Implementation of the Grover Search Algorithm

CHEN Xiaolong

The Hong Kong University of Science and Technology
`xchenfg@connect.ust.hk`

Abstract. Quantum computing has attracted people's attention. With the property of superposition, quantum computers have great advantages over classical computers and many quantum algorithms that outperform classical algorithms have been proposed. This report will dive into the Grover search algorithm and give the implementation of it on the IBM Quantum Experience platform. In addition, the matrix representations of the circuit components will also be given.

Keywords: Quantum Computing · Grover Search Algorithm · IBM.

1 Introduction

Quantum computing has been a hot topic in recent days. Compared with traditional computers, quantum computing has great advantages. The calculations and information processing of our everyday computers are based on the work of Turing and John von Neumann [4,5]. In the classical model, all of the information is stored or represented by bits, which can take the value of either 0 or 1. Thus, one can say that the state of a classical computer is determined by the state of all its bits. A quantum computer also has bits, which are called quantum bits or qubits. A qubit can represent 0, 1 or any linear combination of 0 and 1, which is the property called superposition. This property allows qubits to represent exponentially many logical states at once. For example, n qubits can represent 2^n states at the same time, from $|00\cdots0\rangle$ to $|11\cdots1\rangle$. Based on this powerful feature, a lot of quantum algorithms that can outperform the classical algorithms are proposed, one of which is Grover search algorithm. Classically, the time complexity to search an element in an unsorted database is $O(N)$. However, in quantum computing, Grover proposed an algorithm which can finish this task with $O(N)$ operations [2]. In this report, the basic principle and process of Grover search algorithm will be introduced and an implementation of this algorithm on IBM Quantum Experience platform will also be given. To make things more clearly, the matrix representations of every part of the quantum circuit will also be given.

2 Grover Search Algorithm

Grover search algorithm provides a quadratic speedup over its classical counterparts[3]. In this part, the principle of this algorithm will be discussed. First we need to know the meaning of the coefficients in the superposition. Suppose we have the following superposition, which is the linear combination of $|0\rangle$ and $|1\rangle$.

$$|s\rangle = a|0\rangle + b|1\rangle \tag{1}$$

The coefficients a and b satisfy the equation $a^2 + b^2 = 1$. a^2 represents the probability that we conduct a measurement and the result is $|0\rangle$. Similarly, b^2 represents the probability that a measurement gives the result $|1\rangle$. The key point of Grover search algorithm is that we want to enlarge the probability of obtaining the target result.

2.1 Amplitude Amplification

Take a database consisting of $N = 2^n$ items as an example. We can use n qubits to represent all of them. Since we do not know where the target item is located, we can consider every position has the same probability of being measured out, which gives us the uniform superposition of these qubits.

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \tag{2}$$

The probability for every state of being measured is $1/N$. Using $|\omega\rangle$ to denote the target item, we can generate another vector which is orthogonal to $|\omega\rangle$ by removing $|\omega\rangle$ from $|s\rangle$ and re-scaling. Using $|s'\rangle$ to denote this vector, we can represent it as Fig. 1 [1]. After simple calculations, one can obtain that $\theta = arcsin(\frac{1}{\sqrt{N}})$.

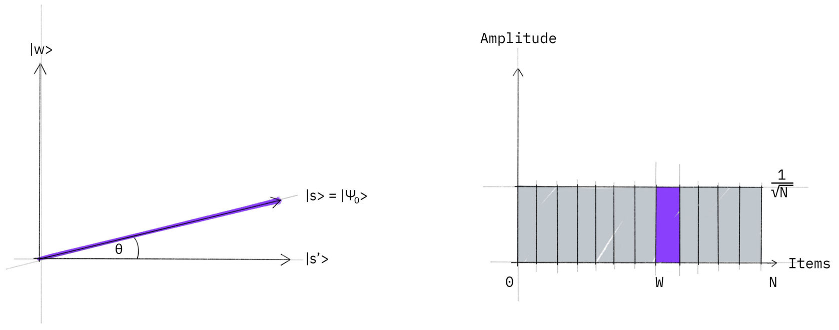


Fig. 1. This figure shows the initial situation of the process.

Then we can start thinking about how to amplify the amplitude of the state $|\omega\rangle$ in $|s\rangle$. By flipping $|s\rangle$ over the axis of $|s'\rangle$, we can convert the coefficient of $|\omega\rangle$ to $-1/\sqrt{N}$ (notice that this operation does not change the probability of obtaining $|\omega\rangle$ after one measurement). Then the geometric graph of the situation can be given by Fig. 2. After that, we do a reflection of this new vector about the original $|s\rangle$, as shown in Fig. 3. From the mathematical perspective, this operation can be viewed as projecting, extending and subtracting, the detail of which will be described later. The operations of the reflection of these two times can be

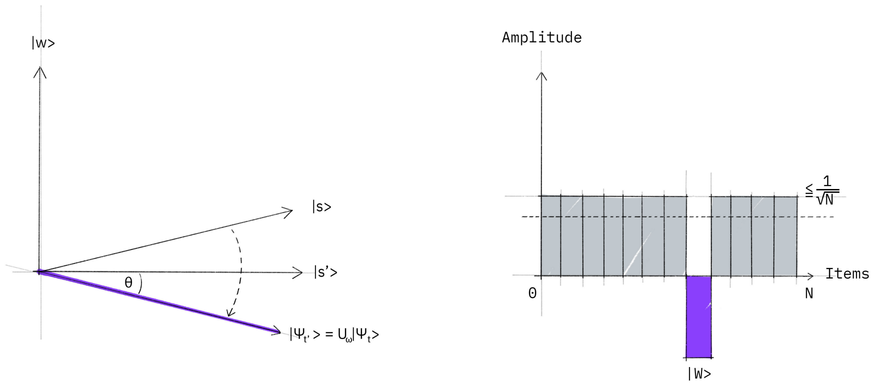


Fig. 2. This figure shows the flip of $|s\rangle$.

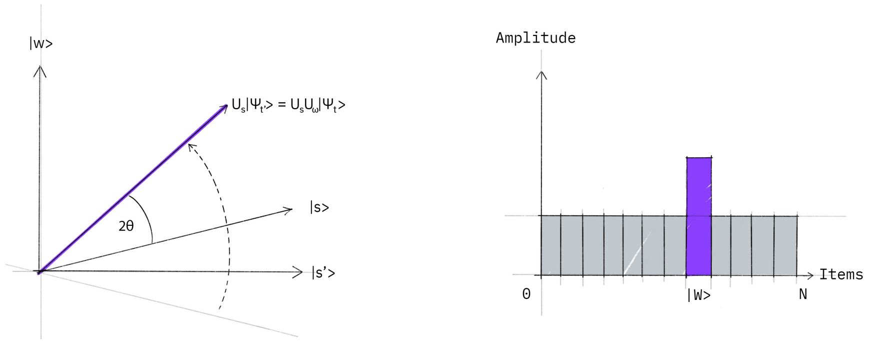


Fig. 3. This figure shows the second reflection.

written as the matrix operators. The first operator is known as the oracle and the second is called diffuser.

2.2 The Oracle

To do the first reflection, we can just change the coefficient of $|\omega\rangle$ to be -1. Here an example of 2 qubits is introduced, where we are trying to find $|01\rangle$. The oracle (U_ω) can be written as

$$U_\omega = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3}$$

In the IBM Quantum Composer, the circuit of this part in this particular example is given by Fig. 4.

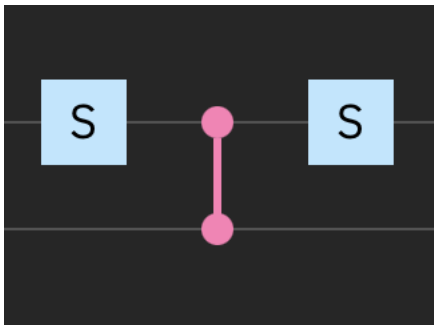


Fig. 4. The oracle used to search $|01\rangle$.

2.3 The Diffuser

The diffuser part is more complicated. As is mentioned before, it can be seen as the projection to $|s\rangle$, extend it twice and use it to subtract a certain vector. Thus, this operator can be written as,

$$U_s = 2|s\rangle\langle s| - I, \tag{4}$$

where I means the identity matrix. Taking the same example as last part, the matrix can be represented as

$$U_s = \frac{1}{2}\begin{bmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{bmatrix}. \tag{5}$$

The circuit of this part is given by Fig. 5

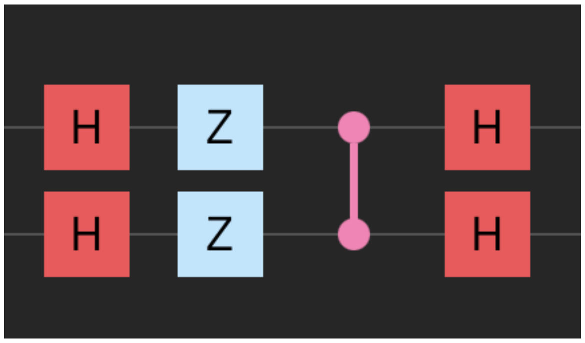


Fig. 5. The circuit of the diffuser used to search $|01\rangle$.

3 Implementation

3.1 Quantum Circuit

Consider the situation where we have 2 qubits and $|01\rangle$ is the target item we are searching for. First we will generate the superposition of two qubits, as shown in Fig. 6. Then the two parts aforementioned in Fig. 4 and Fig. 5 are added to the

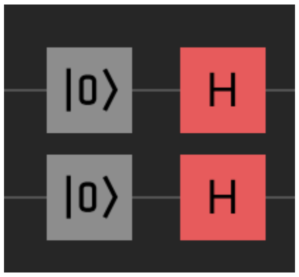


Fig. 6. The circuit of generating the superposition.

circuit. Finally, the measurement will be conducted. The whole circuit of this implementation of iterating the process for one time is shown in Fig. 7.

By repeating the oracle and diffuser components for different times, we have different results. The result of iterating once is shown in Fig. 8. We can see that the circuit finish the task to rather good extent. However, if we iterate the process for 2 times (Fig. 9) and 3 times, the probability of obtaining the correct result decreases. This is reasonable because of the nature of reflections, which will be given in the last section.

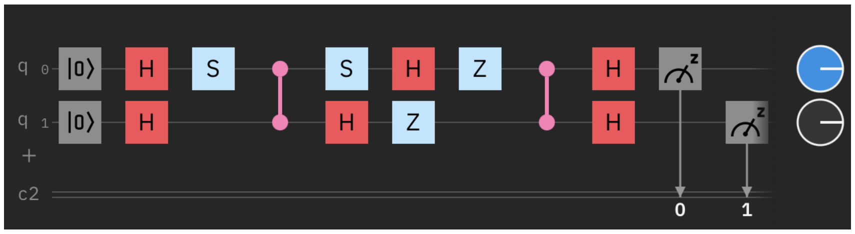
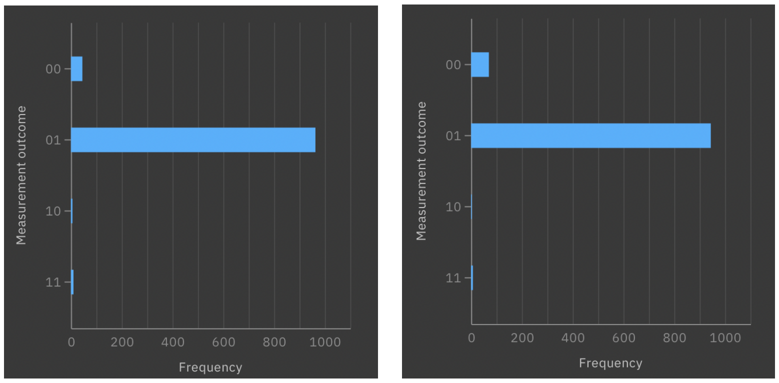
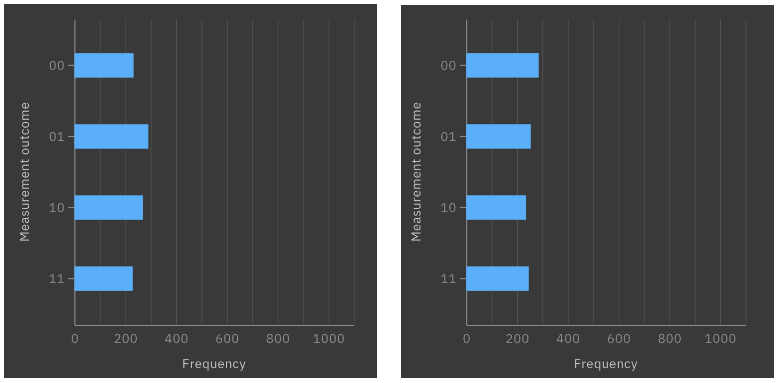


Fig. 7. The whole circuit of the process, iterating one time.



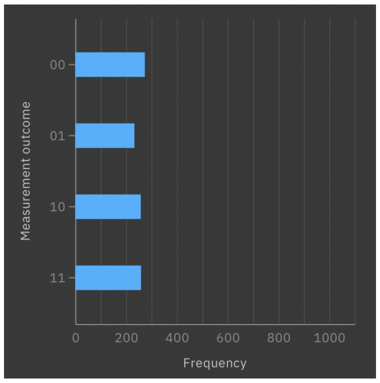
(a) Backend: ibmq_lima (b) Backend: ibmq_quito

Fig. 8. The result of iterating once using different backends.

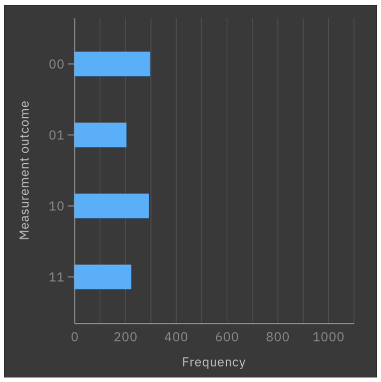


(a) Backend: ibmq_lima (b) Backend: ibmq_quito

Fig. 9. The result of iterating two times using different backends.



(a) Backend: ibmq_lima



(b) Backend: ibmq_quito

Fig. 10. The result of iterating three times using different backends.

3.2 Matrix Computation

Matrix Representations of the circuit In this part, the matrix representations of the components in the circuit will be given. Most basically, the matrices of some relevant gates are given by the following code.

```
import numpy as np
# H Gate
H = 1/np.sqrt(2) * np.array([
    [1, 1],
    [1, -1]
])
# Z Gate
Z = np.array([
    [1, 0],
    [0, -1]
])
# cz gate
cz = np.identity(4)
cz[3, 3] = -1
# S gate
S = np.array([
    [1, 0],
    [0, 1j]
])
```

Then the superposition of these two qubits is generated using the following code.

```
q0 = np.array([1, 0])
q1 = np.array([1, 0])
s = np.kron(H, H)@np.kron(q1, q0)
```

After that, the main components of this algorithm, which are the oracle and the diffuser can be calculated by the code below, and for the convenience to repeat, the product of them is stored as *core*.

```
U_w = np.kron(np.identity(2), S)@cz@np.kron(np.identity(2), S)
diffuser = np.kron(H, H)@cz@np.kron(Z, Z)@np.kron(H, H)
core = diffuser@U_w
```

Finally, we multiply them together.

```
iter_1 = core@s
iter_2 = core@core@s
iter_3 = core@core@core@s
print("Iterate once:", iter_1)
print("Iterate twice:", iter_2)
print("Iterate three times", iter_3)
```

The results are

```
Iterate once: [0.+0.j 1.+0.j 0.+0.j 0.+0.j]
Iterate twice: [-0.5+0.j 0.5+0.j -0.5+0.j -0.5+0.j]
Iterate three times [-0.5+0.j -0.5+0.j -0.5+0.j -0.5+0.j],
```

which conforms what we observe in the quantum circuit experiment.

Matrix Calculation of the theory From the perspective of the theory, the matrix calculation is as follows.

```
# Generating superposition
q0 = np.array([1, 0])
q1 = np.array([1, 0])
s = np.kron(H, H)@np.kron(q1, q0)
# Oracle
Uf = np.identity(4)
Uf[1, 1] = -1
# Diffuser
tmp = s[np.newaxis]
Us = 2*tmp.T@tmp - np.identity(4)
core = Us@Uf
iter_1 = core@s
iter_2 = core@core@s
iter_3 = core@core@core@s
print("Iterate once:", iter_1)
print("Iterate twice:", iter_2)
print("Iterate three times", iter_3)
```

The results are shown below.

```
Iterate once: [-2.22044605e-16 1.00000000e+00 -2.22044605e-16
    -2.22044605e-16]
Iterate twice: [-0.5 0.5 -0.5 -0.5]
Iterate three times [-0.5 -0.5 -0.5 -0.5]
```

4 Discussion

The matrix calculations and circuit simulation lead to the same result. However, it is weird that the algorithm does not perform well when repeated 2 and 3 times. The reason will be given in this part.

When iterating once, the process is shown in Fig. 11. From this figure, we can know that the algorithm has found the answer. The probability of obtaining the result is 1, theoretically. However, if we repeat this process again, based on

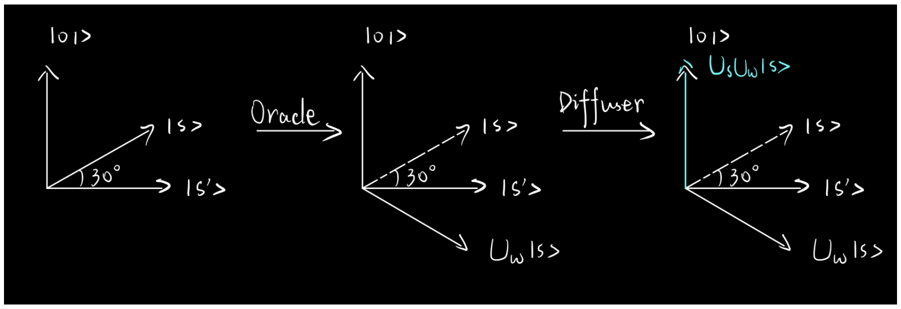


Fig. 11. This figure illustrates the process of implementing the iteration once.

the current situation, we will have Fig. 12. This process makes the probability of obtaining the target item decrease to $\frac{1}{4}$.

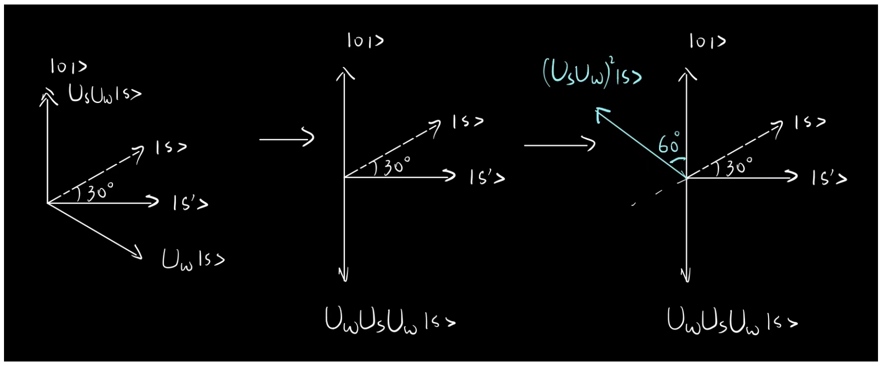


Fig. 12. This figure illustrates the process of implementing the iteration twice.

And the process of iterating 3 times is similar. So far, everything has been consistent and harmonic.

References

1. ANIS, M.S., Abraham, H., AduOffei, Agarwal, R., Agliardi, G., Aharoni, M., Akhalwaya, I.Y., Aleksandrowicz, G., Alexander, T., Amy, M., Anagolum, S., Arbel, E., Asfaw, A., Athalye, A., Avkhadiev, A., Azaustre, C., BHOLE, P., Banerjee, A., Banerjee, S., Bang, W., Bansal, A., Barkoutsos, P., Barnawal, A., Barron, G., Barron, G.S., Bello, L., Ben-Haim, Y., Bennett, M.C., Bevenius, D., Bhatnagar, D., Bhobe, A., Bianchini, P., Bishop, L.S., Blank, C., Bolos, S., Bopardikar, S., Bosch, S., Brandhofer, S., Brandon, Bravyi, S., Bronn, N., Bryce-Fuller, Bucher, D., Burov, A., Cabrera, F., Calpin, P., Capelluto, L., Carballo, J., Carrascal, G., Carriker, A., Carvalho, I., Chen, A., Chen, C.F., Chen, E., Chen, J.C., Chen, R., Chevallier, F., Chinda, K., Cholarajan, R., Chow, J.M., Churchill, S., CisterMoke, Claus, C., Clauss, C., Clothier, C., Cocking, R., Cocuzzo, R., Connor, J., Correa, F., Cross, A.J., Cross, A.W., Cross, S., Cruz-Benito, J., Culver, C., Córcoles-Gonzales, A.D., D, N., Dague, S., Dandachi, T.E., Dangwal, A.N., Daniel, J., Daniels, M., Dartiailh, M., Davila, A.R., Debouni, F., Dekusar, A., Deshmukh, A., Deshpande, M., Ding, D., Doi, J., Dow, E.M., Drechsler, E., Dumitrescu, E., Dumon, K., Duran, I., EL-Safty, K., Eastman, E., Eberle, G., Ebrahimi, A., Eendebak, P., Egger, D., ElePT, Emilio, Espiricueta, A., Everitt, M., Facoetti, D., Farida, Fernández, P.M., Ferracin, S., Ferrari, D., Ferrera, A.H., Fouilland, R., Frisch, A., Fuhrer, A., Fuller, B., GEORGE, M., Gacon, J., Gago, B.G., Gambella, C., Gambetta, J.M., Gammanpila, A., Garcia, L., Garg, T., Garion, S., Garrison, J.R., Garrison, J., Gates, T., Gil, L., Gilliam, A., Giridharan, A., Gomez-Mosquera, J., Gonzalo, de la Puente González, S., Gorzinski, J., Gould, I., Greenberg, D., Grinko, D., Guan, W., Guijo, D., Gunnels, J.A., Gupta, H., Gupta, N., Günther, J.M., Haglund, M., Haide, I., Hamamura, I., Hamido, O.C., Harkins, F., Hartman, K., Hasan, A., Havlicek, V., Hellmers, J., Herok, Ł., Hillmich, S., Horii, H., Howington, C., Hu, S., Hu, W., Huang, J., Huisman, R., Imai, H., Imamichi, T., Ishizaki, K., Ishwor, Iten, R., Itoko, T., Ivrii, A., Javadi, A., Javadi-Abhari, A., Javed, W., Jianhua, Q., Jivrajani, M., Johns, K., Johnstun, S., Jonathan-Shoemaker, JosDenmark, JoshDumo, Judge, J., Kachmann, T., Kale, A., Kanazawa, N., Kane, J., Kang-Bae, Kapila, A., Karazeev, A., Kassebaum, P., Kelso, J., Kelso, S., Khanderao, V., King, S., Kobayashi, Y., Kovi11Day, Kovyrshin, A., Krishnakumar, R., Krishnan, V., Krsulich, K., Kumkar, P., Kus, G., LaRose, R., Lacal, E., Lambert, R., Landa, H., Lapeyre, J., Latone, J., Lawrence, S., Lee, C., Li, G., Lishman, J., Liu, D., Liu, P., Madden, L., Maeng, Y., Maheshkar, S., Majmudar, K., Malyshev, A., Mandouh, M.E., Manela, J., Manjula, Marecek, J., Marques, M., Marwaha, K., Maslov, D., Maszota, P., Mathews, D., Matsuo, A., Mazhandu, F., McClure, D., McElaney, M., McGarry, C., McKay, D., McPherson, D., Meesala, S., Meirom, D., Mendell, C., Metcalfe, T., Mevissen, M., Meyer, A., Mezzacapo, A., Midha, R., Miller, D., Minev, Z., Mitchell, A., Moll, N., Montanez, A., Monteiro, G., Mooring, M.D., Morales, R., Moran, N., Morcuende, D., Mostafa, S., Motta, M., Moyard, R., Murali, P., Müggenburg, J., NEMOZ, T., Nadlinger, D., Nakanishi, K., Nannicini, G., Nation, P., Navarro, E., Naveh, Y., Neagle, S.W., Neuweiler, P., Ngoueya, A., Nicander, J., Nick-Singstock, Niroula, P., Norlen, H., NuoWenLei, O'Riordan, L.J., Ogunbayo, O., Ollitrault, P., Onodera, T., Otaolea, R., Oud, S., Padilha, D., Paik, H., Pal, S., Pang, Y., Panigrahi, A.,

Pascuzzi, V.R., Perriello, S., Peterson, E., Phan, A., Piro, F., Pistoia, M., Piveteau, C., Plewa, J., Pocreau, P., Pozas-Kerstjens, A., Pracht, R., Prokop, M., Prutyanov, V., Puri, S., Puzzuoli, D., Pérez, J., Quant02, Quintiii, R, I., Rahman, R.I., Raja, A., Rajeev, R., Ramagiri, N., Rao, A., Raymond, R., Reardon-Smith, O., Redondo, R.M.C., Reuter, M., Rice, J., Riedemann, M., Rietesh, Risinger, D., Rocca, M.L., Rodríguez, D.M., RohithKarur, Rosand, B., Rossmannek, M., Ryu, M., SAPV, T., Sa, N.R.C., Saha, A., Ash-Saki, A., Sanand, S., Sandberg, M., Sandesara, H., Sapra, R., Sargsyan, H., Sarkar, A., Sathaye, N., Schmitt, B., Schnabel, C., Schoenfeld, Z., Scholten, T.L., Schoute, E., Schulterbrandt, M., Schwarm, J., Seaward, J., Sergi, Sertage, I.F., Setia, K., Shah, F., Shammah, N., Sharma, R., Shi, Y., Shoemaker, J., Silva, A., Simonetto, A., Singh, D., Singh, D., Singh, P., Singkanipa, P., Siraichi, Y., Siri, Sistos, J., Sitdikov, I., Sivarajah, S., Sletfjerding, M.B., Smolin, J.A., Soeken, M., Sokolov, I.O., Sokolov, I., Soloviev, V.P., SooluThomas, Starfish, Steenken, D., Stypulkoski, M., Suau, A., Sun, S., Sung, K.J., Suwama, M., Słowik, O., Takahashi, H., Takawale, T., Tavernelli, I., Taylor, C., Taylour, P., Thomas, S., Tian, K., Tillet, M., Tod, M., Tomasik, M., Tornow, C., de la Torre, E., Toural, J.L.S., Trabing, K., Treinish, M., Trenev, D., TrishaPe, Truger, F., Tsilimigkounakis, G., Tulsi, D., Turner, W., Vaknin, Y., Valcarce, C.R., Varchon, F., Vartak, A., Vazquez, A.C., Vijaywargiya, P., Villar, V., Vishnu, B., Vogt-Lee, D., Vuillot, C., Weaver, J., Weidenfeller, J., Wieczorek, R., Wildstrom, J.A., Wilson, J., Winston, E., WinterSoldier, Woehr, J.J., Woerner, S., Woo, R., Wood, C.J., Wood, R., Wood, S., Wootton, J., Wright, M., Xing, L., YU, J., Yang, B., Yeralin, D., Yonekura, R., Yonge-Mallo, D., Yoshida, R., Young, R., Yu, J., Yu, L., Zachow, C., Zdanski, L., Zhang, H., Zoufal, C., aeddins ibm, alexzhang13, b63, bartek bartlomiej, bcamorrison, brandhsn, charmerDark, deeplokhande, dekel.meirom, dime10, dlasecki, ehchen, fanizzamarco, fs1132429, gadial, galeinston, georgezhou20, georgios ts, gruu, hhorii, hykavitha, itoko, jessica angel7, jezerjojo14, jliu45, jscott2, klinvill, krutik2966, ma5x, michelle4654, msuwama, ntgiwsvp, ordmoj, sagar pahwa, pritamsinha2304, ryancocuzzo, saswati qiskit, septembrr, sethmerkel, shaashwat, sternparky, strickroman, tigerjack, tsura crisaldo, vadebayo49, welien, willhbang, wmurphy collabstar, yang.luh, Čepulkovskis, M.: Qiskit: An open-source framework for quantum computing (2021). https://doi.org/10.5281/zenodo.2573505

2. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing. pp. 212–219 (1996)
3. Mandviwalla, A., Ohshiro, K., Ji, B.: Implementing grover's algorithm on the ibm quantum computers. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 2531–2537. IEEE (2018)
4. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. a correction. Proceedings of the London Mathematical Society **2**(1), 544–546 (1938)
5. Von Neumann, J.: First draft of a report on the edvac. IEEE Annals of the History of Computing **15**(4), 27–75 (1993)