TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 23, No. 5, October 2025, pp. 1247∼1257
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v23i5.26929

Realization of Bernstein-Vazirani quantum algorithm in an interactive educational game

David Gosal, Timothy Rudolf Tan, Yozef Tjandra, Hendrik Santoso Sugiarto
Department of IT and Big Data Analytics, Faculty of Science and Engineering, Calvin Institute of Technology, Jakarta, Indonesia

Article Info

Article history:

Received Feb 21, 2025
Revised Jun 3, 2025
Accepted Aug 1, 2025

Keywords:

Bernstein-Vazirani algorithms
Gamification
Interactive educational game
Quantum algorithms
Quantum contextualization

ABSTRACT

Quantum algorithms are celebrated for their computational superiority over classical counterparts, yet they pose significant learning challenges for non-physics audiences. Among these, the Bernstein-Vazirani (BV) algorithm stands out for its quantum speedup by efficiently identifying a secret binary string. However, the accessibility of such algorithms remains constrained by their inherent technical complexity. To address this educational gap, this paper introduces a gamified, web-based tool that innovatively reinterprets the BV algorithm's complex mathematical settings through an into engaging scenario of identifying broken lamps. Players assume the role of an investigator, utilizing both classical and quantum solvers to identify faulty lamps with minimal queries. By transforming the BV algorithm into an intuitive gameplay experience, the tool helps reducing technical barriers, making quantum concepts much more comprehensible for educators and students than traditional methods that demand rigorous mathematical understanding. Developed using Qiskit, IBM's Python package for quantum computation, and deployed via Flask, a popular Python microframework for building web applications, the game effectively simplifies complex quantum algorithms while demonstrating the practical applications of quantum speedup. This contribution advances quantum education by merging technical depth with interactive design, fostering a broader understanding of quantum principles and inspiring new innovations in gamified learning.

Corresponding Author:

Hendrik Santoso Sugiarto
Department of IT and Big Data Analytics, Faculty of Science and Engineering, Calvin Institute of Technology
Calvin Tower RMCI, St. Industri Raya Kav 1 Blok B14, Kemayoran, Jakarta 10610, Indonesia
Email: hendrik.sugiarto@calvin.ac.id

1. INTRODUCTION

Quantum computing leverages principles from quantum physics such as superposition and entanglement to address complex problems with potential capabilities beyond classical computing. Groundbreaking algorithms like Shor's algorithm for period finding of prime integer factorization [1] and Grover's algorithm, which has been applied to quantum key search in cryptographic systems like advanced encryption standard (AES) and low multiplicative complexity (LowMC) [2] have historically showcased the potential computational power of quantum algorithms. These breakthroughs have ignited the development of quantum algorithms in various domains, including optimization [3], [4], machine learning [5]-[7] scientific simulation [8], and cryptography [9]. The realization of quantum supremacy, notably demonstrated by Google in 2019 [10] and 2024 [11], underscored this potential by solving a problem in a practical running time that would take septillion years of computation for classical computers. Moreover, the recent breakthrough in Majorana quantum chips by Microsoft Azure Quantum [12] significantly enhances the prospects of quantum technology, enabling more stable and scalable quantum computing.

Among the many quantum algorithms, the Bernstein-Vazirani (BV) algorithm [13] stands out as a famous foundational example showcasing quantum speedup through its elegant problem-solving approach. For a detailed explanation on the algorithm, one could consult many well-known references [14]. It addresses the problem of identifying a secret binary string using polynomially fewer queries compared to classical methods, making it a compelling illustration of quantum superiority. Furthermore, various literature had shown the algorithm's application in terms of information security [15], [16]. Various implementations of the BV algorithm on quantum hardware have been explored, for example in trapped ions [17], [18] and superconductor devices [19]. Classical simulations of the BV algorithm are also available across platforms, such as web-based tools [20] and mobile applications [21], designed to demonstrate its quantum principles. However, these resources are often targeted at researchers and experts, requiring prior technical knowledge, making them insufficient to engage the general public.

Despite the increasing availability of quantum algorithm demonstrations on classical devices [22]-[24], many existing tools remain heavily focused on circuit visualization and intricate mathematical formulations, making them less engaging to broader audiences. This gap in user-friendly educational tools limits the potential to inspire and train the next generation of scientists who can integrate quantum technologies, as highlighted in efforts such as [25]. Gamified applications have shown promise in breaking down technical barriers in science, technology, engineering, and mathematics (STEM) education [26], especially in making quantum concepts intuitive and interactive, and equipping educators with tools to introduce quantum technologies even at primary and secondary school levels [27]. Notable conceptualizations of quantum games include quantum chess [28] and simulations of quantum error correction [29]. Furthermore, hackathons, game jams, and student projects from various countries have produced diverse quantum games aimed at educating the public about the fundamentals of quantum mechanics and its applications [30]. However, while these efforts contribute meaningfully to quantum outreach, none explicitly focus on the contextualization and pedagogical unpacking of the BV algorithm. To address this problem, our work introduces a gamified realization of the BV algorithm, designed to convey its core principles and illustrate quantum speedup in an engaging and accessible format, bridging the gap between technical sophistication and public understanding.

This work introduces an innovative web-based interactive educational game that contextualizes the BV algorithm through a relatable scenario involving n broken lamps (corresponding to the n-digit secret binary string). In this game, players act as investigators, querying an oracle (corresponding to the special binary function in the BV algorithm's oracle setting) to determine which lamp is faulty. While performing the classical logical deduction requires several queries, the game would demonstrate the excellence of the BV quantum algorithm to obtain the correct answer with only a single query from the oracle based on quantum principles. By embedding this concept within a familiar narrative and intuitive gameplay, the intricate mathematical formalism can be avoided and thus the game simplifies the BV algorithm's technical concepts into an engaging and intuitive format, making quantum computing accessible to a broader audience. By focusing on user-friendly design and contextual gameplay, the game bridges the gap between technical demonstrations and public education, contributing to both the growing body of quantum educational resources as well as the development of accessible quantum computing demonstrations. The remainder of this paper discusses the design and implementation of the game, its educational objectives, and its potential impact in making quantum computing more comprehensible and engaging.

2. RESEARCH METHOD

The research approaches include reviews of the literature, design, and development of the web-based game application. The design of the quantum algorithm implementation is described using flowcharts. In this research, the quantum error correction and ancilla qubits are not taken into account. The web-app realization of BV's algorithm is done using Qiskit library and Flask. The web-app is deployed through DOM Cloud where the web can be accessed from any computer. This web-app compares how human, classical computer and quantum computer solves the problem. In this section, we first describe the BV problem, and then explain how to solve it utilizing the classical and quantum algorithm.

2.1. Bernstein-Vazirani problem

Let $s = s_0s_1 \ldots s_{n-1}$ be an n-bit binary string and f be a Boolean function $f : \{0,1\}^n \to \{0,1\}$ which depends on s, defined as:

$$f(x) = s \cdot x = s_0x_0 + s_1x_1 + \ldots + s_{n-1}x_{n-1} \mod 2,$$

where $x_0, \ldots, x_{n-1}$ are the bits of x. In the BV problem, f is called the oracle and the problem objective is to find s by leveraging the oracle's outputs without knowing its implementation details. In this setting, one must astutely determine the binary string x to be asked to the oracle so that the query result $f(x)$ may bestow useful information regarding the secret number s.

In the best classical solution of this problem, one needs to consult the oracle at least n times. For instance, we first query $f(100\ldots0)$, which reveals s_0. Next, we query $f(010\ldots0)$ to find s_1, and so on, until $f(000\ldots1)$ reveals s_{n-1}. Thus, n oracle queries are necessary to find all bits of s. There is no way to reduce this number without introducing errors in the algorithm.

In the quantum case, the function $f(x) = s \cdot x$ is implemented using the unitary operator U_f, which acts on a system of $n+1$ qubits, defined as:

$$U_f|x\rangle|j\rangle = |x\rangle|j \oplus f(x)\rangle,$$

where $x \in \{0,1\}^n$, j is a bit, and $\oplus$ is the exclusive OR (XOR) operation (sum modulo 2). This operator uses two registers: the first of size n qubits and the second of size 1 qubit. Although U_f can be used as many times as needed, it is only used once in the BV algorithm.

In the original problem setting, the binary operations within the oracle function are abstract and lack a tangible interpretation beyond their use as a quantum demonstration tool. In this work, we propose a gamified contextualization that brings the technical implementation to life, making it more engaging and relatable. In our game, users are placed in a scenario involving multiple lamps, some of which are broken yet visually identical to the functional ones. The task is to identify the defective lamps by interacting with an oracle. The oracle can toggle the lamps' connection to an electrical source based on the user's query and evaluate whether each lamp is capable of lighting up. While the user cannot directly observe which lamps light up, the oracle provides feedback by revealing only the parity of the number of operational lamps. The specific details of how the oracle operates in this contextualized scenario are provided in Table 1. A more detailed explanation of the AND bitwise operator within the oracle contextualization is provided in Table 2.

Table 1. BV gamified contextualization

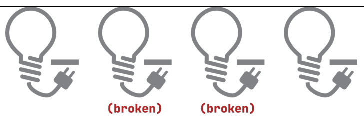
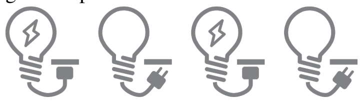
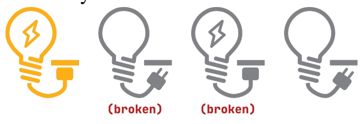
BV concept	Contextualization	Illustrative instance
Secret binary string s, each bit is unknown to the user.	n untoggled lamps: 0 corresponds to broken lamp and 1 to functional lamp (both appear to be off since they are untoggled).	 Secret word $s = 1001$; appears as 4 untoggled lamps that look alike.
Binary string x, queried to the oracle.	Electricity setup of the n lamps queried to the oracle: 0 means the lamp is untoggled and 1 toggled.	 Queried word $x = 1010$; appears as a configuration of toggling the lamps to electricity.
AND-bitwise-operator in the oracle applied to s and x.	The actual lights of each lamp's internal condition (either broken or functional) and its electricity availability (either toggled or not). See Table 2 for more explanations.	 Bitwise AND operation of s and x. Only functional and toggled lamps would realize factual light.
XOR operation.	The parity of the actual number of lamps with lights on: 0 means even and 1 odd.	Return 1 since there is 1 (odd) factual lamp with on condition.

Table 2. Contextualization of the AND bitwise operator between s and x inside the oracle function

AND	Untoggled ($x_i = 0$)	Toggled ($x_i = 1$)
Broken ($s_i = 0$)	Lamp is off ($s_i x_i = 0$)	Lamp is off ($s_i x_i = 0$)
Functional ($s_i = 1$)	Lamp is off ($s_i x_i = 0$)	Lamp is on ($s_i x_i = 1$)

2.2. Classical solver

Figure 1 outlines the flow diagram of a classical computer which deterministically infers the secret number based on interactions with the oracle. In this section, we only describe the best classical algorithm to solve BV problem. The process begins with the user inputting the length of the secret number (n) and initializing a counter variable (i) to be zero. For each bit position i (with $1 \leq i \leq n$), a specific pattern, `asked_bit`, is defined to be a binary string of n length where the i-th bit is 1 while all the others are 0. This pattern is sent to the oracle, and the oracle would return a response indicating whether the result is even or odd. Based on the oracle's response, the solver would guess the i-th bit iteratively. Specifically, the i-th bit of the guessed sequence (`guessed_bit`) is equal to 0 if the response is even, and 1 if it is odd. The counter i is then incremented, and the process is repeated for all bit positions until the guessed sequence reaches the specified length n. Finally, the complete `guessed_bit` sequence is returned as the output. This algorithm efficiently determines the secret number one bit at a time by leveraging the oracle's feedback.

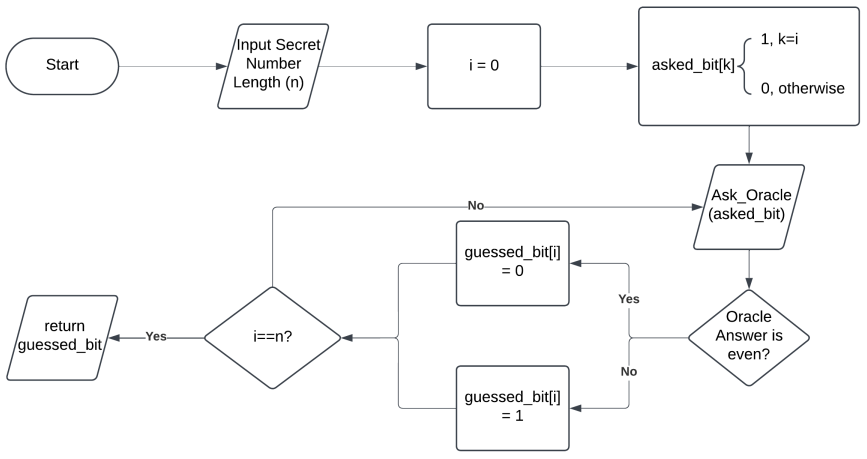


Figure 1. Classical solver flowchart

2.3. Quantum solver

In this section, we apply the original BV algorithm [13] inside our quantum solver. Figure 2 describes the process of a quantum solver to determine the guessed bit sequence. The algorithm begins with the user providing the length of the secret number (n). An initial quantum state is prepared as the input (`asked_bit`), using n qubits all initialized to zero. The quantum oracle is queried using this state, producing a measurement that reflects the probability distribution of potential solutions. From the measurement results, the `guessed_bit` sequence is determined by selecting the outcome with the highest probability (using argmax). Finally, the `guessed_bit` sequence is returned as the solution. This process leverages quantum computation to efficiently explore and identify the correct sequence.

On the other hand, the quantum oracle is implemented separately, as described in Figure 2. The process begins by preparing the initial quantum state $|0\rangle^{\otimes n}|1\rangle$. In this state, the first n qubits are initialized to $|0\rangle$, and the $(n+1)$-th qubit, which serves as an auxiliary qubit, is initialized to $|1\rangle$. Next, the Hadamard gates H are applied to all $n+1$ qubits, denoted by $H^{\otimes(n+1)}$, transforming the system into an equal superposition of all possible states. The oracle is then applied as a unitary operator U_f, which incorporates the information regarding the secret number into the quantum system. The unitary operator U_f is implemented using controlled-NOT (CNOT) gates, that entangle the qubits and encode the oracle's logic. Following this, Hadamard gates are applied again to all qubits, represented as $H^{\otimes(n+1)}$, creating quantum interference patterns that reveal the solution to the encoded problem. Finally, the first n qubits are measured in the computational basis, yielding the secret number $s_0 s_1 \ldots s_{n-1}$.

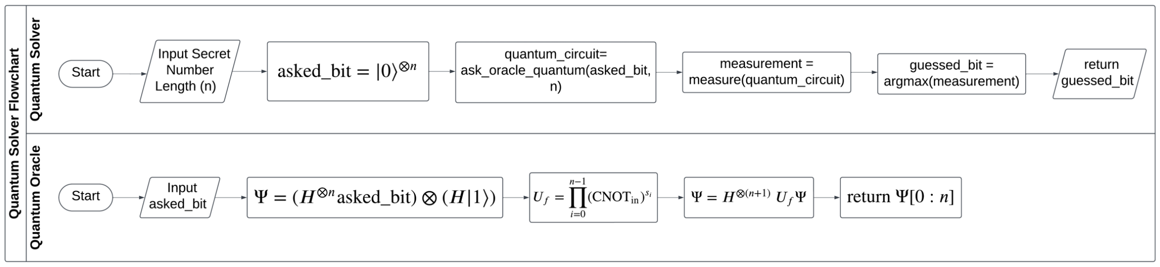

Quantum Solver Flowchart
Quantum Solver
Start
Input Secret Number Length (n)
asked_bit = $|0\rangle^{\otimes n}$
quantum_circuit= ask_oracle_quantum(asked_bit, n)
measurement = measure(quantum_circuit)
guessed_bit = argmax(measurement)
return guessed_bit
Quantum Oracle
Start
Input asked_bit
$\Psi = (H^{\otimes n}\text{asked\_bit}) \otimes (H|1\rangle)$
$U_f = \prod_{i=0}^{n-1} (\text{CNOT}_{\text{in}})^{s_i}$
$\Psi = H^{\otimes(n+1)}\, U_f \Psi$
return $\Psi[0:n]$

Figure 2. Quantum solver flowchart

2.4. Web-app design

Figure 3 illustrates the flowchart of the web application. The user begins at the index page, which serves as the starting point. From there, they navigate to the homepage where the user is required to input several parameters, such as the length of the secret number, the way the secret number is determined, and the type of solver to be used. The application provides two ways of determining the secret number: 1) the secret number can be manually predetermined by the user, or 2) it can also be randomly generated. Once all the necessary inputs had been entered, the user is directed to the appropriate game page. Upon completing the game, the user is presented with two options: they can either choose to play again, which redirects them back to the homepage, or exit, which returns them to the index page.

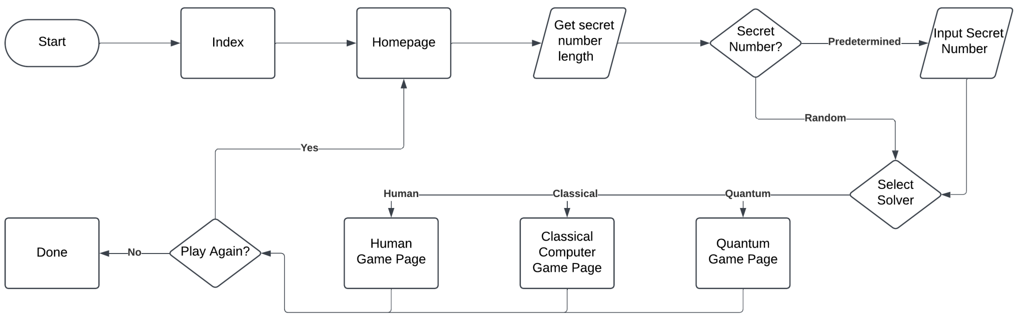
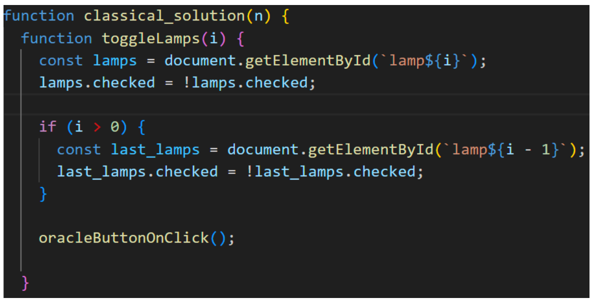


Figure 3. Web-App design flowchart

3. RESULT AND DISCUSSION

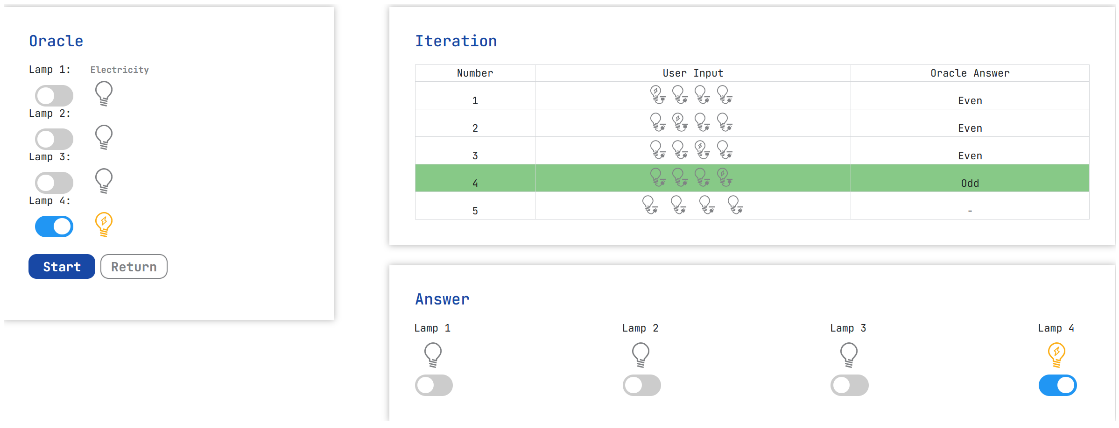
3.1. Classical solver web-app realization

The implementation of the classical solver using Python code is provided in Figure 4. Specifically, the solver makes n queries to the oracle, where n denotes the length of the secret number. For each query, the oracle evaluates the current guess by returning whether the result is ”even” or ”odd.” If the oracle returns an ”even” response, the solver appends a 0 to the guessed number. Conversely, if the oracle responds with ”odd,” the solver appends a 1. This method is computationally efficient, with a time complexity of $O(n)$, as it requires only n queries to the oracle to determine the secret number. In contrast, a naive brute-force approach would involve generating and checking all $O(2^n)$ possible binary strings of length n, making it exponentially slower as n increases. The $O(n)$ complexity of the classical solver highlights its advantage in efficiency, particularly for large values of n, where the brute-force method becomes impractical. Figure 5 illustrates the solution display of the classical solver. Figure 5(a) presents the result for the secret number $s = 0001$, while Figure 5(b) demonstrates the outcome for the secret number $s = 111100$. From the screenshots, it is evident that the solver is able to iteratively guess each correct bit by asking the oracles n times.



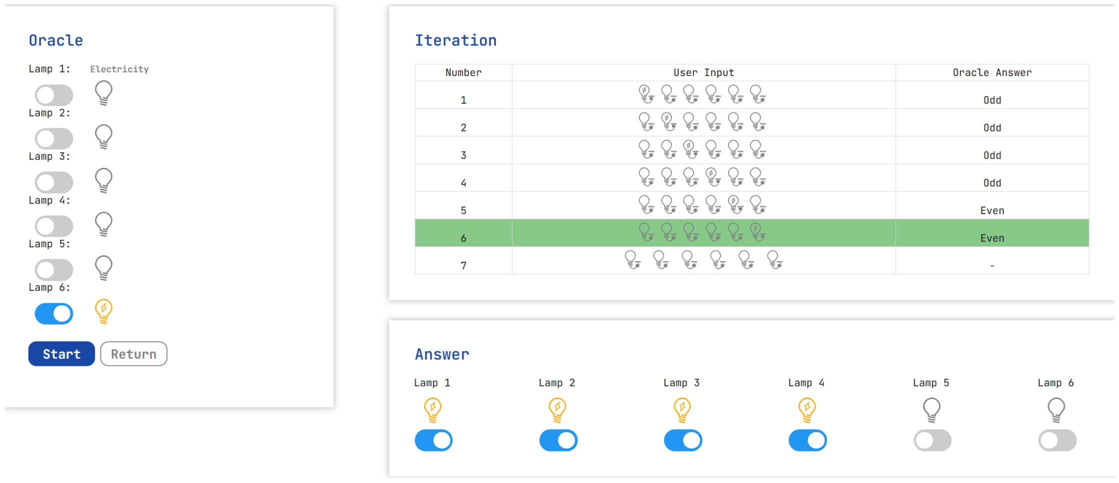
```
function classical_solution(n) {
  function toggleLamps(i) {
    const lamps = document.getElementById(`lamp${i}`);
    lamps.checked = !lamps.checked;

    if (i > 0) {
      const last_lamps = document.getElementById(`lamp${i - 1}`);
      last_lamps.checked = !last_lamps.checked;
    }

    oracleButtonOnClick();

  }
```

Figure 4. Classical code



Number	User Input	Oracle Answer
1	● ○ ○ ○	Even
2	○ ● ○ ○	Even
3	○ ○ ● ○	Even
4	○ ○ ○ ●	Odd
5	○ ○ ○ ○	-

(a)



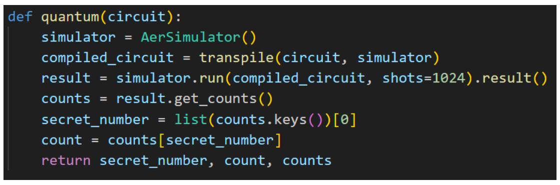
Number	User Input	Oracle Answer
1	● ○ ○ ○ ○ ○	Odd
2	○ ● ○ ○ ○ ○	Odd
3	○ ○ ● ○ ○ ○	Odd
4	○ ○ ○ ● ○ ○	Odd
5	○ ○ ○ ○ ● ○	Even
6	○ ○ ○ ○ ○ ●	Even
7	○ ○ ○ ○ ○ ○	-

(b)

Figure 5. Realization of the classical solver for (a) $s = 0001$ and (b) $s = 111100$

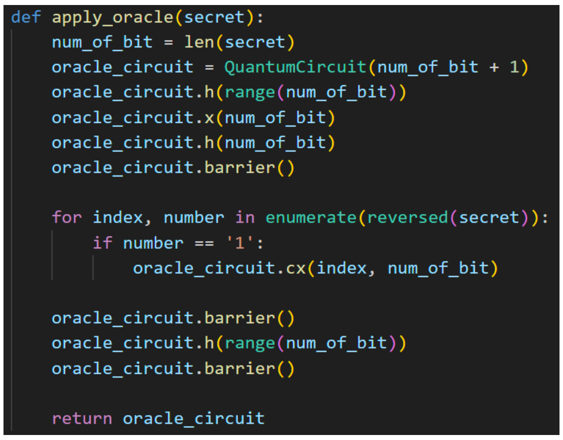
3.2. Quantum solver web-app realization

In this section, we expose how the quantum algorithm is simulated using classical devices. In our works, all quantum simulation computations are performed using IBM's Qiskit python package. The implementation of the BV algorithm using the Qiskit library simulation is described in Figure 6. Specifically, in Figure 6(a), the function begins by determining the length of the secret binary number using `len(secret)`. A `QuantumCircuit` is then initialized with `length + 1` qubits. Next, the `apply_oracle` function is called to construct a subcircuit (the oracle) that encodes the secret into the quantum system. Afterward, a measurement operation is applied to the first `length` qubits. Finally, the function returns the `QuantumCircuit` object that has been measured. Meanwhile, in Figure 6(b), the quantum circuit is again initialized with `length + 1` qubits. The Hadamard gate (`.h`) is applied to all qubits to create a superposition of all possible states. The oracle is implemented using a series of CNOT gates (`.cx`) applied among bits according to the secret number information. Consequently, another Hadamard gate (`.h`) is applied to the `length` qubits, completing the quantum operation. The function returns the quantum circuit with all applied operations ready for measurement. Once the circuit has been fully constructed and prepared to encode the secret number, the function `quantum()` (as seen in Figure 6(c)) takes it and uses a quantum simulator to execute the circuit and retrieve the results. First, the function initializes an instance of AerSimulator, which is a classical simulation tool provided by Qiskit for running quantum circuits without needing access to a physical quantum computer. The circuit is then transpiled for the simulator using `transpile(circuit, simulator)`, optimizing the circuit's instructions to match the simulator's requirements.



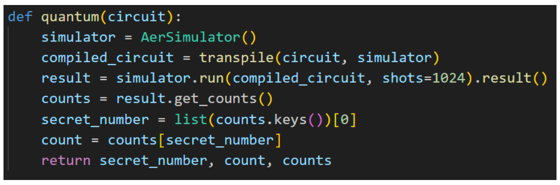
```python
def quantum(circuit):
    simulator = AerSimulator()
    compiled_circuit = transpile(circuit, simulator)
    result = simulator.run(compiled_circuit, shots=1024).result()
    counts = result.get_counts()
    secret_number = list(counts.keys())[0]
    count = counts[secret_number]
    return secret_number, count, counts
```

(a)



```python
def apply_oracle(secret):
    num_of_bit = len(secret)
    oracle_circuit = QuantumCircuit(num_of_bit + 1)
    oracle_circuit.h(range(num_of_bit))
    oracle_circuit.x(num_of_bit)
    oracle_circuit.h(num_of_bit)
    oracle_circuit.barrier()

    for index, number in enumerate(reversed(secret)):
        if number == '1':
            oracle_circuit.cx(index, num_of_bit)

    oracle_circuit.barrier()
    oracle_circuit.h(range(num_of_bit))
    oracle_circuit.barrier()

    return oracle_circuit
```

(b)

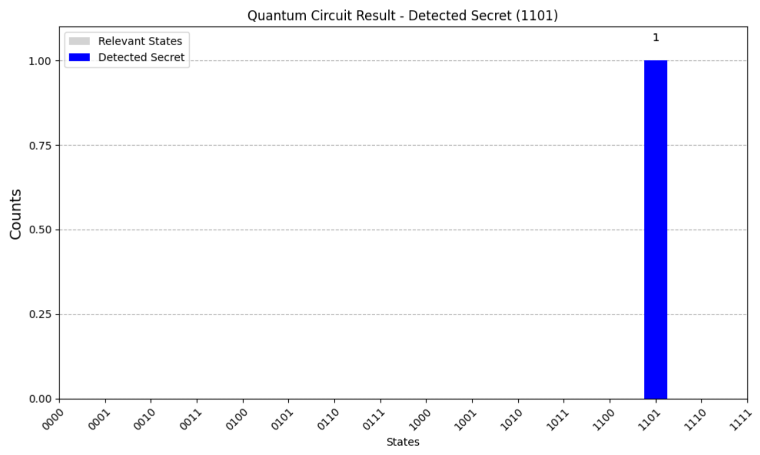


```python
def quantum(circuit):
    simulator = AerSimulator()
    compiled_circuit = transpile(circuit, simulator)
    result = simulator.run(compiled_circuit, shots=1024).result()
    counts = result.get_counts()
    secret_number = list(counts.keys())[0]
    count = counts[secret_number]
    return secret_number, count, counts
```

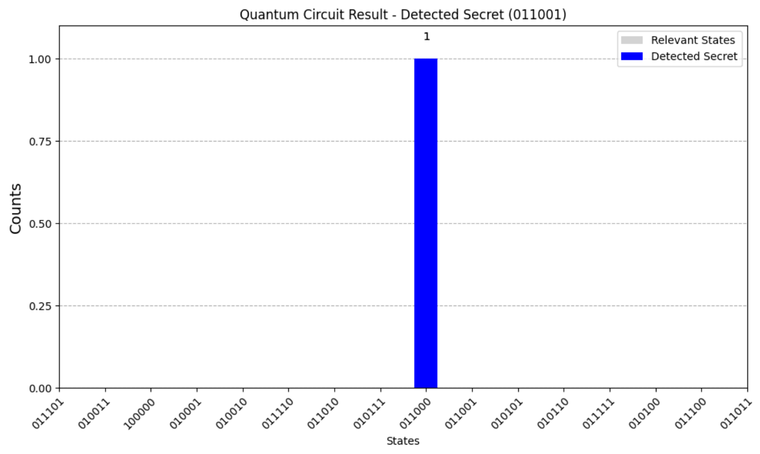
(c)

Figure 6. Quantum code of (a) quantum circuit initialization code, (b) quantum oracle code, and (c) quantum simulation code

The function then retrieves the secret number by acquiring the count values of each measurement simulation shot. It also retrieves the frequency of this result, stored in count. Finally, the function returns the secret number, count, and the full counts dictionary, effectively decoding the secret number from the quantum simulation and providing a record of the simulation's outcome. This completes the process of encoding the secret, running the circuit, and extracting the result. In Figure 7, the quantum solver could find the correct answer in one shot of the simulation. Figure 7(a) displays the plot for the secret number $s = 1101$, while Figure 7(b) presents the plot for the secret number $s = 011001$. The plot shows that no matter the length of the secret number, the result will always be the same, pointing to the correct answer.



(a)



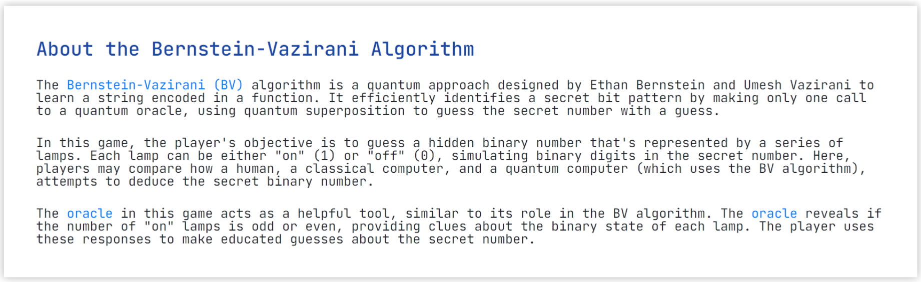
(b)

Figure 7. Measurement histograms; (a) plot secret number 1101 and (b) plot secret number 011001

This shows that the advantage of quantum algorithms, such as the BV algorithm, is their $O(1)$ complexity. This means that the algorithm requires only a single execution to determine the secret binary number, regardless of its length. In comparison, classical algorithms for solving the same problem have a complexity of $O(n)$, where n is the length of the secret binary number. In other words, the best classical algorithm requires a linear iteration over each bit to find the result. Therefore, quantum algorithms provide a significant efficiency improvement, particularly for problems with large-scale inputs, making them a highly superior solution in certain scenarios.

3.3. Overall web-app realization

The web application brings the BV algorithm to life through an interactive game where players decode a binary ”secret number,” creatively represented as a series of lamps. Built using Flask and hosted on DOM Cloud at citbvgame.domcloud.dev, the app allows players to customize their experiences by selecting the secret number’s length, whether it is randomly generated or predefined, and the solver type: human, classical computer, or quantum computer. Figure 8 provides snippets of the web application interface. Figure 8(a) displays the ”about” section, providing an overview of the web app, while Figure 8(b) illustrates the ”rules and input” interface, detailing user instructions and input options. This interactive setup highlights the distinct approaches taken by humans, classical computers, and quantum computers to solve the problem, showcasing the efficiency and elegance of quantum computing. By combining theoretical concepts with an intuitive, game-based interface, the app makes learning about quantum algorithms engaging and accessible.



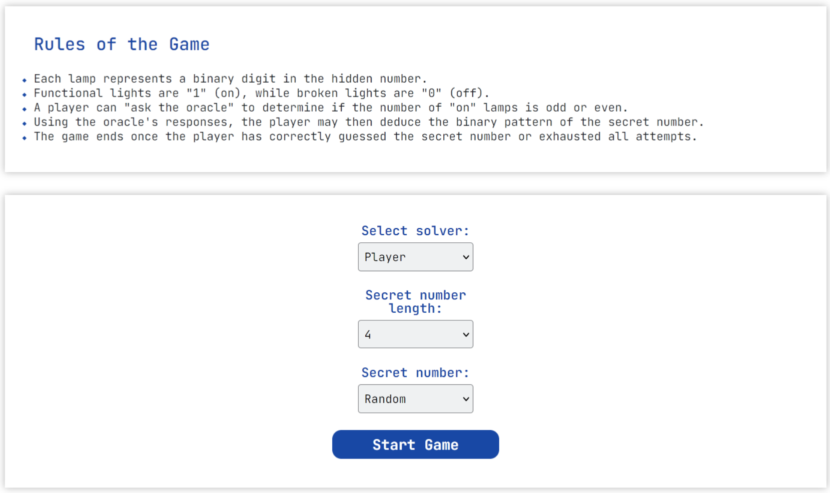
About the Bernstein-Vazirani Algorithm

The Bernstein-Vazirani (BV) algorithm is a quantum approach designed by Ethan Bernstein and Umesh Vazirani to learn a string encoded in a function. It efficiently identifies a secret bit pattern by making only one call to a quantum oracle, using quantum superposition to guess the secret number with a guess.

In this game, the player's objective is to guess a hidden binary number that's represented by a series of lamps. Each lamp can be either "on" (1) or "off" (0), simulating binary digits in the secret number. Here, players may compare how a human, a classical computer, and a quantum computer (which uses the BV algorithm), attempts to deduce the secret binary number.

The oracle in this game acts as a helpful tool, similar to its role in the BV algorithm. The oracle reveals if the number of "on" lamps is odd or even, providing clues about the binary state of each lamp. The player uses these responses to make educated guesses about the secret number.

(a)



Rules of the Game

- Each lamp represents a binary digit in the hidden number.
- Functional lights are "1" (on), while broken lights are "0" (off).
- A player can "ask the oracle" to determine if the number of "on" lamps is odd or even.
- Using the oracle's responses, the player may then deduce the binary pattern of the secret number.
- The game ends once the player has correctly guessed the secret number or exhausted all attempts.

Select solver: Player

Secret number length: 4

Secret number: Random

(b)

Figure 8. Web application interface; (a) about and (b) rules and input

4. CONCLUSION

This paper presents a gamified implementation of the BV algorithm to make quantum algorithms more understandable and enjoyable. By narrating the algorithm into a relatable scenario, this game effectively bridges the technical gap between the quantum algorithm and the general public. Players experience the fundamental principles of quantum speed-up in an intuitive and engaging manner, making the core concepts of quantum computing more approachable to diverse audiences, including educators, students, and enthusiasts with little or no technical background. The implementation leverages the Qiskit-Aer library for simulation and is effectively deployed on DOM Cloud using the Flask framework, illustrating a practical pathway for integrating quantum algorithms into modern web-based applications. Future developments could expand this gamified education approach on other more complex quantum algorithms, such as Grover's search and Shor's factoring algorithms. Quantum operations such as diffuser and reflector in Grover algorithm; period finding and phase estimation in Shor algorithm are intuitively difficult to learn, and an interactive game may help bridge this education gap. This work contributes to quantum education by transforming a difficult quantum algorithm into a fun interactive game. Through various tools like this game, we can support the global effort to prepare for the wider adoption of these technologies by training a new generation of quantum innovators.

FUNDING INFORMATION

This work was funded by PT. Lancs Arche Consumma (MOU.003/CIT/XI/2022).

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	**C**	**M**	**So**	**Va**	**Fo**	**I**	**R**	**D**	**O**	**E**	**Vi**	**Su**	**P**	**Fu**
David Gosal			✓	✓		✓	✓	✓	✓	✓	✓			
Timothy Rudolf Tan			✓	✓		✓	✓	✓	✓	✓	✓			
Yozef Tjandra		✓			✓	✓	✓		✓	✓		✓	✓	✓
Hendrik Santoso Sugiarto	✓				✓	✓	✓		✓	✓		✓	✓	✓

C	: **C**onceptualization	I	: **I**nvestigation	Vi	: **Vi**sualization
M	: **M**ethodology	R	: **R**esources	Su	: **Su**pervision
So	: **So**ftware	D	: **D**ata Curation	P	: **P**roject Administration
Va	: **Va**lidation	O	: Writing - **O**riginal Draft	Fu	: **Fu**nding Acquisition
Fo	: **Fo**rmal Analysis	E	: Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] H. Y. Wong, ”Shor’s Algorithm,” *Introduction to Quantum Computing. Springer*, Cham, 2024, pp. 289-298, doi: 10.1007/978-3-031-36985-8_29.

[2] S. Jaques, M. Naehrig, M. Roetteler, and F. Virdia, “Implementing grover oracles for quantum key search on aes and lowmc,” in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30*, Springer, 2020, pp. 280–310, doi: 10.1007/978-3-030-45724-2_10.

[3] J. R. Finžgar, A. Kerschbaumer, M. J. Schuetz, C. B. Mendl, and H. G. Katzgraber, “Quantum-informed recursive optimization algorithms,” *PRX Quantum*, vol. 5, no. 2, p. 020327, 2024.

[4] A. Abbas *et al.*, “Challenges and opportunities in quantum optimization,” *Nature Reviews Physics*, pp. 1–18, 2024, doi: 11245.1/e3a25092-f400-4b62-9456-672855c76e8c.

[5] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, “Challenges and opportunities in quantum machine learning,” *Nature computational science*, vol. 2, no. 9, pp. 567–576, 2022, doi: 10.1002/wcms.1580.

[6] Y. Tjandra and H. S. Sugiarto, “An evolutionary algorithm design for pauli-based quantum kernel classification,” in *VLDB Workshops*, 2023.

[7] Y. Tjandra and H. S. Sugiarto, “Metaheuristic optimization scheme for quantum kernel classifiers using entanglement-directed graphs,” *ETRI Journal*, vol. 46, no. 5, pp. 793–805, 2024, doi: 10.4218/etrij.2024-0144.

[8] M. Motta and J. E. Rice, “Emerging quantum computing algorithms for quantum chemistry,” *Wiley Interdisciplinary Reviews: Computational Molecular Science*, vol. 12, no. 3, p. e1580, 2022, doi: 10.1002/wcms.1580.

[9] R. Wolf, ”Quantum Key Distribution,” in *Lecture Notes in Physics, vol. 988. Cham: Springer International Publishing*, 2021, doi: 10.1007/978-3-030-73991-1.

[10] F. Arute *et al.*, “Quantum supremacy using a programmable superconducting processor,” *Nature*, vol. 574, no. 7779, pp. 505–510, Oct. 2019, doi: 10.1038/s41586-019-1666-5.

[11] N. Zobrist *et al.*, “Quantum error correction below the surface code threshold,” *Nature*, vol. 638, no. 8052, pp. 920–926, Feb. 2025, doi: 10.1038/s41586-024-08449-y.

[12] M. Aghaee *et al.*, “Interferometric single-shot parity measurement in InAs–Al hybrid devices,” *Nature*, vol. 638, no. 8051, pp. 651–655, Feb. 2025, doi: 10.1038/s41586-024-08445-2.

[13] C. Easttom, *Quantum Computing Fundamentals, [First edition]*. Addison-Wesley Professional, 2023.

[14] R. Portugal, “Basic quantum algorithms,” *arXiv*, 2022, doi: 10.48550/arXiv.2201.10574.

[15] K. Nagata, D. N. Diepm, A. Farouk, and T. Nakamura, *Simplified quantum computing with applications*. IOP Publishing, 2022, doi: 10.1088/978-0-7503-4700-6.

[16] H. Xie and L. Yang, “Using bernstein–vazirani algorithm to attack block ciphers,” *Designs, Codes and Cryptography*, vol. 87, pp. 1161–1182, 2019, doi: 10.1007/s10623-018-0510-5.

[17] S. Fallek, C. Herold, B. McMahon, K. Maller, K. Brown, and J. Amini, “Transport implementation of the bernstein–vazirani algorithm with ion qubits,” *New Journal of Physics*, vol. 18, no. 8, p. 083030, 2016, doi: 10.1088/1367-2630/18/8/083030.

[18] K. Wright *et al.*, “Benchmarking an 11-qubit quantum computer,” *Nature communications*, vol. 10, no. 1, p. 5464, 2019, doi: 10.1038/s41467-019-13534-2.

[19] B. Pokharel and D. Lidar, “Demonstration of algorithmic quantum speedup,” *Physical Review Letters*, vol. 130, no. 21, p. 210602, 2023, doi: 10.1103/PhysRevLett.130.210602.

[20] T. Chu, “Simulation and visualization of the bernstein-vazirani quantum protocol,” 2023, accessed on 11 December 2024, [Online]. Available: https://bernstein-vazirani.streamlit.app.

[21] S. Kim, "Bernstein-vazirani algorithm: quantum circuit for everyone!" [Online]. Available: https://apps.apple.com/vn/app/bernstein-vazirani-algorithm/id6457204574?platform=iphone.
[22] A. Chattopadhyay and V. Menon, "Fast simulation of grover's quantum search on classical computer," *arXiv*, 2020.
[23] A.-G. Tudorache, V.-I. Manta, and S. Caraiman, "Implementation of the bernstein-vazirani quantum algorithm using the qiskit framework," *Bulletin of the Polytechnic Institute of Iași. Electrical Engineering, Power Engineering, Electronics Section*, vol. 67, no. 2, pp. 31-40, 2021, doi: 10.2478/bipie-2021-0009.
[24] A. Wicaksana, A. Anthony, and A. W. Wicaksono, "Web-app realization of shor's quantum factoring algorithm and grover's quantum search algorithm," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, pp. 1319–1330, 2020, doi: 10.12928/telkomnika.v18i3.14755.
[25] S. M. Mackay, E. W. Tan, and D. S. Warren, "Developing a new generation of scientist communicators through effective public outreach," *Communications Chemistry*, vol. 3, no. 1, p. 76, 2020, doi: 10.1038/s42004-020-0315-0.
[26] Y. Gui, Z. Cai, Y. Yang, L. Kong, X. Fan, and R. H. Tai, "Effectiveness of digital educational game and game design in stem learning: a meta-analytic review," *International Journal of STEM Education*, vol. 10, no. 36, 2023, doi: 10.1186/s40594-023-00424-9.
[27] Z. C. Seskir, S. R. Goorney, and M. L. Chiofalo, "Educating to the "culture" of quantum technologies: A survey study on concepts for public awareness." *European Journal of STEM Education*, vol. 9, no. 1, p. 3, 2024, doi: 10.20897/ejsteme/14193.
[28] C. Cantwell, "Quantum chess: Developing a mathematical framework and design methodology for creating quantum games," *arXiv*, 2019, doi: 10.48550/arXiv.1906.05836.
[29] J. R. Wootton, "Getting the public involved in quantum error correction," *arXiv*, 2017, doi: 10.48550/arXiv.1712.09649.
[30] Z. C. Seskir *et al.*, "Quantum games and interactive tools for quantum technologies outreach and education," *Optical Engineering*, vol. 61, no. 8, pp. 081 809–081 809, 2022, doi: 10.1117/1.OE.61.8.081809.

BIOGRAPHIES OF AUTHORS

David Gosal is currently pursuing his Bachelor's degree in IT and Big Data Analytics education at Calvin Institute of Technology, Indonesia, since 2021. With a keen interest in data science, he is building his knowledge and skills in data-driven decision making, machine learning, and advanced analytical techniques. He aspires to contribute to the industry as a data scientist, leveraging his expertise to provide insights and solutions for real-world challenges. He is dedicated to expanding his horizons and is always eager to learn from others and collaborate on meaningful projects. He can be contacted at email: dgjy2019@gmail.com.

Timothy Rudolf Tan is currently pursuing a Bachelor's degree in IT and Big Data Analytics at Calvin Institute of Technology, Indonesia, since 2022. With a strong interest in human-computer interaction and software engineering, he is focused on enhancing user-centric design and building efficient software systems. As a student, he is committed to acquiring in-depth knowledge and practical skills to excel in these fields. He is eager to explore innovative technologies and collaborate on projects that bridge the gap between technology and user experience. He can be contacted at email: ttan12@students.calvin.ac.id.

Yozef Tjandra received his research Master's degree in Mathematics from Monash University, Australia, focusing on the area of probabilistic combinatorics. Since 2019, he is a lecturer in IT and Big Data Analytics Department in Calvin Institute of Technology, Indonesia. He has been working in various mathematical and computational related projects conducted for both educational and commercial purposes. His research interests range from enumerative combinatorics, applications of machine learning, optimization and quantum machine learning. He can be contacted at email: yozef.tjandra@calvin.ac.id.

Hendrik Santoso Sugiarto obtained his Ph.D. in Physics of Complex Systems from Nanyang Technological University. His research interests ranged from phase transition (in nature and society), nonlinear dynamics, complex network, probabilistic graphical modeling, deep learning, and quantum machine learning; in which he has published several research articles on those topics. He has various professional experiences: as a research scientist at a smart nation research center in Singapore, as a data scientist at the leading startup company in Indonesia, and as a visiting scholar at the largest research institute in Switzerland. Currently, he is the head of IT and Big Data Analytics Department in Calvin Institute of Technology, Indonesia. He can be contacted at email: hendrik.sugiarto@calvin.ac.id.