Kvantno računanje

Josipa Šimić
1191245326

Demonstracija rada Shorovog algoritma pomoću Qiskit simulatora

1. Općenito o algoritmu

Poznato nam je da Shorov algoritam rješava problem faktorizacije prirodnog broja $N = p \cdot q$ $(p, q \in \mathbb{P})$, slijedeći navedene korake:

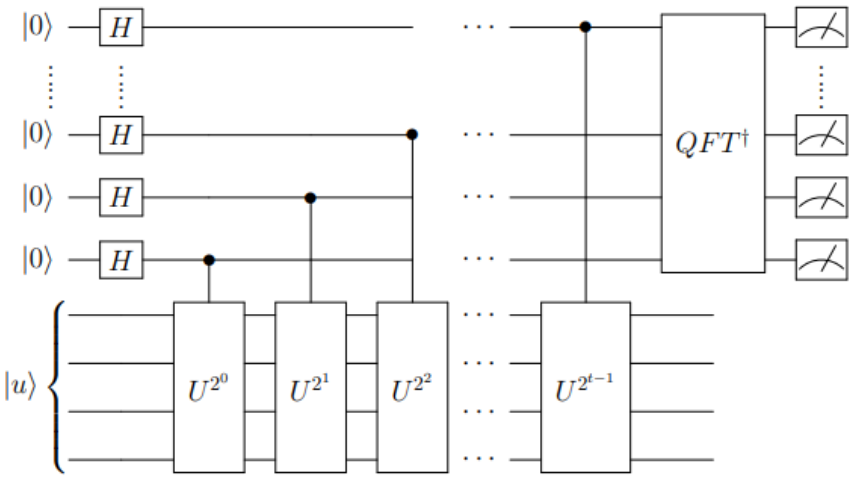
1) izaberi proizvoljan $a \in \mathbb{N}$
2) ako vrijedi $\text{NZM}(a, N) > 1$: nađi novi a
   inače:
   a) nađi najmanji $r \in \mathbb{N}$ $td.\ a^r \equiv 1\ (mod\ N)$
   b) ako je r neparan: nađi novi a
      inače:
      i) definiramo $x \equiv a^{\frac{r}{2}}\ (mod\ N)$
      ii) ako vrijedi $x + 1 \equiv 0\ (mod\ N)$: nađi novi a
      inače: vrati $\{\text{NZM}(x + 1,\ N), \text{NZM}(x - 1,\ N)\}$

Barem jedan od prostih faktora broja N se nalazi u skupu koji algoritam vraća.

Kako bi (barem otprilike) izgledala implementacija ovog algoritam?

Vidimo da su svi koraci algoritma trivijalni za implementirati u bilo kojem programskom jeziku, osim koraka 2. a) – ne samo da nije trivijalno, nego taj korak nije ni moguće implementirati na klasičnom računalu s očuvanjem polinomijalne složenosti algoritma. Međutim, za utjehu možemo simulirati generalnu ideju implementacije kakva bi bila na kvantnom računalu, i vidjeti demonstraciju rada algoritma na dovoljno malenom broju, kao što je 15.

Korak 2. a), to jest pronalaženje reda r od a modulo N, je zapravo proces kvantne estimacije faze (QPE), prikazan sljedećim kvantnim sklopom:



U implementaciji trebamo koristiti Hadamardov operator za početnu inicijalizaciju qubita, kontrolirani mod N operator (unitarni operator koji kontrolirano djeluje na $|u\rangle$) , te operator inverzne Fourierove transformacije.

2. Implementacija u Pythonu korištenjem Qiskit biblioteke

Inicijalizaciju ulaznih qubita simuliramo pomoću Qiskitovih metoda za primjenu Hadamardovih vrata:

```
def InitializeQubits(circuit, n, m):

  circuit.h(range(n))
  circuit.x(n+m-1)
```

(*Shorov algoritam.py*, linije 4 – 7)

n je broj ulaznih $|0\rangle$ qubita za krajnje mjerenje, dok je m broj qubita čiji tenzorski produkt određuje $|u\rangle$. Primijetimo da smo zadnji qubit unutar $|u\rangle$ negirali, tako da poprimi vrijednost $|1\rangle$, budući da nam je upravo faza tog vektora od interesa u traženju reda r.

Metoda za kontrolirani modularni operator je specifično implementirana za simuliranje ostataka modulo 15, i generalno ju nije lako simulirati za proizvoljan N, stoga ju preuzimamo iz Qiskit bilježnice[1]:

```
def CircuitMod15(a, power):

  if a not in [2,4,7,8,11,13]:
    raise ValueError("Broj a mora biti jednak 2,4,7,8,11 ili 13!")
  U = QuantumCircuit(4)
  for iteration in range(power):
    if a in [2,13]:
      U.swap(2,3)
      U.swap(1,2)
      U.swap(0,1)
    if a in [7,8]:
      U.swap(0,1)
      U.swap(1,2)
      U.swap(2,3)
    if a in [4, 11]:
      U.swap(1,3)
      U.swap(0,2)
    if a in [7,11,13]:
      for q in range(4):
        U.x(q)
  U = U.to_gate()
  U.name = "%i^%i mod 15" % (a, power)
  c_U = U.control()
  return c_U

def ModularExponentiation(circuit, n, m, a):

  for x in range(n):
    exponent = 2**x
    circuit.append(CircuitMod15(a, exponent),
            [x] + list(range(n, n+m)))
```

(*ShorovAlgoritam.py*, linije 11 – 41)

Na kraju koristimo i inverz kvantne Fourierove transformacije za prvih n qubita:

```
from qiskit.circuit.library import QFT

def QFTInverse(circuit, qubits):

   circuit.append(QFT(len(qubits), do_swaps=True).inverse(), qubits)
```

(*ShorovAlgoritam.py*, linije 45 – 47)

te sve skupa koristimo u izvršavanju kvantne estimacije faze:

```
def ExecuteQPE(n, m, a):

   shor = QuantumCircuit(n+m, n)

   # Hadamard i negacija
   InitializeQubits(shor, n, m)
   shor.barrier()

   # modularni operator
   ModularExponentiation(shor, n, m, a)
   shor.barrier()

   # transponirana kvantna Fourierova transformacija
   QFTInverse(shor, range(n))

   # mjerimo izlaz
   shor.measure(range(n), range(m))

   return shor
```

(*ShorovAlgoritam.py*, linije 49 – 67)

Algoritam konkretno testiramo za $n = m = 4, a = 7$ (i *već spomenuto* $N = 15$):

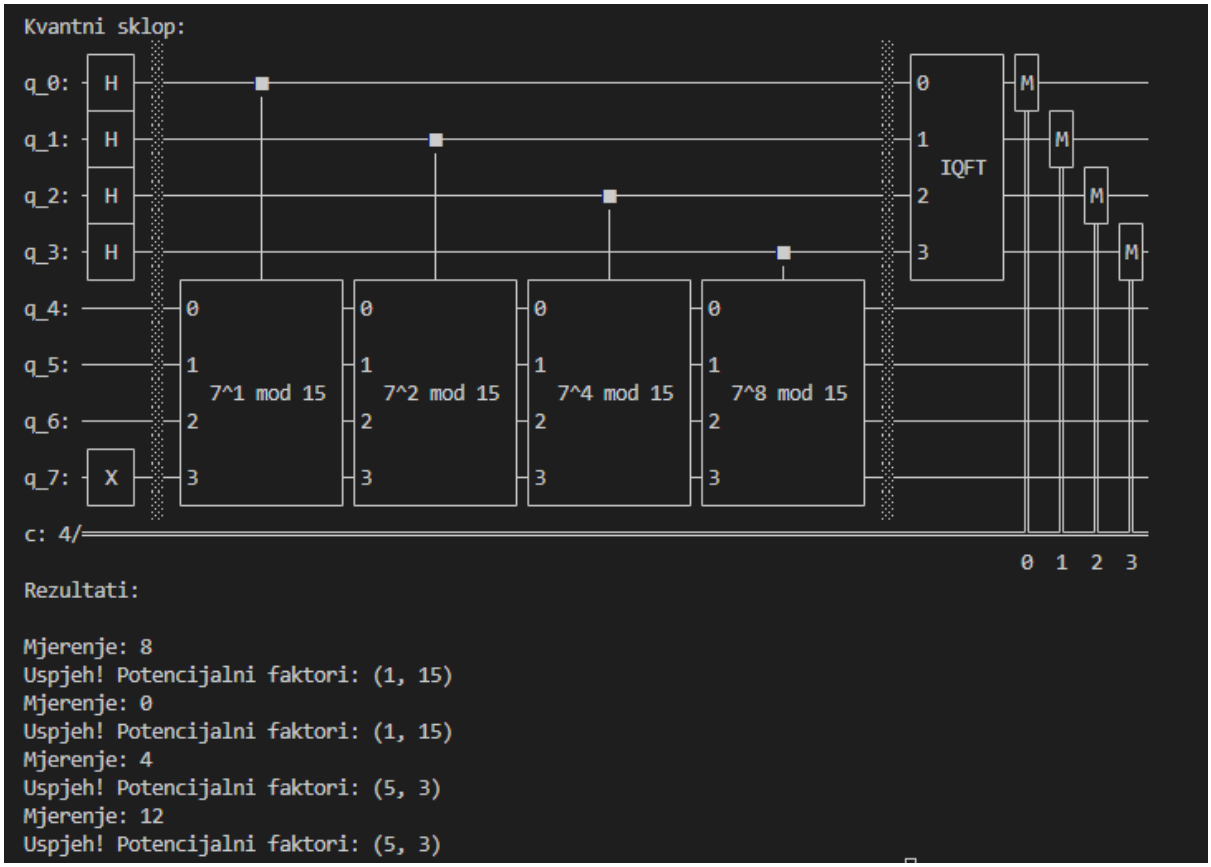
```
# pokretanje kvantnog algoritma
n = 4; m = 4; a = 7
mycircuit = ExecuteQPE(n, m, a)
```

(*ShorovAlgoritam.py*, linije 69 – 73)

3. Pokretanje programa

Program se može pokrenuti unutar bilo koje razvojne okoline koja podržava programski jezik Python, uz uvjet instalirane qiskit biblioteke. Program se također može pokrenuti i direktno u terminalu, uz isti uvjet. U izlazu program iscrtava konstruirani kvantni sklop, kao i Shorov post-proces koji prihvaća/odbija dobivena mjerenja prema pravilima iz algoritma.

Izlaz:



4. Literatura

[1]https://qiskit.org/textbook/preface.html

https://qiskit.org/learn/summer-school/introduction-to-quantum-computing-and-quantum-hardware-2020/

Matija Kazalicki: *Kvantno računanje*, Zagreb