Quantum Mastermind

Gegham Zakaryan[1]

[1]American University of Armenia
[1]College of Science and Engineering
[1]CS339 - Quantum Computing

May 20, 2024

Abstract

This project is a quantum version of the classic game Master Mind played by two players. Given 2 different colors, the first player – the keeper, secretly forms a sequence of n colored pins, where several may share the same color. The task of the second player – the guesser, to disclose the hidden sequence with minimal guesses.

This project works with the following version (**version 2**): each guess is graded by the keeper with a single digit – the number of correct pins in their correct positions. The game stops when the grade of the most recent guess is n.

Given a guess sequence g of size n and the grade s from the keeper, the given algorithms and steps will produce the next guess efficiently.

Contents

1 Time Complexity

Estimate the complexity of the move search in different stages of the game.

Let's consider how many guesses we can possibly have initially, given no information at all. Denote that number by $N_0 = 2^n$, where n is the length of the sequence in the game.

We will pick a guess $g = g_1 g_2 ... g_n$ randomly, give it to the keeper, and get the number k back. Then, let's denote N_1 as the number of guesses that make sense, given the score k for the sequence g. We can count this number by calculating the number of ways we can count the number of different sequences that can be created by flipping k bits in the sequence g, and denote this number by N_1.

$$N_1 = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Let's use Stirling's approximation to approximate this number:

$$N_1 \approx \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{\sqrt{2\pi k}(\frac{k}{e})^k \sqrt{2\pi(n-k)}(\frac{n-k}{e})^{n-k}}$$

Since $\binom{n}{k}$ attains its maximum value at $k = \frac{n}{2}$, then the number N_1 will also attain its maximum value there.

$$N_1 \approx \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{\sqrt{2\pi n/2}(\frac{n/2}{e})^k \sqrt{2\pi(n-n/2)}(\frac{n-n/2}{e})^{n-n/2}} = \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{2\pi\frac{n}{2}(\frac{n}{2e})^n} = \sqrt{\frac{2}{\pi}}\frac{2^n}{\sqrt{n}}$$

Following this, we can calculate the ratio of the past and current number of possibilities.

$$\frac{N_0}{N_1} = \sqrt{\frac{\pi}{2}}\sqrt{n} \approx \sqrt{n}$$

From this, it follows that we can reduce the input of the algorithm after each step by a factor of $\sqrt{n}$, which will greatly affect the time complexity of the algorithm.

2 Quantum System

Suggest a system of qubits that describes the game, and define the game states and state vectors.

Let n be the length of the sequence in the game. We will keep n qubits describing our guess, n auxiliary qubits storing the XOR results and the Hamming distances, and some auxiliary qubits for interim calculations.

Initialization of the state and state vectors

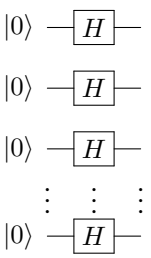
Start with n qubits in the state $|0\rangle$ and additional auxiliary qubits as required.

$$|0\rangle^{\otimes n} \otimes |0\rangle^{\otimes n}$$

Create Superposition

Apply Hadamard gates to the first n qubits to create an equal superposition of all possible n-bit states:

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$$



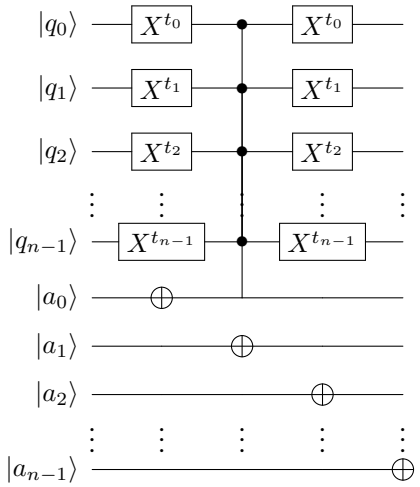
3 Results

Design quantum gates that implement the operations of the sequential classical algorithm. Apply the designed gates to superposition states aiming at parallelization of the move search.

Let n be the length of the picked guess $g = g_0 g_1 \dots g_{n-1}$. Let s be the score we got from the keeper.

We will use XOR operations to compare each qubit of the superposition state with the guess bit sequence $g = g_0 g_1 \dots g_{n-1}$ and calculate the best candidates for the next guess. The best candidates will be sequences that have a Hamming distance s from a possible state of the superposition. For example, if we are playing the game with n=4 bits, the guess is g=1011, and the score is s=1, then, from the 16 possible states of the superposition of 4 qubits, we desire to get one of the following qubits when we measure it: 0011, 1111, 1001, 1010.

- For each qubit q_i and the corresponding bit g_i in the guess sequence:
  - If $g_i = 1$, apply an X gate to flip q_i.
  - Apply a CNOT gate from q_i to an auxiliary qubit initialized to $|0\rangle$.
  - If $g_i = 1$, apply another X gate to revert q_i.

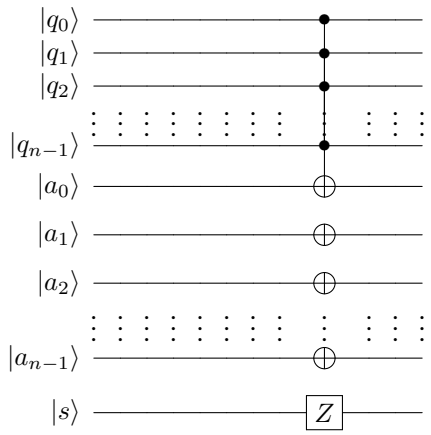


4. Sum the Hamming Distance

Use quantum addition to sum the results in the auxiliary qubits, which represent the XOR results, to compute the Hamming distance. This requires several auxiliary qubits and a quantum adder circuit.

5. Mark States with Desired Hamming Distance

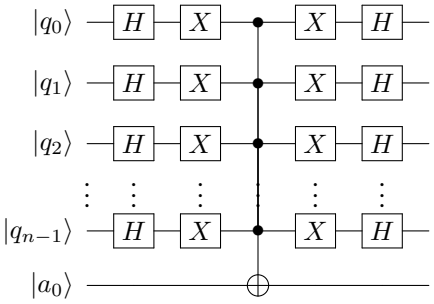
Use quantum comparison to mark states where the auxiliary qubits encode the desired Hamming distance s. This can be done using a comparator circuit that flags states with Hamming distance s and then applies a phase flip.



6. Amplitude Amplification

Apply the Grover diffusion operator to amplify the amplitude of the marked states.

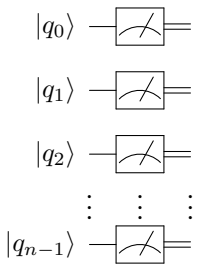
- Apply Hadamard gates to all input qubits.
- Apply X gates to all input qubits.
- Apply a multi-controlled Z gate.
- Apply X gates to all input qubits again.
- Apply Hadamard gates to all input qubits again.



Repeat the marking and diffusion steps as needed.

7. Measurement

Measure the first n qubits. The resulting bit string should have a high probability of having a Hamming distance of the desired value s from the given sequence $t = t_0 t_1 \dots t_{n-1}$.



4 Next Steps

Since the algorithm gives the next guess only, given a previous guess with its score, it's not clearly apparent how to solve the problem completely. To solve the problem, here are the steps we take with this algorithm:

- Initially, we pick a completely random guess $g0$ and get the score $s0$ from the keeper.
- Then, we use our algorithm to get the next guess $g1$.
- We give this guess to the keeper and get $s1$.
- Instead of passing $g1$ and $s1$ to the algorithm immediately, we do something different. First, we pass $g0$ and $s0$, but we don't measure the qubits in the last step. This allows us to keep only the states that have the Hamming distance $s0$. After that, we run the algorithm on those qubits with $g1$ and $s1$. This allows us to have much less (a factor of $\sqrt{n}$ less) states to work with. After this is done, we finally measure the qubits.
- For each next step, we do the same: we run the algorithm for $g0, 1, ...n$ and $s0, 1, ...n$ instead of only gn and sn.

References

[1] Dr. Suren Khachatryan. Review Session on the Quantum Mastermind problem. Unpublished review session, May 2024. CS339 Quantum Computing, Spring 2024, American University of Armenia.

[2] Lvzhou Li, Jingquan Luo, and Yongzhen Xu. Playing mastermind on quantum computers, 2023.