Application of BFS algorithm on classical and quantum computers based on modern technologies of e-education

Nuray Shiraliyeva

Abstract

Everyone knows how big the role of computers is in our daily life. In this era we live in, especially in recent years, the Internet and computers have already become the most valuable parts of our lives. The power behind this development of technology belongs to science. It should noted that thanks to technical sciences, we were able to move today's technology so far. Quantum mechanics is a very large field of science, and this topic, which we cannot finish no matter how much we talk about it, has begun to speak for itself in the world of technology. This basically called the key to unlimited power, and the breadth-search algorithm (BFS) will reveal how this power pushes the limits of our brain's thinking.

Keywords: Quantum theory, technology, science, unlimited power

The famous physicist Richard Feynman:"If you don't understand quantum mechanics, don't worry, because nobody does."Sometimes when we see some laws of nature outside of the laws of nature that we witness around us, we cannot remember it and see that event as an illusion. Are we not sure that the moon is there even when we are not looking? But this is not the case in quantum mechanics...This is where this state called the superposition state arises and this unlimited power arises from here. We talked about the importance of science in technology, but already F=-F: technology must come to the aid of science. For years, this great theory-quantum theory, which even great physicists like those that Einstein could not explain, is trying to explain with the help of technology. Briefly, if we look at quantum theory, it will be clear what this unlimited power is. If we have to give an example, we can give this example: Think of 2 people, when the society observes these 2 people, they behave as if they are 2 separate individuals doing different things, but the moment they stop observing, these people collaborate and come up with a very big project. It sounds very logical, but now consider this situation between photon particles, how much communication is possible between the photon particles that make up the light? The experience goes like this: scientists get together and take two photons and two plates, each of which has holes and photons can easily move through these holes. Scientists send these photons under their observation, and these photons create an image on the wall individually, like particles, as it should be, it is normal, but the scientists leave the device that sends the photons to the opposite board on and take a break, when they come back, they cannot believe their eyes, it is ours. They faced with a vision that considered a miracle for scientists and a horror for scientists. The photons reflected on the wall in the form of different waves...

Seeing that this was the case, the scientists placed a camera and left the room, and decided to follow the movement of photons through the camera so that they could understand what was happening that they did not see at that moment. What are you waiting for? Again, he was not aware that these photons were being observed (which is an impossibility anyway, so it should be) that they would show a wave function, right? Photons behave like particles, as if they sense that they observe by that camera...

Welcome to the quantum world! After the experiment, I must note that this phenomenon fully explained and still considered one of the biggest problems of physics. Seeing that this is the case, the world of science tries to explain this theory by applying it to technology, contrary to what we used to. Normally, some theory presented and we then apply it to the technology. Now, the most exciting part of the work, when talking about the quantum mechanism so far, we mentioned many different theorems, we talked about experiments that push the limits of our minds, mechanisms that work very differently from the macrocosm. When talking about all this, there was only one question in our mind? What does the phenomenon of inertia, wave-particle duality, quantum entanglement, even radiation mean for us? What is the real life equivalent of these? Quantum computers are today's equivalent of these ways that we have come from Max Planck. Many doors will open before us. First, we must remember that the classic computers we use today cannot

switch from one calculator to another. It is a very advanced calculator. The computers we use have limits. They were able to bring humanity forward, but they could not overcome these limits. Even if we continue to use them, there are many problems waiting to solve that they cannot overcome, so we need a huge innovation. We must not forget this, because when we think of quantum computers, everyone thinks of a new, more powerful version of the classical computers we use, but quantum computers are a completely new concept. That is, the comparison of quantum computers and classical computers is similar to the comparison of the classical mechanism of physics with the quantum mechanism. That is, there is no similarity or we are not able to perceive it. However, what are these quantum computers? Frankly, it is the most complicated mechanism ever created by humankind. The underlying reason for this is that these computers connected with all the complex theorems revealed by the quantum mechanism. A breadth-first search (BFS) algorithm used to search a tree or graph data structure for a node that meets a set of criteria. It starts at the root or graph of the tree and searches/visits all nodes at the current depth level before moving to the nodes at the next depth level. Breadth-first search can used to solve many problems in graph theory. This algorithm itself is a slow algorithm because it approaches each node separately and walks one by one. This algorithm used on a real quantum computer. However, when we build a quantum circuit in python and simulate it, we can see that it is much faster than the classical circuit.
If we go over the code that will appear with the simulation:
1. Import matplotlib.pyplot as plt
2. Import networkx as nx
Explanation: Python libraries used to create analyze and visualize graph and network structure. Through these libraries, graphs can created according to graph theory and can be simulated in the form of animation.
3. def bfs(graph,start):
4. isited=set()
5.queue=[start]
6. While queue :
7. vertex=queue.pop(0)
8. if vertex not in visited :
9. isited.add(vertex)
10.queue.extend(set(graph[vertex])-visited)
The first line executes the BFS function on the given graph, the set given structure is created to hold the traversable nodes, the tail structure of the BFS algorithm, if there is an element in the queue in the 6th line, the function will continue to work, the elements are removed from the queue, if the element has not been visited before, is visited and added to the set. Nodes not visited by outgoing links added to the queue. If we were to show an example of how this code would work with graphs:
Graph=defaultdict(list)
Graph[1].append(2)
Graph[1].append(3)
Graph[2].append(4)
Graph[3].append(5)
Let the starting node be 1, in which case the BFS algorithm will work like this:
1. First 1 will be visited and added to the set called visited.
2 and 3, which are neighbors of 2.1, will be added to the queue. (queue=[2,3])
3. 2 is taken from Queue and it is added to the visited set as it is found not to be in the visited set, and 4, which is a neighbor of 2, is also added to the queue. (queue=[3,4])
4. 3 is taken from the queue and enters the visited set, in this case, its neighbor 4 should be added to the queue, but since 4 is already in the queue, it is not needed. (queue=[4])
5. In the same way, 4 is taken from the queue and its neighbor 5 is added to the visited queue. (queue=[5])

6. 5 is taken from Queue and goes to visited.
7. And there are no more unvisited nodes left, so the BFS work is finished.
Suppose the first 4 elements in the queue have been visited and added to visited, the first step is complete. Then the neighbor 5 came and we sent him to visited, the second step was completed. These first-second steps are done in the code with vertex operation.
Queue.extend(set(graph[vertex])-visited)
This line is one of the most important parts of the BFS algorithm. This line is the command that creates the basic logic of the BFS algorithm and serves to keep BFS constantly updated. Here, the neighbors of a node running in the BFS queue are added to the queue and the available node neighbors are obtained with graph[vertex].
11. plt.clf()
12. Nx.draw(graph, with_labels=True)
13. plt.draw()
14. plt.pause(0.5)
15. Return visited
Explanation: The previously drawn plot is deleted, plotting is done with the networkx library, the matplotlib library is used to update the plot,
The drawing process continues at 0.5 second intervals, returning the specified data containing the nodes that have been cycled as a result of the BFS algorithm.
16.graph = nx.digraph()
17.graph.add_edge(0, 1)
18. graph.add_edge(0, 2)
19. graph.add_edge(1, 2)
20. graph.add_edge(2, 0)
21. graph.add_edge(2, 3)
22. graph.add_edge(3, 3)
A directed graph is created, the vertices of the graph and the relationships between the vertices are determined.
23.start = 2
The first knot 2 is selected as the starting point.
24.nx.draw(graph, with_labels=True)
25.plt.show()
A graph is drawn, used to display the image on the screen
26.bfs(graph, start)
The generated graph and starting node are given as parameters of the BFS function.
BFS in quantum computers
The same simulation for quantum computers:
Import matplotlib.pyplot as plt
Import networkx as nx
From qiskit import quantumregister, classicalregister, quantumcircuit, Aer, execute
We introduce the matplotlib and networkx libraries to create the network between mathematical graphs and graphs, and at the same time, we introduce 4 classes from the kitkit created separately for quantum theory.

```
Def bfs(graph, start):
    Visited = set()
    Queue = [start]
    While queue:
        Vertex = queue.pop(0)
        If vertex not in visited:
            Visited.add(vertex)
            Queue.extend(set(graph[vertex]) - visited)
```

We introduce the BFS algorithm for quantum computers in the same way as we introduce it for classical computers. The first line executes the BFS function on the given graph, creates a set given structure that will hold the visited nodes, the tail structure of the BFS algorithm, in the 6th line, if there is an element in the queue, the function will continue to work, the elements are removed from the queue, if the element has not been visited before, it is visited and is added to the set. Nodes not visited by outgoing links added to the queue.

```
Plt.clf()
            Nx.draw(graph, with_labels=True)
            Plt.draw()
            Plt.pause(0.5)
    Return visited
```

Here we introduce the graph drawing process, just like in classic computers. The previously drawn graph is deleted, drawing graphs is done with the networkx library, the matplotlib library is used to update the plot,

The drawing process continues at 0.5 second intervals, returning the specified data containing the nodes that have been cycled as a result of the BFS algorithm.

```
Graph = nx.digraph()
Graph.add_edge(0, 1)
Graph.add_edge(0, 2)
Graph.add_edge(1, 2)
Graph.add_edge(2, 0)
Graph.add_edge(2, 3)
Graph.add_edge(3, 3)
```

A directed graph is created, the vertices of the graph and the relationships between the vertices are determined.

```
Start = 2
```

2 is taken as the starting element.

The main difference of quantum computers:

```
Q = quantumregister(2)
C = classicalregister(2)
Qc = quantumcircuit(q, c)
```

In this code, a quantum circuit is created using the IMB Qiskit library. First, two quantum registers and two classical registers are taken and they are combined in the quantumcircuit object. This object, which we call quantumcircuit, contains 2 quantum bits and 2 classical bits. These are just the first steps to get the circuit working.

```
Qc.h(q[0])
Qc.h(q[1])
Qc.barrier()
```

Explanation : The qc.barrier() method is a tool used to improve code readability and circuit design. This method ensures stabilization of the states of all quantum bits (qubits) at a certain stage of the circuit. Let me try to explain with an example. Suppose there are two quantum bits in a circuit called Qubit 0 and Qubit 1. They can be in any state of 0 or 1 in this circuit. Also, these two Qubits work independently of each other. If qc.barrier() is not used, for example, the first Qubit can be in state 0 and the second Qubit in state

1. But in this case, when the first qubit is in the 0 state, and the second qubit is in the 1 state, the operation of the circuit is disturbed and gives wrong results.

```
For i in range(len(graph)):
   Qc.h(q[0])
   Qc.h(q[1])
   Qc.barrier()
   For u, v in graph.edges():
      If u == i:
         Qc.cx(q[0], q[1])
      If v == i:
         Qc.cx(q[1], q[0])
   Qc.barrier()
   Qc.h(q[0])
   Qc.h(q[1])
   Qc.barrier()
   Qc.measure(q, c)
```

For i in the range (len(graph)):
This loop visits all the nodes in the graph and performs each step of the search process.
Qc.h(q[0]) qc.h(q[1]) qc.barrier()
These lines allow the superposition of two qubits using a Hadamard gate. The instructions qc.h(q[0]) and qc.h(q[1]) apply a Hadamard gate on each qubit, and the qubits are in the 1 and 0 states at the same time. Then the qc.barrier() instruction completes the loop up to this point by determining the superposition state of the qubits.

Graph.edges() for u, v:
This loop visits all the edges of the graph.
If u == i: qc.cx(q[0], q[1])
This condition ensures that a CNOT (CX) gate is applied between two qubits when node u is equal to node i. In some cases, qubits are connected to each other, which makes it easier to find the desired element.
If v == i: qc.cx(q[1], q[0])
This condition ensures that a CNOT (CX) gate is applied between the other two qubits when node v is equal to node i.
Qc.barrier()
This line indicates that all CNOT operations have completed and will proceed to the next step.
Qc.h(q[0]) qc.h(q[1]) qc.barrier()
These lines allow superposition of qubits using a Hadamard gate.
Qc.measure(q, c)
This line performs the measurement of the qubits and stores the results in classical bits c.
This piece of code searches using Grover's algorithm. The searched element is found in the given graph and the results are stored in classical bits.

```
Backend = Aer.get_backend('gasm_simulator')
   Job = execute(qc, backend=backend, shots=1024)
   Counts = job.result().get_counts(qc)
   For key in counts:
      If key == '00':
         Bfs_node = i
   Bfs(graph, bfs_node)
```

Explanation : These lines run the predefined quantum circuit by calling the BFS algorithm, taking the results.
First, an instance is created for the quantum backend called Aer. A predetermined quantum circuit qc is then made to operate on this backend using an implementation function. The count parameter specifies how many times the specified loop will be run and is assigned to the counts variable where the results are collected.
Then the counts variable looped and when a result is obtained with the key 00, the bfs_node variable is defined and this value is passed as a parameter to the bfs function and the BFS algorithm is started.
Thus, the BFS algorithm implemented using the results of the quantum circuit.

Conclusion

The two code examples will perform the same function, but the speed difference between them will be visible through simulation. If the application of quantum theory is extended to computers, the great revolution it will bring to the Internet and information security will shift the wars from weapons and arm power to 0 s and 1s. The importance of this theory in politics can be expressed by this sentence alone. This great power, which can calculate all probabilities at the same time in the case of superposition, can calculate all the probabilities of a social media account password in a very short time, if we give a simple example, and can break it. For quantum researchers, the goal is to implement this rapid and massive innovation first in their own country and then around the world. The application of quantum theory in technology is a project that we have to work on for a very long time. Moreover, according to scientists' calculations, it will take at least 10 more years before we can get this project somewhere. 10 years is a long time, and during these 10 years, we should at least recognize this theory and apply it in small steps to protect our country from danger.

References

[1] Theoretical Concepts of Quantum Mechanics- edited by Mohammad Reza Pahlavani, Complementarity in Quantum Mechanics and Classical Statistical Mechanics pg 1, by Luisberis Velazquez Abad and Sergio Curilef Huichalaf
[2] Departamento de Física, Universidad Católica del Norte Chile
[3] Theoretical Concepts of Quantum Mechanics- edited by Mohammad Reza Pahlavani,
The Physical Nature of Wave/Particle Duality pg 23 by Marcello Cini Università La Sapienza, Roma
[4] Learning python by Mark Lutz-Module Coding Basics pg 54geekforgeeks.org