PERMANENT-BASED APPROACH TO THE FINAL PROBLEM OF IBM FALL 2020 QUANTUM CHALLENGE.

JAN TUŁOWIECKI

BEIT
j.tulowiecki[at]beit.tech

Abstract. I introduce a permanent-based approach to the final problem of IBM Fall 2020 Quantum Challenge and prove its correctness. I describe a way to implement it as a quantum circuit and discuss the combinatorial consequences of such approach. The optimized solution obtained the score of 6574 and the 7th place in the final challenge ranking.

1. Problem description

A square $n \times n$ board is given with m stars placed on it ($0 \leq m \leq n^2$). We call the board *solvable*, if and only if there exists a selection of k rows and $n-k-1$ columns ($0 \leq k \leq n-1$) such that each star belongs to at least one of the selected rows / columns. Otherwise we call the board *unsolvable*. Examples are presented on Fig. 1 and 2 respectively.

	1	2	3	4
a	★	★		
b			★	
c			★	
d	★			★

Figure 1. Example of a solvable board with $n = 4, m = 6$ by taking rows a and d and the column 3.

	1	2	3	4
a		★		
b			★	
c	★			
d	★		★	★

Figure 2. Example of unsolvable board with $n = 4, m = 6$. No set of 3 rows and columns can contain all of the stars.

We are given 2^N such boards with a guarantee that exactly one of them is unsolvable. We are challenged to (probabilistically) find the unsolvable board (with probability higher than any other board).

The IBM Quantum Challenge Problem description fixes the parameters to follow $n = 4, m = 6$ and $N = 4$ and requires contestants to use qRAM and Grover's algorithm.

2. Solution overview

The diagram in the Fig. 3 presents a general circuit that will solve the problem. Any compound gate may use ancilla qubits as desired to obtain an implementation with a lower cost. Used optimizations are described later.

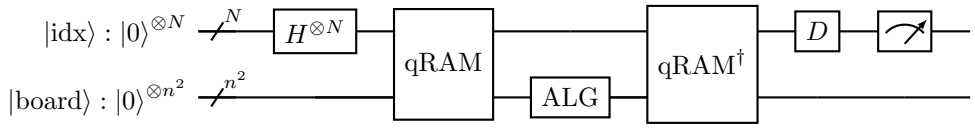


FIGURE 3. A general scheme for solution. ALG is an oracle introducing phase difference if the board is unsolvable. D is the Grover diffusion operator.

Let's start with some definitions.

Definition 1. *of Matrix permanent*
Let $A = [a_{i,j}]_{n\times n}$ be a real-valued matrix. Then

$$\mathbf{perm}(A) = \sum_{\sigma\in S_n}\prod_{k=1}^{n} a_{k,\sigma(k)}.$$

Definition 2. *of Board Matrix*
Let B be an $n \times n$ board with m stars placed on it. We define Board Matrix *M_B to be 0-1 matrix (0 and 1 are treated as real numbers) such that $M_B(i, j) = 1$ if and only if a star is placed in the i-th row and j-th column of the board.*

We start the solution with a theorem which our algorithm will be based on. Proof of this theorem is presented in the appendix A.

Theorem 1. *A board B is solvable if and only if* $\mathbf{perm}(M_B) = 0$.

This theorem immediately entail a simple algorithm for a single instance of the problem. We compute the relevant matrix permanent and simply check if it's a zero.

The following sections will discuss high level implementation and optimization of each element of the diagram presented in Fig. 3. When discussing details, I will focus on constraint specific to IBM Quantum Challenge (i.e. $n = 4, m = 6$ and $N = 4$).

2.1. qRAM and qRAM† implementation.

We are given an array of $2^N = 16$ arrays, each containing $m = 6$ pairs describing star location for all the boards. We need to obtain the state $\frac{1}{4}\sum_{k=0}^{15}|k\rangle \otimes |\text{board}[k]\rangle$, where board$[k]$ is the bitmask representing vectorized matrix M_B. After initial Hadamard gate we have an equal superposition in $|\text{idx}\rangle$. Now we transfer the input data into board qubits using multi-controlled Toffoli gate, controlling with $|\text{idx}\rangle$ and with target qubit representing the requested position.

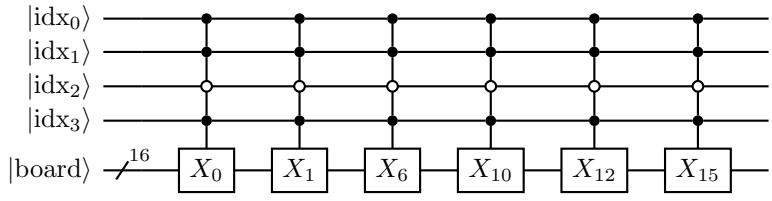


FIGURE 4. Transfer from classical data to qRAM example. Board from the Fig. 1 will be transferred onto index $1101_{(2)} = 13_{(10)}$. X_k is bit flip gate applied to k-th qubit representing the board.

A simple trick using ancilla qubits allows us to greatly reduce number of Toffoli gates which are an intermediary gates during multi-controlled Toffoli gate decomposition. We first compute all four combinations of the first two qubits of the index into separate ancillas. Then, by iterating four combinations of the last two qubits (with one additional ancilla), we can cheaply translate all the boards into qRAM. We can use Gray codes to reduce some of the X gates required to implement 0-controlled bits.

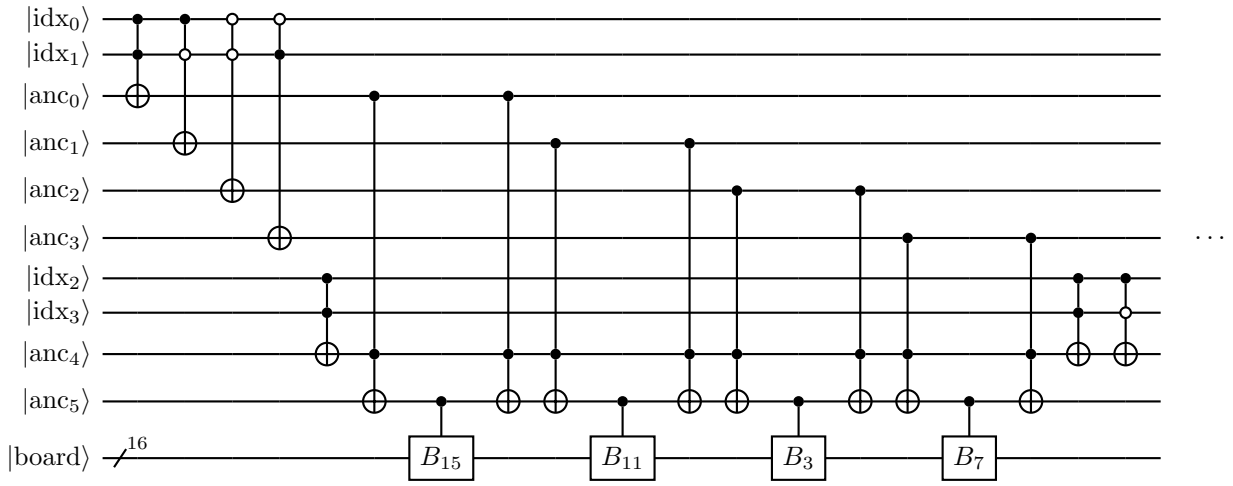


FIGURE 5. Fast transfer of board positions into qRAM. B_n is a sequence of 6 X gates flipping $|\text{board}\rangle$ qubits according to star locations for n-th board. Qubits for $|\text{idx}_2\rangle$ and $|\text{idx}_3\rangle$ are also iterated using Gray codes. This approach requires a total of $4 \cdot 2 + 4 \cdot 2 + 4 \cdot 4 \cdot 2 = 48$ Toffoli gates.

Of course, qRAM† can be realized by exactly the same circuit as the one above. In total, we have $2 \cdot 48 = 96$ Toffoli and $2 \cdot 16 \cdot 6 = 192$ CX gates to fully operate all qRAM-related instruction.

2.2. **Algorithm.** We start with a vital observation that will allow us to greatly reduce the number of required 2 qubit gates.

Proposition 1. *Let B be a board that has $n = 4$ and $m = 6$. Then* $\mathbf{perm}(M_B) \leq 2$.

Proof. By the Bregman–Minc inequality, the only possible cases having $\mathbf{perm}(M_B) > 0$ are:

- one row with three stars, three rows with one star. Thence, $\mathbf{perm}(M_B) \leq (3!)^{\frac{1}{3}} < 2$.
- two rows with two stars, two rows with one star. Thence, $\mathbf{perm}(M_B) \leq (2!)^{\frac{1}{2}} \cdot (2!)^{\frac{1}{2}} = 2! = 2$.

□

If we now try to introduce phase change to the states corresponding to the unsolvable boards (i.e. having $\mathbf{perm}(M_B) > 0$) we can do it by computing the permanent by its definition. For each permutation having stars alongside it, we simply introduce phase change equal to $\frac{2\pi}{3}$. Thus, by Prop. 1 we will change the phase by at most $\frac{4\pi}{3}$. In both cases, the phase change is close enough to π to significantly increase the probability of finding the unsolvable board.

If we changed phase by π, we expect around 47.27% of the samples to find the correct, unsolvable board with one Grover iteration. With phase change equal to $\frac{2\pi}{3}$ and $\frac{4\pi}{3}$ this probability drops to 37.01%, which is still much higher than if we sampled from the uniform distribution (probability of 6.25%).

Now we only need to process the permutations in some smart order to reduce the number of Toffoli gates. We will split the board into two halves (upper and lower two rows), and iterate through pairs (i, j) of columns in the first half. There are $\binom{n}{2} = \binom{4}{2} = 6$ such column pairs. Now for each column pair, we have two *boxes* in board halves, which we join into ancillas by diagonals (please refer to example at Fig. 6). Now we simply execute a controlled phase gate on all four combinations of boxes diagonals, effectively computing value of four permutations.

	1	2	3	4
a		a		b
b		b		a
c	c		d	
d	d		c	

FIGURE 6. Boxes defined by selecting column 2 and 4. We use qubits corresponding to squares with the same letter as control qubits for CCX with four ancillas as target ($|\text{anc}_a\rangle$, $|\text{anc}_b\rangle$, $|\text{anc}_c\rangle$, $|\text{anc}_d\rangle$). Then we execute controlled phase shift by $\frac{2\pi}{3}$ between ancillas corresponding to different boxes (i.e. $(CP\,|\text{anc}_a\rangle\otimes|\text{anc}_c\rangle)\cdot(CP\,|\text{anc}_a\rangle\otimes|\text{anc}_d\rangle)\cdot(CP\,|\text{anc}_b\rangle\otimes|\text{anc}_c\rangle)\cdot(CP\,|\text{anc}_b\rangle\otimes|\text{anc}_d\rangle)$).

This way, we can compute all phase shifts from permutations using $6\cdot(2\cdot 4\times\text{Toffoli}+4\times CP) = 48\times\text{Toffoli}+24\times CP$ gates. Toffoli gates are counted twice to uncompute ancillas.

2.3. **Grover diffusion operator.** This operator is created using the simple decomposition of $CCCZ$ using a single ancilla. Conjunction of qubits $|\text{idx}_0\rangle$ and $|\text{idx}_1\rangle$ is computed into said ancilla. Then, CCZ gate is executed on $|\text{idx}_2\rangle, |\text{idx}_3\rangle$ and that ancilla.

We don't have to uncompute the last Toffoli gate (which would result in not uncomputing the ancilla) – it will cause the diffuser not to diffuse fully correctly and approximately halve the success rate, finishing at expected probability of around 19.43%.

Thus, diffuser requires one Tofolli gate, and one CCZ gate.

3. Further optimization

At this point, we have a very solid circuit that solves the problem using 145 Tofolli, 192 CX, 24 Controlled-Phase, and one CCZ gate. Decomposition of all those gates yields a total of $6\cdot 145+1\cdot 192+2\cdot 24+6\cdot 1 = 1116$ CX gates. We can greatly reduce this number by a neat trick, namely the Margolus gate.

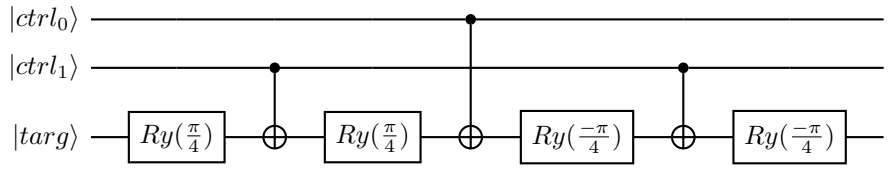


FIGURE 7. Margolus gate that approximates Toffoli gate up to an additional phase change. It requires only 3 CX gates.

Margolus gate is a gate that has similar effect to Toffoli gate – the only difference is the introduced phase change to some of the states in the superposition. Since we use all of the 145 Toffoli gates to mark ancilla qubits, we can simply substitute them with Margolus gates – the phase difference that is Margolus's byproduct is removed during ancilla uncompute!

This simple observation saves us $3\cdot 145 = 435$ CX gates, which single-handedly puts us below the $10,000$ point mark with only 681 CX gates.

3.1. **Intra- and inter- routine optimizations.** Finally, to obtain the result of 6574 we need one more observation: two neighboring Margolus gates that share the target qubit don't have to execute touching Ry gates. If they share the second control qubit as well, we don't have to execute 3 gates per single Margolus (4 Ry gates and two CX in total), gaining us 2 and 24 points respectively.

This gives a great advantage especially with regard to the qRAM subroutine which targets $|\text{anc}_5\rangle$ with $|\text{anc}_4\rangle$ constantly (even though the first control qubit is different).

In the ALG subroutine we can also find this observation useful (which may be less visible at first glance) – when subsequent boxes share a column. Those cases are rarer, only-target sharing is more common in this case.

Finally, we execute some inter-routine optimizations. Those are simply reductions between the final Margolus gates of the qRAM routine and initial Margolus gates of the ALG routine, and later between ALG and qRAM† and between qRAM† and D. This saves only a handful of single qubit gates and is the last optimization implemented in the solution.

Appendix A. Proof of correctness

In this appendix I intend to prove Theorem 1.

Proof. First, we show that if B is solvable, then $\mathbf{perm}(M_B) = 0$.

Let $R = \{r_1, r_2, ..., r_k\}$ and $C = \{c_1, c_2, ..., c_l\}$ be the set of rows and columns in the solution of board B and $k + l = n - 1$. Then, for any permutation $\sigma \in S_n$, we have $\mathbf{card}(R \cup \sigma^{-1}[C]) \leq k + l = n - 1$. Let's pick $s \notin R \cup \sigma^{-1}[C], 1 \leq s \leq n$. Then obviously $s \notin R$ and $\sigma(s) \notin C$. It means, that we just found a square (i.e. at position $(s, \sigma(s))$) that must not be occupied by a star.

It means that the corresponding product for this permutation in the permanent's definition must be zero.

Thence, $\mathbf{perm}(M_B) = \sum_{\sigma \in S_n} 0 = 0$.

Now let's prove that if $\mathbf{perm}(M_B) = 0$ then B is solvable.

Since $\mathbf{perm}(M_B) = 0$, it means that for each permutation $\sigma \in S_n$ at least one element $M_B(s, \sigma(s)) = 0$. Let's fix $\tau = \mathbf{Arg\,max}_{\sigma \in S_n} \mathbf{card}(\{i : M_B(i, \sigma(i)) = 1\})$, in other words τ is one of the permutations that has the most stars alongside on the board.

Now we will construct the solution by the application of the following algorithm:

First, we select all rows i such that $M_B(i, \tau(i)) = 1$, i.e. having a star along the selected permutation τ (we selected at most $n - 1$ rows). If the number of selected rows is less than $n - 1$, we select all but one (arbitrarily selected) remaining row. The only not selected row will be further called x for clarity. Then, we repeatedly execute the following routine (until the board is solved):

(1) find an uncovered star at coordinates (i, j),
(2) select column j,
(3) deselect row $\tau^{-1}(j)$.

If the step (1) cannot be executed, it means that the board is solved and we don't need to execute the routine.

Otherwise, the step (2) is always executable (column j couldn't have been selected previously).

For the step (3) to be not executable, row $\tau^{-1}(j)$ must not be selected. But it would mean that either a) this row was the only row not selected at the beginning (i.e. $\tau^{-1}(j) = x$), or b) that it has been already deselected by some previous iteration. The first case we will consider in the section A.1. The second case is impossible – if some other star had deselected row $\tau^{-1}(j)$ it must have been a star in the column j – thus the star at (i, j) is already covered.

If every step of the routine is executable under condition that board is not yet solved, it means that either:

- routine could be applied *ad infinitum*,
- algorithm stops with a valid solution to the board.

The first case is not possible – since every execution of the routine decreases number of selected rows by one, thus maximal number of executions is $n - 1$.

The other case means that we proved that $\mathbf{perm}(M_B) = 0$ then B is solvable, which is the case.

□

A.1. **Analysis of the a) case in the proof.** To show that this case is impossible, we will try to trace back the origin of the problem and construct a permutation τ' which will have more stars alongside it than τ, which will be a contradiction.

Let $(i_k, j_k)_{k=1}^{s}$ be a sequence of star positions that were selected by the iterations of the routine, with $(i_s, j_s) = (i, j)$ with $j = \tau(x)$. We will prove the following:

Proposition 2. *There exist a subsequence $(i_{k_l}, j_{k_l})_{l=1}^{t}$ of (i_k, j_k) such that:*

- $k_t = s$,
- $\forall l \in \{1, 2, ..., t - 1\} : i_{k_{l+1}} = \tau^{-1}(j_{k_l})$.
- $\forall l \in \{1, ..., t\} : i_{k_l} \neq x$.

We will assume that we selected the longest one amongst all having those properties (one like that exists, since the sequences are finite).

We construct this subsequence from the end to the beginning. We start with $k_t = s$, so the first condition holds. To check the third condition for $k_t = s$, we consider that if $i = i_{k_t} = x$ we would have a star at position $(i, j) = (x, \tau(x))$ which contradicts with the row selection at the beginning of the algorithm.

Now, for each $l \in \{t-1, t-2, \ldots, 1\}$, we construct k_l inductively as follows.

We can see that star at position $(i_{k_{l+1}}, j_{k_{l+1}})$ was not covered during k_{l+1}-th iteration. It couldn't be that $i_{k_{l+1}} = x$ (the third condition). So the star at iteration k_{l+1} must be in a row that was deselected at some previous iteration, namely k_l. We see that, by definition of the routine, we deselected $\tau^{-1}(j_{k_l}) = i_{k_{l+1}}$ by our requirement, which proves the second condition for this l.

Also, if we had $i_{k_l} = x$, we define new permutation τ', such that:

- $\forall p \in \{l, l+1, \ldots, t\} : \tau'(i_{k_p}) = j_{k_p}$,
- $\tau'(y) = \tau(y)$ for all remaining elements y.

By the second condition for the subsequence: $\tau[\{i_{k_l}, i_{k_{l+1}}, \ldots, i_{k_t}\}] = \{\tau(i_{k_l}), \tau(i_{k_{l+1}}), \ldots, \tau(i_{k_t})\} = \{\tau(x), j_{k_l}, j_{k_{l+1}}, \ldots, j_{k_{t-1}}\} = \{j, j_{k_l}, j_{k_{l+1}}, \ldots, j_{k_{t-1}}\}$. On the other hand, $\tau'[\{i_{k_l}, i_{k_{l+1}}, \ldots, i_{k_t}\}] = \{j_{k_l}, j_{k_{l+1}}, \ldots, j_{k_t}\} = \{j_{k_l}, j_{k_{l+1}}, \ldots, j_{k_{t-1}}, j\}$. In other words, we permuted τ at indices $i_{k_l}, i_{k_{l+1}}, \ldots, i_{k_t}$ to obtain τ'. Thus, τ' is also a permutation.

Let's count stars alongside indices $k_l, k_{l+1}, \ldots, k_t$ for τ' and τ. For τ', every (i_{k_p}, j_{k_p}) contains a star by construction, so it has $t - l + 1$ stars for those indices. On the other hand, for τ we had $M_B(i, \tau(i)) = M_B(x, \tau(x)) = 0$ which is the case for $i = i_{k_t} = x$. Thus τ has at most $t - l$ stars on the considered set of rows. This contradicts with τ having the most stars between all permutations.

This construction shows that $i_{k_l} \neq x$ which proves the third condition for this l. Example sequence (i_{k_l}, j_{k_l}) is presented at Fig. 8.

But our subsequence was the longest one, so we must have traced back up to the selection from the only not covered row in the initial setup, namely $i_{k_1} = x$. It contradicts with the third condition. Thus, no such sequence is possible and the case is finally resolved.

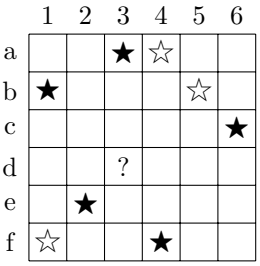


FIGURE 8. Example case for $n = 6$ and $\tau = \{(a,3), (b,1), (c,6), (d,5), (e,2), (f,4)\}$. We are tracing back the star at $(i,j) = (b,5)$. $x := \tau^{-1}(5) = d$ is the only not selected row at the beginning. We traced back that star at $(b,5)$ was uncovered because previously star at $(f,1)$ deselected row b. Furthermore, star at $(a,4)$ deselected row f, which was itself in a row deselected by some other star at column 3. Could this star be placed at $(d,3)$? No – it would create permutation $\tau' = \{(a,4), (b,5), (c,6), (d,3), (e,2), (f,1)\}$ which is the desired contradiction.

Appendix B. Consequences of the permanent-based approach

Computation of the 0-1 matrix permanent is $\#P$-complete, thus it's unlikely that there exists an algorithm that computes it in polynomial time. Our solution, which computes the value directly from definition, has complexity of $O(n \cdot n!)$ (in the quantum setup, $O(nlogn \cdot n!)$ due to required n-qubit Toffoli gate).

Ryser formula for permanent would yield the same result in $O(n \cdot 2^n)$ by using inclusion-exclusion principle. In the challenge with n fixed to 4, this formula does not give any advantage (or at least I didn't figure out how to implement it using less than 168 CX gates).

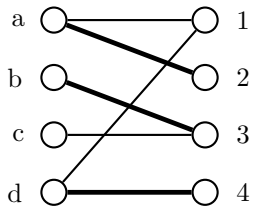
Both solutions are exponential. We can do better, however. Notice that we only need to know whether permanent is zero or not, not necessarily its exact value. In other words, finding any permutation that has n stars alongside it suffice to claim the board unsolvable. Finding one such permutation rather than counting all of them is much easier problem and can be solved in $O(\sqrt{n} \cdot m)$ using maximal (in terms of cardinality) bipartite matching algorithm, for example Hopcroft-Karp.

Definition 3. *Board graph*

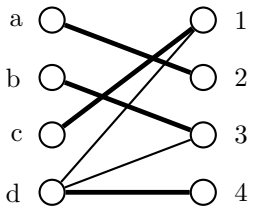
Given $n \times n$ board B with m stars, we define $G_B(V, E)$ – a board graph – such that:

- $V = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\}$, *i.e.* $\mathbf{card}(V) = 2n$
- $E = \{(a_i, b_j) \in V^2 : M_B(i, j) = 1\}$, *i.e.* $\mathbf{card}(E) = m$.

If there exist a perfect matching in this graph then the board is unsolvable – selected edges precisely define the permutation with n stars. This gives us a fast, polynomial time algorithm for the problem. During competition, I didn't believe it could be implemented using fewer gates than the brute-force, given that $n = 4$ but I'd certainly use it provided that n is large.



(A) Graph representing board from Fig. 1. Maximal cardinality matching has size 3, thus board is solvable.



(B) Graph representing board from Fig. 2. Existence of the perfect matching proves its unsolvability.

Figure 9. Example board graphs with highlighted maximal cardinality matchings.