American Journal of Physics

PAPERS | JANUARY 01 2025

Introducing quantum computing to high school students with Grover's search algorithm

Special Collection: Celebrating the International Year of Quantum Science and Technology

Mark S. Hannum

Check for updates

Am. J. Phys. 93, 78–87 (2025)
https://doi.org/10.1119/5.0228847

View Online
Export Citation

Articles You May Be Interested In

Searching a quantum database with Grover's search algorithm

Am. J. Phys. (June 2021)

Designing Grover's quantum searching algorithm for 2-qubit and 3-qubit states of quantum systems

AIP Conf. Proc. (December 2024)

Similarity between Grover's quantum search algorithm and classical two-body collisions

Am. J. Phys. (January 2003)

25 February 2026 08:46:17

AAPT PHYSICS EDUCATION®

Introducing quantum computing to high school students with Grover's search algorithm

Mark S. Hannum[a)]
Thomas Jefferson High School for Science and Technology, Alexandria, Virginia 22312

(Received 15 September 2024; accepted 21 November 2024)

The noisy intermediate-scale quantum (NISQ) era is progressing rapidly and giving both physics researchers and quantum computing companies (and their investors) strong indications that we could be on the cusp of the next quantum revolution. In addition to pushing science forward, we cannot ignore the development of the future workforce. This paper provides a detailed description of a quantum computing activity used in a high school physics course that applies Grover's algorithm to solve Boolean satisfiability problems. The details of the algorithm are first presented before the activity is described. Based on informal feedback from students, a case is made that the described activity is an effective means of introducing the field of quantum computing to high school students that builds interest in the field to support the development of a future workforce. © *2025 Published under an exclusive license by American Association of Physics Teachers.*
https://doi.org/10.1119/5.0228847

I. INTRODUCTION

The need for a qualified quantum-enabled workforce[1,2] and a plan for how to educate it[3,4] have been articulated by the National Science Foundation (NSF), the Quantum Coordinating Office (QCO), and many other individuals and organizations. A large curricular reform is occurring at undergraduate institutions[5–8] and at the K-12 level.[9] This paper describes a classroom activity for high school students taught during the 2023–2024 school year as a part of a Quantum Mechanics course. The activity was also implemented in a more informal summer camp during the summer of 2024 at a different location. It prompts students to explore the fundamental constructs of quantum computing through an investigation using Grover's search algorithm. This algorithm provides a useful introduction to all of the fundamental concepts important to quantum computing like qubits, superposition, gates (operators), and the probabilistic nature of quantum information. The learning opportunity also attempts to motivate students with an engaging application and connections to a larger class of Boolean satisfiability problems.

II. STUDENT DESCRIPTION

Thomas Jefferson High School for Science and Technology is a public STEM Magnet High School* that draws students from the large and diverse geographic region of Northern Virginia. The students in the described course are mostly 12th grade students who have completed one year of Advanced Placement Physics C and are currently enrolled in a year-long quantum mechanics and electrodynamics course. The course follows a spins-first approach to teaching quantum and uses the McIntyre textbook.[10] Most students have not taken linear algebra, but they are well-versed in calculus. The class meets for just under four hours per week, and the described activity occurred around week seventeen of the course as the culminating activity of a two-week unit on quantum computing. There were 60 students in two sections, taught by two instructors.

Despite the academic credentials of these students, the activity described in Sec. IV is accessible to high school physics students with at least an algebra II background. This was demonstrated to be true during the summer of 2024 during a two-week-long summer camp experience where an additional 24 students completed the activity. The summer camp students were in the 10th, 11th, and 12th grades with most of them having completed a standard introductory year of physics, some at the Advanced Placement 1, 2, and C levels. This summer camp group of students more closely resembled motivated (but not necessarily advanced) physics students from around the country. The summer camp students had different mathematical backgrounds than the students in the formal class. Figure 1 contains student responses from a survey question about their previous math courses.

III. GROVER'S ALGORITHM

If the reader is already familiar with quantum computing and/or Grover's algorithm it is suggested they skip to Sec. IV, which describes the learning activity given to students. This section provides a detailed discussion of Grover's algorithm and a small example of its implementation.

Grover's algorithm was proposed as a quantum search algorithm able to extract an entry (or index) from an unsorted list of length N in approximately $\sqrt{N}$ operations.[11] This provides a quantum speed-up over classical linear searching, which is order N.[12] It is important for beginning students to understand that what we casually refer to as "searching" is probability manipulation. A better view of the algorithm is that it randomly selects one (or more) of the entries in the list to extract, but the algorithm stacks the probability of selection in favor of the entry we are searching for. Leveraging the superposition of quantum states and probability amplitude amplification, the algorithm is a staple of most quantum computing courses. The algorithm itself is easy to implement and has a well-documented usefulness, even in the noisy intermediate-scale quantum (NISQ) era.[13,14]

*Public magnet schools are schools that receive public funding, are often run by local school boards, but have special curriculum that allows students access to more advanced topics, resources, and experiences.

 © 2025 Published under an exclusive license by AAPT

25 February 2026 08:46:17

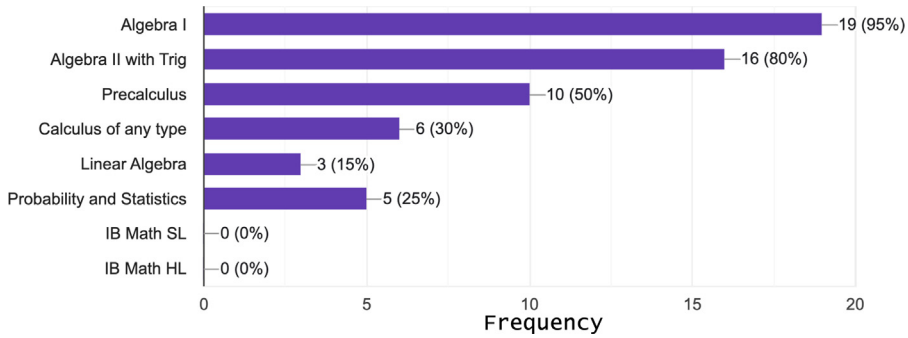


Fig. 1. Self-reported prior math courses from summer students.

A. The Heart of Grover's

The heart of Grover's algorithm is a computing tool called an Oracle. An Oracle, in both classical and quantum computing is a tool that changes computational problems from "finding a solution" to "confirming a solution." The use of Oracles is popular in theoretical computer science and algorithm development but is put to very practical use inside Grover's algorithm. A simple example of how an Oracle changes a computational problem is illustrated with a polynomial equation,

$$x^3 - 5x^2 - 12x + 36 = 0. \tag{1}$$

We could write a computer program that finds all solutions to Eq. (1). An implementation of the Newton–Raphson method[15] might be a valid approach, but that might not converge to all solutions. However, if we employ an Oracle, the problem changes to confirm if a value of x satisfies the equation. For example, writing a program that confirms that $x = 2$ is a solution to Eq. (1). This seems like it would be a terribly inefficient way to find all zeros of the polynomial as it would require an exhaustive search through all possible inputs. Even limiting the search to the set of integers on some closed domain could be time consuming. Classically, this is true, but as will be demonstrated later, Grover's algorithm makes use of quantum superposition to simultaneously pass all possible states into an Oracle operator, which then confirms which input states satisfy the logical conditions of the problem in one (giant) step.

B. The algorithm

There are three basic steps to constructing Grover's search algorithm (GSA).

(1) Generate a uniform superposition state representing all possible items in our unstructured list.
(2) Apply the Grover Oracle U_G, which tags the state(s) for which we are searching.
(3) Amplify the probability of randomly selecting the tagged state by applying the diffusion operator U_D.

The details of GSA are well motivated and described in the seminal textbook by Nielson and Chuang,[16] but it is worthwhile to provide some of the mathematical formalism now. If the unsorted list has length N, we require n qubits, where $2^n = N$. It is standard to initialize all qubits in an algorithm to be in the $|0\rangle$ state. After initialization, a Hadamard gate is applied to each qubit. Hadamard gates have the following effect on each basis state:

$$H|0\rangle = \frac{1}{\sqrt{2}}[|0\rangle + |1\rangle],$$
$$H|1\rangle = \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle]. \tag{2}$$

The action of the Hadamard on a basis state is to put that basis state into a superposition. The net result of applying Hadamards to all n qubits individually (Eq. (3)) is the system is placed into a state where all of the possible outcomes are equally probable (Eq. (4)),

$$|\psi\rangle = H^{\otimes n}|(|0\rangle \otimes |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle)\rangle, \tag{3}$$

$$|\psi\rangle = \frac{1}{\sqrt{N}}[|00\ldots0\rangle + |00\ldots1\rangle + |00\ldots10\rangle + \cdots + |11\ldots1\rangle]. \tag{4}$$

Each binary number in Eq. (4) has n digits representing the tensor product of the individual qubits.

The goal of GSA is to select one (or more) of the states in this superposition. This brings us to the second step in the algorithm, the application of the Oracle. The Oracle gate U_G labels a "winner" state $|w\rangle$ by flipping its phase, or in other words, flipping the sign of its probability amplitude. Returning to Eq. (4) to highlight the $|w\rangle$ state, the action of the Grover Oracle on the superposition is given by the following equation:

$$U_G|\psi\rangle = \frac{1}{\sqrt{N}}[|00\ldots0\rangle + |00\ldots1\rangle + \cdots - |w\rangle + \cdots + |11\ldots1\rangle]. \tag{5}$$

For beginning students, it is easy to get deterred by the mathematics of Eqs. (3)–(5). In practical use, the generation of a Grover Oracle has been automated by the most popular quantum computing libraries such as IBM's Qiskit.[17] What is more important is to understand conceptually what the Oracle accomplishes. The Grover Oracle takes in the superposition states representing all possible entries, and *confirms* the winning $|w\rangle$ by flipping its phase and leaving all other states unchanged. Unlike a classical application of an Oracle, the Grover Oracle checks all of possible states simultaneously in one application.

It is possible to write out this Oracle as an operator (matrix) using the general recipe

$$U_G = I - 2|w\rangle\langle w|. \tag{6}$$

Here, I is the identity matrix of dimension $n \times n$ and $|w\rangle\langle w|$ is the outer product of the state being searched for $|w\rangle$, which

25 February 2026 08:46:17

produces a matrix also of dimension $n \times n$. Decomposing this Oracle operator into the basis gates of the machine we are using can be difficult, especially for beginning students; however, that is exactly what is done automatically by the high-level quantum libraries. In Sec. IV, an example is provided of how to construct an Oracle for searching a small list of states, and later the construction of a *logical phase Oracle* is detailed in Sec. IV B because it serves as the central idea explored by the students in the assigned activity.

The final stage of the algorithm is the application of the diffusion operator. The diffusion operator amplifies the likelihood of randomly selecting the winning state out of the superposition of all states by reflecting the total state $|\psi\rangle$ about the mean value of the probability amplitude of each component of the superposition. Like the Grover Oracle, there is a general recipe for constructing the Diffuser, which is given by

$$U_D = 2|\psi\rangle\langle\psi| - I. \tag{7}$$

A full derivation of Eqs. (6) and (7) can be found in Ref. 16 as exercises in Chap. 6. Looking at Eq. (7), we can see that it is formed from the general superposition state $|\psi\rangle$, which implies it is completely general. We can reliably decompose this operator in a reusable gate set, which will become evident in the next section. However, how does the Diffuser amplify the likelihood of randomly selecting the tagged state? This process is best depicted graphically and is shown in Fig. 2. Part (a) of the figure shows the N components of the superposition states. Note that initially, each has a probability amplitude $1/\sqrt{N}$. Part (b) of Fig. 2 demonstrates the action of the Oracle where the state we are searching for has its probability amplitude negated. The process of negation also shifts the overall mean downward. Finally, in part (c), we see how each state is inverted about the mean which decreases the probability amplitudes of all non-tagged states and dramatically increases the amplitude of the w state. The inversion about the mean can also be described as an amplitude amplification because the Diffuser has now amplified the likelihood of randomly selecting the state for which we are searching and suppressed the possibility of the non-tagged states.

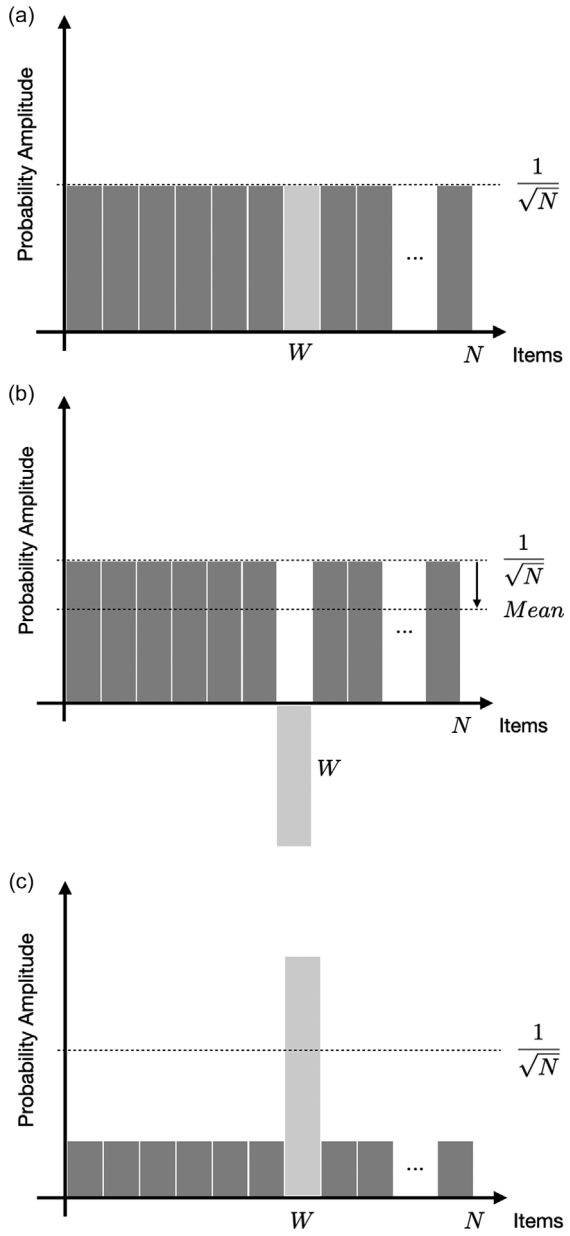


Fig. 2. Amplitude amplification of the Grover Diffuser.

C. A small example of GSA

The most instructive way the actions of the Oracle and Diffuser can be observed is to work through a small example. The smallest example consists of searching a list of four items, which we can represent using only two qubits. The four states that can be represented by two qubits are $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. For example, let us search this list for the state $|w\rangle = |11\rangle$. Figure 3 depicts the circuit we will use.

We initialize the qubits into the $|0\rangle$ states on the left. When working with Dirac notation and gates it is also useful to index the qubits from top to bottom, indexing from zero. The top wire in the figure is qubit (0), and the second wire is qubit (1). If we consider the total state of all qubits at any point in the circuit as $|\Psi\rangle$, we apply Hadamard gates to each initialized qubit and show the result in Eq. (8). The subscripts in the equation are the qubit indices,

$$\begin{aligned} |\Psi_1\rangle &= H_1 \otimes H_0 |0_1 0_0\rangle \\ &= \frac{1}{2}[|0_1 0_0\rangle + |0_1 1_0\rangle + |1_1 0_0\rangle + |1_1 1_0\rangle]. \end{aligned} \tag{8}$$

Next, we apply the Oracle, which tags the state we are searching for. This is accomplished with the controlled-Z gate (CZ). The CZ gate is a two-qubit gate, requiring a control qubit and a target qubit. Its action be summarized in Table I.

Applying the CZ gate with the 0 qubit as the control and the 1 qubit as the target results in Eq. (9). It is important to note that if we were to write the CZ gate as a matrix, we

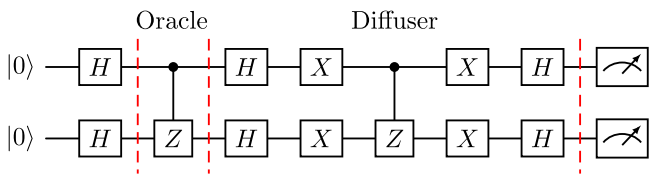


Fig. 3. GSA Circuit to find state $|11\rangle$ depicting both the Oracle and Diffuser gate decomposition.

25 February 2026 08:46:17

Table I. Action of the controlled-Z gate.

Control	Target	CZ Result
$\|0_c\rangle$	$\|0_t\rangle$	$\|0_t0_c\rangle$
$\|0_c\rangle$	$\|1_t\rangle$	$\|1_t0_c\rangle$
$\|1_c\rangle$	$\|0_t\rangle$	$\|0_t1_c\rangle$
$\|1_c\rangle$	$\|1_t\rangle$	$-\|1_t1_c\rangle$

would find it to be the predicted result of Eq. (6). Moving to the Diffuser, we again apply Hadamards to each qubit with the result shown in Eq. (10). After these Hadamards, we then apply X-gates. These gates are the quantum equivalent to the classical NOT gate. Applying X-gates to each of the qubits produces Eq. (11),

$$\begin{aligned}|\Psi_2\rangle &= CZ|\Psi_1\rangle \\ &= \frac{1}{2}[|0_1 0_0\rangle + |0_1 1_0\rangle + |1_1 0_0\rangle - |1_1 1_0\rangle], \end{aligned} \tag{9}$$

$$\begin{aligned}|\Psi_3\rangle &= H_1 \otimes H_0|\Psi_2\rangle \\ &= \frac{1}{2}[|0_1 0_0\rangle + |1_1 0_0\rangle + |0_1 1_0\rangle - |1_1 1_0\rangle], \end{aligned} \tag{10}$$

$$\begin{aligned}|\Psi_4\rangle &= X_1 \otimes X_0|\Psi_3\rangle \\ &= \frac{1}{2}[|1_1 1_0\rangle + |0_1 1_0\rangle + |1_1 0_0\rangle - |0_1 0_0\rangle]. \end{aligned} \tag{11}$$

We can quickly finish out the rest of the calculations working our way down the circuit,

$$\begin{aligned}|\Psi_5\rangle &= CZ|\Psi_4\rangle \\ &= \frac{1}{2}[-|1_1 1_0\rangle + |0_1 1_0\rangle + |1_1 0_0\rangle - |0_1 0_0\rangle], \\ |\Psi_6\rangle &= X_1 \otimes X_0|\Psi_5\rangle \\ &= \frac{1}{2}[-|0_1 0_0\rangle + |1_1 0_0\rangle + |0_1 1_0\rangle - |1_1 1_0\rangle], \\ |\Psi_7\rangle &= H_1 \otimes H_0|\Psi_6\rangle \\ &= 0|00\rangle + 0|01\rangle + 0|10\rangle - 1|11\rangle. \end{aligned} \tag{12}$$

The final line of Eq. (12) is written in a way to emphasize that three of the probability amplitudes of the states have been reduced to zero, which forces the amplitude of the tagged state $|11\rangle$ to be 1. When we conclude the algorithm by measuring both qubits there is only one possible outcome—the state that we were searching for. What is not directly apparent, but could be quickly confirmed, is that the matrix multiplication of all the tensors in each step of the Diffuser (detailed in Eqs. (10)–(12)) would yield the same result as Eq. (7).

D. Vector interpretation and complexity class

Before describing the specific assignment it is instructive to provide a visual representation of GSA beyond Fig. 2. A vector model of the algorithm can help conceptualize the actions of the Oracle and the Diffuser without the complexities of Dirac notation or quantum circuits.

Consider the uniform superposition state $|\psi\rangle$ given in Eq. (8) as a vector in an N dimensional complex vector space. We then construct two orthogonal vectors, one being the winner state $|w\rangle$ and the other the state $|\psi'\rangle$, which we generate by removing $|w\rangle$ from the superposition $|\psi\rangle$. We can depict these three vectors in part (a) of Fig. 4.

Applying the Grover Oracle U_G to the state $|\psi\rangle$ negates the component of $|\psi\rangle$ along the $|w\rangle$ axis, flipping the vector across the $|\psi'\rangle$ axis as in part (b) of Fig. 4. The Diffuser operator U_D then flips the state vector about the original vector $|\psi\rangle$ as shown in part (c) of Fig. 4. The final state vector $U_D U_G|\psi\rangle$ has a larger overlap with the $|w\rangle$ axis than the original $|\psi\rangle$ state, which indicates that all of the probability amplitudes of the components of the superposition state $|\psi\rangle$ were reduced except the $|w\rangle$ component. Because of normalization rules, the element in the $|w\rangle$ direction must, therefore, increase, making it more probable that the state $|w\rangle$ could be chosen at random from the set of all possible states.

In Sec. III C, it was shown that after just one application of an Oracle and Diffuser, the desired state was recovered with 100% certainty. For larger problems, U_G and U_D must be applied iteratively until some maximum probability of selecting the $|w\rangle$ state is achieved—but how many times must the process be iterated? In Fig. 4(a), we can decompose $|\psi\rangle$ into the $|w\rangle$ and $|\psi'\rangle$ basis,

$$|\psi\rangle = \sin\theta|w\rangle + \cos\theta|\psi'\rangle. \tag{13}$$

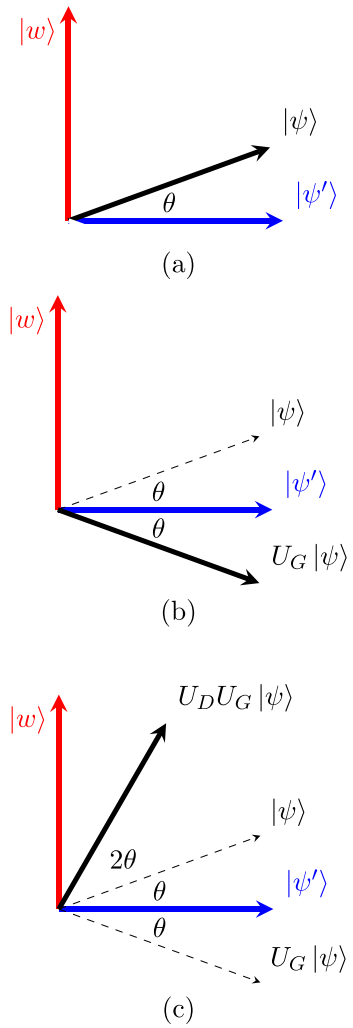


Fig. 4. Vector model of Grover's algorithm.

25 February 2026 08:46:17

We then can solve Eq. (13) for the angle θ by performing the inner product of both sides of the equation with $\langle w|$ and recalling that $|\psi\rangle$ is the superposition of all possible states. This inner product $\langle w|\psi\rangle$ is just the probability amplitude of the state $|w\rangle$ as shown in Eq. (4). Assuming N is sufficiently large, an approximation for θ is shown in the following equation:

$$\begin{aligned}
\langle w|\psi\rangle &= \sin\theta, \\
\frac{1}{\sqrt{N}} &= \sin\theta, \\
\arcsin\left(\frac{1}{\sqrt{N}}\right) &= \theta, \\
\frac{1}{\sqrt{N}} &\approx \theta.
\end{aligned} \tag{14}$$

In Fig. 4(c), we can see that after one iteration of GSA the state has been rotated by 2θ toward the $|w\rangle$ axis. To calculate the optimal number of iterations (n), we can model the process by the following equation:

$$\begin{aligned}
\frac{\pi}{2} &= \theta + n(2\theta) \\
&\approx \frac{1}{\sqrt{N}} + n\left(\frac{2}{\sqrt{N}}\right), \\
n &\approx \frac{\pi}{4}\sqrt{N} - \frac{1}{2}, \\
n &\approx \sqrt{N}.
\end{aligned} \tag{15}$$

Equation (15) provides the time complexity class of GSA and evidence of the quantum speed-up.

IV. ACTIVITY DESCRIPTION

Section III provided an overview of the components and construction of GSA. We now focus on two classroom-tested activities for high school students that explore a broader class of use cases beyond the canonical unstructured list search. Boolean satisfiability problems (or SAT problems) have a long history including a famous one supposedly posed by Einstein.[18] On the surface, SAT problems can take the form of engaging riddles, but their importance in solving optimization problems raises their status beyond simple riddles.[19,20] In this section, we will start with a step-by-step example of applying GSA to a SAT problem, then we will conclude with a description of two class activities with two levels of difficulty.

A. A physics dinner party

We start with a simple example, and then describe how students extend to discover more fundamental phenomena. Imagine that five physics students, Adam, Maita, Mark, Elissa, and William, are planning a party and are trying to determine a menu of complementary fruits and snacks. They need to figure out what type of fruit selections they can make that meet all of their particular eating preferences. Their preferences are listed in the Table II.

We assign an integer label for each type of food. Apples = 1, bananas = 2, cherries = 3, and dates = 4, which we use to convert the preferences into logical statements as shown in Table III. The table makes use of standard logical symbols ¬ (NOT) and ∨ (OR).

Table II. Physics student fruit preferences.

Teacher	Preferences
Adam	Likes apples or bananas or cherries or dates.
Maita	Will not eat apples or will not eat dates.
Mark	Likes cherries or will eat anything except dates.
Elissa	Likes dates or will eat anything except cherries.
William	Likes apples or cherries, but doesn't like bananas or dates.

B. Constructing a logical oracle

In Sec. III C, we stated that constructing the Grover Oracle in the machine basis gates was difficult for beginning students. Generating logical Oracles, like the ones used for SAT problems, is worth discussing. Construction of these oracles can be automated by storing the logical clauses in a text file in the Conjunctive Normal Form,[21] which is a simple restatement of the logical clauses from Table III but formatted without the logical symbols. Importing CNF files into Qiskit or another package is then possible.[22] It is instructive for students to explore this process by hand before moving to the automated process.

When solving SAT-type problems with GSA, we need to expand the size of the Grover Oracle as shown in Fig. 5. The oracle must use C computational qubits, one for each of the logical variables (4 in the Party Problem). It must also accommodate A ancilla qubits, with A equal to one more than the number of logical clauses. It is very important to note that the Diffuser U_D still only operates on the computational qubits.

We can directly translate the logical clauses given in Table III into gates as shown in Fig. 6. Note the four computational qubits C corresponding to the four types of food and the six ancilla qubits A that store information if the conditions of each logical clause are satisfied. It is these ancilla qubits that make the logical oracle work.

Figure 6 makes use of controlled (dark circles) and anti-controlled (open circles) X-gates. These gates have a similar structure to the controlled-Z gate described in Sec. III C with a few additions. These are multi-controlled—meaning that all the control qubits must either be 1 (controlled) or 0 (anti-controlled) as a condition to apply the X-gate to the target qubit. Reading Fig. 6 from left to right, the first gate is a multi-anti-controlled X representing the first logical clause $(1\vee2\vee3\vee4)$ This gate only triggers when all four of the computational qubits are $|0\rangle$, or written as a product state $|0000\rangle$, which we read right to left. Because a superposition of all possible combinations of the computational bits is being passed into this oracle, We can think of this first gate as a

Table III. Physics student preferences as logicals.

Teacher	Preferences
Adam	$1\vee2\vee3\vee4$
Maita	$\neg1\vee\neg2$
Mark	$3\vee\neg4$
Elissa	$\neg3\vee4$
William	$1\vee\neg2\vee3\vee\neg4$

25 February 2026 08:46:17

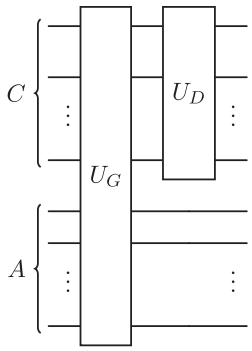


Fig. 5. Structure of a logical Grover Oracle.

filter. It lets all states through except $|0000\rangle$. In this context, "filtering" means that all states other than $|0000\rangle$ still have a chance of having their probability amplitude negated by the oracle. The second gate is a multi-controlled X-gate and represents the second logical clause $(\neg 1 \vee \neg 2)$ It filters out the state $|0011\rangle$, once again reading the qubits right to left. The remaining gates up to the central Z-gate represent the remainder of the logical clauses from the physics party problem. It is this central Z-gate that tags all the states that pass through the filter gates with negation and sets the stage for the diffusion operator.

All of the gates to the right of the Z-gate are necessary as an *Uncompute* step. Because GSA can be iterative, we must erase any temporary calculations stored in the ancilla—which is accomplished by applying the logical clause steps in reverse. This returns all of the values of the ancilla bits to $|0\rangle$ because X-gates are both hermitian and unitary, meaning $X = X^{\dagger}$ and $X \cdot X = I$. To complete the calculations, we apply an extended Diffuser operator shown in Fig. 7. This Diffuser is a generalization of the one depicted in Fig. 3 and defined by Eq. (7).

When we run this circuit on a quantum simulator (with simulated noise) it produces results shown in Fig. 8. We clearly see five states with much higher probabilities of occurrence corresponding to the five Boolean states that fit all the individual student requirements. These results are interpreted in Table IV.

C. A note on coding environments for beginners

After reading the preceding introduction to the student activity, there might be some concern that students do not have the prerequisite coding skills to implement the

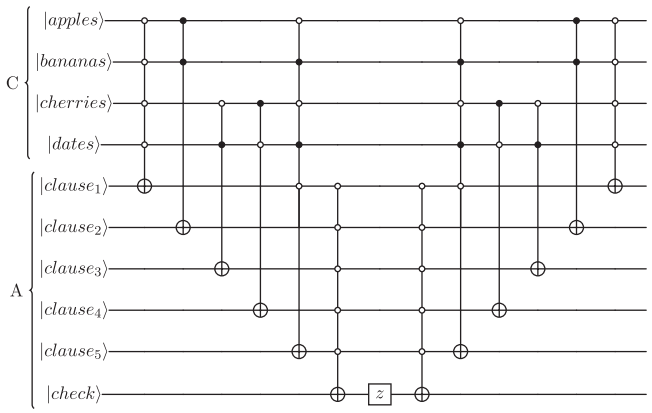


Fig. 6. Logical Grover Oracle for dinner party problem.

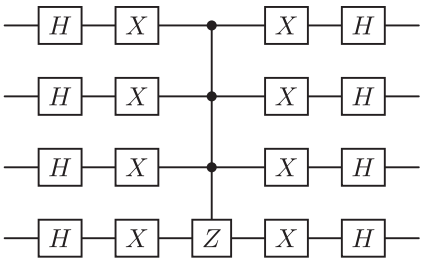


Fig. 7. Extended Diffuser operator.

algorithm. Quantum coding libraries have progressed to the point that most of the process can be automated, but even highly automated libraries require a baseline level of comfort with Python. To address this gap, two different activities are presented that enable students to explore Grover's algorithm and demonstrate proficiency using two different levels of coding skill. The first is a low-code activity, and the second is suitable for students with more Python experience.

D. Low/no code assignment

Students with low, or no Python experience can engage with the GSA by using one of two drag-and-drop coding platforms. The two most user-friendly graphical coding platforms are IBM Circuit Composer[†] and Quirk.[‡] The IBM Circuit Composer can be exported and run on one of IBM's real quantum computers but requires an IBM account. Additionally, jobs submitted to one of the real quantum computers are subject to long queue times and students may have to wait hours for results. Quirk not only does not require an account but is also only a quantum simulator and it allows for multi-controlled gates, which are necessary to construct both logical Oracles and the Grover Diffuser. It is because of this that we recommend Quirk for use with the Low/No code assignment.

The base-level activity is straightforward, students were asked to find two more SAT problems that could be solved using Grover's algorithm with a logical oracle using the physics student party example as a model. Students were encouraged to write their own, but there are also many good online sources for easy SAT problems[23–25] that can serve as guidance. We also allowed students to use generative AI to make unique ones. Students must find problems online with some care—it is very easy to fall into a SAT problem that is solvable but contains too many variables for students to construct by hand making the assignment too tedious. Using one of the many generative AI sources lets you precisely control the number of variables and logical clauses and keeps the overall problem tractable. From experience, SAT problems with 5–6 variables, and no more than 10 logical clauses are best. Quirk has a maximum of 16 qubits, so the balance of variables and logical clauses must follow the equation

$$\text{Variables} + \text{Clauses} + 1 \leq 16. \tag{16}$$

The students were given one week to find problems, then we organized a SAT problem swap during class where students exchanged their problems. After exchanging problems, students spent the rest of class time solving them using Grover's search algorithm. Sample problems as well as all

[†]At time of writing https://quantum.ibm.com
[‡]At time of writing https://algassert.com/quirk

25 February 2026 08:46:17

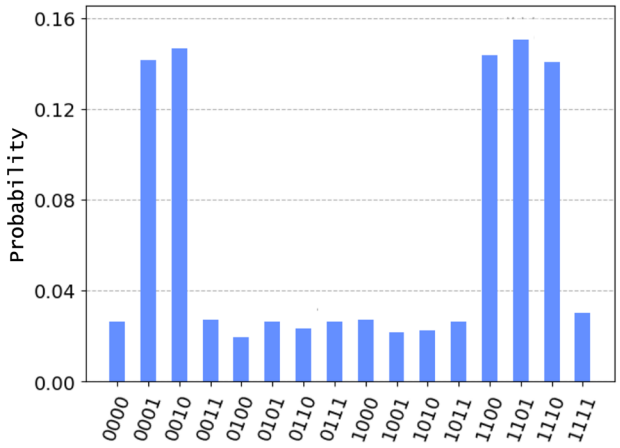


Fig. 8. Output of GSA for dinner party problem.

other supplementary material can be found in the GitHub repository cited in Ref. 26.

E. High code assignment

To complete the *High Code Assignment*, students will not only need some baseline skill using Python but they will also need an IBM account to access the full Qiskit library. We also strongly encourage students to use Jupyter Notebooks for their code such as Google CoLab. Shell code was provided to students for much of the problem, which can be viewed in the Github repository.[26]

Returning to the dinner party example. How much more complicated would it be if the physics students started to invite members of the math department to their party? Each additional invite would come with an additional logical clause. *How many more people could the Physics students invite before there would be too many constraints and no single solution exists?* This becomes the central question to explore.

We should further cast the problem statement with some limiting cases:

- The total number of variables is fixed at nine—i.e., there are nine types of fruit/food/etc. to choose from.[§]
- Each person we invite must have exactly three constraints that detail their preferences and requirements.
- Logical clauses (for each additional person) should be chosen at random, but
  - all variables must have equal probability of negation;
  - each new person added must provide a unique logical clause;
  - individual clauses cannot contain tautologies where a variable is both included and excluded. For example, $(1 \vee 5 \vee \neg 5)$;
  - logical clauses must be formatted in Conjunctive Normal Form for comparability with IBM Qiskit.

The assignment is to write a Python program that implements GSA using n logical clauses and vary n, $1 \leq n \leq 80$. After each execution of GSA, check if at least one solution exists that meets all the conditions of the n logical clauses. This task is referred to as a 3SAT problem.[27] To generate reasonable statistics, each n was repeated 20 times and a probability of success was calculated for each n. As evidence of successful implementation of GSA, the students produced a plot similar to Fig. 9, which depicts the likelihood of a solution as a function of the number of logical clauses.

[§]Nine variables were chosen because at the time IBM Qiskit subroutines limited total variables to this value. Since this assignment was initially given to students this limit has been removed.

Table IV. Interpreting GSA outcomes.

Result	Snack combinations
0001	Only serve apples
0010	Only serve bananas
1100	Serve dates and cherries
1101	Serve dates, cherries and apples
1110	Serve dates, cherries, and bananas

What is very interesting about this graph is that it displays an elongated phase transition between the existence of a solution being very probable $(n \leq 35)$ and the existence of a solution being very improbable $(55 \leq n)$. This phase transition is one of the hallmarks of SAT problems.[28] The sharpness of the transition scales with the number of variables and has a theoretical phase transition point given by the following equation:[29]

$$\frac{L}{N} \approx 4.24, \tag{17}$$

where N= number of variables and L = number of logical clauses.

It also gives us a gateway to move the conversation back to a direct physics application—Thermodynamics! Thermodynamic models such as the Ising model can be recast into 3SAT structures.[30] This connection can be leveraged for further investigation by students into the applications of GSA anywhere the Ising model is used.

There are very good classical algorithms for finding solutions to 3SAT problems,[31] *so why use a quantum algorithm*? This forms the second main question of investigation for the High Code Assignment. The students were asked to alter their original GSA code to measure the run time of each Grover Oracle for each value n. This is a rather simple benchmarking process and when completed, students produced plots similar to Fig. 10. The error in run times is shown by calculating the standard error across the 20 iterations for each n.

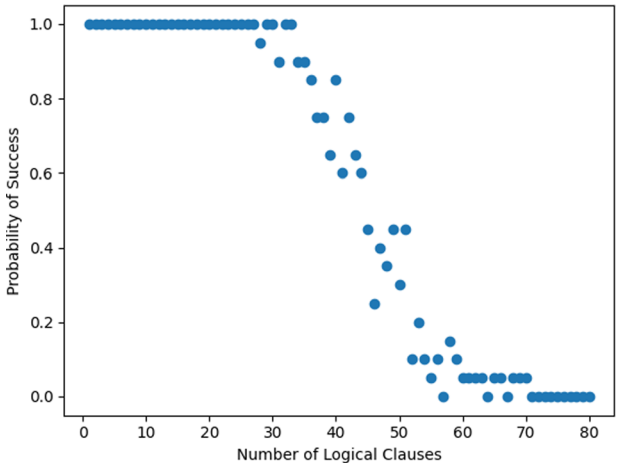


Fig. 9. 3SAT Grover search showing phase transition.

25 February 2026 08:46:17

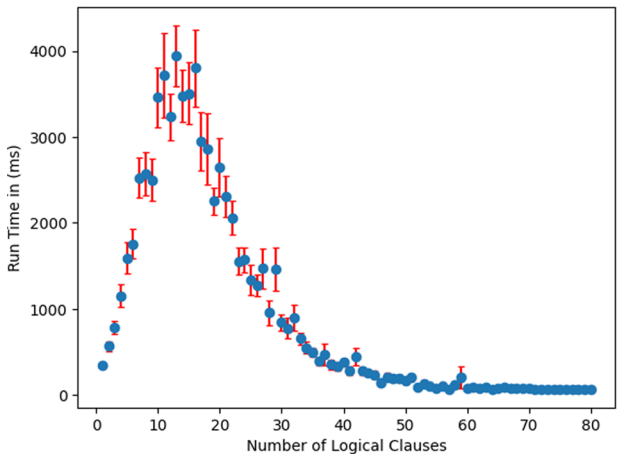


Fig. 10. Time complexity of GSA Oracle applied to 3SAT.

What is of particular interest is that the longest run times correspond to relatively easy $n \approx 15$ problems, then drop quickly as the number of logical clauses increases. This is quite different from the classical algorithms where run times increase to a maximum just after the phase transition.[32] Students were asked to produce one more plot of Grover Oracle gate depth to uncover the underlying cause of the time complexity. Gate dept of the Grover Oracle can be extracted using a simple, built in method in Qiskit. Figure 11 displays the gate depth of the Oracle data.

Students were able to infer from Fig. 11 that as the number of logical clauses increases, rather than adding additional complexity to the Oracle, it actually provides more opportunities for internal gate simplifications. This is a fundamental property of GSA applied to 3SAT problems.

V. STUDENT FEEDBACK

Although a formal assessment of learning outcomes was not conducted, a post-activity survey was provided to the students to collect anecdotal information about the engagement and effectiveness of the activities. All anonymized survey data can be found in the Github repository.[26] Students were asked to reflect on their experience by answering some focusing questions. What follows are representative responses from students.

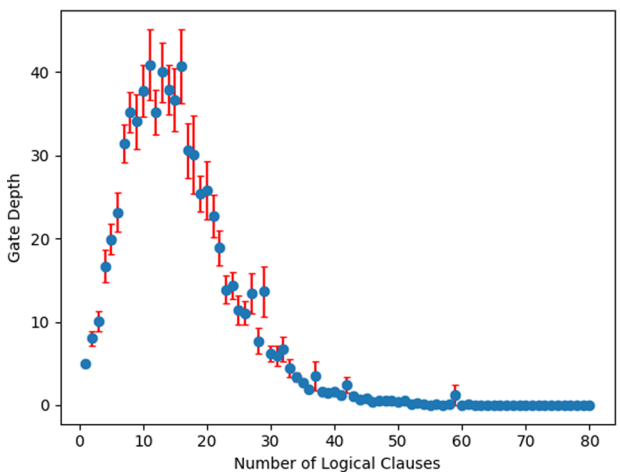


Fig. 11. Gate depth of GSA Oracle applied to 3SAT.

(1) **Before starting the GroverLab, what was your perception of the usefulness of quantum computing?**
I thought that quantum computing wasn't useful because it seemed too theoretical to be applicable.
I did not understand how quantum computing could use quantum mechanical properties to advance computer science algorithms and other types of problems as they are already pretty efficient

(2) **After completing the GroverLab, what is your current perception of the usefulness of quantum computing?**
I definitely have a better perception of the types of questions quantum computing could be competitive against classical computing after this lab especially after directly seeing the time differences to solve the same problems.
I still don't think quantum computing is particularly useful at this time or for most everyday tasks, but after seeing an application that solved a (not stupid) problem in a completely different way and time-frame than the classical approach, I definitely think that it has a lot more potential.

(3) **The GroverLab involved solving 3SAT problems by converting logical clauses into a quantum operator. How were these logical statements used to construct the Oracle?**
The logical statements constructed the Oracle by putting constraints on what the solution had to achieve in order for it to be a valid solution. It essentially specified which values the Oracle was supposed to tag–the specific Oracle matrix thus depended on these logical statements. Instead of directly identifying the solution states and flipping them, we can instead model each 3SAT clause as a gate and layer these gates to create the oracle.

(4) **Do you have any feedback on how Grover's algorithm was presented and taught throughout the course and through the GroverLab project?**
I liked the way that Grover's Algorithm was taught in the course, and I appreciated not having to take a test and instead getting to work on a lab that boosted my understanding much better.
I think letting us code the lab by ourselves was nice because I felt accomplished after we figured it out.
I actually really enjoyed doing the lab, which was surprising since I am very much not into computer science. The main thing for me was seeing quantum computing as a more tangible concept and not something people just talk about as a buzzword, which definitely increases my interest in the field and makes me want to read more about it.

The students in the summer course where also asked to rate the mathematical complexity of the summer camp, including a specific reference to the GSA activity. Their responses are shown in Fig. 12.

VI. CONCLUDING REMARKS

The pending quantum computing revolution will require K-12 teachers to consider ways to expose all students to more quantum topics than what is currently in state and local education standards. If we do not do this we will not only leave students behind, we will endanger the progress of the field. We only have to look at the current AI landscape as a cautionary tale and see who is participating and who is being

25 February 2026 08:46:17

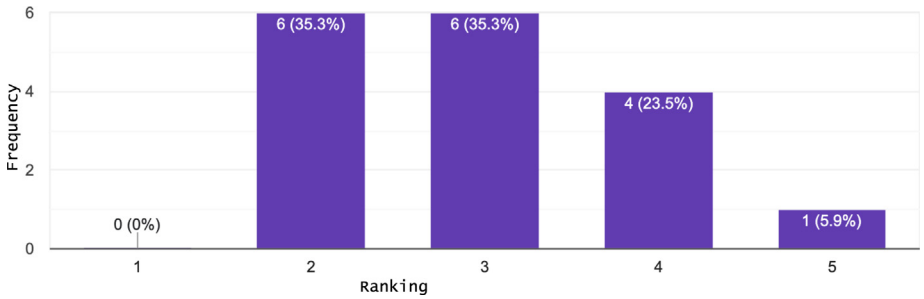


Fig. 12. Self-reported math difficulty 1 = not challenging and 5 = very challenging.

left out, and the negative impacts.[33,34] My institution is lucky to have the flexibility to run a stand-alone course on Quantum Mechanics. Certainly, replicating this is not a reasonable goal, nor a sustainable solution to prime the quantum workforce at every high school. Looking at the self-reported outcomes of the students who participated in the summer camp version of the activity, we can see that the fundamentals of quantum computing, and its applications such as Grover's search algorithm are accessible to motivated high school students. Using activities like the ones described in this paper in the context of traditional physics, computer science, chemistry, or mathematics courses is the only way forward. More rigorous research should be conducted measuring the gains of quantum-focused learning with high school students. However, based on the informal feedback from students, we have evidence that the integration of Quantum Computing topics into high school courses is possible, accessible, and produces motivational gains for students.

ACKNOWLEDGMENTS

The author would like to express his gratitude to Dr. William Linch (TJHSST) for his invaluable technical and editorial support in writing this manuscript and for his assistance during the design phase of the student-facing assignment. The author also acknowledges the contributions of Paul Kosek (TJHSST) in developing the Python code and Dr. Adam Smith (TJHSST) as co-instructor of the high school course. Finally, the feedback from all students who completed the assignment has been indispensable and will be used to continuously enhance the experience for future students.

AUTHOR DECLARATIONS

Conflict of Interest

The author declares no conflicts of interest related to the information and materials presented in this manuscript.

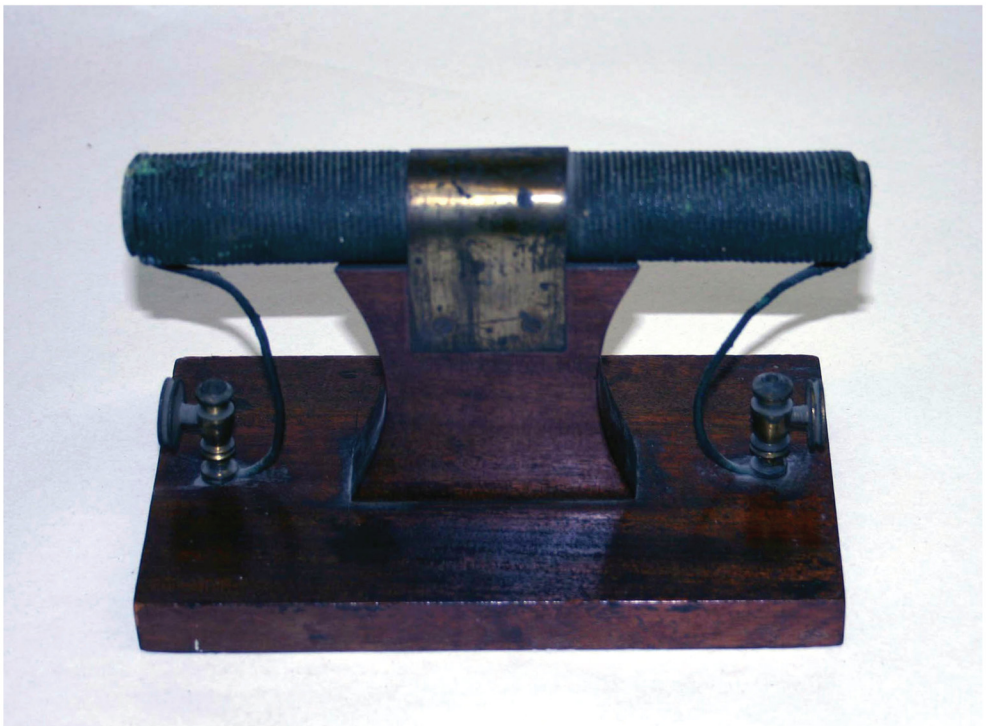
Note: This paper is part of the special issue celebrating the International Year of Quantum Science and Technology.

[a)]ORCID: 0000-0002-9054-5981.

[1]C. Hughes, D. Finke, D.-A. German, C. Merzbacher, P. M. Vora, and H. J. Lewandowski, "Assessing the needs of the quantum industry," IEEE Trans. Educ. **65**(4), 592–601 (2022).
[2]M. Brooks, "Quantum computers: What are they good for?," Nature **617**(7962), S1–S3 (2023).
[3]Quantum Information Science, *QIST Workforce Development* (National Coodinating Office, 2022).
[4]NQC Office, *Summary of the Quantum Workforce: Q-12 Actions for Community Growth Meeting* (White House Office of Science and Technology Policy, National Quantum Coordinating Office, 2022).
[5]J. K. Perron, C. DeLeone, S. Sharif, T. Carter, J. M. Grossman, G. Passante, and J. Sack, "Quantum undergraduate education and scientific training," arXiv:2109.13850 (2021).
[6]J. C. Meyer, G. Passante, S. J. Pollock, and B. R. Wilcox, "Today's interdisciplinary quantum information classroom: Themes from a survey of quantum information science instructors," Phys. Rev. Phys. Educ. Res. **18**(1), 010150 (2022).
[7]A. Asfaw *et al.*, "Building a quantum engineering undergraduate program," IEEE Trans. Educ. **65**(2), 220–242 (2022).
[8]M. F. J. Fox, B. M. Zwickl, and H. J. Lewandowski, "Preparing for the quantum revolution: what is the role of higher education?," Phys. Rev. Phys. Educ. Res. **16**(2), 020131 (2020).
[9]*QIS Key Concepts for High School Physics* (National Q-12 Education Partnership, 2023).
[10]D. H. McIntyre, *Quantum Mechanics: A Paradigms Approach* (Cambridge U.P., Cambridge, 2023), p. 590.
[11]L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing—(STOC '96)* (Association for Computing Machinery, 1996), pp. 212–219.
[12]D. E. Knuth, *The Art of Computer Programming, Volume 3: Sorting and Searching*, 2nd ed. (Addison-Wesley, Reading, MA, 1998).
[13]A. Mandviwalla, K. Ohshiro, and B. Ji, "Implementing Grover's algorithm on the IBM quantum computers," in *IEEE International Conference on Big Data (Big Data)* (IEEE, New York, 2018), pp. 2531–2537.
[14]M. AbuGhanem, "Comprehensive characterization of three-qubit Grover search algorithm on IBM's 127-qubit superconducting quantum computers," arXiv:2406.16018 (2024).
[15]W. H. Press, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. (Cambridge U.P., New York/Cambridge, UK, 2007), p. 1235.
[16]M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge U.P., Cambridge, 2010), Chap. VI.
[17]A. Javadi-Abhari *et al.*, "Quantum computing with Qiskit," arXiv:2405.08810 (2024).
[18]J. Stangroom, "Einstein's riddle: Riddles, paradoxes, and conundrums to stretch your mind," in *Collab With Internet Archive* (Bloomsbury, New York, 2009), p. 154.
[19]E. Ábraám and G. Kremer, "Satisfiability checking: Theory and applications," in *Software Engineering and Formal Methods*, edited by R. De Nicola and E. Kühn (Springer, Berlin Heidelberg, 2016), pp. 9–23.
[20]Z. Zhu, C. Fang, and H. G. Katzgraber, "Borealis—A generalized global update algorithm for Boolean optimization problems," Optim. Lett. **14**(8), 2495–2514 (2020).
[21]M. Ben-Ari, "First-order logic: Terms and normal forms," in *Mathematical Logic for Computer Science* (Springer, London, 2012), pp. 167–183.
[22]LogicalExpressionOracle, see https://docs.quantum.ibm.com/api/qiskit/0.19/qiskit.aqua.components.oracles.LogicalExpressionOracle for "IBM Quantum Documentation" (accessed September 7, 2024).
[23]Zebra puzzles—Brainzilla, see https://www.brainzilla.com/logic/zebra/ (accessed November 7, 2024).
[24]Zebra puzzles—Aha! puzzles, see https://www.ahapuzzles.com/logic/zebra/ (accessed November 7, 2024).
[25]ZebraPuzzles.com: New zebra puzzles every day, see <https://www.zebrapuzzles.com/> (accessed November 7, 2024).
[26]M. Hannum, see https://github.com/MarkHannumQuantum/Grover-Algorithm-SATProblems for "Grover-Algorithm-SAT-Problems" (2024) (accessed November 1, 2024).

25 February 2026 08:46:17

[27]M. Motoki and R. Uehara, *Unique Solution Instance Generation for the 3-Satisfiability (3SAT) Problem* (Tokyo Institute of Technology, 1999).
[28]W. Zhang, "Phase transitions and backbones of 3-SAT and maximum 3-SAT," in *Principles and Practice of Constraint Programming—CP 2001*, edited by T. Walsh (Springer, Berlin, Heidelberg, 2001), Vol. 2239, pp. 153–167.
[29]Z. Mu and H. H. Hoos, "On the empirical time complexity of random 3-SAT at the phase transition," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)* (AAAI Press, Philadelphia, PA, 2015), pp. 367–373.
[30]A. Lucas, "Ising formulations of many NP problems," Front. Phys. **2**, 1–5 (2014).
[31]T. Sato and R. Kojima, "MatSat: A matrix-based differentiable SAT solver," arXiv:2108.06481 (2021).
[32]K. Xu and W. Li, "The SAT phase transition," arXiv:cs/0005024 (2000).
[33]R. A. Shams, D. Zowghi, and M. Bano, "AI and the quest for diversity and inclusion: A systematic literature review," AI Ethics (published online 2023).
[34]R. Crowell, "Why AI's diversity crisis matters, and how to tackle it," Nature (published online 2023).



Electro-Magnet, with Three Poles

This oddity intrigued me, so I built my own: Thomas B. Greenslade, Jr., "The Three-Pole Electromagnet", *Phys. Teach.*, 49, 496 (2011). An iron rod is wound with insulated wire that travels in one direction along one half of the rod, and then is wound in the other direction until the far end is reached. As a consequence, when the ends of the wire are connected to a source of EMF, the two ends of the bar have similar magnetic poles, while the middle has the reverse polarity. You can check this out by passing a small compass down the length of the coil. This example is at Washington and Lee University. (Picture and text by Thomas B. Greenslade, Jr., Kenyon College)

25 February 2026 08:46:17