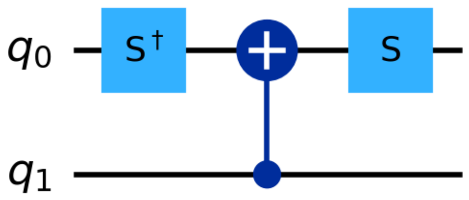
Lil Luna

There were two different approaches that we used for most of the problems. Problems 1,2,3,4,5,8 and 9, we were able to decompose the given unitary matrix into either exact combination of Clifford+T gate sets, or into some combination of Clifford gates + non Clifford R_z gates and we used the Ross-Selinger algorithm to replace the R_z gates with Clifford+T gate sets. For unitaries that we were not able to decompose, we used a general KAK decomposition and some other methods to minimize T gates, detailed in the section of Problem 6.

1 Problem 1

Since $Y = SXS^\dagger$, $CY = (I \otimes S)CX(I \otimes S^\dagger)$. The corresponding circuit is



2 Problem 2

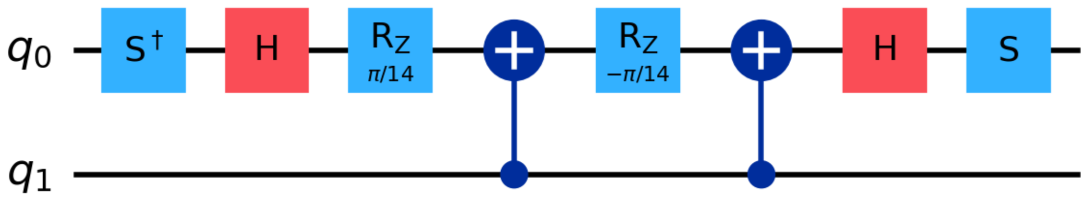
Since $R_y(\theta) = e^{-i\theta Y/2} = \cos(\theta/2)I - i\sin(\theta/2)Y$, and $Y = SXS^\dagger = SHZHS^\dagger$, it follows that

$$CR_y(\pi/7) = SHCR_z(\pi/7)HS^\dagger$$

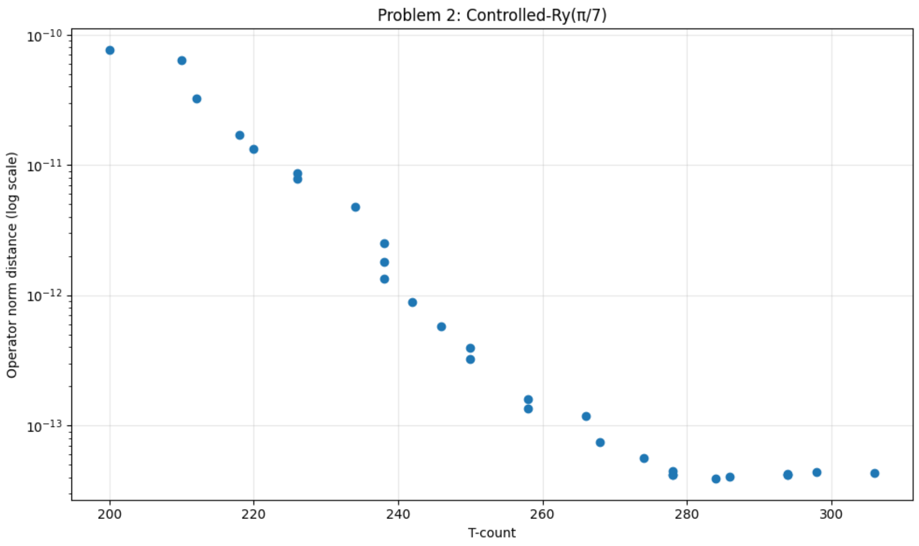
Then we can use the identity

$$CR_z(\theta) = CX(I \otimes R_z(-\theta/2))CX(I \otimes R_z(\theta/2))$$

we can decompose the matrix purely in terms of R_z gates and some Clifford gates. The corresponding circuit is



Then we just use the Ross-Selinger algorithm to replace the R_z rotations with an optimal (within some tolerance) Clifford+T gate set.



3 Problem 3

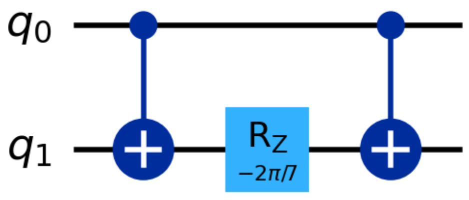
As the Pauli matrices anticommute, $XZ = -ZX \implies XZX = -Z$. Using this, we can show that

$$CX(I \otimes Z)CX = |0\rangle\langle 0| IZI + |1\rangle\langle 1| XZX = (|0\rangle\langle 0| - |1\rangle\langle 1|)Z = Z \otimes Z$$

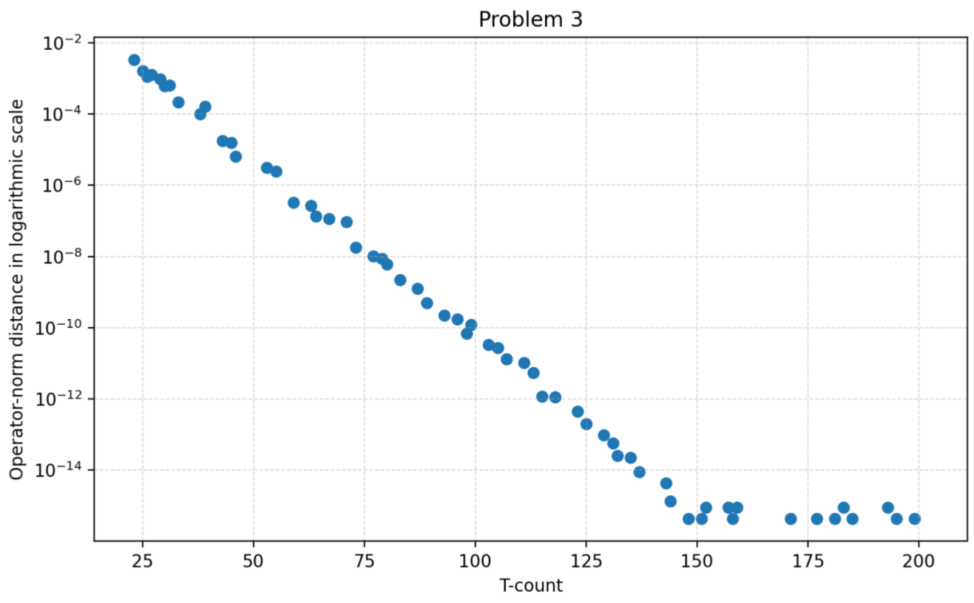
So,

$$e^{i(\pi/7)Z\otimes Z} = CX(I \otimes R_z(-2\pi/7))CX$$

The corresponding circuit is



Again we use the Ross-Selinger algorithm to get a Clifford+T gate set approximation to replace the R_z rotation. A plot of operator norm distance vs. T count is shown below



4 Problem 4

Explicitly writing out H_1 gives

$$H_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The matrix that diagonalizes this is

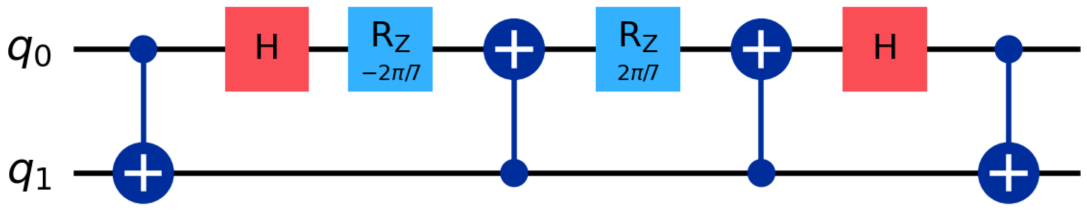
$$V = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

which is exactly equal to $(H \otimes I)CX$. The diagonalized matrix is $D = diag(0, 2, -2, 0)$, and so

$$e^{i(\pi/7)H_1} = e^{i(\pi/7)V^\dagger DV} = V^\dagger e^{i(\pi/7)D} V$$

$$= CX(H \otimes I)diag(1, e^{2i\pi/7}, e^{-2i\pi/7}, 1)(H \otimes I)CX$$

But $diag(1, e^{2i\pi/7}, e^{-2i\pi/7}, 1) = CXdiag(1, 1, e^{-2i\pi/7}, e^{2i\pi/7})CX = CXCR_z(4\pi/7)CX$. After using a similar decomposition of CR_z as used in problem 2, the circuit becomes



5 Problem 5

Using the identity

$$XX + ZZ + YY + I = 2SWAP$$

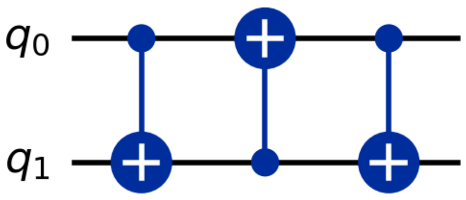
we get

$$e^{i(\pi/4)(XX+YY+ZZ)} = e^{i(\pi/4)(2SWAP-I)}$$

Since $SWAP$ and I commute, this is equal to $e^{i(\pi/2)SWAP}e^{i\pi/4}$. Ignoring the global phase factor, and using Euler's formula,

$$e^{i(\pi/2)SWAP} = iSWAP$$

Ignoring the phase factor again, SWAP can be decomposed into three alternating CNOT gates, with the corresponding circuit being



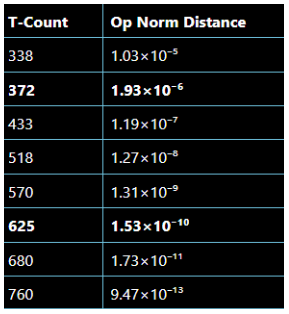
This is the optimized minimum T count circuit.

6 Problem 6

The solution of this problem and problem 10 has the following pipeline:

- We used KAK decomposition to decompose the unitaries into a combination of single qubit unitaries and at most 3 CNOT gates. Qiskit's TwoQubitBasisDecomposer implements this.
- Each single-qubit unitary needs to be decomposed into rotations. The choice of Euler basis matters. Different bases were tested on and the one that produced the smallest number of non Clifford R_z rotations was chosen.
- After Euler decomposition, many Rz gates have angles that are multiples of $\pi/4$. These are already Clifford gates. We keep only the non Clifford R_z, and merge adjacent Clifford gates.
- For each non Clifford R_z, we use the Ross-Selinger algorithm to get an approximate Clifford+T gate set.

This is a general procedure we used for any unitary matrices that we were not able to decompose on pen and paper ourselves, specifically the matrices in problems 6 and 10.Below are some of the range of values we scanned through



T-Count	Op Norm Distance
338	1.03×10^{-5}
372	**1.93×10^{-6}**
433	1.19×10^{-7}
518	1.27×10^{-8}
570	1.31×10^{-9}
625	**1.53×10^{-10}**
680	1.73×10^{-11}
760	9.47×10^{-13}

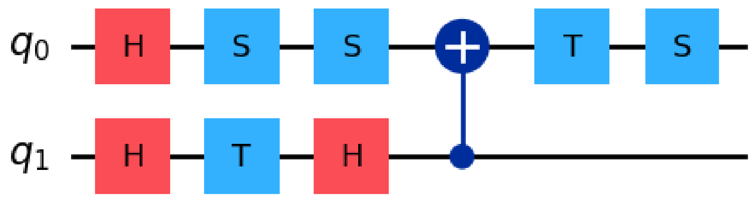


Figure 1: 2-T solution to problem 7. Fidelity of 0.979

7 Problem 7

Problem 7 focuses on synthesizing a circuit to reach a given randomized statevector from the $|00\rangle$ state. Because the statevector is random, there is little inherent structure to exploit. However, because we only care about the transition from the $|00\rangle$ state, only one column of the unitary operator is predetermined. We split this problem into two approaches: lowest T and highest fidelity.

7.1 Lowest T

Because the statevector has lower dimensionality than a unitary, it is actually possible (with aggressive pruning) to search all possible circuits using only ≤ 2 T gates up to a circuit depth of 10. During this process, we build a library of sequences that either a) perform the identity, or b) generate a state we have already visited. Any sequence that contains one of these as a subsequence can be pruned from the search.

This method results in the optimal circuit with depth ≤ 10 and T count ≤ 2. Our circuit is shown in figure **??**, and has fidelity 0.979.

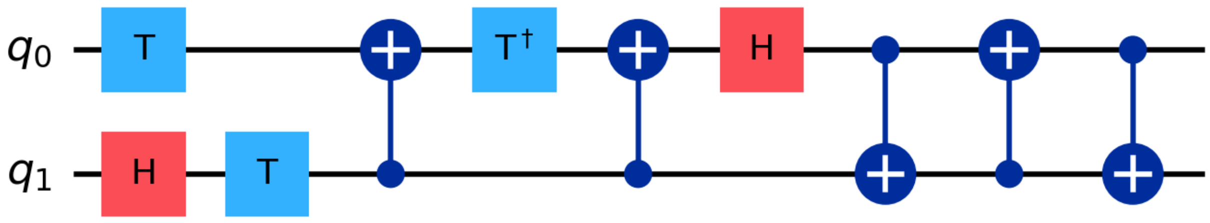
7.2 Highest Fidelity

For this problem, we used the first unitary we synthesized that generates the desired state, and considered it a random unitary with no structure. Thus, we applied our arbitrary-unitary method.

See the section High Fidelity Arbitrary Unitaries.

8 Problem 8

We found this mostly by trial and error. The first column being all ones implied Hadamard must be applied to both qubits. The complex i entries looked like a form of CS gate which can be formed in terms of $T, T^\dagger$ and CX. Later we realized a SWAP gate was needed to fix some bit reversal.

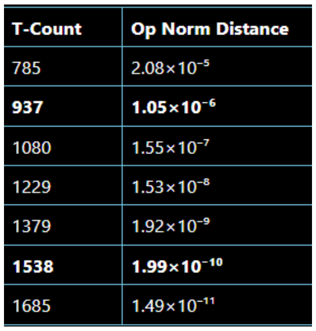


9 Problem 9

We deduced that the 1/2 comes from a Hadamard spreading the state and the i comes from some sort of controlled phase interaction. We wrapped a CS gate inside Hadamards, and noticed the mapping was bit reversed, so we applied the SWAP gate to match the unitary matrix.

10 Problem 10

See the Problem 6 section. Below are some of the range of values we scanned through



T-Count	Op Norm Distance
785	2.08×10^{-5}
937	**1.05×10^{-6}**
1080	1.55×10^{-7}
1229	1.53×10^{-8}
1379	1.92×10^{-9}
1538	**1.99×10^{-10}**
1685	1.49×10^{-11}

11 Problem 11

Problem 11 specifies a 4-qubit *diagonal* unitary

$$U\,|x\rangle = e^{i\varphi(x)}\,|x\rangle\,, \qquad x\in\{0,1\}^4,$$

where all given phases $\varphi(x)$ are integer multiples of $\pi/4$. Define $\omega := e^{i\pi/4}$ and quantize the phase table into

$$p(x) := \mathrm{round}\left(\frac{\varphi(x)}{\pi/4}\right) \pmod 8 \in \mathbb{Z}_8, \qquad \Rightarrow \qquad U\,|x\rangle = \omega^{p(x)}\,|x\rangle\,.$$

Since $p(x)\in\mathbb{Z}_8$, the unitary can be synthesized *exactly* (up to global phase) using Clifford+T.

We compile U using the *phase polynomial* formalism: diagonal Clifford+T circuits correspond to $\pi/4$ phases conditioned on linear Boolean parities produced by CNOT networks. We extract a $\mathbb{Z}_8$ phase representation from the truth table and run `rmsynth` to reduce the number of $\pi/4$ phase injections (minimizing T-count). Each remaining phase term is implemented with a parity gadget: compute the relevant parity onto a target wire using a CNOT ladder, apply T or $T^\dagger$ (repeated as needed) to realize $R_z\left(\frac{\pi}{4}k\right)$, then uncompute the ladder. This yields a circuit using only $\{H, T, T^\dagger, \mathrm{CNOT}\}$.

11.1 Results

Our final compiled circuit achieved:

- T-count: 4
- Fidelity: 1.0
- Operator-norm distance: $3.389203244510956\times10^{-16}$

12 Bonus Problem 12

In this bonus task, we are given a JSON file `challenge12.json` specifying a list of m pairwise commuting n-qubit Pauli strings. Each term consists of a Pauli string $P_j\in\{I,X,Y,Z\}^{\otimes n}$ and an odd coefficient $k_j\in\{1,7\}$ with angle unit $\pi/8$. The target unitary is

$$U \;=\; \prod_{j=1}^{m} \exp\Bigl(-i\,\frac{\pi}{8}\,k_j\,P_j\Bigr)\,.$$

Because all P_j commute pairwise, the product is order-independent and defines an exact unitary. We are required to use only the gate set

$$\{H,\ T,\ T^\dagger,\ \mathrm{CNOT}\}.$$

12.1 Compilation strategy

The key idea is to exploit commutativity by compiling the set of terms as a structured object. For each Pauli exponential, we implement a *basis change* that maps non-Z Paulis into the Z basis, apply the required diagonal phase using a parity gadget, and then undo the basis change.

Step 1: Basis change to Z-type Paulis. For each qubit on which P_j acts nontrivially, we apply Clifford basis changes using only $\{H, T, T^\dagger\}$:

- $X \mapsto Z$ via H since $HXH = Z$.
- $Y \mapsto Z$ via an $S^\dagger H$-type change, but since $S^\dagger$ is not allowed we implement it as $(T^\dagger)^2 H = T^\dagger T^\dagger H$.

This converts P_j into a product of Z operators (up to a sign), so the exponential becomes diagonal in the computational basis.

Step 2: Parity compute–phase–uncompute. If the transformed operator is Z on a subset of qubits, we compute the parity of that subset onto a target qubit using a CNOT ladder, apply a single-qubit phase using only T or $T^\dagger$, and then uncompute the ladder. Up to global phase,

$$\exp\Big(-i\,\frac{\pi}{8}\,kZ\Big)\ \equiv\ T^k,$$

so only T or $T^\dagger$ (possibly repeated) are needed for $k \in \{1, 7\}$.

Step 3: Gate-set validation. After synthesis and optimization, we validate that the final OpenQASM contains only $\{H, T, T^\dagger, \mathrm{CNOT}\}$ gates and no S or $S^\dagger$ gates.

12.2 Results

Our final compiled circuit achieved:

- T-count: 1963
- Fidelity: 1.0
- Operator-norm distance: 2.633315×10^{-13}

13 High Fidelity Arbitrary Unitaries (Nathan)

To address the first problem with no real structure, we created a pipeline to generate a circuit in the $H, T, T\dagger, S, S\dagger, CNOT$ basis from an arbitrary unitary operator. As shown in [**??**], non-optimal compilation methods can achieve an operator norm of $\frac{1}{\epsilon}$ only by using $O(\epsilon)$ T gates (as opposed to optimal Solovay-Kitaev, which has error $\frac{1}{2^{cT}}$). With that in mind, we expect our error to scale between $O(\frac{1}{T})$ and $O(\frac{1}{2^{cT}})$, and therefore allow a user-specified ϵ during search.

Our method for high fidelity arbitrary unitaries has the following steps:

1. Conversion to `Clifford` $+ T$ basis
2. Discretization of cz angles using `pygridsynth`
3. Optimizaiton via `rmsynth`, identity search, commutative aggregation, and `qiskit` transpile.
4. Final identity search to force conversion to desired basis

The first two items are prerequisites for using `rmsynth` and increase the efficiency of commutative aggregation. The `pygridsynth` library implements the Ross Selinger algorithm. The optimization loop works as follows: for each method in `rmsynth`, `id search`, `aggregation`, `qiskit-transpile`, we compare the circuit before and after application. If the T-count or total gate count decrease, we keep the new circuit. The number of iterations is tunable, but we found 10 iterations to converge such that further iterations made no difference.

For the `rmsynth` method, we selected continuous chunks of diagonal operators broken apart by H gates. For each chunk, we note the entanglement structure, solve for the phase polynomial, run it through `rmsynth`, then rebuild the resulting chunk from the optimized phase polynomial and reimplement the entanglement with CNOTs.

For `id search`, we pre-compute a dictionary of all sequences in our basis of length ≤ 15 that result in identity or that have a lower T-count representation. We then scan across the circuit in linear time, removing/replacing sequence matches.

For `aggregation`, we note that many 2-gate identities that could occur fail because of a gate in between. This step utilizes commutation rules to group gate-types with themselves, to maximize the chances of annihilation. This aggregation also allows larger chunks during future `rmsynth` passes and higher hit rates for longer identities in future `id search` passes.

For `qiskit-transpile`, we transpile to $\texttt{Clifford} + T$ basis with optimization level 3.

From our experience, `rmsynth` usually provides significant savings during the first iteration, and `id search` provides the bulk of remaining savings, with improvement tapering exponentially each iteration.