Assignment 1

June 3, 2023 1:41 PM

Khaled Gaber #1004144302

Question 0. *Unitary matrices are invertible, and the conjugate transpose is the inverse. Certain gates, like Hadamard, are their own inverses. What are some gates that are their own inverses, versus some that are not. How does this property manifest itself when we produce the U_f for an arbitrary function?*

Answer 0. Other gates that are their own inverses are the Pauli Gates (X, Y, Z), and Phase gates. Some gates that are not their own inverses are the CNOT and Toffoli gates. This property manifests itself in Uf for an arbitrary function during their computation, as it simplifies the computation. E.g. in a function that uses Hadamard gates, then we know that an even number of these applications can be removed, simplifying the computation.

Question 1. *What is the |+> state and how can it be produced?*

Answer 1. The |+> state is a specific quantum state, which is a superposition of |0> and |1> states with equal probability and the same phase. It can be produced by applying a Hadamard gate to the |0> state.

Question 2. *What is the |-> state and how can it be produced?*

Answer 2. The |-> state is a specific quantum state, which is a superposition of |0> and |1> states with equal probability and opposite phase. It can be produced by applying a Hadamard gate to the |1> state.

Question 3. *Comment on the probabilities of measuring |0> and |1> for the states -|+>, +|+> and the analogous |-> states.*

Answer 3. The global phase of a quantum state (the overall multiplicative factor) does not affect the outcome of measurements. Therefore, the states |+>, -|+>, and +|+> all have the same probabilities of being measured as |0> and |1>, which are both 1/2. The same applies to the |->, -|->, and +|-> states.

Question 4. *What does H(-|+>) and H(+|+>) map to? Ditto for the analogous |-> states.*

Answer 4. The Hadamard gate applied to the |+> and |- > states maps back to the |0> and | 1> states respectively, regardless of the global phase. So H(-|+>) and H(+|+>) both map to | 0>, while H(-|->) and H(+|->) both map to |1>.

Question 5. *Read about the Bernstein-Vazirani algorithm and comment on it's similarities with the Deutsch* algorithms. Show how it operates.*

Answer 5. Both algorithms exploit the parallelism inherent in quantum computing to solve a problem faster than classical algorithms. They both use a black box function (oracle) to hide certain properties of the function that they are trying to determine. The Bernstein-Vazirani algorithm is designed to find a secret number hidden within a black box function. Given a function that takes in a string of bits and returns the bitwise product of that string and some hidden string, the algorithm can find the hidden string in a single query to the function.

An example of how it operates:

1. Prepare two quantum registers, initially in the state |0>

2. Apply a Hadamard gate to each qubit in the first register, and a Hadamard gate to the qubit in the second register, which brings us to the state (|0> + |1>) for each qubit in the first register and the state (|0> - |1>) for the second register
3. Query the oracle: the qubits in the first register are entangled with the qubit in the second register, which is flipped if the corresponding bit in the hidden string is 1
4. Apply a Hadamard gate to each qubit in the first register again, causing constructive interference for the correct result
5. Measure the first register to give the hidden string

Question 6. *Construct a U_f for the Boolean function "OR". Apply |-> to the control input and |+> to the data input and write the full response. Prove that U_f is Unitary and report the inverse of U_f.*

Answer 6.

f(x, y) = x OR y.

The unitary transformation U_f acting on the two-qubit state |x, y> for this function would be:

$$U_f\ |x, y\rangle = |x, y \oplus (x \text{ OR } y)\rangle.$$

We take |-> for the control qubit x and |+> for the data qubit y, the initial state is:

$$(1/\sqrt{4})(|00\rangle - |10\rangle + |01\rangle + |11\rangle).$$

Upon applying U_f, the states |01>, |10>, and |11> will change because f(x, y) = 1 for these states, so the final state becomes:

$$(1/\sqrt{4})(|00\rangle - |10\rangle + |10\rangle + |01\rangle).$$

To prove that U_f is unitary, we have to show that U_f U_f(T) = U_f(T) U_f = Identity Matrix. Since U_f only changes the value of specific states and does not change their phase, it is its own inverse, i.e., U_f = U_f(T), and hence, U_f is unitary.

Question 7. *Construct a U_f for an arbitrary three input Boolean function (you may define it), and repeat the above.*

We define a function f(a, b, c) = a XOR b XOR c

The unitary transformation U_f acting on the three-qubit state |a, b, c> for this function would be:

$$U_f\ |a, b, c\rangle = |a, b, c \oplus ((a \text{ XOR } b) \text{ XOR } c)\rangle.$$

We take |-> for the control qubit a and |+> for the data qubits b and c, the initial state is:

$$(1/\sqrt{8})(|000\rangle - |100\rangle + |010\rangle + |110\rangle - |001\rangle - |101\rangle + |011\rangle + |111\rangle).$$

Upon applying U_f, the states |001>, |010>, |100>, |110>, and |111> will flip because f(a, b, c) = 1 for these states, hence the final state becomes:

$(1/\sqrt{8})(|000> - |100> - |010> - |110> + |001> + |101> - |011> - |111>)$.

To prove that U_f is unitary, we have to show that U_f U_f(T) = U_f(T) U_f = I. Since U_f only flips the sign of specific states and does not change their phase, it is its own inverse, i.e., U_f = U_f(T), and so, U_f is unitary.

Question 8. *Explain how all classical algorithms can be executed via a Quantum Computer, and show a general synthesis technique for this using class concepts.*

Answer 8. All classical algorithms can be reduced to some combination of AND, OR, & NOT circuits. We have quantum computer equivalents for these AND OR and NOT circuits, therefore we can recreate any classical algorithm as a quantum algorithm. Based on class concepts the general synthesis technique involves creating a truth table for inputs and outputs, then recreating it as a quantum algorithm.

Question 9. *What is the key function for search, classically and in a quantum computer. Show the definitions of these functions.*

Answer 9. The key function for search in a classical context is basically the comparison function, where you compare each item with the item you are searching for. If you have a function f(x) such that f(x) = 0 for all x except for when x is the solution, where f(x) = 1, then f(x) effectively serves as the classical search function.

In a quantum computer, we use the same type of function f(x) but we use it in a phase oracle. The phase oracle applies a phase flip to the quantum state that corresponds to the solution. Thus, the oracle function U_f is defined such that U_f |x> = (-1)^(f(x)) |x>. If x is a solution, its phase gets flipped; otherwise, its state remains the same.

Question 10. *Apply the control bit for the above key function quantum oracle in the superposition |-> and derive the output as a function of the input |x>.*

Answer 10. If we take the control bit for the phase oracle in the superposition |->, and the state |x> as the input state, then the output state will be (-1)^(f(x)) |->. This means that if x is a solution, the control bit will flip from |-> to |- > = -|->; otherwise, the control bit remains the same.

Question 11. *Suppose there are 2^Q possible elements to be searched, show how to generate the uniform superposition of the above via a quantum circuit.*

Answer 11. If there are 2^Q possible elements to be searched, we can start with a quantum register of Q qubits, all initialized to the state |0>. We then apply a Hadamard gate to each qubit. The Hadamard gate transforms the |0> state to (|0> + |1>)/sqrt(2) and the |1> state to (|0> - |1>)/sqrt(2), so applying a Hadamard gate to each qubit will put the quantum register in a uniform superposition of all possible 2^Q states.

Question 12. *Apply (11) to the data input of (10), and show how the resulting function implements the first operator for the Grover iteration.*

Answer 12.

The first operator of Grover iteration is the application of the oracle function U_f which flips the phase of the target state.

Now we apply step 1 to the superposition from Q11. The superposition after applying the Hadamard gates is:

-

$$(1/\sqrt{(2^Q)})\ \Sigma|x\rangle \text{ from } x=0 \text{ to } 2^Q - 1.$$

Applying the oracle function U_f from Q10 to this state gives us:

$$(1/\sqrt{(2^Q)})\ \Sigma((-1)^{(f(x))})|x\rangle,$$

where the phase of the state corresponding to the solution is flipped. This is the state after the first step of the Grover iteration.

This application of the oracle function basically "marks" the target item we are searching for by inverting its phase. The state is still a superposition of all possible states, but the state corresponding to the solution has been given a phase flip.

Question 13. *Consider two dimensional vector (sx,sy), and show how to construct the rotation matrix that rotates general vector (x,y) about (sx,sy). Relate this to the Grover diffusion operator.*

Answer 13.

1. Translate the plane so (sx, sy) is the new origin.
2. Our vector is now (x - sx, y-sy)
3. Multiply the general rotation matrix by our new vectors

$$\begin{bmatrix} \cos\theta & -\sin\theta \end{bmatrix} \begin{bmatrix} x - s_x \end{bmatrix}$$
$$\begin{bmatrix} \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y - s_y \end{bmatrix}$$

4. New coordinates are:

$$x' = \cos\theta * (x - sx) - \sin\theta * (y - sy)$$

$$y' = \sin\theta * (x - sx) + \cos\theta * (y - sy)$$

5. Translate the plane back to origin (0,0)

$$x'' = x' + sx = \cos\theta * (x - sx) - \sin\theta * (y - sy) + sx$$

$$y'' = y' + sy = \sin\theta * (x - sx) + \cos\theta * (y - sy) + sy$$

This relates to the Grover diffusion operator because it performs a similar rotation, but in the space of quantum states.

Question 14. *If a database has 2^64 entries and one solution, what is the angle that |s> makes with |s->? How many iterations are required by Grover Search? Explain what happens when you go one over and one less than these # of iterations. Redo this analysis for a 2^3 entry database, and compare.*

Answer 14.
$\theta = \arccos((N-M) /(\sqrt{N}\ \sqrt{(N-M)}))$

For 2^64: M = 1, N = 2^64
$\theta = \arccos((2^{64}-1) /(\sqrt{(2^{64})}\ \sqrt{(2^{64}-1)})) = 2.366*10^{-7}$ rad
From class, the number of iterations = $\sqrt{N}$, so for M = 1 solution and N = 2^64, # Iterations = $\sqrt{2^{64}} = 2^{32} = 4294967296$

If we go one over, then we will 'overshoot' our desired solution state, and reduce the success probability

If we go one below, then we will not have yet converged onto our solution state, and also reduce the success probability

For 2^3: M = 1, N = 2^3
θ = arccos((2^3-1) /(√(2^3) √(2^3-1))) = 0.36137 rad
From class, the number of iterations = √N, so for M = 1 solution and N = 2^3, # Iterations = √2^3 = 2.828

The difference between the required number of iterations for the two show that Grover's algorithm has a lower time complexity compared to classical searching algorithms