Optimizing quantum circuits with classical thinking

Craig Gidney
Google Quantum AI

QPL/MFPS 2018

Goal: explain section 3-D of arXiv:1805.03662

Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity

Ryan Babbush,[1,*] Craig Gidney,[2] Dominic W. Berry,[3] Nathan Wiebe,[4]
Jarrod McClean,[1] Alexandru Paler,[5] Austin Fowler,[2] and Hartmut Neven[1]

[...]

D. Subsampling the Coefficient Oracle

In this section we introduce a technique for initializing a state with L unique coefficients (provided by a classical database) with a number of T gates scaling as $4L + \mathcal{O}(\log(1/\epsilon))$ where ϵ is the largest absolute error that one can

[...]

Key ideas we'll cover

1. Cost of error corrected quantum computation

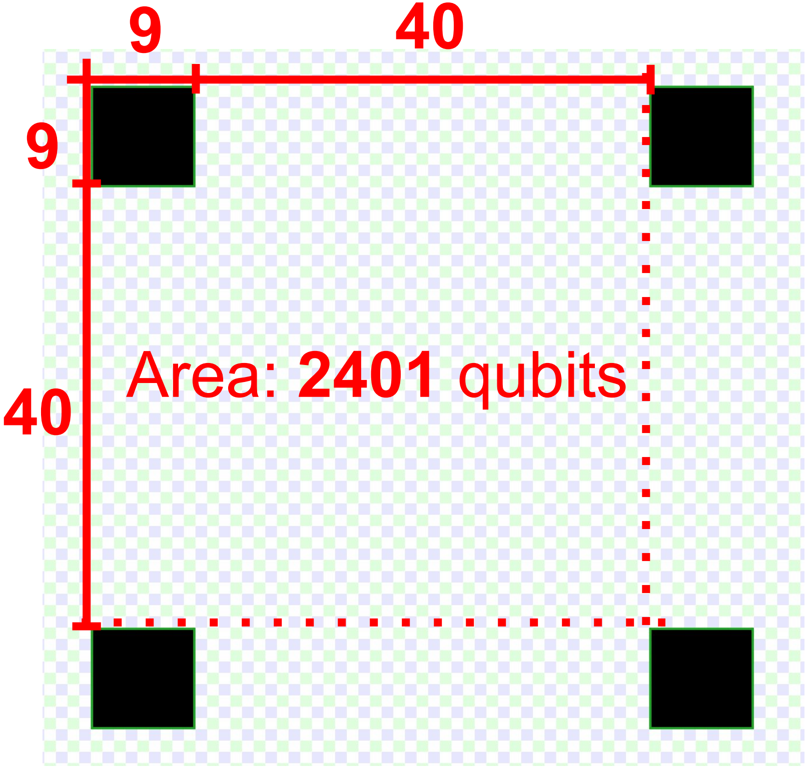
2. Preparing phase-insensitive superpositions == random sampling

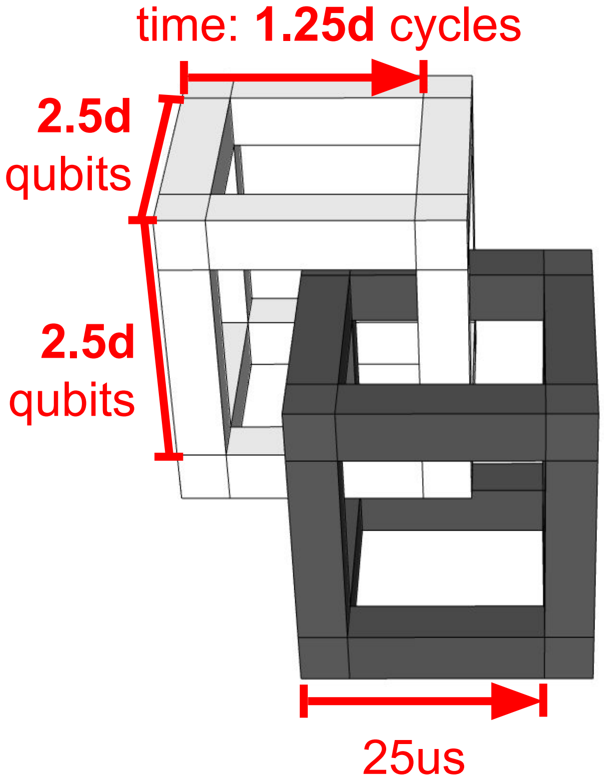
3. Fast proportionate sampling

4. Putting it all together for *savings!*

Part 1
The cost of error corrected quantum computation

""""Real world"""" parameters: d_{code}≈20, t_{cycle}≈1us

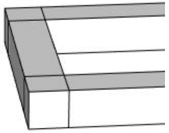




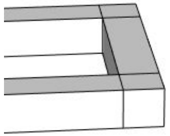
Google

Basic error-corrected operations

initialization: cheap



measurement: cheap

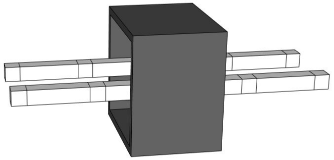


Time

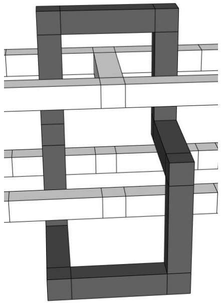
NOT gate: free

this space intentionally left blank

Sqrt(NOT) gate: cheap

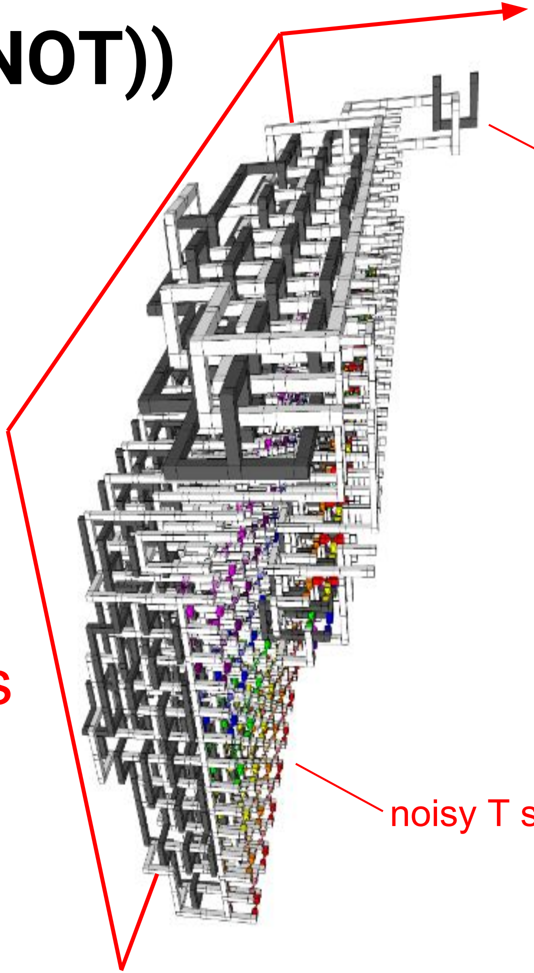


Controlled-NOT: cheap

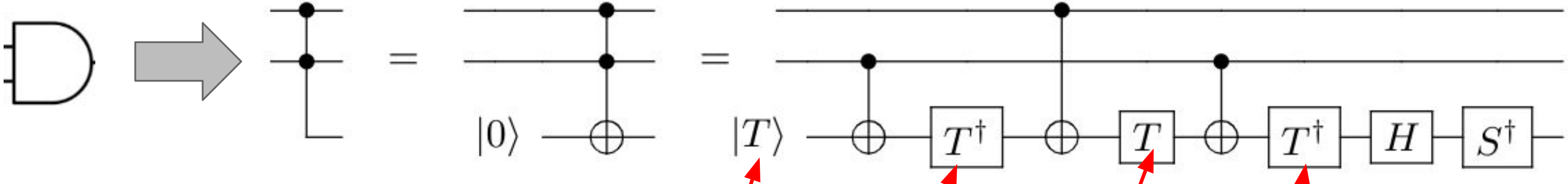


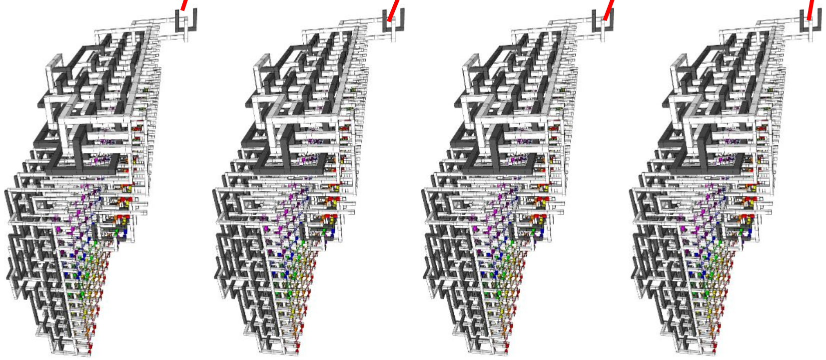
Not so cheap: Sqrt(Sqrt(NOT))

T state factory:



Quantum AND gate: expensive!





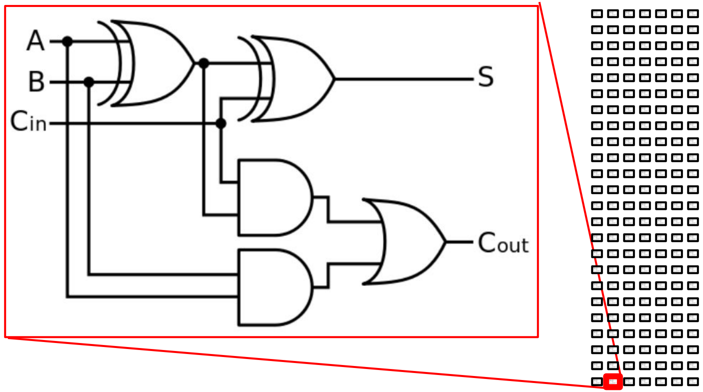
OR, NAND, NOR, etc
are similarly expensive.

150us + 150us + 150us + 150us = **0.6ms**

Google

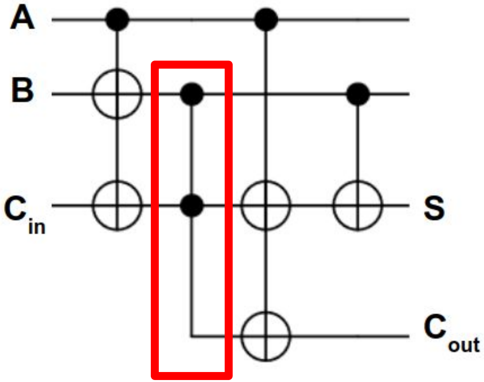
Wildly differing costs

Classical perspective on gate costs



FullAdder isn't even a whole instruction.

Quantum perspective on gate costs



FullAdder takes a half millisecond.

Another cost: reading data under superposition

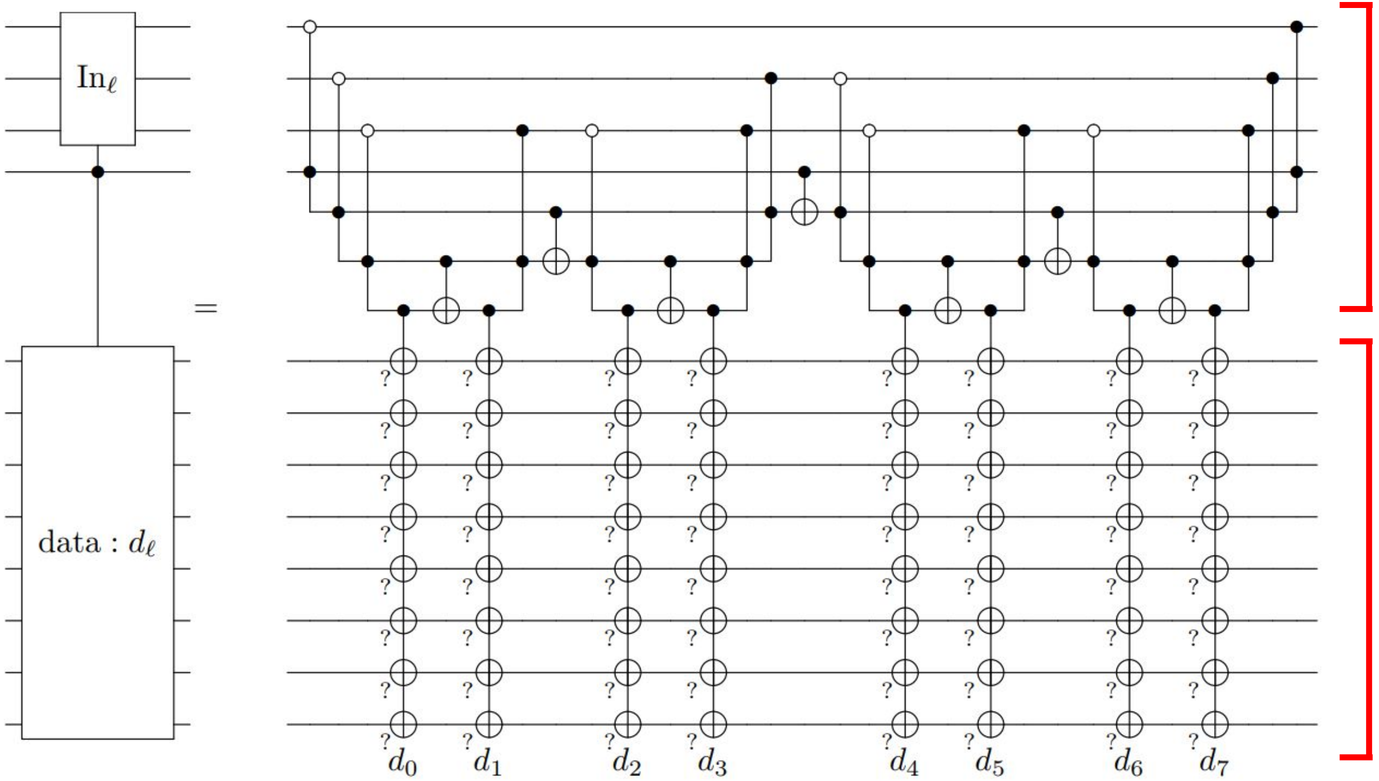
- RAM takes O(N) space to store.

- N AND gates is expensive, but N logical qubits are even more expensive.

- Instead of storing data in qubits, hardcode it into a circuit ("QROM").

- QROM circuit needs AND gates.

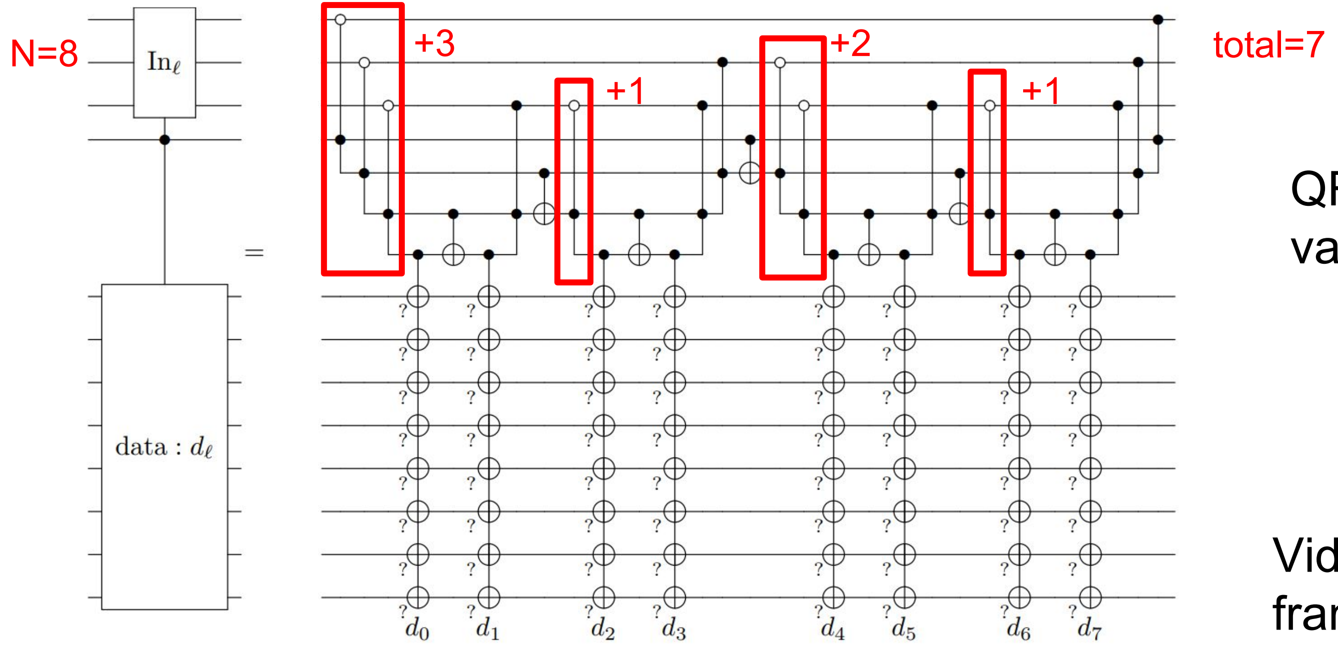
Reading data under superposition: QROM circuit



Iterate over possible index values.

Encode data into presence/absence of CNOT targets.

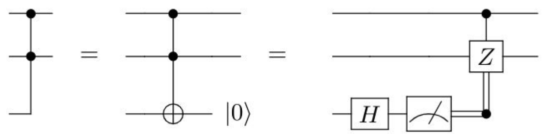
Reading data under superposition: Expensive!



QROM query over N values: N-1 AND gates

Video games render frames faster than we hope to do QROM reads

Note: uncomputing AND is ~cheap



Part 2
Preparing quantum states

Google

The Preparation Problem

Given precomputed coefficients for a superposition, prepare such a superposition

$$[a_0, a_1, a_2, \ldots, a_{N-1}]$$

$$\downarrow$$

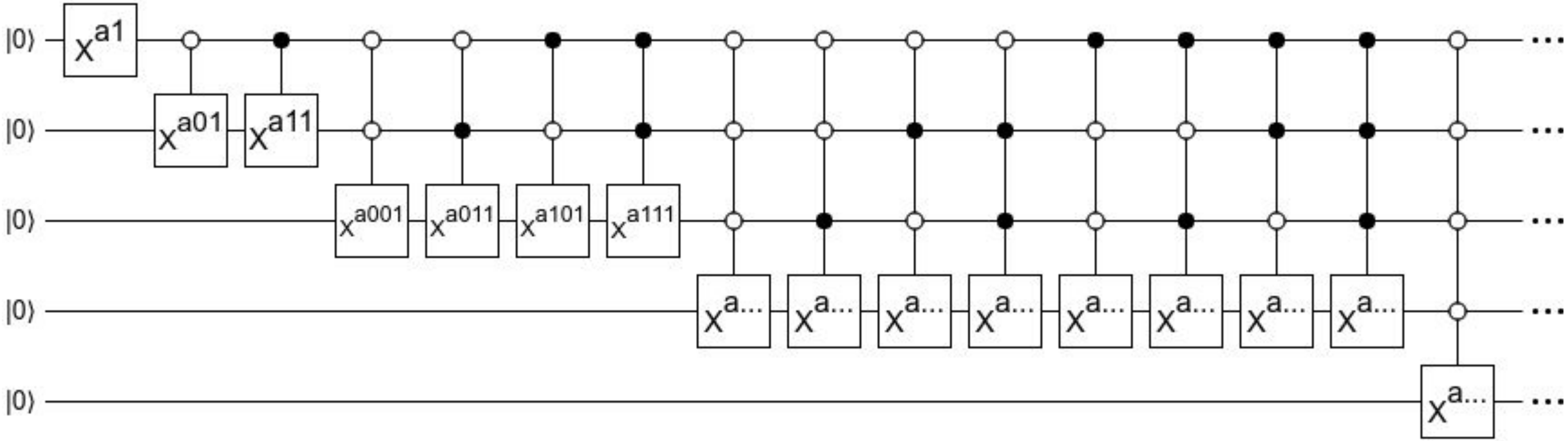
$$a_0|0\rangle + a_1|1\rangle + a_2|2\rangle + \ldots + a_{N-1}|N-1\rangle$$

Previous Approach

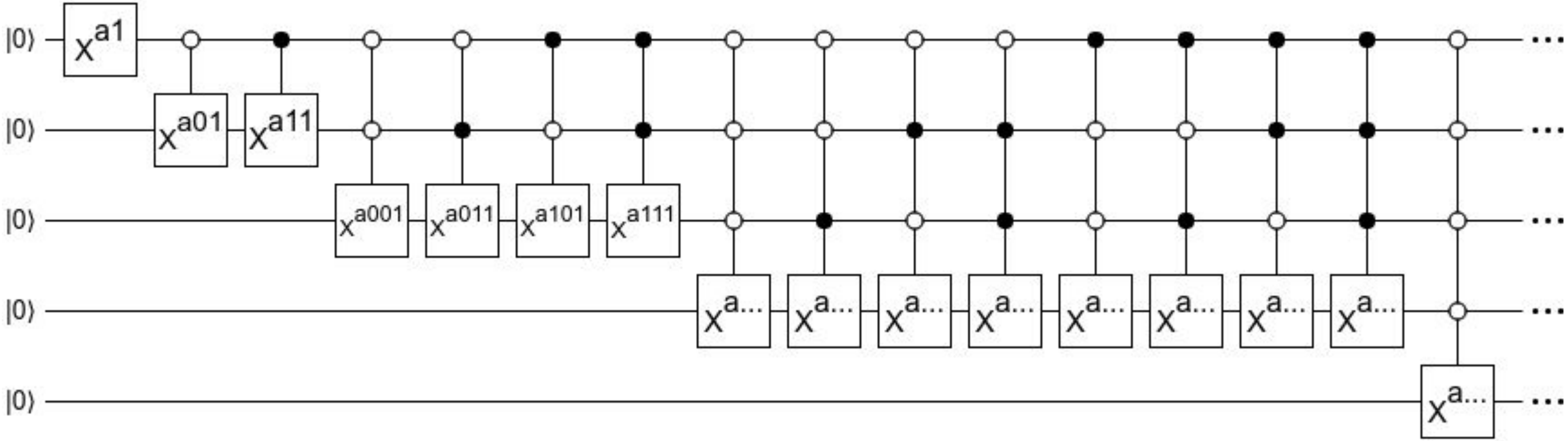
Set ON-vs-OFF proportion of a qubit just right with a precise rotation.

Conditioned on first qubit, set another qubit's ON-vs-OFF proportion just right.

Etc.



Cost of Previous Approach



Uses N-1 precise rotations.

Cost of precise rotation ≈ 12 AND gates. (≈50 T gates)

Roughly 3/4 of a second at N=100.

Google

Key insight: sometimes junk is okay

You were asked to prepare a superposition:

$$\sum_k a_k |k\rangle$$

But if its usage is insensitive to phase error, you can prepare this instead:

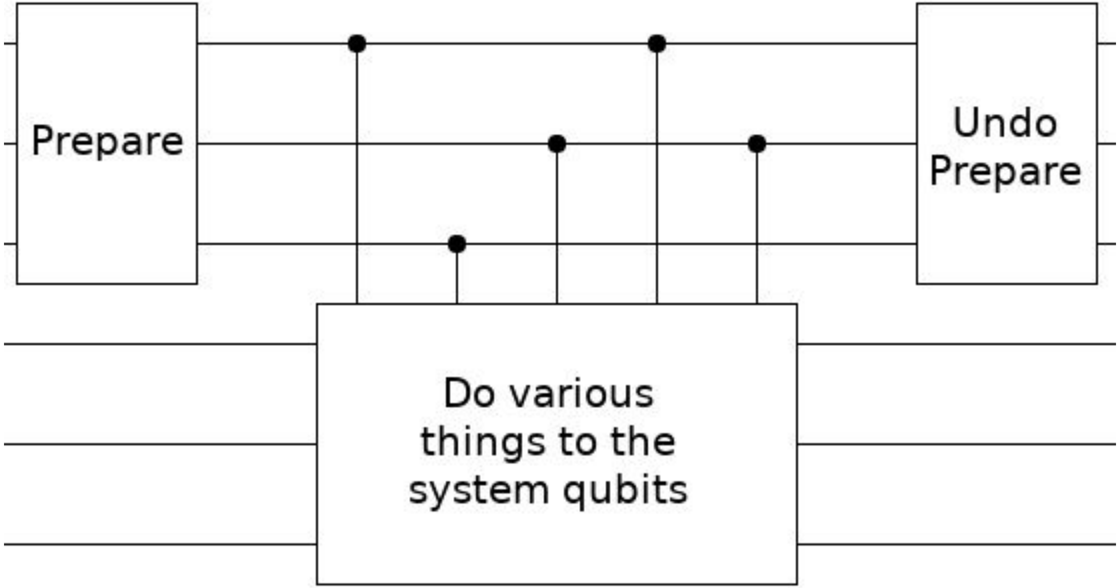
$$\sum_k a_k |k\rangle |\mathrm{temp}_k\rangle$$

i.e. just get the probabilities right:

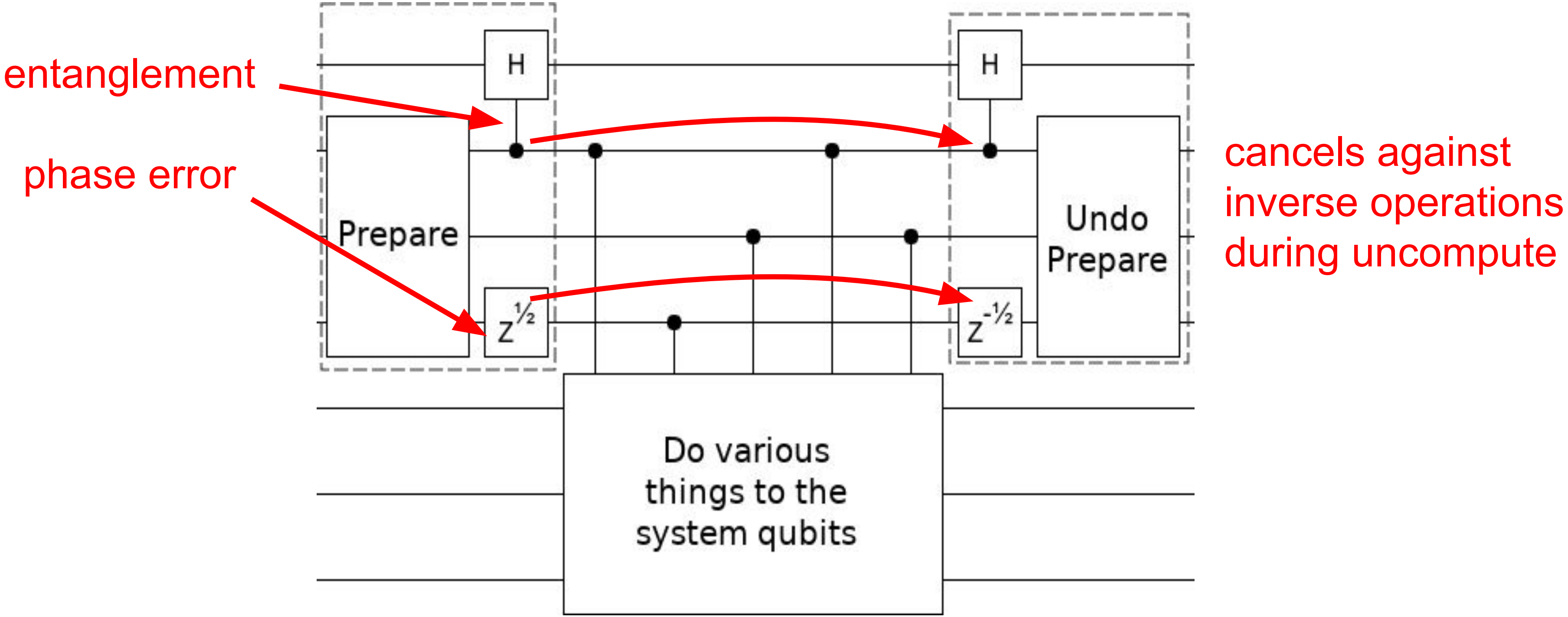
$$\forall k, \langle k|\psi|k\rangle = |a_k|^2$$

Key insight: sometimes junk is okay

Context: prepared superposition is only used as a control



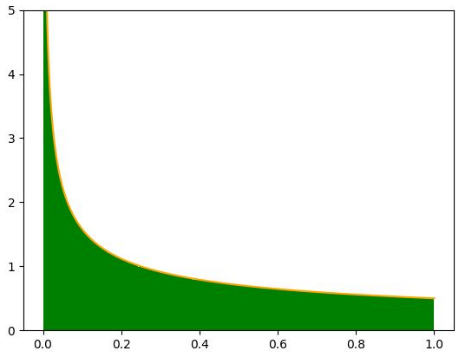
Key insight: sometimes junk is okay



Example: Preparing $\sum_{k=1}^{N} \frac{1}{\sqrt[4]{k}} |k\rangle|\text{temp}_k\rangle$

Step 1: What's the probability distribution?

$$P(k) \propto |a_k|^2 \propto \left|\frac{1}{\sqrt[4]{k}}\right|^2 = \frac{1}{\sqrt{k}}$$



Step 2: Create a classical sampling method.

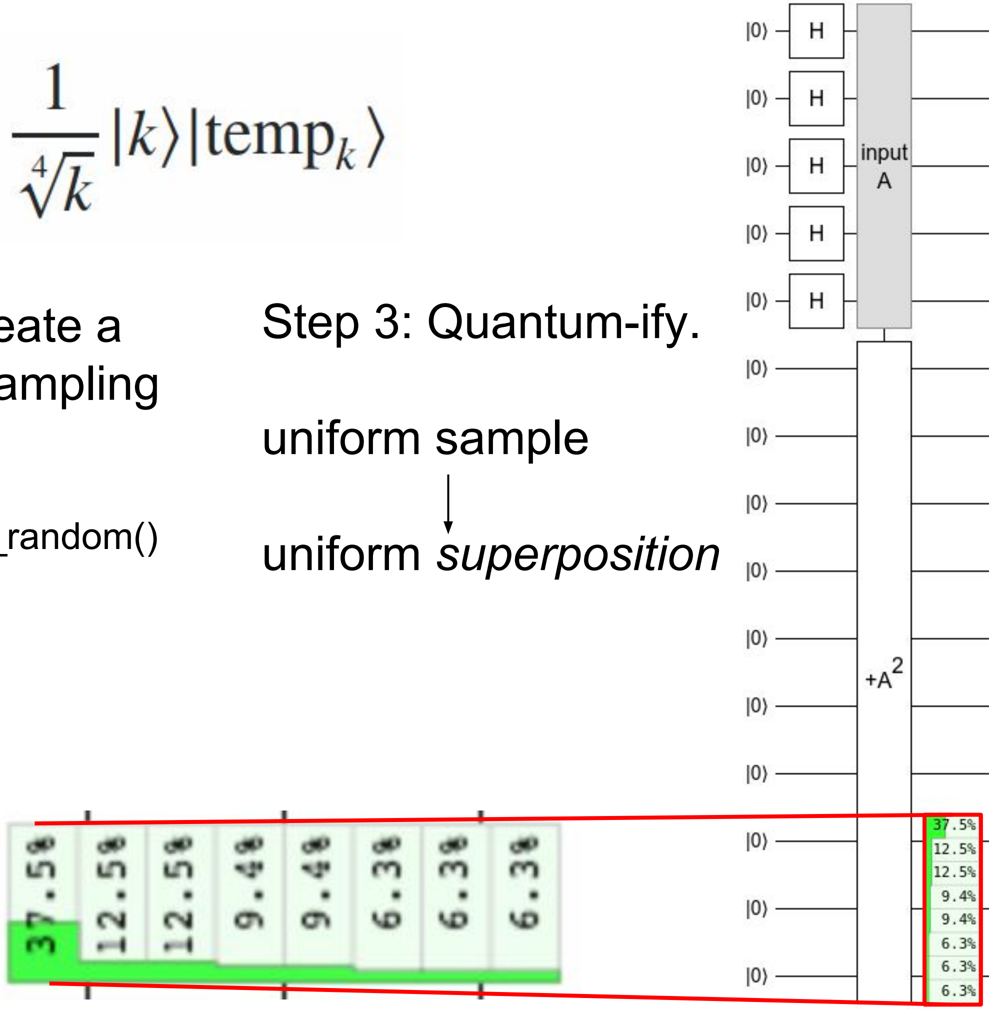
```
u = uniform_random()
return u**2
```

Step 3: Quantum-ify.

uniform sample

↓

uniform *superposition*

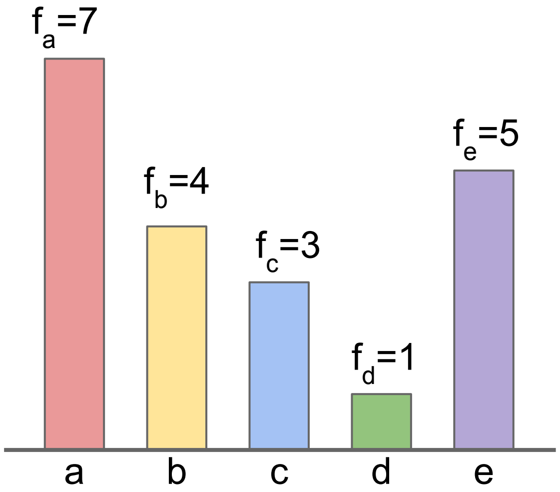


Part 3
Sampling hard-coded probability distributions

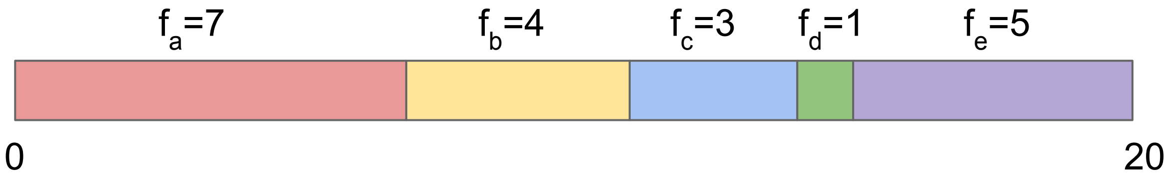
Fitness proportionate selection

Common step in genetic algorithms

Given: a list of items with fitnesses



Goal: sample items with twice as much fitness twice as often

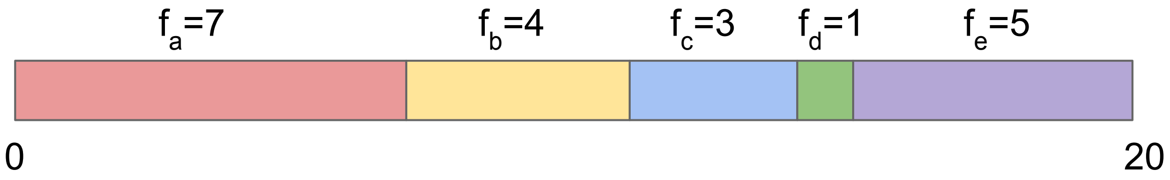


Google

Common Fitness-Proportionate Selection Methods

https://jbn.github.io/fast_proportional_selection/

	Classical Sampling Cost
Linear Walk	O(N)
Bisecting Search	O(lg N)
Stochastic Acceptance	$O(p_{max} N)$



Common Fitness-Proportionate Selection Methods

https://jbn.github.io/fast_proportional_selection/

	Classical Sampling Cost	Quantum Preparation Cost
Linear Walk	$O(N)$	$O(N \lg(1/\varepsilon))$
Bisecting Search	$O(\lg N)$	$O(N \lg(1/\varepsilon))$
Stochastic Acceptance	$O(p_{max} N)$	Not Reversible

Search trees don't help quantum cost. Under superposition, you must do the operations for **every** path.

Common Fitness-Proportionate Selection Methods

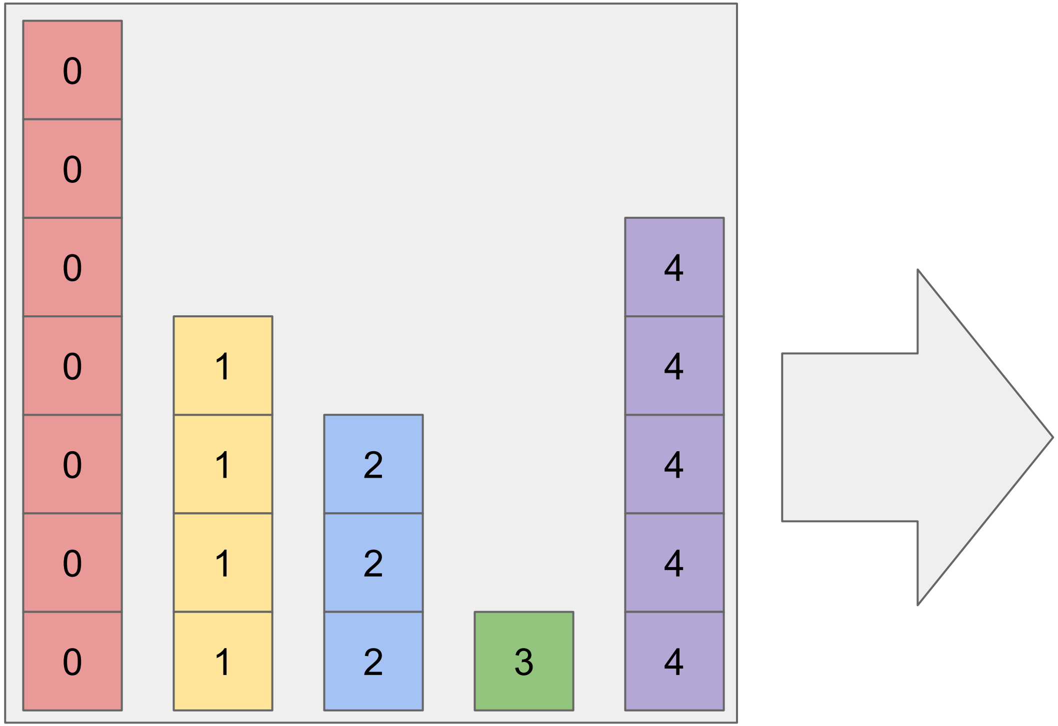
https://jbn.github.io/fast_proportional_selection/

	Classical Sampling Cost	Quantum Preparation Cost
Linear Walk	$O(N)$	$O(N \lg(1/\varepsilon))$
Bisecting Search	$O(\lg N)$	$O(N \lg(1/\varepsilon))$
Stochastic Acceptance	$O(p_{max} N)$	Not Reversible
Alias Sampling*	$O(1)$	$O(N + \lg(1/\varepsilon))$

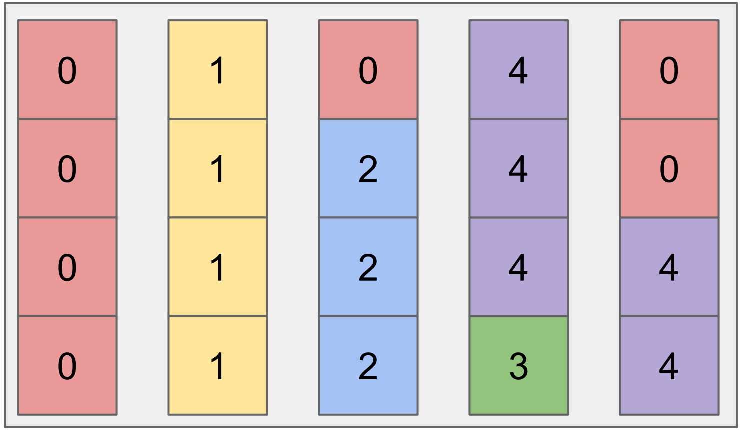
*Walker 1974: "New fast method for generating discrete random numbers with arbitrary frequency distributions"

Google

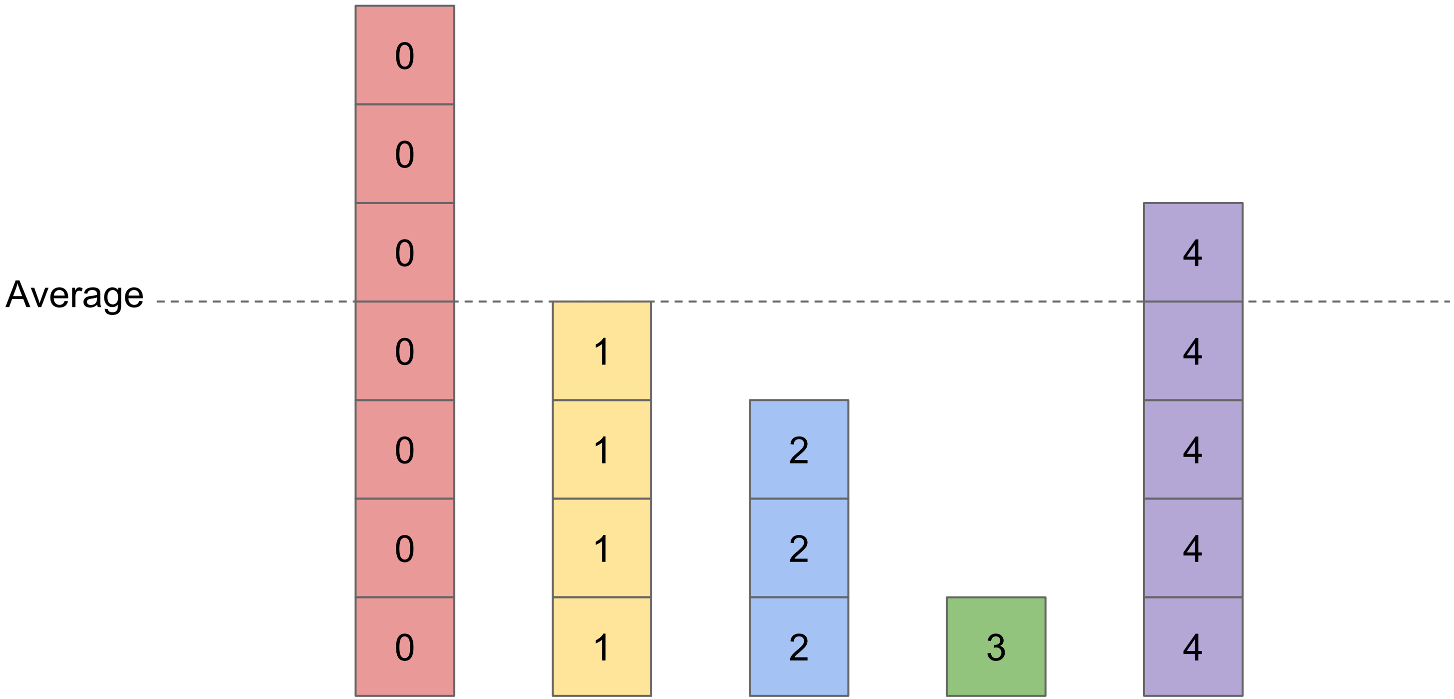
Alias sampling: repacking histograms



Pick initial item uniformly at random, then probabilistically switch to an alternate item.

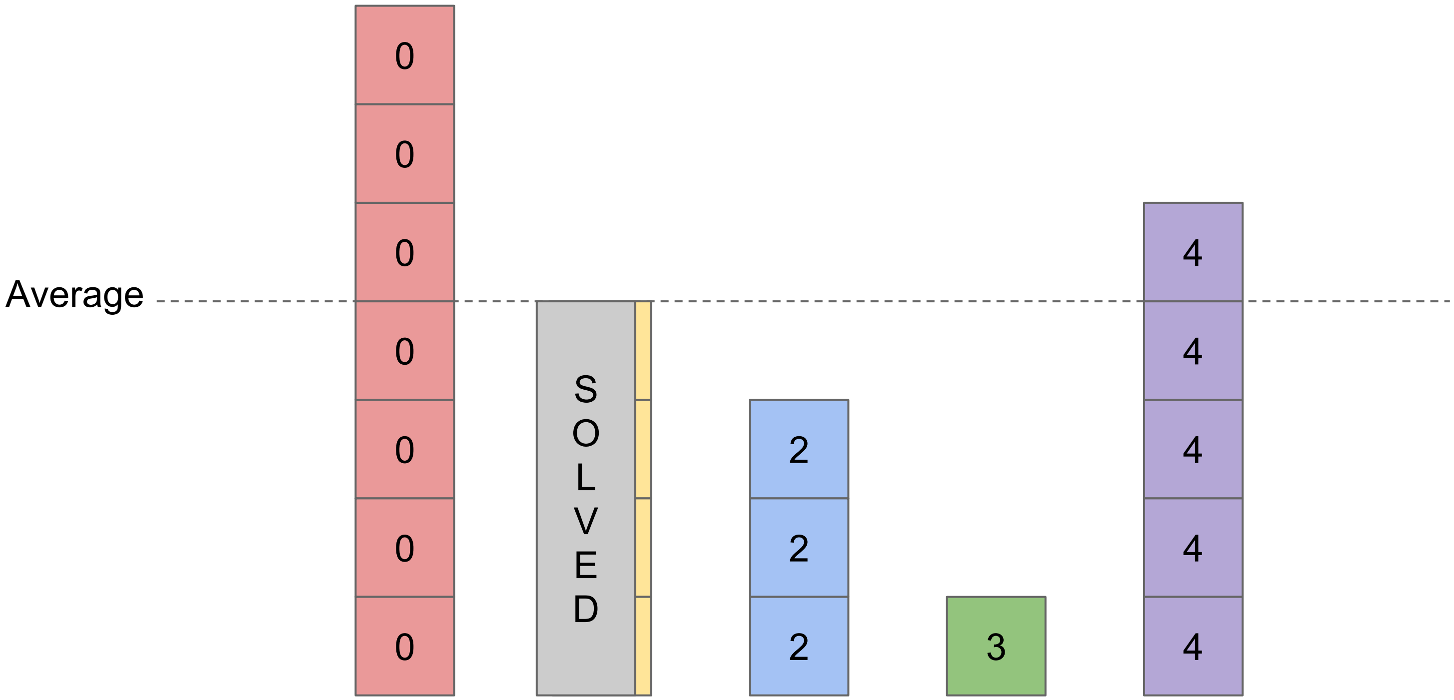


How to repack a histogram

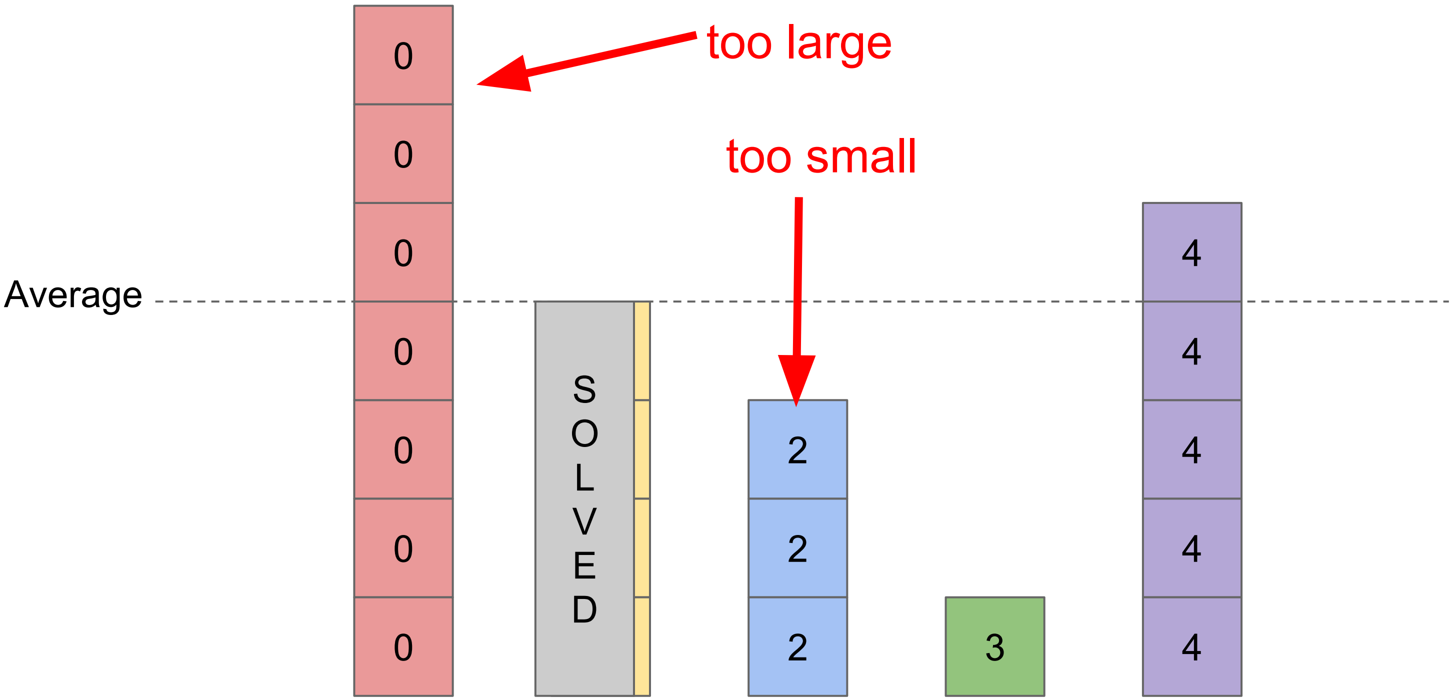


Google

How to repack a histogram

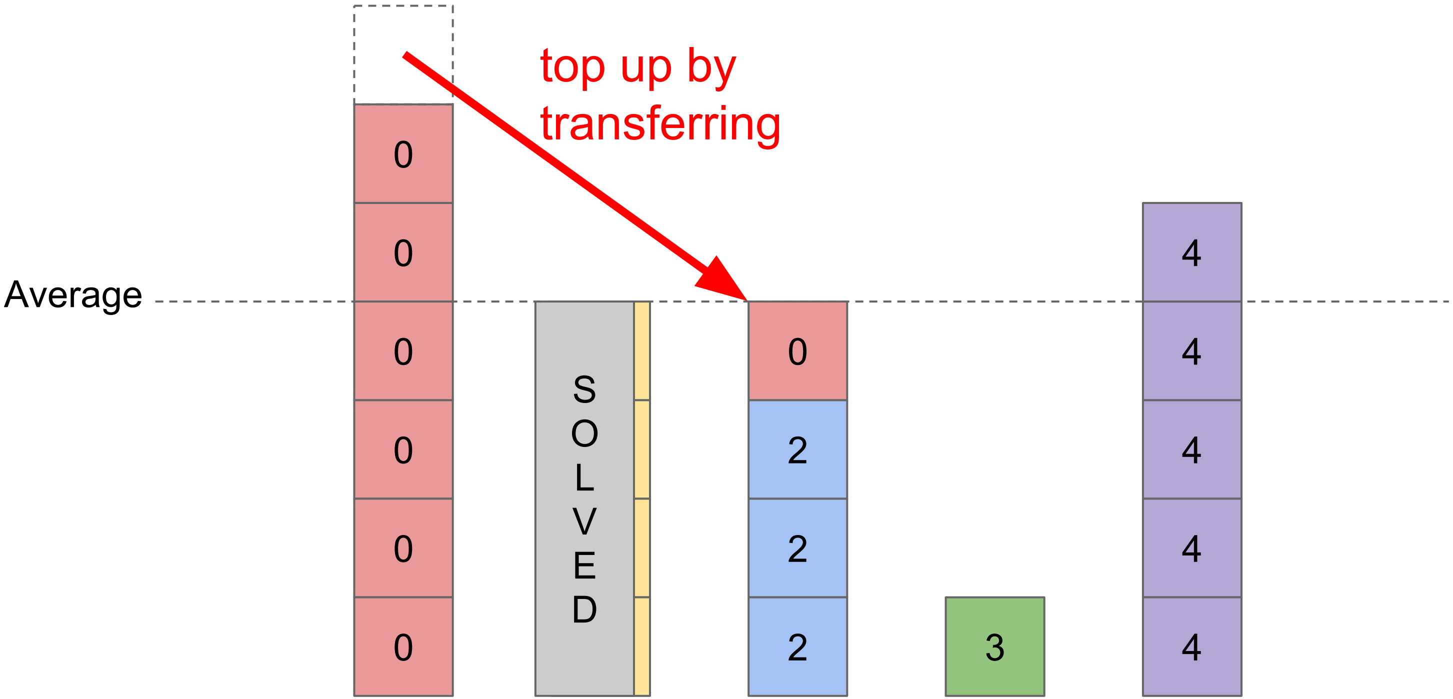


How to repack a histogram



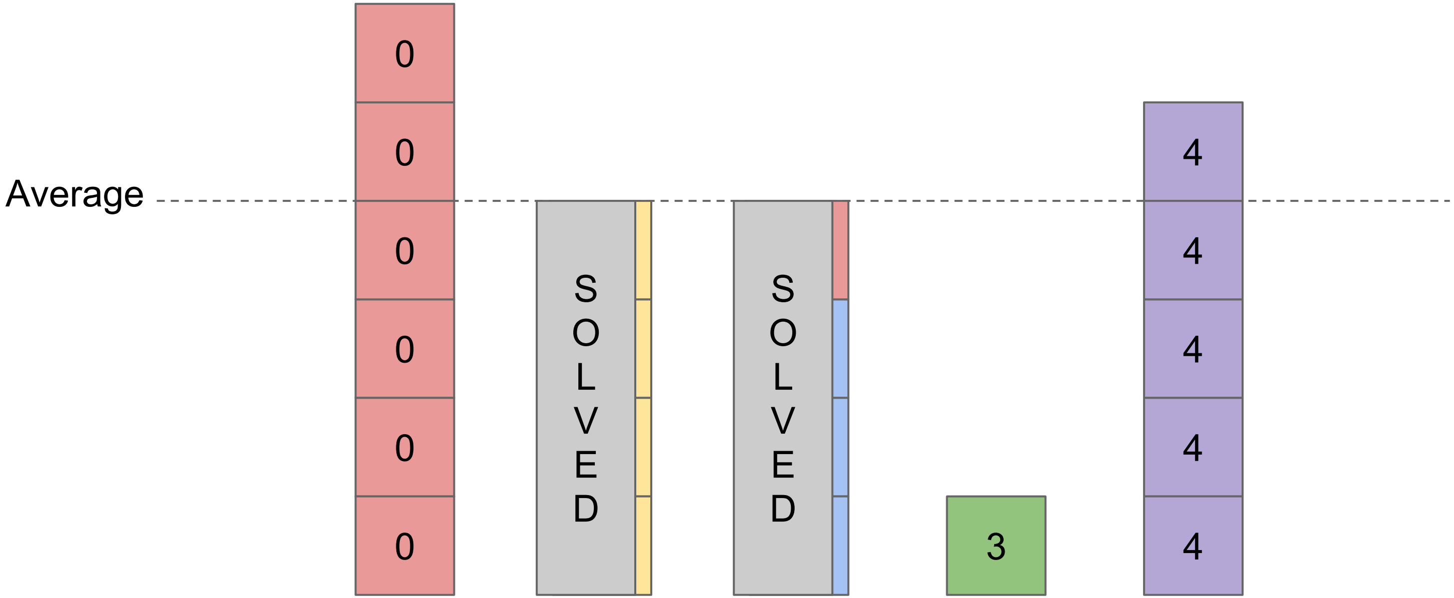
Google

How to repack a histogram



Google

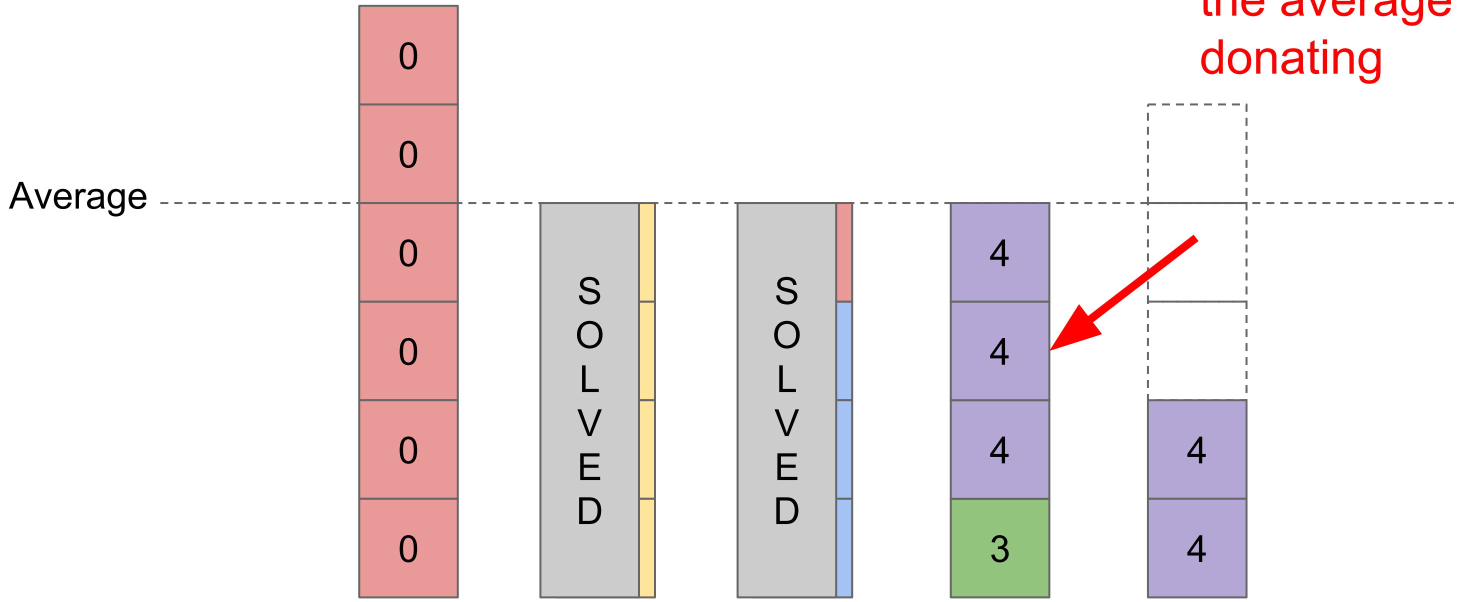
How to repack a histogram



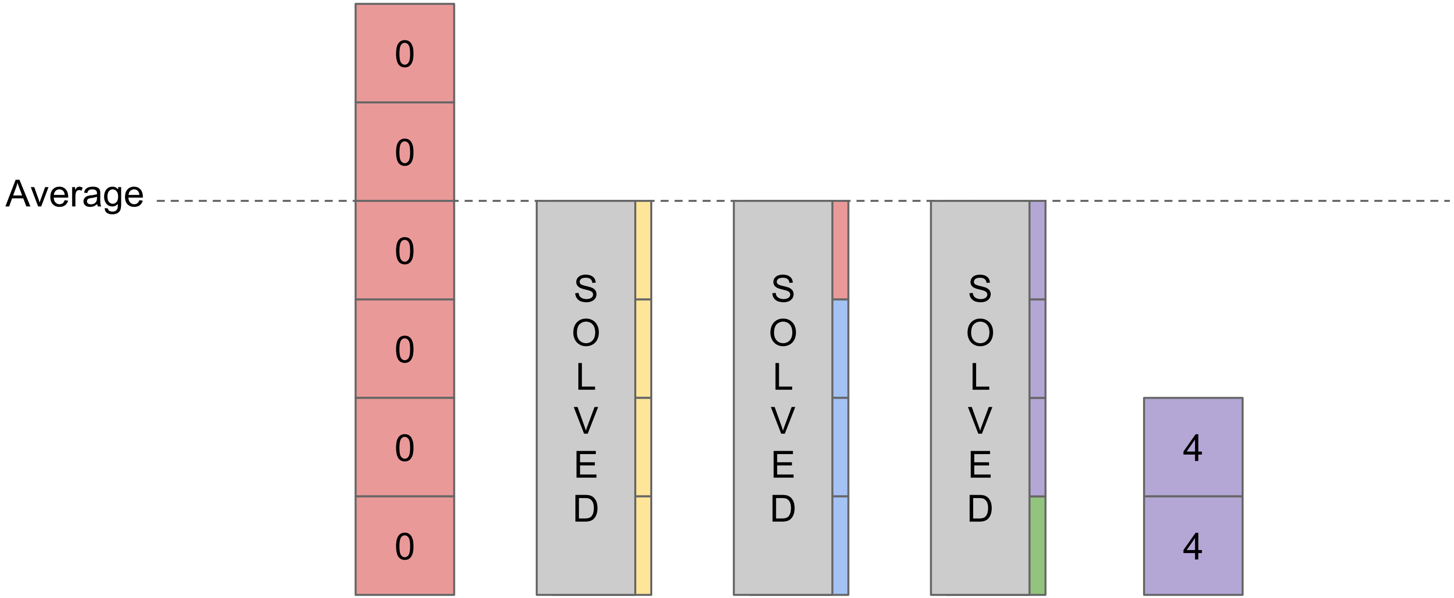
Google

How to repack a histogram

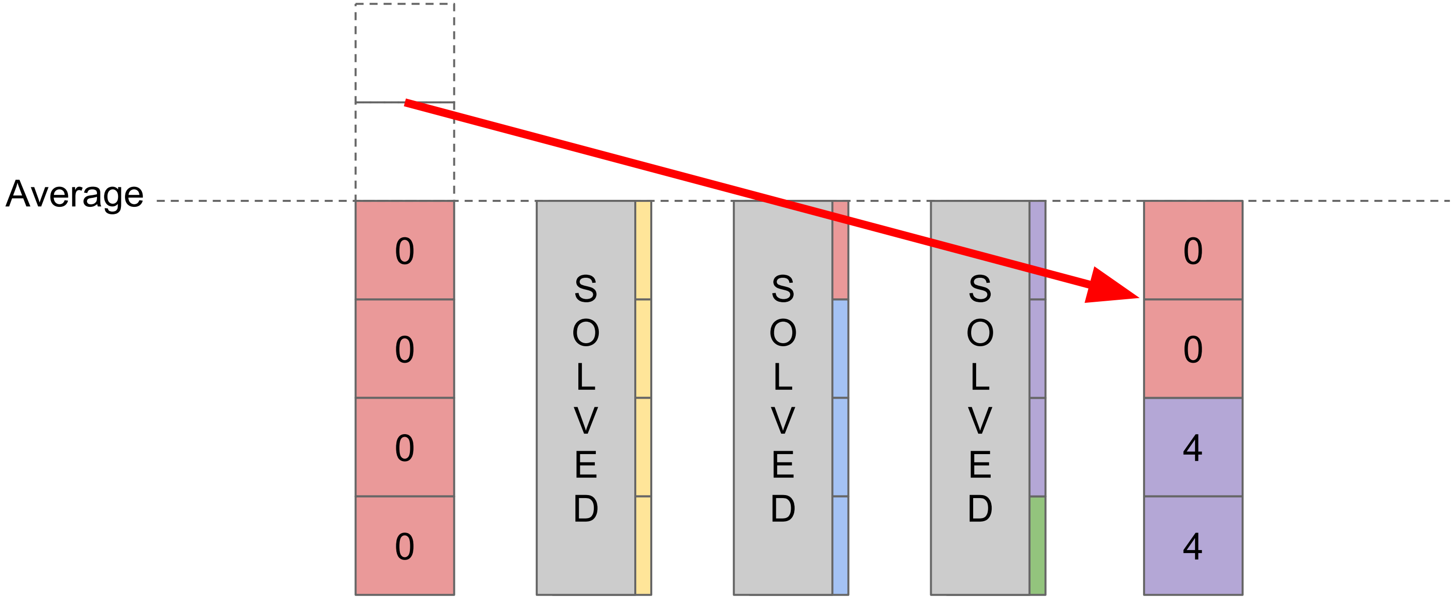
It's okay to undershoot the average when donating



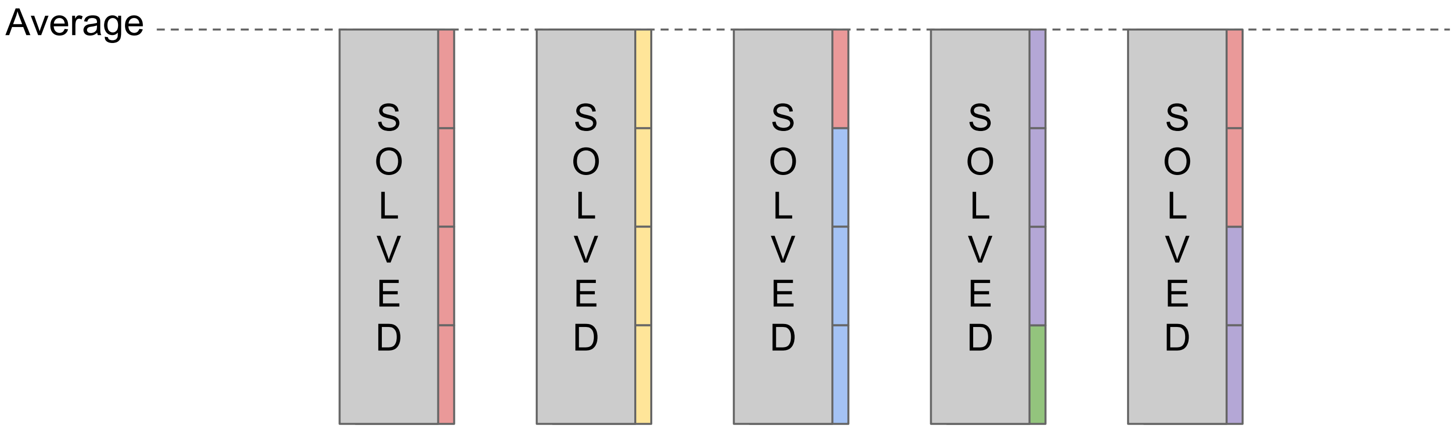
How to repack a histogram



How to repack a histogram



How to repack a histogram

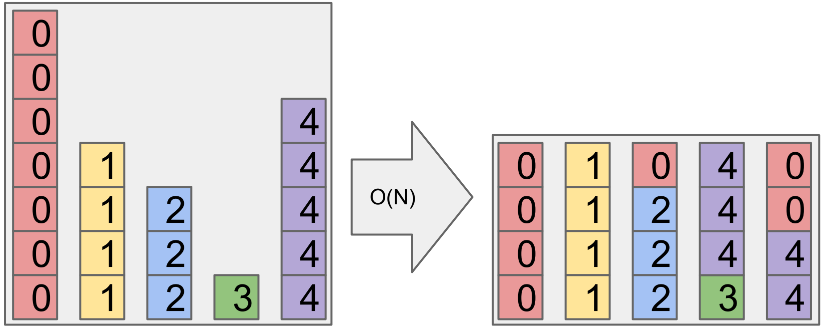


Repacking costs

Linear time using Vose's algorithm

Doesn't affect runtime of quantum algorithm (classically precomputed)

All approximations happen here. Sampling adds *zero* additional error!



Part 4
Putting it all together

Google

Using alias sampling to prepare a superposition

Classical Sampling

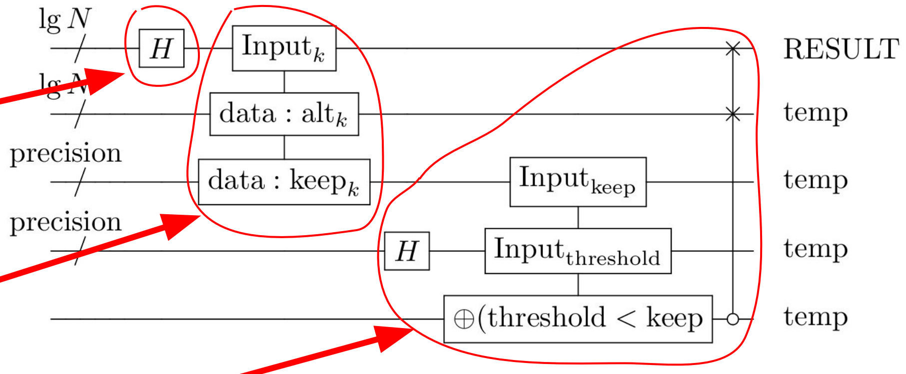
```
def alias_sample(alternates,
                 keep_weights,
                 precision):

    # Pick an item uniformly at random.
    n = len(alternates)
    k = randint(n)

    # Look up alternate item and keep chance.
    alt = alternates[k]
    keep = keep_weights[k]

    # Potentially switch to alternate item.
    threshold = randint(2**precision)
    kept = threshold < keep
    return k if kept else alt
```

Quantum Preparation



Cost of alias preparation

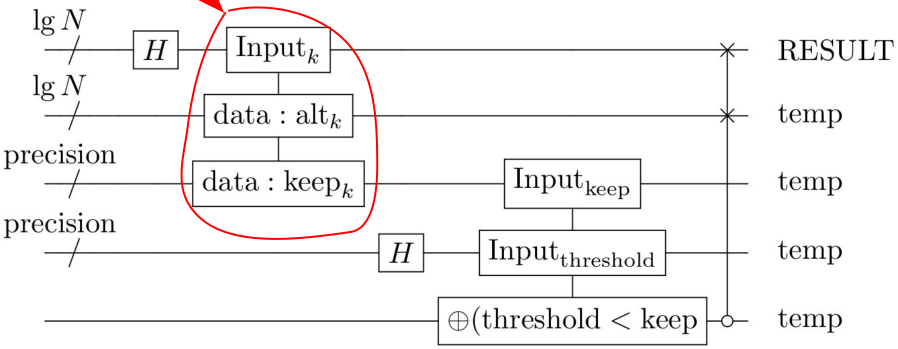
Preparing a uniform superposition costs O(lg N + lg 1/ε)

QROM lookup uses **N-1 AND gates (dominant cost)**

Compare+swap costs O(lg N + lg 1/ε)

Runs at ≈20Hz given N=100.

(an order of magnitude faster)



Part 5
Wrap-up

What we covered: section 3-D of arXiv:1805.03662

Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity

Ryan Babbush,[1,*] Craig Gidney,[2] Dominic W. Berry,[3] Nathan Wiebe,[4]
Jarrod McClean,[1] Alexandru Paler,[5] Austin Fowler,[2] and Hartmut Neven[1]

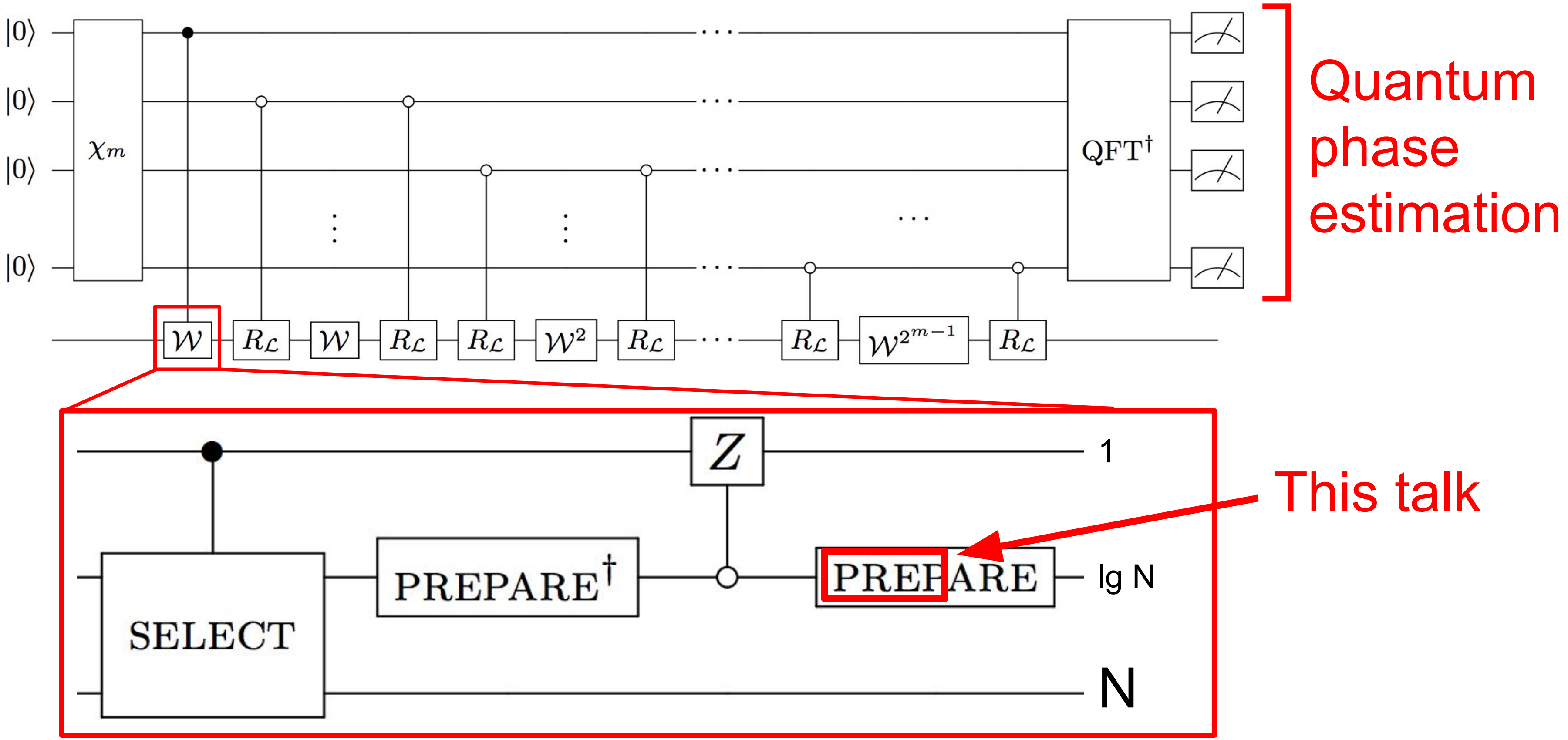
[...]

D. Subsampling the Coefficient Oracle

In this section we introduce a technique for initializing a state with L unique coefficients (provided by a classical database) with a number of T gates scaling as $4L + \mathcal{O}(\log(1/\epsilon))$ where ϵ is the largest absolute error that one can

[...]

Preparation is a small part of a larger algorithm



Google

Estimated costs of the overall algorithm

problem		physical qubits		execution time (hours)	
System	N Spin-Orbitals	$p = 10^{-3}$	$p = 10^{-4}$	$p = 10^{-3}$	$p = 10^{-4}$
Hubbard model	72	1.7×10^6	5.3×10^5	4.6	2.6
Hubbard model	128	2.4×10^6	7.8×10^5	15	8.4
Hubbard model	200	3.8×10^6	1.0×10^6	40	21
Hubbard model	800	1.5×10^7	4.2×10^6	6.7×10^2	3.7×10^2
Electronic structure	54	1.7×10^6	4.7×10^5	0.85	0.44
Electronic structure	128	2.9×10^6	9.5×10^5	10	5.7
Electronic structure	250	5.1×10^6	1.4×10^6	58	30
Electronic structure	1024	2.3×10^7	5.6×10^6	2.8×10^3	1.4×10^3

Contrast with previous work*, which had:
- Execution times in months
- Using hundreds of millions of physical qubits
- Assuming 10 **nanosecond** T gates instead of 150us T gates

*Reiher et al: "Elucidating reaction mechanisms on quantum computers"

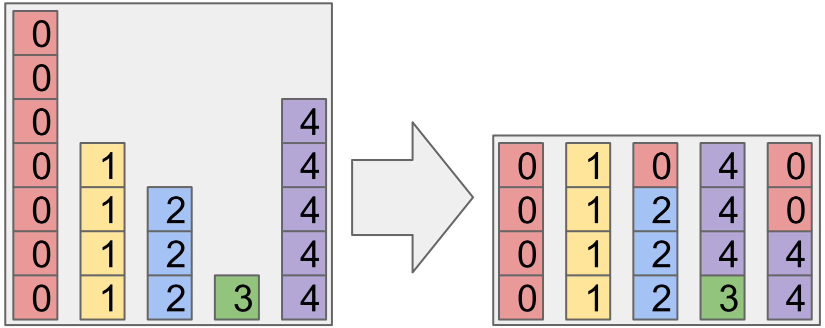
Key Takeaways

- Quantum algorithms start with a constant factor penalty of a billion (if not more).

- When a quantum subroutine is phase-insensitive, try porting classical methods.

- Random sampling methods seem to port particularly well.

- Alias sampling dominates bisecting search sampling yet is less well known.



Thanks for listening!