QCI(Quantum Computing Inc.)

QCi is a photonic quantum computing company .They provide affordable quantum machines.The primary goal of their machines is to enable high-performance computation, particularly for optimization problems, artificial intelligence, and cybersecurity.

QCi offers different options to access its flagship product - Dirac-3. These are:

- ❖ Trial Cloud Access: Free, with a 10-minute time allocation.
- ❖ Hourly Cloud Access: Paid, charged per hour.
- ❖ Cloud with Concierge: Paid hourly access that includes technical assistance from an application scientist.
- ❖ Premium Access: Installs Dirac-3 locally, avoiding queue delays.

Dirac-3

Dirac-3 is QCi's highest-performing quantum machine, designed for complex optimization problems.Dirac-3 uses **qudits** as the unit of quantum information, with each state represented in ddd-dimensions.It can solve problems involving integers and continuous numbers, surpassing binary constraints.

Working principles

Entropy Computing and Noise

Entropy refers to randomness or uncertainty, often introduced by noise or information loss.Traditional quantum computing minimizes entropy to preserve quantum coherence, the property that allows qubits to work in superposition and entanglement.Instead of minimizing entropy, in contrary, EQC leverages noise and loss as computational resources, turning the problem into a solution.EQC uses the coupling of quantum states to an environment (an entropy source) for computation. Entropy, seen as randomness, is utilized to explore a vast number of computational pathways efficiently.

The Challenge in Traditional Quantum Computing:

Scaling computational power is hindered by noise. Current solutions involve error-corrected logical qubits, which require many physical qubits (hundreds to thousands) for each logical qubit, making the system complex and resource-intensive. Further, Error correction demands an error rate below 1%, achievable only in cryogenic and highly isolated environments, creating scalability bottlenecks.

Why Noise is a Problem in Traditional Quantum Computing

Decoherence:

- ○ Noise leads to qubits losing their quantum properties due to environmental interactions.

Error Correction:

- Logical Qubits: Error-corrected qubits perform actual computation.
- Physical Qubits: Many are required to support one logical qubit, making the system resource-heavy.
- Cryogenic environments and high isolation are needed for low error rates, which hinder scalability.

EQC Embraces decoherence and loss as part of the computation process.Utilizes entropy to explore computational pathways rather than suppressing it.

Interaction - and Measurement - Free Quantum Zeno Gates:

Enables universal computation.

Experimental Demonstration:

Leverages the quantum Zeno effect for interaction-free, all-optical switching.

The Role of Vacuum Fluctuations

- **What Are Vacuum Fluctuations?**
  - In quantum physics, even "empty" space (vacuum) contains random energy spikes due to the uncertainty principle.
  - These fluctuations manifest in effects like the Casimir force, caused by changes in vacuum energy near surfaces.

Dirac Diagrams and Quantum States

- **Dirac Diagrams**:
  - Represent quantum states and their interactions.
  - **State Vectors ($|s\rangle$)**: Indicate the state of a quantum system.
  - **Operators**: Represent how states evolve or interact, often as matrices.

Dirac-3 Specifications

Solver Type: Qudit-Constrained Discrete Optimization.

- **Hardware Type**: Hybrid analog machine combining quantum optics and digital electronics.
- **Maximum Variables**: 949.

Definitions

- **Cryogenics**: The use of extremely low temperatures to stabilize quantum states and minimize disruptions.
- **Vacuum Fluctuations**: Short-term, random changes in vacuum energy caused by the uncertainty principle.
    - Virtual particles are created and annihilate each other in pairs.
- **Quantum Zeno Effect**: Frequent measurement prevents a quantum system from changing its state.

Optimization Techniques

1.Discrete Solver Optimization

Definition: Optimization where the decision variables are discrete (e.g., integers or binary values) rather than continuous.
Solution Space: The solution space consists of distinct, separate points within the discrete set.
Objective: The goal is to find the best solution by optimizing a specific objective function within the discrete space.
Example Problems:

- ➢ **Scheduling**: Finding the optimal schedule for tasks or events.
- ➢ **Graph Theory Problems**: Problems like the traveling salesman problem, shortest path problems, or network flow problems.
- ➢ **Resource Allocation**: Allocating limited resources (e.g., time, money, or equipment) optimally across different tasks or entities.

2.Non-Convex Optimization:

Definition: Optimization involving objective functions or constraints that are non-convex, meaning the problem has multiple local minima and maxima.
Convex vs Non-Convex:

- ○ A **convex function** ensures that any line segment between two points on the curve lies above or on the curve, which guarantees a single global optimum.
- ○ A **non-convex function** can have multiple local minima and maxima, making it difficult to guarantee finding the global optimum.

Challenge: The presence of multiple local minima and maxima makes it more challenging to find the global minimum or maximum.

Applications: Common in machine learning (e.g., training neural networks) and quantum computing.

Combinatorial Optimization:

Definition: Optimization problems where the solution involves selecting or arranging discrete objects or components in the best possible way.
Solution Space: The solution space consists of a finite set of possibilities, often with a large number of combinations.
Objective: The goal is to find the optimal combination or arrangement of elements that maximizes or minimizes an objective function.
Example Problems:

- **Traveling Salesman Problem (TSP)**: Finding the shortest route that visits a set of cities and returns to the starting point.
- **Knapsack Problem**: Selecting a subset of items to maximize total value without exceeding a weight limit.
- **Graph Coloring**: Assigning colors to vertices in a graph such that adjacent vertices have different colors while minimizing the number of colors used.
- **Network Flow Problems**: Optimizing the flow of materials or information through a network.

3.Quadratic Optimization:

Definition: Optimization problems where the objective function is quadratic, typically involving terms with squared variables.
Objective Function: The general form of a quadratic optimization problem is:
Minimize 12xTQx+cTx\text{Minimize } \frac{1}{2} x^T Q x + c^T xMinimize 21xTQx+cTx
Where xxx is the vector of decision variables, QQQ is a matrix representing quadratic terms, and ccc is a vector for linear terms.
Properties:

- Involves both linear and quadratic terms in the objective function.
- Can be either convex (if QQQ is positive semi-definite) or non-convex (if QQQ is indefinite).

Applications:

- **Portfolio Optimization**: Selecting a mix of investments to maximize return while minimizing risk (variance).
- **Control Systems**: Optimizing system performance while maintaining stability.
- **Support Vector Machines**: Used in machine learning for classification tasks, where the objective function involves quadratic terms.

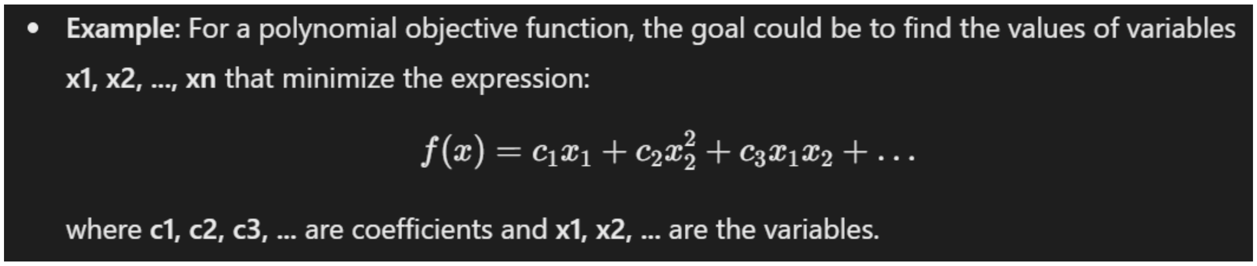
4.Two-Body Interaction:

Definition: Optimization problems where the variables or entities interact in pairs, typically involving pairwise interactions between them.
Context: Often used in fields like physics, quantum mechanics, and optimization problems in machine learning or network theory.
Objective: The goal is to optimize interactions between pairs of variables or entities, taking into account their mutual influence on the system.
Example Applications:

- **Quantum Mechanics**: Modeling the interaction between two particles, such as in the calculation of energy levels or wave functions.
- **Quantum Computing**: Two-body interactions in quantum circuits, where qubits interact in pairs to generate entanglement and perform computations.
- **Machine Learning**: Pairwise similarity or interaction between data points, such as in collaborative filtering or recommendation systems.
- **Molecular Dynamics**: Modeling the interaction between pairs of atoms or molecules in simulations of chemical reactions or physical systems.

Objective Function:

- **Definition**: An **objective function** is a mathematical expression that represents the goal of an optimization problem. It defines how we evaluate a solution to the problem based on certain variables.
- **Optimization Goals**:
  - **Minimization**: The goal is to find the smallest possible value of the objective function (i.e., minimize the cost, energy, or penalty).
  - **Maximization**: The goal is to find the largest possible value (i.e., maximize profit, efficiency, etc.).
  - **In Dirac-3**: By default, the system minimizes the objective function, but to maximize, you multiply the objective function by **-1**.



- **Example:** For a polynomial objective function, the goal could be to find the values of variables **x1, x2, ..., xn** that minimize the expression:

$$f(x) = c_1 x_1 + c_2 x_2^2 + c_3 x_1 x_2 + \dots$$

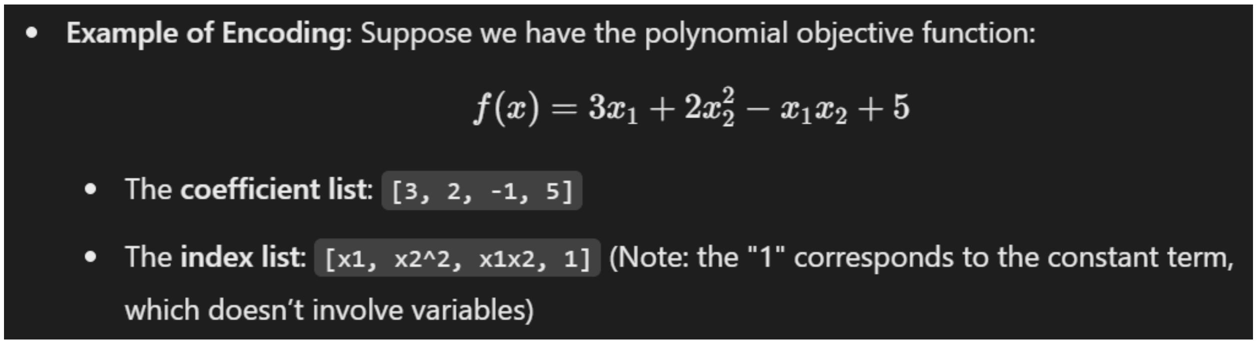
where **c1, c2, c3, ...** are coefficients and **x1, x2, ...** are the variables.

2. Energy Minimization:

- **Energy**: In Dirac-3, the **energy** refers to the value computed from the objective function. It quantifies the "cost" or "penalty" of a specific configuration of variables.
- **Goal**: The optimization process tries to **minimize the energy**, meaning it searches for the configuration of variables that results in the **smallest possible energy** value. This represents the **best solution** to the optimization problem.
- **Stochastic Nature**:
  - Dirac-3 is a **stochastic solver**, meaning that the solutions it finds are probabilistic, and the result might not be the same each time the algorithm runs.
  - To deal with this randomness, you run the algorithm multiple times and choose the solution that occurs most frequently or has the best value (i.e., the minimal energy).
- **Energy Interpretation**: The lower the energy, the better the configuration of the system. For example, in a cost optimization problem, the lower the energy, the lower the cost.

3. Polynomial Encoding:

- **Purpose**: The objective function in Dirac-3 is often expressed as a **polynomial**. This means that the function is made up of terms involving **variables** (like **x1, x2**) raised to different powers (squared, cubed, etc.).
- **Encoding the Polynomial**:
  - Each **term** in the polynomial has two parts:
    - **Coefficient**: A numerical value (e.g., **5**, **-2**, **3.14**).
    - **Indices**: The **variables** involved and their corresponding powers (e.g., **x1**, **x2^2**, **x1x2**).
  - The **encoding** involves creating a **list of coefficients** and a **list of indices**:
    - **List 1 (Coefficients)**: Contains the numerical values of the polynomial terms.
    - **List 2 (Indices)**: Contains the variables and their corresponding powers (e.g., **x1**, **x2^2**).



- **Example of Encoding**: Suppose we have the polynomial objective function:

$$f(x) = 3x_1 + 2x_2^2 - x_1x_2 + 5$$

  - The **coefficient list**: `[3, 2, -1, 5]`
  - The **index list**: `[x1, x2^2, x1x2, 1]` (Note: the "1" corresponds to the constant term, which doesn't involve variables)

- **Goal**: Dirac-3 uses these encoded lists to understand the structure of the polynomial and compute the energy for different variable assignments.

Summary:

- **Objective Function**: A function that defines the goal of the optimization problem (to be minimized or maximized).
- **Energy Minimization**: The process of finding the optimal solution by minimizing the value of the objective function (energy). Dirac-3 uses randomness (stochastic nature) and multiple runs to find the best result.
- **Polynomial Encoding**: The objective function can often be expressed as a polynomial. The function is encoded with a list of coefficients and a list of variable indices (with powers) to guide the optimization algorithm.

Q1: Dirac-3 uses a constrained discrete number optimization approach powered by entropy quantum computing. How does the system handle noise and decoherence differently compared to traditional quantum systems?

- Elizabeth Y.

<u>Answers</u>

A1: the way it works is completely different. In the traditional Quantum system, noise and decoherence are challenges that could lead to loss of information. which is why quantum error correction is used. This requires significant overhead- if you have read about physical qubit and logical qubit. Many physical qubits are needed to protect one logical qubit. This is a major issue in scaling those systems. In Dirac-3, we are using EQC(Entropy Quantum Computing) which embraces and leverages noise and decoherence instead of trying to isolate the system from it. I am not sure how exactly that is happening but here it's not an issue instead an advantage. more thoughts on this is welcome.

- Hope A.

Q2: How does it use Entropy and noise as a method for computation?

-Amir F.

<u>Answers</u>

A1: maybe this has the answer: https://arxiv.org/abs/2407.04512

Q3: Is there a particular reason for the selection of qudits encoding in dirac-3 in comparison to the common multi-qubit encoding?

- Peniel Y.

<u>Answers</u>

A1: The selection of qudit encoding in Dirac-3 (or similar quantum systems) over traditional multi-qubit encoding is often driven by several specific advantages that qudits

provide for certain types of quantum applications. Here are some of the key reasons:
Higher-Dimensional Quantum States: Qudits (e.g., 3-level systems in Dirac-3) represent more information per quantum system compared to qubits, which require multiple qubits to represent the same number of states.
Resource Efficiency: Qudits reduce the number of quantum systems needed, making quantum systems more resource-efficient, especially in complex problems requiring multiple states.
Simplified Quantum Circuits: Qudit encoding streamlines circuit design by directly representing higher-dimensional states, avoiding the complexity of multi-qubit systems.
Problem Matching: Qudits naturally align with problems needing higher-dimensional state spaces (e.g., quantum cryptography, multi-level systems), making them more efficient for specific applications.
Performance and Fault Tolerance: Qudits can improve performance by reducing gate depth and error potential, offering better fault tolerance in quantum operations.
Flexibility in Quantum State Representations: Qudits offer more compact and efficient representations, particularly in systems suited for multi-level states (e.g., ions, photons).
Implementation with Quantum Hardware: Some quantum platforms (like trapped ions and photons) naturally handle qudits, leveraging their ability to control more than two energy levels or modes.
Potential for Advanced Quantum Algorithms: Qudits enable advanced algorithms that exploit higher-dimensional Hilbert spaces, improving performance in areas like quantum error correction, simulation, and optimization.

-**Hope A.**

A2: in the context of the problem we will be working on: the use of qudit encoding in Dirac-3 for problems like discrete solver optimization provides several advantages over traditional multi-qubit encoding:
Compact Representation: In discrete optimization problems, variables often have multiple possible states. With qudits, a single quantum system can represent multiple states (e.g., 3 states in Dirac-3), directly matching the problem's need for multiple discrete values. For a multi-qubit system, you would need several qubits to represent the same number of states, which increases complexity and resource requirements.
Reduced Complexity: For problems like Quadratic Unconstrained Binary Optimization (QUBO), multi-qubit systems can lead to large, complex circuits with more gates. Qudits, on the other hand, reduce the number of quantum resources and operations needed by encoding the problem's variables directly in higher dimensions, leading to more efficient circuit design.
Efficient Search Space Exploration: In discrete optimization, you typically search for the best combination of variable states. Qudits naturally support a broader search space because they can represent more states simultaneously, speeding up the exploration compared to multi-qubit systems, which require multiple qubits to represent a larger state space.

Matching the Problem Domain: Many optimization problems, such as integer programming or resource allocation, deal with variables that can take integer values. Qudits are naturally suited to these problems because they can directly encode multiple integer values in a single quantum system. In contrast, multi-qubit encoding may require complex transformations to map these integer values into binary representations, complicating the problem-solving process.

Improved Fault Tolerance: In optimization tasks, especially when looking for optimal solutions, qudit-based systems can provide better fault tolerance. The higher-dimensional states can be more resilient to noise and errors, which is critical in optimization tasks where precision is important.

-Hope A.

Q4: It is said Dirac3 can solve problems involving integers and continuous numbers, surpassing binary constraints. What are these binary constraints?

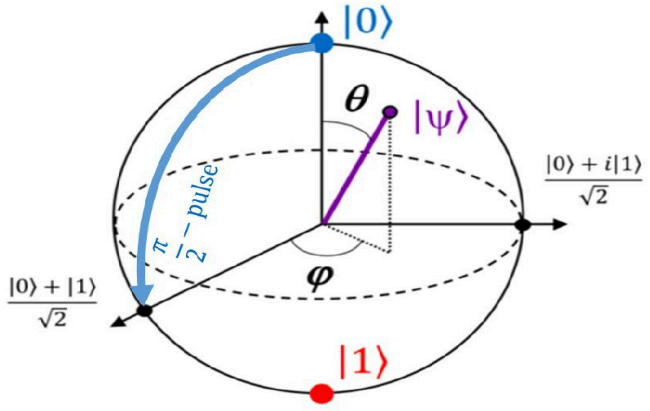
- Nahom G.

Answers

A1: based on the description for qudits, the dirac-3 employs encoding system of integers like |3> kind ... which means the binary constraints may refer to binary numbers being defined as |0> and |1> only.

-Peniel Y.

A2: Classical computing uses sequences of 0s and 1s to describe systems, but quantum mechanics uses rays in Hilbert spaces to describe systems which can be superpositions of several single states. So instead of vectors representing integer coefficients in classical computing, in quantum mechanics the complex coefficients, that when normed squared represent probability distributions, have continuous probability density interpretation so the state vectors represent form a continuous distribution of points named bloch sphere.



$|0\rangle$
θ
$|\psi\rangle$
$\frac{|0\rangle + i|1\rangle}{\sqrt{2}}$
$\frac{\pi}{2} - pulse$
φ
$\frac{|0\rangle + |1\rangle}{\sqrt{2}}$
$|1\rangle$

$$|\psi\rangle = cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle$$

There are infinite points in a Bloch sphere representing state vectors |psi>.

- Osvaldo P.

Q5: Why is it hard to solve non-convex objective functions? How can we solve them(non-convex) and the trade offs for the proposed solutions?

- Nahom G.

<u>Answers</u>

A1: One problem with non convex functions is that they can have accumulation of local maxima or minima that do not have neighbor points. This means that it is hard to approximate functions by sequences of points that belong to the domain of such functions. In summary it is hard to find convergent solutions.

- Osvaldo P.

Q6: One of the fundamental principles in keeping a persistent quantum state is the Quantum Zeno effect(I really recommend u guys to head back and check out the philosophical background btw, cool question raised by Zeno). My question is how often do the measurements need to be done(the interval duration) or in general ,are there principles dictating the measurements?

-Peniel Y.

Q7: What does the data file look like when we create it?

Answer:
The data file contains the following components:

1. **Objective Function Coefficients**:
   - These are numbers that define the "cost" or "energy" associated with selecting a feature.
   - Example:
     For an objective function $2x_1+3x_2-x_3$, the coefficients are 2,3,−1
2. **Variables**:
   - Represent the features you are selecting (e.g.,x_1,x_2,x_3).
   - Example: In the Iris dataset, x_1 might represent "sepal length," x_2 "sepal width," and so on.
3. **Constraints**:
   - Rules that limit the selection of features.
   - Example: If we want to select no more than 2 features, the constraint is:
     $x_1+x_2+x_3 \leq 2$
4. **Variable Types**:
   - Define whether variables are binary (e.g., 000 or 111) or continuous.

Sample Data File for Feature Selection Problem:

makefile

Copy code
```
Objective:
2 3 -1
Variables:
x1 x2 x3
Constraints:
x1 + x2 + x3 <= 2
x1, x2, x3 ∈ {0,ww 1}
```

Q8: How does quantum computing provide an advantage in Feature Selection?

Answer:
Quantum computing is useful for solving optimization problems like feature selection, especially when the problem size grows exponentially.

1. **Why Classical Struggles:**
    - For n features, there are 2^n subsets to evaluate.
    - Example: For 100 features, there are 2^{100} combinations, which is infeasible for classical methods.
2. **Quantum Advantages:**
    - **Parallelism**: Quantum computers process multiple states at once, allowing faster exploration of possible solutions.
    - **Optimization Algorithms**: Algorithms like **QAOA (Quantum Approximate Optimization Algorithm)** efficiently find good or near-optimal solutions.

Example:

- If we need to select features to minimize an error function with 20 variables, a quantum computer can explore these combinations faster than a classical brute-force method.

Q9: What are integer problems, and why are they important here?

Answer:
Integer problems involve variables that can only take discrete integer values (e.g., 0 or 1 for feature selection).

1. **Finite Integers**:
    - The possible values are finite. For binary variables, they are {0,1}
2. **Importance in Feature Selection:**
    - Each feature is either selected (1) or not (0).

Example Problem:

- For x_1,x_2,x_3, we want to select features that minimize an error function like: Objective=$2x_1+3x_2-x_3$
- Constraints: $x_1+x_2+x_3 \leq 2$.

Q11.**How can feature selection be formulated as a problem suitable for Dirac3?**

Feature selection can be formulated as a problem for Dirac3 by representing it as an optimization task. The goal is to select a subset of features that optimizes a specific objective, such as maximizing the predictive performance of a model or minimizing the complexity of the selected features, while adhering to certain constraints.

In this context:

1. **Variables**: Represent whether a feature is selected or not (e.g., binary variables where $x_i=1$ if the feature i is selected, and $x_i=0$ otherwise).
2. **Objective Function**: Define the mathematical function to be minimized or maximized, such as a trade-off between model accuracy and feature count.
   Example: f(x)=−accuracy(x)+λ⋅feature count(x)f(x) = - \text{accuracy}(x) + \lambda \cdot \text{feature count}(x)f(x)=−accuracy(x)+λ⋅feature count(x).
3. **Constraints**: Add restrictions like limiting the total number of selected features or ensuring selected features meet specific criteria.

Dirac3 can then solve this as an integer optimization problem by encoding the variables, coefficients, and constraints into its solver format. This allows exploration of combinations of features that minimize the objective function, leveraging quantum algorithms for efficiency in complex, large-scale problems.

Github: https://github.com/Hope-Alemayehu/QPFS-using-Dirac3

Week 1: Run Initial Experiment on Feature Selection Using Dirac3

Why This Task?

- It's manageable within a week.
- Results will provide valuable insights into Dirac3's performance and usability.

Steps to Complete This Task:

1. **Preparation**:
   - Review Dirac3's documentation to set up the environment and understand feature selection capabilities.
   - Select a small, clean dataset for feature selection (e.g., a public dataset in genomics or finance).
   - Preprocess the dataset to ensure compatibility with Dirac3.

2. **Experiment Setup**:
    - Load the dataset into Dirac3.
    - Use Dirac3's feature selection functionality to analyze the dataset.
    - Experiment with different configurations or hyperparameters if supported.
3. **Execution**:
    - Run the experiment and monitor the process.
    - Record execution time, output features, and any errors or anomalies encountered.
4. **Analysis**:
    - Compare the selected features with those obtained from classical methods (e.g., using sklearn or another classical ML library).
    - Assess the quality of the features selected based on domain knowledge or by running a simple downstream model (e.g., logistic regression).
5. **Documentation**:
    - Summarize the process, findings, and challenges.
    - Prepare a short report to discuss results in the next team meeting.

Deliverable:

- A short report or presentation covering:
    - Dataset used.
    - Experiment setup.
    - Key results (e.g., selected features, execution time, comparison with classical methods).
    - Observations and insights.
    - Recommendations for next steps (e.g., tweaking Dirac3 settings or scaling up).

- **Qci-developer guide**
    - https://quantumcomputinginc.com/learn/module/introduction-to-dirac-3
    - https://quantumcomputinginc.com/learn/module/introduction-to-dirac-3/dirac-3-developer-beginner-guide
    - https://quantumcomputinginc.com/learn/lessons
    - Official Document