DMF Work Experience: Quantum Computing for Engineering Design

## Introduction

This document outlines the steps for applying a quintessential quantum algorithm to an abstracted engineering design problem. We will cover the actual code required to run a (simulated) version of Grover’s algorithm for unstructured search as applied to a tile placement problem. The intention is to introduce you to standard coding practices, as well as learn some of the specifics about this quantum approach, and the considerations we must make as engineering designers looking to apply these burgeoning techniques. Note that this exercise was developed using code from <https://github.com/OliverSchiffmann/comparingGateAndAnnealing> and Google’s Gemini Advanced 2.5 Pro.

## Overview of Problem and Approach

1. **The Problem:** Imagine an 8x8 grid (like a small chessboard). We want to place two small tiles on it. There are rules: both tiles *must* be on the far right edge (the 'eastern wall', where x=7), and they can't be in the same square. See below for problem diagram.

A red square with green squares and green squares

AI-generated content may be incorrect.

1. **Why Quantum?** Classically, you'd check every possible pair of positions. For a big grid, this takes time. Grover's algorithm is a quantum search algorithm that can potentially find the correct positions faster than a classical computer *in theory* (though current quantum computers have limitations).
2. **Qubits:** Instead of classical bits (0 or 1), we use qubits. Qubits can be 0, 1, or *both at the same time* (superposition). We use qubits to represent the x and y coordinates of our tiles. Since coordinates go from 0 to 7, we need 3 qubits for each coordinate (binary 000 to 111). So, 3 for x1, 3 for y1, 3 for x2, 3 for y2 = 12 qubits total for positions.
3. **Superposition (initialize\_state)**: We start by putting all position qubits into superposition using the Hadamard (H) gate. This means the quantum computer explores *all possible pairs of positions simultaneously*.
4. **The Oracle (build\_oracle)**: This is the clever part. The oracle checks if a specific arrangement of tiles follows our rules (x=7 for both, y1 different from y2). If an arrangement is a *solution*, the oracle "marks" it by flipping its *phase* (a quantum property). It doesn't tell us the answer directly, just marks it. We use extra 'helper' qubits (c\_q) to perform the checks. Crucially, we 'uncompute' the checks at the end to reset the helper qubits.
5. **The Diffuser (build\_diffuser)**: After the oracle marks solutions, the diffuser amplifies the importance (amplitude) of the marked states and shrinks the importance of the non-marked states. Think of it like turning up the volume on the correct answers and turning down the volume on the wrong ones.
6. **Iteration**: We repeat the Oracle and Diffuser steps several times (num\_iterations). Each time, the correct answers get more and more amplified. The ideal number of iterations is related to the square root of the total possibilities.
7. **Measurement**: Finally, we measure the position qubits. Because the solution states have been amplified, we are now very likely to measure a state that represents a valid placement of the tiles according to our rules.
8. **Simulation (AerSampler)**: Since we don't have easy access to a real quantum computer, we use AerSampler from Qiskit Aer. This is a powerful classical program that *simulates* how a quantum computer would behave for this circuit.
9. **Results (results\_to\_excel)**: The simulation runs the circuit many times (num\_shots). The results tell us how often each possible tile arrangement was measured. We process these results (converting binary codes to coordinates) and save the most frequent (most likely correct) arrangements to an Excel file. The highest frequency results should be the valid solutions where both tiles are at x=7 but have different y coordinates.

## Implementing Approach Using VS Code

### Prerequisites: Install Python and VS Code

1. **Install Python:**
   * Go to the official Python website: <https://www.python.org/downloads/>
   * Download the latest stable version for your operating system (Windows, macOS, Linux).
   * Run the installer.
   * **Important for Windows Users:** During installation, make sure to check the box that says "Add Python X.X to PATH" (where X.X is the version number). This makes Python accessible from the command line.
   * Follow the installer prompts to complete the installation.
   * **Verify Installation:** Open a new terminal or command prompt (Terminal on Mac/Linux, Command Prompt or PowerShell on Windows) and type python --version (or python3 --version on some Mac/Linux systems). You should see the Python version number printed. If you just installed and added to PATH on Windows, you might need to close and reopen your terminal.
2. **Install Visual Studio Code (VS Code):**
   * Go to the official VS Code website: <https://code.visualstudio.com/download>
   * Download the installer for your operating system (Windows, macOS, Linux).
   * Run the installer and follow the prompts. Default settings are usually fine.
   * **Install Python Extension (Recommended):** Once VS Code is installed, open it. Go to the Extensions view (click the square icon on the left sidebar or press Ctrl+Shift+X). Search for "Python" (published by Microsoft) and click "Install". This provides features like code completion, debugging, and linting for Python.

### Project Setup and Running the Script

1. **Create a Project Folder:** Make a new folder on your computer specifically for this exercise (e.g., grover\_tiling\_project).
2. **Save Files:**
   * Go to <https://github.com/OliverSchiffmann/LearningGroversForEngDes> and download all the files.
   * Save the Python code as grover\_tiling\_simplified.py inside the folder you just created.
   * Save the requirements text as requirements.txt in the *same* folder.
3. **Open Folder in VS Code:**
   * Launch VS Code.
   * Go to File -> Open Folder...
   * Navigate to and select the project folder you created (e.g., grover\_tiling\_project).
4. **Create and Activate Virtual Environment:**
   * Open a terminal *within* VS Code: Go to Terminal -> New Terminal.
   * **Create:** In the terminal panel that appears at the bottom, type the following command and press Enter:
     + python -m venv venv (use python3 -m venv venv if python doesn't point to your Python 3 installation on Mac/Linux)
     + This creates a subfolder named venv containing a private copy of Python.
   * **Activate:** Run the activation script. The command depends on your operating system and shell:
     + **Windows (Command Prompt):** .\venv\Scripts\activate and press Enter.
     + **Windows (PowerShell):** .\venv\Scripts\Activate.ps1 and press Enter. (If you get an error about execution policies, you might need to run Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope Process first and confirm).
     + **Mac/Linux (bash/zsh):** source venv/bin/activate and press Enter.
   * You should see (venv) appear at the beginning of your terminal prompt, indicating the virtual environment is active.
5. **Install Libraries:**
   * Make sure the (venv) prompt is visible in the VS Code terminal.
   * Type the following command and press Enter:
     + pip install -r requirements.txt
   * This command reads the requirements.txt file and installs Qiskit, Pandas, and any other listed libraries into your active virtual environment.
6. **Exercise: Fixing the Grover's Algorithm Script (grover\_tiling\_fill\_blanks.py)**

* **Objective:** Your first main task is to get a Python script working that uses Grover's quantum algorithm to solve our simple tiling problem. You've been given the script grover\_tiling\_fill\_blanks.py which contains the core quantum logic, but it has some errors and missing pieces that need fixing! Your goal is to correct the script so it runs successfully and produces the correct results file.
* **Prerequisites:** By now, you should have completed Steps 3-7 above. This means you have:
  + Created a project folder for this exercise.
  + Saved both grover\_tiling\_fill\_blanks.py and requirements.txt inside that folder.
  + Opened the project folder in VS Code.
  + Created and activated a Python virtual environment (you should see (venv) at the start of your VS Code terminal prompt).
  + Installed the necessary libraries by running pip install -r requirements.txt.
* **Your Task:** The provided script, grover\_tiling\_fill\_blanks.py, contains **10 intentional errors or omissions ("blanks")**. These are marked in the code with comments like # --- BLANK X --- (where X is a number from 1 to 10). To complete this exercise, you need to:
  + Open the grover\_tiling\_fill\_blanks.py file in your VS Code editor.
  + Carefully read through the code, paying special attention to the lines marked with # --- BLANK X ---.
  + Use the surrounding code, comments, and hints (if provided) to understand what mistake has been made or what code is missing.
  + Edit the script to correct the errors. The blanks cover a variety of issues, including basic Python syntax, how to use Qiskit functions for setting up quantum registers and applying gates, and implementing the specific logic steps of Grover's algorithm (like the Oracle and Diffuser).
* **Expected Outcome:** Once you have correctly fixed all 10 blanks, the script should run properly. When you execute it (as described in Step 9), you should see the following happen:
  + The script runs from start to finish without any error messages.
  + Status updates (like "Initializing state...", "Building Oracle...", "Running simulation...") are printed to the VS Code terminal.
  + The Top 10 results (tile placements with the highest frequency) are printed to the terminal.
  + An Excel file named grover\_tiling\_results.xlsx is created in your project folder, containing the full list of results found by the algorithm.

1. **Run the Python Script:**
   * Ensure the (venv) prompt is still active in the terminal.
   * Type the following command and press Enter:
     + python grover\_tiling\_simplified.py (or python3 grover\_tiling\_simplified.py if needed)
2. **Check the Output:**
   * The script will print status messages to the VS Code terminal as it runs.
   * When finished, it will create an Excel file (e.g., grover\_tiling\_results.xlsx) inside your project folder.
   * Open this Excel file using a spreadsheet program (like Microsoft Excel, Google Sheets, LibreOffice Calc). You should see the possible tile placements and their measured frequencies (or probabilities). The tile pairs with the highest frequencies represent the most likely valid solutions found by the quantum algorithm.

## (Extension) Plotting Results for Visualisation

### Prerequisites:

1. You need the grover\_tiling\_results.xlsx file produced by the (fixed) Grover script (grover\_tiling\_fill\_blanks.py).
2. You **must edit** the plot\_grover\_results\_blanks.py script: Find the file\_path variable near the bottom (in the if \_\_name\_\_ == "\_\_main\_\_": section) and replace the placeholder path with the actual path to the *folder* containing your grover\_tiling\_results.xlsx file. Ask for help if you're unsure how to find this path on your computer.
3. Ensure you have installed the necessary Python packages using the provided requirements.txt file in your environment.

**Your Task:** Just like the first script, this plotting script isn't quite right! It contains **5 intentional errors or "blanks"** that prevent it from running correctly or producing the best possible plots.

Your goal is to:

* Open plot\_grover\_results\_blanks.py in VS Code (or your preferred editor).
* Locate the comments marked # --- BLANK X --- (where X is a number from 1 to 5).
* Read the code and comments around each blank to understand what might be wrong. The issues in this script relate to reading data correctly, setting up the 3D plots properly, mapping data accurately, and controlling the script's flow.
* Modify the code to fix the errors.

**Expected Outcome:** Once you have successfully fixed all 5 blanks, running the plot\_grover\_results\_blanks.py script should:

1. Load the data from your Excel file without errors.
2. Print some summary information to the terminal (like data shape and total frequency).
3. Display **two** separate 3D plot windows on your screen *at the same time*.
4. Print a message in the terminal asking you to press Enter.
5. Wait until you press Enter in the terminal.
6. Automatically close *both* plot windows after you press Enter.

Below is a screenshot of one of the plots created by a fully corrected script:

A graph of a bar graph

AI-generated content may be incorrect.