *s in any row or col.
⇒ *'ed positions can be colored w/ $\{1, 2, \ldots, n\}$ s.t. no number appears twice in any row or column.

Remark: A completely naive greedy algorithm can easily get in trouble.

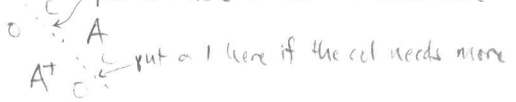
eg  or 

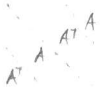
Proof: Induction in n. n=0 trivial.
Suffices to place n's s.t. in remaining positions, $\leq n-1$ *s in any row or col.
"Worst case" (???) is every row & col has exactly n *s.
Then consider the bipartite graph (the biadjacency matrix is the *s)



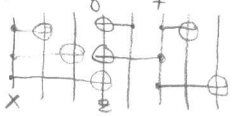
This is a regular graph with degree n on both sides.
∴ ∃ a perfect matching, ie. a coloring that reduces everybody by one *.
(Proof Hall's Matching Thm: For a set S of rows, the # of incident edges is $n \cdot |S|$. This is $\leq n \cdot |N(S)|$, the # of incident edges to N(S).
Thus $|S| \leq |N(S)|$ ∀S, so matching thm. applies.) ~sort of

This is true b/c we can always embed the bipartite graph in a larger one in which every vertex has degree =n, and the restriction of a matching on the larger graph in particular covers every row/col of the smaller graph.

Here's how: 

Repeat over & over again, up to n-1 times.  etc.

This can be suboptimal:
- eg. $|00000\rangle + |11111\rangle$  vs.
- using teleportation, multiple CNOTs can be applied from 1-qubit in one round

(example) 

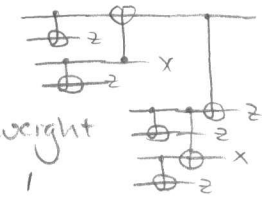
Open: Come up w/ a faster encoding method for general CSS states

② Verifying ancillas

[A] Piecemeal, one stabilizer at a time (error detection or correction)

[B] All at once (error detection)

for a $d=3$ perfect CSS code



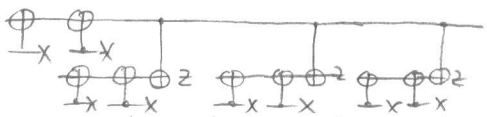
Error orders →	X error weight 0	1	2	3	Z error weight 0	1
unverified ancilla $\overline{0}$	0	1	1	1	0	1 ✓
$\overline{0}$ Z	0	1	2	2	0	1
$\overline{0}$ Z Z	0	1	2	3 ✓	0	1 ✓

for a $d=7$ perfect CSS code

	X error wt 0	1	2	3	4
$\overline{0}$ —	0	1	1	1	1
$\overline{0}$ Z	0	1	2	2	2
$\overline{0}$ Z Z	0	1	2	3	3
$\overline{0}$ Z Z Z	0	1	2	3	4 ✓

	Z wt 0	1	2	3
$\overline{0}$ —	0	1	1	1
$\overline{0}$ X	0	1	2	2
X X	0	1	2	3

eg.



(can apply a random ckt to reduce X/Z asymmetry)

- can be optimized: use different prep'n ckts to create different correlated errors that can't cancel out

do not use full error rejection

[C] Start by encoding into an error-detecting code

eg. X X X X
Z Z 1 1
1 1 Z Z

- then apply Steane's prep ckt
- finally, or occasionally, check for errors
- decode

[D] Use the "slow measure" decoding trick from last time to avoid any/some verification

→ reduces overhead, also threshold

(massive ancilla verification probably gives best thresholds)