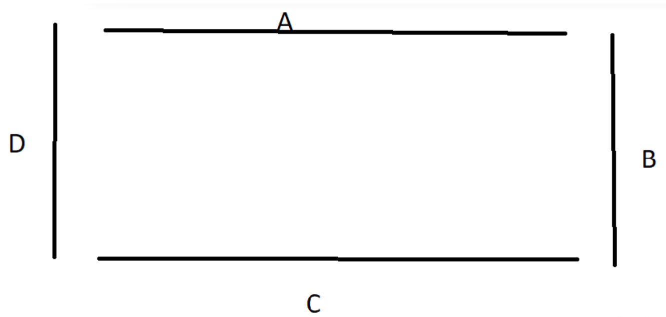
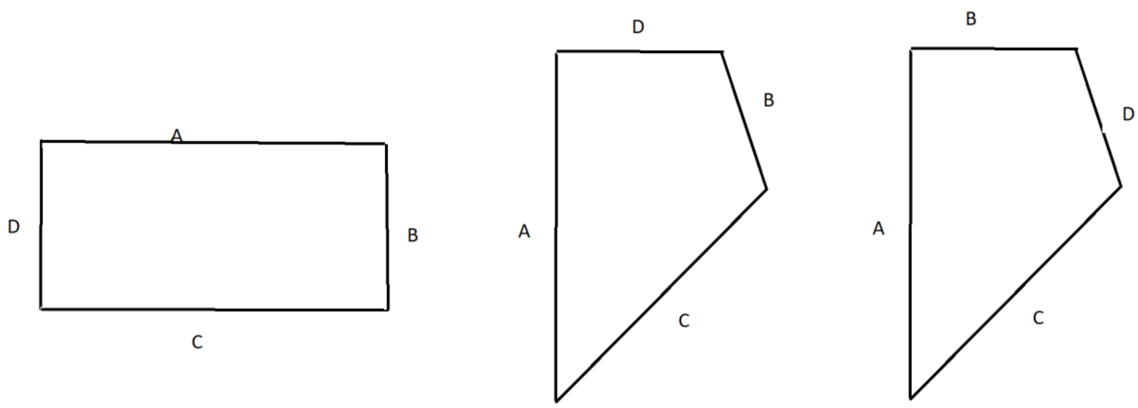
Identifying Rectangles using Quantum Computing

For this problem, there are 4 sides of random length. The goal is to find if it is possible to make a rectangle with these side lengths. Using the basic identity that a rectangle consists of two pairs of equal sides; it is possible to reframe this as a Boolean satisfiability problem.



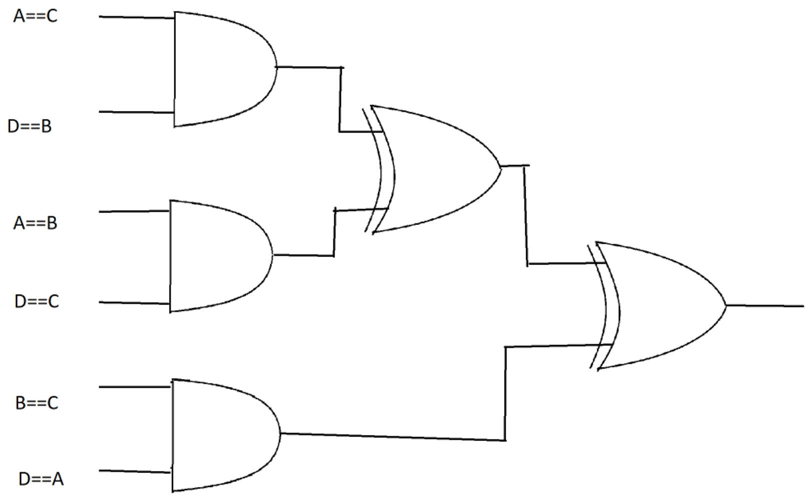
This can be split into three discrete possibilities. If any single possibility is can be determined to be a rectangle, the program should return 1. To evaluate if it is a rectangle, we can use the rule of rectangles that states all rectangles have 2 pairs of equal sides.



These are the three possibilities of side arrangements. We can convert this to the following Boolean statement.

(A == C && D == B) || (A == B && D == C) || (B == C && D == A)

A classical circuit to solve this problem can be written as follows:



This does not include storing the integer values and doing the initial operation which would use bits proportional to the number of digits of the number in binary. However, this is separate and dependent on the numbers used, while the circuit shown above is constant across all inputs. This classical circuit will increase in complexity exponentially if dimensions are increased – a rectangular prism instead of a rectangle. A quantum circuit can be achieved using half the number of qubits. We can do this by encoding the side lengths as vectors instead of Boolean values. This is possible because qubits can be between 1 and 0. We can then use Grover's algorithm to diffuse the result into a 1 if rectangle and a 0 if not a rectangle. Grover's algorithm is a quantum algorithm that can search through an unsorted database or a list of items to find a specific item much faster than classical algorithms as it provides a quadratic speedup. However it has more applications than just the search algorithm and can also be used to solve this type of Boolean satisfiability problem. It consists of two parts: an oracle and a diffuser.

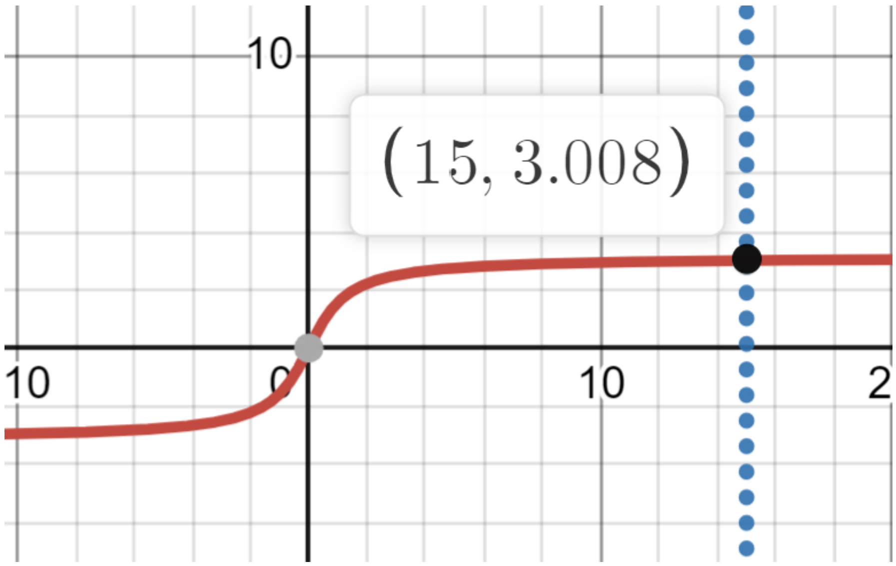
The oracle changes the phase of the target item (i.e., flips its sign), while leaving the other states unchanged. This is done by encoding the target item into a separate quantum state and using a task specific algorithm to apply the phase shift. In this case it should change the phase of the target qubit only if it is a rectangle and leave it as is if it is a not a rectangle. Similar to the Boolean statements for the classical algorithm, there are similar arithmetic with a quantum computer. A qubit is a vector space and the RZ gate can be used on a qubit in superposition to do a phase rotation by a specific angle. The quantum solution presented here uses the below functions to determine the phase rotation. There are two functions because there are two rectangle possibilities in this quantum circuit. We can define these statements as:

(A - C) + (D - B)

(A - B) – (D - C)

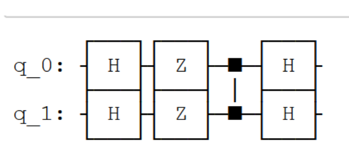
At least one of these statements will be 0 is a rectangle can be made. If a rectangle cannot be made it will not be 0. However it cannot be between -1 and 1 because the problem definition stated side lengths were integers and differences will also be integers. Slotting these functions into the inverse tangent function can help get a phase rotation that is 0 if it is a rectangle and close to pi if not a rectangle. This is because the inverse tangent function has a horizontal asymptote of pi/2 and passes through the origin.

However, the tangent function would need to be modified to better fit the task. if the function is vertically stretched by a factor of 2, the asymptote will be raised to 3 radians. Since this is near π radians but not π radians, it is slightly probabilistic with a small error rate.

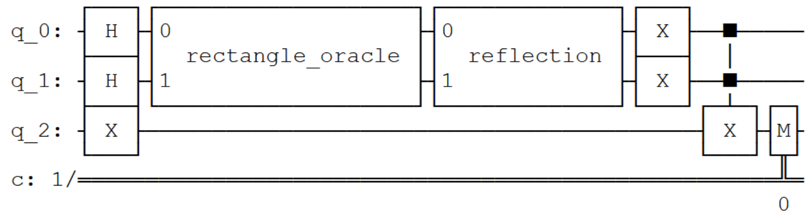


However since the RZ gate only takes in integers the closest integer value to the asymptote is 3 radians. The closest integer x value that surpasses 3 is 15 so we can horizontally compress the graph by a factor of 15 so the smallest input of 1 can surpass 3. Note that if it is -1 it would be -3 but that would still provide the same rotation as it is close to pi degrees rotation.

The diffusion operator or reflection operator amplifies the amplitude of the target state relative to the other states in the superposition. It does this by collapsing the superposition by using a Hadamard gate then reversing the state by using a Pauli Z gate. After this the CZ gate is added which behaves similar classical AND gate where a target qubit's state is reversed if both qubits are in the one state. after this superposition is revived by adding another Hadamard gate. Often the diffusion operator and the oracle function are repeated the square root of n times where n is the number of elements in unstructured database. However, due to the algorithm in this case only operating on two qubits it is only performed once.



The Grover's algorithm provides outputs such that measure to 00, 01, 10, and 11. 01, 10 and 11 are all the different types of rectangles listed earlier. 00 means a rectangle is not. In order to get the binary result of 1 if rectangle and 0 if not rectangle, another qubit is needed. To do this we can add a Pauli X gate on all the qubits to reverses the states of the qubits. As a result, the states 00 will become 11. We can the use the Toffoli gate on both qubits, targeting the third qubit. The third qubit will be 1 as the Pauli X gate flipped its state. the Toffoli gate applies a Pauli X gate on Qubit 3 if qubit 1 and qubit 2 are 1 (no rectangle). The Pauli X gate would make the third qubit 0. If both are one of the results is 0 (is a rectangle), the third qubit will remain 1.



While this algorithm may be more scalable than a classical approach as it uses the Grover's algorithm and half as many bits, it is also probabilistic. This means that the majority of runs the result will display correctly, but there is an approximately 1.1% of the time that the program will fail. This can be mitigated by running the program multiple times. However, since the failure percentage is relatively low the probability of a majority wrong (3 wrong) within 5 tries is 1.7×10^{-6}.