Title: TBA - Quantum Information Seminar

Speakers: Daniel Gottesman

Collection/Series: Quantum Information

Subject: Quantum Information

Date: April 16, 2025 - 11:00 AM

URL: https://pirsa.org/25040124

Low-Depth Quantum Symmetrization

Daniel Gottesman
Joint work with Zhenning Liu, Andrew Childs
arXiv:2411.04019

Institute for
Robust Quantum
Simulation

Bosonic and Fermionic States

Suppose we have a system consisting of a group of bosons.

- Bosons can be in the same state and are symmetric under interchange of particles.
- Fermions must be in different states and are anti-symmetric under interchange of particles.

Second-quantized representation: # particles in each mode

- Boson example: $|0120\rangle$
- Fermion example: $|0111\rangle$

First-quantized representation: give mode for each particle

- Boson example: $|2\rangle|3\rangle|3\rangle + |3\rangle|2\rangle|3\rangle + |3\rangle|3\rangle|2\rangle$
- Fermion example:

$$|2\rangle|3\rangle|4\rangle - |3\rangle|2\rangle|4\rangle - |4\rangle|3\rangle|2\rangle - |2\rangle|4\rangle|3\rangle + |4\rangle|2\rangle|3\rangle + |3\rangle|4\rangle|2\rangle$$

First or Second Quantized?

Which representation is better? First or second quantization?

Suppose we have n particles in m modes.

First quantized representation:

- Size is $n \log m$ for bosons or fermions
- Useful for computations about each particle separately
- Requires explicit symmetrization or antisymmetrization

Second quantized representation:

- Size is $m \log n$ for bosons or m for fermions
- Useful for computations which involve all particles
- Symmetrization and antisymmetrization are automatic

Both can be useful in certain contexts.

Creating (Anti-)Symmetrized States

Suppose we want to run a quantum simulation in a first-quantized representation. To create the initial state, we might have a list of particle locations, e.g. (2, 3, 3) or (2, 3, 4).

Given this list, our goal is to create the symmetrized or anti-symmetrized states corresponding to these lists:

$$(2,3,3) \mapsto |2\rangle|3\rangle|3\rangle + |3\rangle|2\rangle|3\rangle + |3\rangle|3\rangle|2\rangle$$

$$(2,3,4) \mapsto |2\rangle|3\rangle|4\rangle - |3\rangle|2\rangle|4\rangle - |4\rangle|3\rangle|2\rangle - |2\rangle|4\rangle|3\rangle + |4\rangle|2\rangle|3\rangle + |3\rangle|4\rangle|2\rangle$$

Note:

(Anti-)symmetrization of arbitrary quantum states is a computationally very powerful operation. It is also not reversible.
But: (anti-)symmetrization of a sorted classical list is not as powerful and is reversible.

Strictly Increasing Lists

For fermions, all numbers in the list must be different. If we assume the initial list is sorted, it is a strictly increasing list (SIL). We can also have a SIL for bosons.

Start with SIL $l = (l_1, l_2, \ldots, l_n)$ with $l_i < l_{i+1}$.

We wish to create the state

$$\sum_{\sigma \in S_n} |\sigma(l)\rangle \text{ (for bosons)}$$

or

$$\sum_{\sigma \in S_n} (-1)^{\sigma} |\sigma(l)\rangle \text{ (for fermions)}$$

We sum over all permutations for a SIL. Our goal is to do this in polylog (n) depth.

This is reversible since given any $|\sigma(l)\rangle$, we can sort it to recover $|l\rangle$.

Sorting, Unsorting, etc.

Sorting is going to play a central role in this algorithm.

A classical reversible sorting algorithm SORT does the following:

$$SORT : |\sigma(l)\rangle|0\rangle \mapsto |l\rangle|rec(\sigma)\rangle \text{ with } l \text{ a SIL.}$$

It leaves behind in an ancilla register a record $rec(\sigma)$ of the original order, which just depends on the permutation σ.

We can also do the inverse UNSORT:

$$UNSORT : |l\rangle|rec(\sigma)\rangle \mapsto |\sigma(l)\rangle|0\rangle.$$

Other options:

$$SHUFFLE : |\sigma(l)\rangle|rec(\tau)\rangle \mapsto |\tau(\sigma(l))\rangle|rec(\tau)\rangle.$$

$$UNSHUFFLE : |\sigma(l)\rangle|rec(\tau)\rangle \mapsto |\tau^{-1}(\sigma(l))\rangle|rec(\tau)\rangle.$$

Inverse Symmetrization

Let us focus first on the symmetrization of an SIL.

If we SORT the output state of symmetrization, what do we get?

$$SORT : \sum_{\sigma} |\sigma(l)\rangle |0\rangle \mapsto \sum_{\sigma} |l\rangle |rec(\sigma)\rangle = |l\rangle \sum_{\sigma} |rec(\sigma)\rangle$$

Note that the ancilla $\sum_{\sigma} |rec(\sigma)\rangle$ has no dependence on l. So we can creating this ancilla state and then apply UNSORT with it to our SIL:

$$UNSORT : |l\rangle \sum_{\sigma} |rec(\sigma)\rangle \mapsto \sum_{\sigma} |\sigma(l)\rangle |0\rangle$$

Superposition of Permutations

How can we create the ancilla state?

Since it doesn't depend on the actual list, we can create it by starting with a superposition of random unsorted lists.

$$\sum_{r_1} |r_1\rangle \sum_{r_2} |r_2\rangle \sum_{r_3} |r_3\rangle \ldots |0\rangle \approx \sum_{r} \sum_{\sigma} |\sigma(r)\rangle$$

Here r is an SIL with random entries. Why is this only approximate? If $r_i = r_j$, then sorting won't give us an SIL. But if we take each sum over r_i to be over a large range of numbers, then the chance of this is small and the superposition is close to one over only permutations of SILs only.

$$SORT : \sum_{r} \sum_{\sigma} |\sigma(r)\rangle |0\rangle \mapsto \sum_{r} |r\rangle \sum_{\sigma} |rec(\sigma)\rangle$$

Discard the first register to create the desired ancilla.

[Berry et al. 2018]

Symmetrizing an SIL Summarized

1. Create superpositions of all numbers over a wide range.
2. Check that all numbers are different (if not, redo step 1).
3. Apply SORT to this superposition of random lists.
4. Discard the sorted list and keep the ancilla with the sorting record.
5. Apply UNSORT to the SIL input and the ancilla.

Steps 1, 2, and 4 are low depth. If we use a low-depth classical sorting algorithm, steps 3 and 5 are also low depth.

For antisymmetrization, just insert a step between 4 and 5 to apply a phase $(-1)^{\sigma}$ (i.e., ± 1 depending if the permutation is even or odd).

Non-Strictly Increasing Lists

But for bosons, there is the possibility that two or more particles are in the same mode. We can still assume the list of particle positions is sorted initially but now the list is a non-strictly increasing list (NSIL):

NSIL $l = (l_1, l_2, \ldots, l_n)$ has $l_i \leq l_{i+1}$.

The Berry et al. algorithm doesn't work any more, because UNSORT is no longer reversible:

$$UNSORT : |l\rangle |rec(\sigma)\rangle \mapsto |\sigma(l)\rangle |0\rangle.$$

Different permutations can produce the same output if they permute two identical l_is.

We can fix up the algorithm by creating an ancilla which separates out the permutations that leave l invariant. But this still doesn't work if we have a superposition over different lists l since the ancilla retains information about the repetition pattern of l.

Lower Exceeding Sequences

We need another method to create the correct ancilla. We take advantage of an equivalence between permutations and lower exceeding sequences:

> A lower exceeding sequence (LES) is a sequence of n numbers $s_i \in \{1,\ldots,n\}$ such that $s_i \leq i$.

Examples:

- (1,2,2), (1,1,2), and (1,2,1) are LESs.
- (2,1,2), (1,3,2), and (1,2,4) are not LESs.

Notes:

- The number of LESs is $n!$, the same as the number of permutations.
- Generating a superposition of all LESs is straightforward.

[Alonso and Schott, 1996]

Permutation to LES

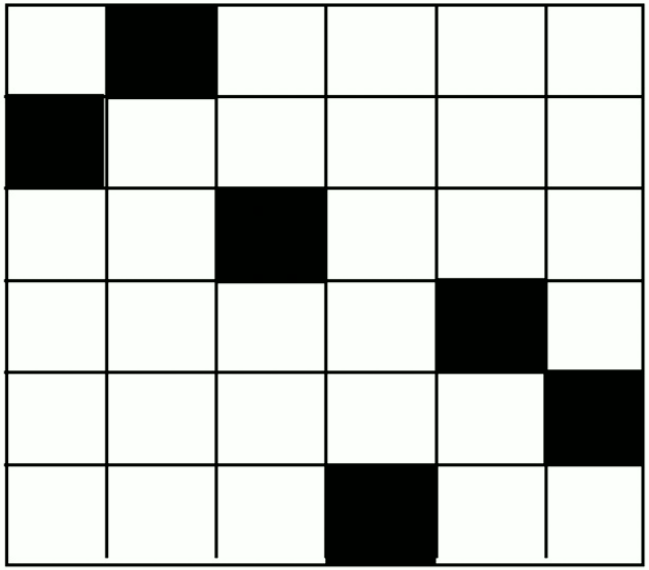
Given a permutation σ, we can represent it as a sequence $\sigma(12\ldots n)$.

E.g.: $\sigma(123456) = 465312$

Color box $\sigma(i)$ in the ith row

The jth element s_j of the LES is the # of filled boxes between the bottom left and the filled box in the jth column (inclusive).

In the example, the LES is:



Note: This will always be an LES since there can be at most j filled boxes up to column j.

Permutation to LES

Given a permutation σ, we can represent it as a sequence $\sigma(12\ldots n)$.

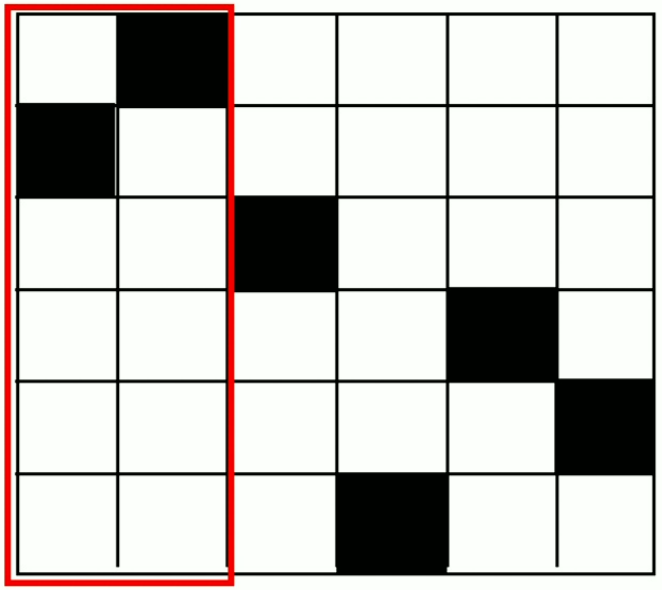
E.g.: $\sigma(123456) = 465312$

Color box $\sigma(i)$ in the ith row

The jth element s_j of the LES is the # of filled boxes between the bottom left and the filled box in the jth column (inclusive).

In the example, the LES is:

1 2



Note: This will always be an LES since there can be at most j filled boxes up to column j.

Permutation to LES

Given a permutation σ, we can represent it as a sequence $\sigma(12\ldots n)$.

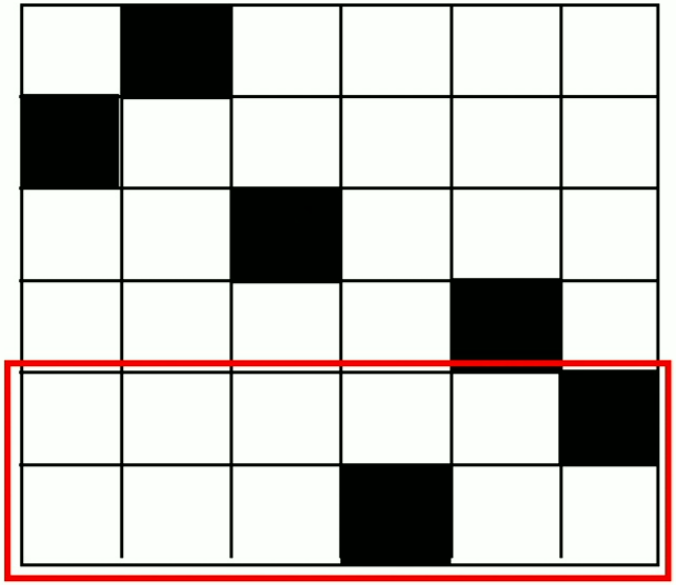
E.g.: $\sigma(123456) = 465312$

Color box $\sigma(i)$ in the ith row

The jth element s_j of the LES is the # of filled boxes between the bottom left and the filled box in the jth column (inclusive).

In the example, the LES is:

1 2 1 1 2 2

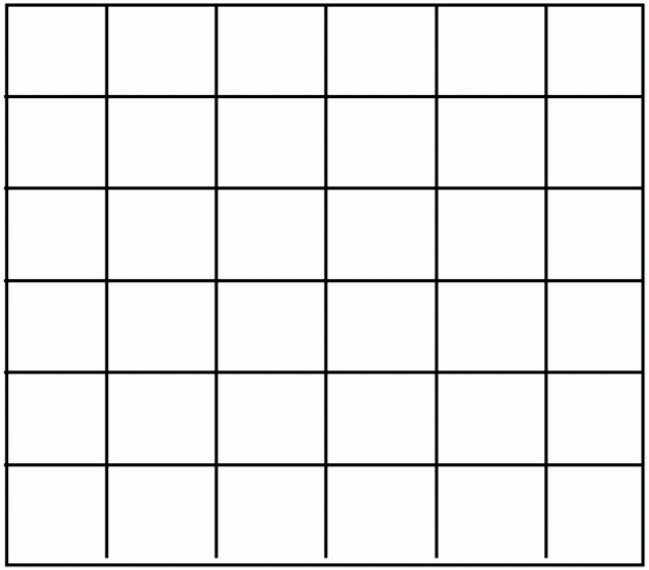


Note: This will always be an LES since there can be at most j filled boxes up to column j.

LES to Permutation

Given an LES, fill in boxes starting with the rightmost column. For column j, color the box in row s_j not counting any already-filled rows.

Example: LES (1, 2, 1, 1, 2, 2)



Note: This will always produce a permutation and the same one that gives this LES.

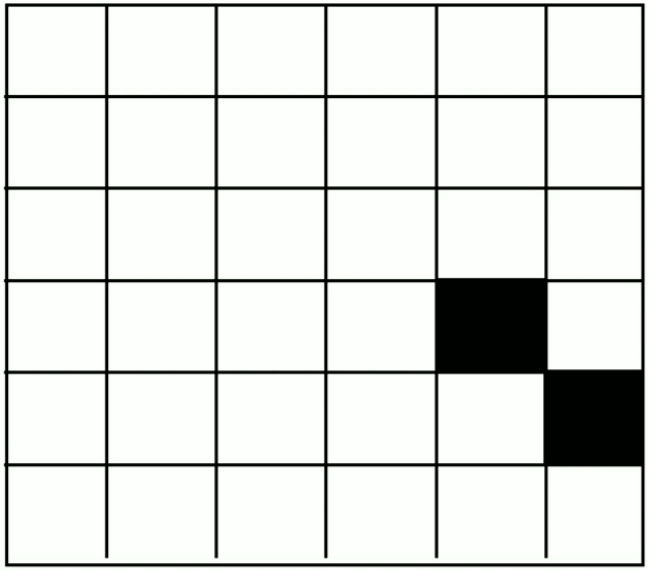
LES to Permutation

Given an LES, fill in boxes starting with the rightmost column. For column j, color the box in row s_j not counting any already-filled rows.

Example: LES (1, 2, 1, 1, 2, 2)

$s_6 = 2$, so row 2

$s_5 = 2$, so row 3 (1 filled below already)



Note: This will always produce a permutation and the same one that gives this LES.

LES to Permutation

Given an LES, fill in boxes starting with the rightmost column. For column j, color the box in row s_j not counting any already-filled rows.

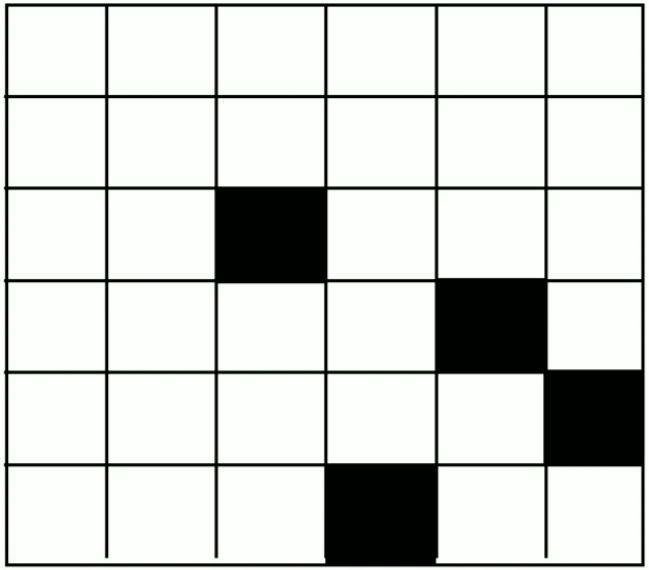
Example: LES $(1, 2, 1, 1, 2, 2)$

$s_6 = 2$, so row 2

$s_5 = 2$, so row 3 (1 filled below already)

$s_4 = 1$, so row 1

$s_3 = 1$, so row 4 (since first three rows already filled)



Note: This will always produce a permutation and the same one that gives this LES.

LES to Permutation

Given an LES, fill in boxes starting with the rightmost column. For column j, color the box in row s_j not counting any already-filled rows.

Example: LES $(1, 2, 1, 1, 2, 2)$

$s_6 = 2$, so row 2

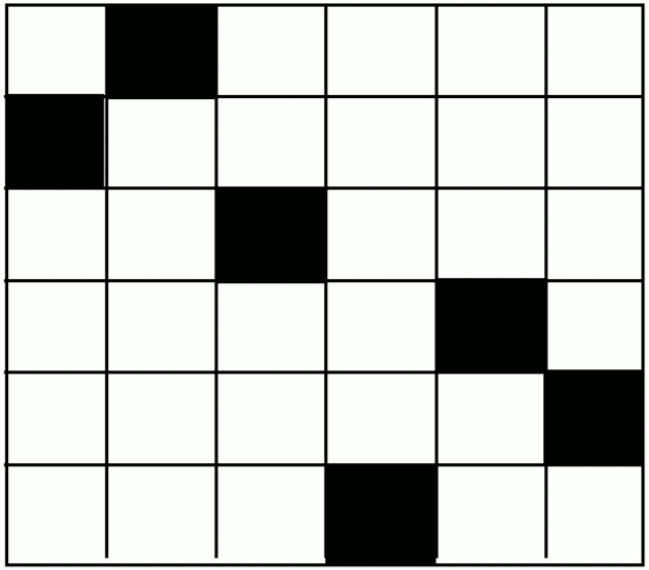
$s_5 = 2$, so row 3 (1 filled below already)

$s_4 = 1$, so row 1

$s_3 = 1$, so row 4 (since first three rows already filled)

$s_2 = 2$, so row 6

$s_4 = 1$, so row 1



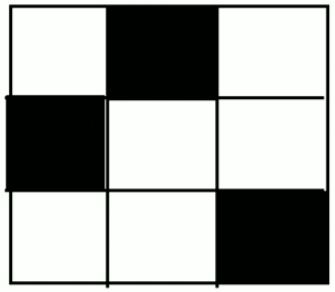
Note: This will always produce a permutation and the same one that gives this LES.

Low-Depth LES to Permutation

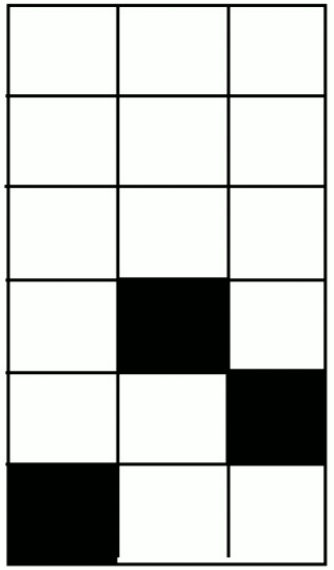
To convert an LES to a permutation in polylog depth, divide and conquer: Build the left and right halves of the diagram separately and then merge them.

Merge by inserting rows of left half into empty rows of right half.

This can be done via a sorting algorithm: sort rows, with ith row from the left after jth row from right if $i + f_j \geq j$, where f_j is the number of filled rows on right below row j.



(1,2,1)



(1,2,2)

Example: LES (1, 2, 1, 1, 2, 2)

[Irreversible: Alonso and Schott 1996, Reversible: new]

Permutations of NSILs

If we have an NSIL, there will be some permutations that leave the list l invariant. These permutations form a subgroup H_l.

E.g.: for list $l = (2,3,3)$, the subgroup $H_l = \{e, (23)\}$ (i.e., the identity plus the swap of the second and third list elements).

> An arbitrary permutation σ can be written as $\sigma' h$, where $h \in H_l$ and σ' is a coset representative of H_l.

The subgroup H_l is equivalent to subset of LESs with the following properties:

- If $l_{i-1} < l_i < l_{i+1}$ (list increasing) then the LES element $s_i = i$.
- If $l_{a-1} < l_a = l_{a+1} = \cdots = l_b < l_{b+1}$ (list constant) then $a \leq s_i \leq i$ for $a \leq i \leq b$.

Thus, given a pattern of repetitions within a list, we can generate a superposition of LESs that correspond to all elements of H_l.

Symmetrization of NSILs

1. Generate a superposition of all LESs, convert to perm.: $\sum_{\sigma} |\sigma(1\ldots n)\rangle$
2. SORT: $\sum_{\sigma} |rec(\sigma)\rangle\,|1\ldots n\rangle$
3. SHUFFLE: $|l\rangle \sum_{\sigma} |rec(\sigma)\rangle\,|1\ldots n\rangle \mapsto \sum_{\sigma} |\sigma(l)\rangle\,|rec(\sigma)\rangle\,|1\ldots n\rangle$
4. UNSORT on ancilla: $\sum_{\sigma} |\sigma(l)\rangle\,|0\rangle\,|\sigma(1\ldots n)\rangle$
5. Decompose $\sigma = \sigma' h$: $\sum_{\sigma'} |\sigma'(l)\rangle\,|0\rangle \sum_{h\in H_l} |\sigma' h(1\ldots n)\rangle$
6. SORT list l: $\sum_{\sigma'} |l\rangle\,|rec(\sigma')\rangle \sum_{h\in H_l} |\sigma' h(1\ldots n)\rangle$
7. UNSHUFFLE ancilla: $\sum_{\sigma'} |l\rangle\,|rec(\sigma')\rangle \sum_{h\in H_l} |h(1\ldots n)\rangle$
8. UNSORT: $\sum_{\sigma'} |\sigma'(l)\rangle\,|0\rangle \sum_{h\in H_l} |h(1\ldots n)\rangle$

Symmetrization of NSILs

1. Generate a superposition of all LESs, convert to perm.: $\sum_{\sigma} |\sigma(1\ldots n)\rangle$
2. SORT: $\sum_{\sigma} |rec(\sigma)\rangle\,|1\ldots n\rangle$
3. SHUFFLE: $|l\rangle \sum_{\sigma} |rec(\sigma)\rangle\,|1\ldots n\rangle \mapsto \sum_{\sigma} |\sigma(l)\rangle\,|rec(\sigma)\rangle\,|1\ldots n\rangle$
4. UNSORT on ancilla: $\sum_{\sigma} |\sigma(l)\rangle\,|0\rangle\,|\sigma(1\ldots n)\rangle$
5. Decompose $\sigma = \sigma' h$: $\sum_{\sigma'} |\sigma'(l)\rangle\,|0\rangle \sum_{h\in H_l} |\sigma' h(1\ldots n)\rangle$
6. SORT list l: $\sum_{\sigma'} |l\rangle\,|rec(\sigma')\rangle \sum_{h\in H_l} |\sigma' h(1\ldots n)\rangle$
7. UNSHUFFLE ancilla: $\sum_{\sigma'} |l\rangle\,|rec(\sigma')\rangle \sum_{h\in H_l} |h(1\ldots n)\rangle$
8. UNSORT: $\sum_{\sigma'} |\sigma'(l)\rangle\,|0\rangle \sum_{h\in H_l} |h(1\ldots n)\rangle$
9. Convert to LES, uncreate the superposition: $\sum_{\sigma'} |\sigma'(l)\rangle\,|0\rangle\,|0\rangle$

Applications

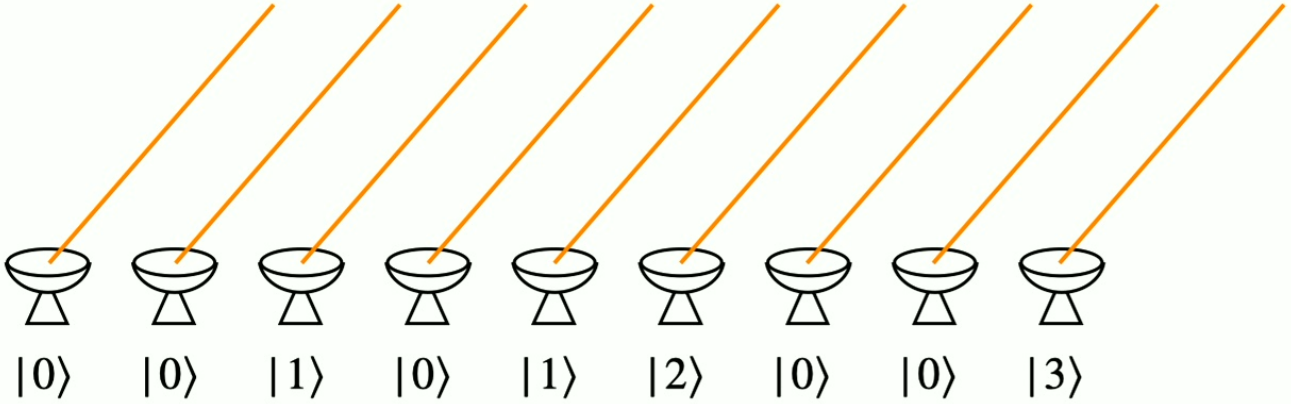
- Creating Dicke states: $\sum_{|r|=k} |r\rangle$

  There are other ways of creating Dicke states, although ours is better in some ways than many.

- Creating initial state for a quantum simulation in first-quantized representation. Advantageous for many modes and few particles.

- Arrays of quantum telescopes collect photons and bring them together to do a quantum computation. Collect light in second-quantized representation but computation may be advantageous in first-quantized representation.

Telescope Arrays



An array of telescopes collect light; we can imagine capturing individual photons in a quantum memory. With many telescopes, we may get many photons at the same time.

This naturally leads to a second-quantized representation, with n_i photons arriving at telescope i.

Fourier Transform

If we have a single photon arriving at a time and we want to do interferometry to reconstruct the direction of the photon, we should do the Fourier transform.

> First we convert the state (with 1 photon at telescope i and 0 photons elsewhere) to the representation $|i\rangle$ (the photon is at telescope i).

But really it is arriving at a superposition of locations:

$$\sum_j e^{2\pi i d j \sin\theta/\lambda} |j\rangle$$

(d distance between telecopes, λ wavelength of light)

The quantum Fourier transform will convert this to

$$\sum_{v\in\mathbb{Z}} |md \sin\theta/\lambda + mv\rangle$$

(m number of telecopes)

which lets us determine the direction θ the photon is coming from with precision $\lambda/(md)$.

Multiple Photons

If we have multiple photons, we would like to do the Fourier transform on each one individually.

This would use the state

$$\left(\sum_{j_1} e^{2\pi i d j_1 \sin\theta_1/\lambda}|j_1\rangle\right) \otimes \left(\sum_{j_2} e^{2\pi i d j_2 \sin\theta_2/\lambda}|j_2\rangle\right) \otimes \left(\sum_{j_3} e^{2\pi i d j_3 \sin\theta_3/\lambda}|j_3\rangle\right) \otimes \cdots$$

(assuming photon i comes from direction θ_i).

But actually, this is not the correct state because of Bose statistics, which give a different combinatoric factor to (unsorted) lists $(j_1, j_2, \ldots, j_m)$ with repetitions in them.

Moreover, this a first-quantized representation and the telescope array collects a second-quantized representation. How can we change between them?

Second to First Quantization

Suppose we have a second-quantized representation $|a_1 a_2 \ldots a_m\rangle$.

- We can convert this directly into an NSIL by stepping through i and writing i to the list a_i times.
- Then we use the symmetrization algorithm on the NSIL.

The output is the correct symmetrized state in first-quantized representation.

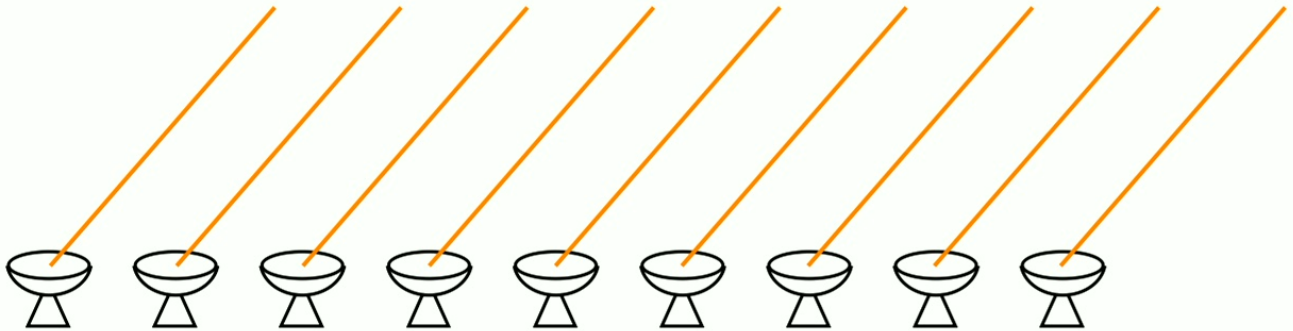
If we now perform a quantum Fourier transform on each photon separately, the combinatoric factor works out correctly and we get the output (suppressing scaling factors within the kets):

$$\sum_{\sigma} |\theta_{\sigma(1)}\rangle |\theta_{\sigma(2)}\rangle |\theta_{\sigma(3)}\rangle \cdots$$

Measurement gives us a list (in random order) of angles from which the photons are coming with precision $\lambda/(md)$.

Summary

- We have presented an algorithm to symmetrize a sorted list of n numbers in depth O(polylog n).
- It works coherently on superpositions of sorted lists with no garbage left over.
- It can convert from second-quantized representation to first-quantized representation (and the inverse algorithm converts the other way).
- Applications include creating Dicke states, preparing states for simulations of bosons, and processing quantum data received in quantum telescope arrays.



Permutation to LES

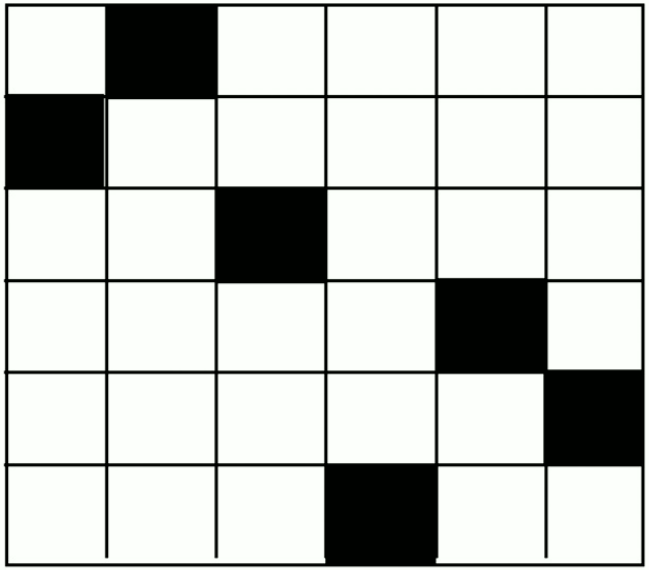
Given a permutation σ, we can represent it as a sequence $\sigma(12\ldots n)$.

E.g.: $\sigma(123456) = 465312$

Color box $\sigma(i)$ in the ith row

The jth element s_j of the LES is the # of filled boxes between the bottom left and the filled box in the jth column (inclusive).

In the example, the LES is:



Note: This will always be an LES since there can be at most j filled boxes up to column j.

Superposition of Permutations

How can we create the ancilla state?

Since it doesn't depend on the actual list, we can create it by starting with a superposition of random unsorted lists.

$$\sum_{r_1} |r_1\rangle \sum_{r_2} |r_2\rangle \sum_{r_3} |r_3\rangle \ldots |0\rangle \approx \sum_{r} \sum_{\sigma} |\sigma(r)\rangle$$

Here r is an SIL with random entries. Why is this only approximate? If $r_i = r_j$, then sorting won't give us an SIL. But if we take each sum over r_i to be over a large range of numbers, then the chance of this is small and the superposition is close to one over only permutations of SILs only.

$$SORT : \sum_{r} \sum_{\sigma} |\sigma(r)\rangle |0\rangle \mapsto \sum_{r} |r\rangle \sum_{\sigma} |rec(\sigma)\rangle$$

Discard the first register to create the desired ancilla.

[Berry et al. 2018]