Quantum searching

Algoritmul lui Grover

# Descriere:

Algoritmul lui Grover eficientizează problema de căutare pentru un set de date neordonate folosind mai puţini paşi decât oricare alt algoritm clasic. Acest algoritm a fost conceput de Lov Grover in 1996. Algoritmul găseşte un singur element marcat într-un set de n elemente în complexitatea O(log(N)). Un algoritm clasic de căutare ar avea complexiatea O(N/2).

# Aplicații

Algoritmul lui Grover este aplicabil petru orice problemă care poate fi redusă la verificarea validității unei valori date, de exemplu căutarea într-un set de date, factorizarea numerelor întregi, rezolvarea formulelor booleene și Traveling Salesman Problem.

Algoritmul Grover poate fi utilizat pentru a decoda chei criptografice considerate sigure, pentru diverse atacuri brute-force asupra criptografiei cu cheie simetrică, inclusiv atacuri de coliziune și atacuri pre-imagine. De exemplu, ar putea decoda o cheie criptografică simetrică de 128 de biţi folosind brute-force în aproximativ 264 iteraţii şi o cheie de 256 de biţi în ≈2128 iteraţii. Acest lucru prezintă un risc pentru securitatea tranzacțiilor online și comunicațiilor.

Mai poate fi utilizat pentru a rezolva probleme de optimizare care sunt dificile sau imposibil de rezolvat cu un computer clasic. Poate găsi mult mai eficient cel mai scurt drum între două puncte sau poate determina ce investiţii vor fi cele mai profitabile. De asemenea, poate fi folosit pentru detectarea activităţilor online illegale sau pentru diagnosticarea unor afecțiuni cum ar fi identificarea celulelor canceroase.

În domeniul învățării automate (machine learning) acest algoritm poate fi folosit pentru a eficientiza găsirea unor soluţii sau datelor relevante în baze mari de date.

# Limitări

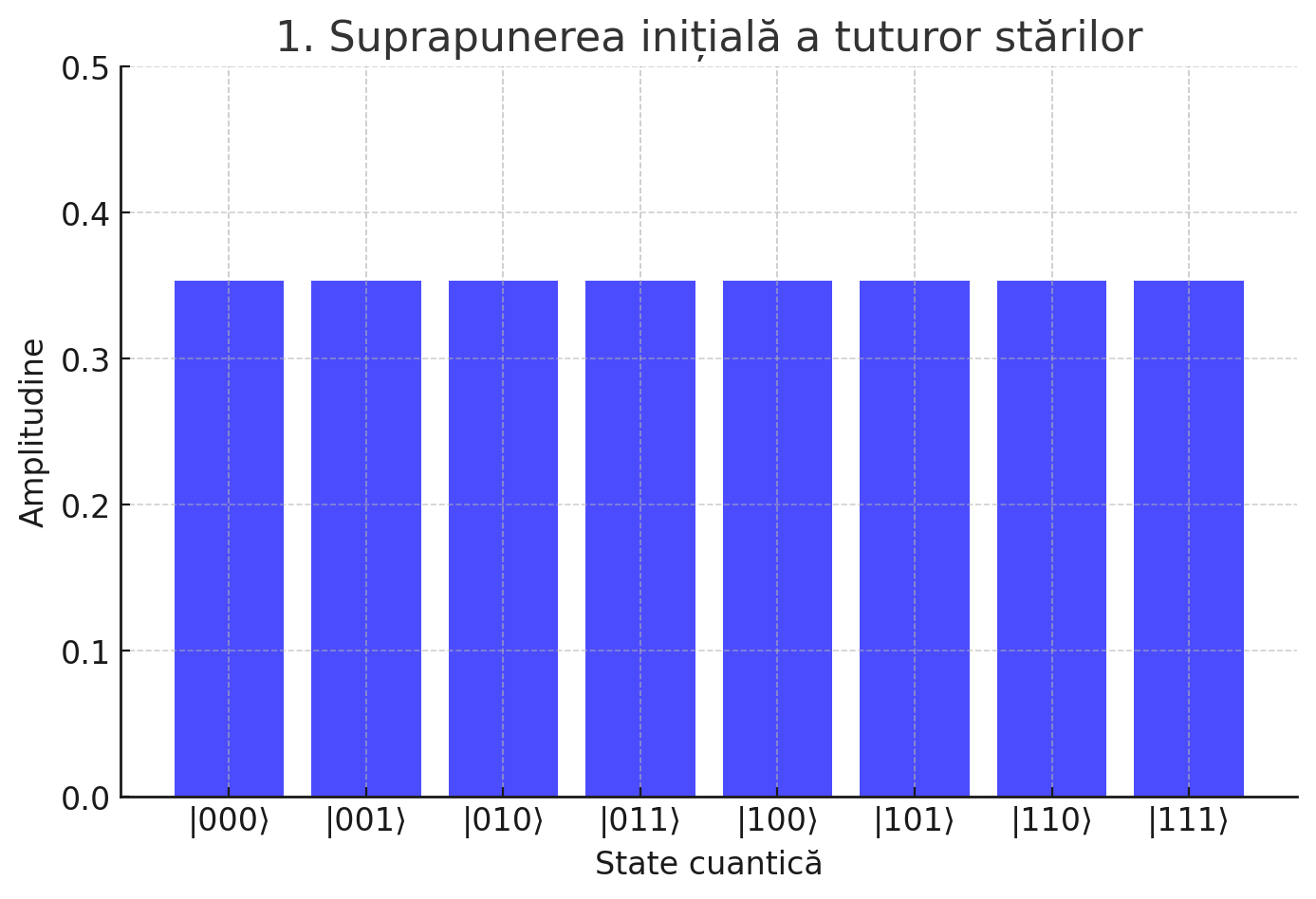
Acest algoritm nu este încă practic deoarece necesită un număr mare de qubiţi pentru unele aplicaţii, iar in actualitate, calculatoarele cuantice nu sunt destul de avansate. Algoritmul lui Grover oferă un avantaj semnificativ doar pentru problemele care au un spațiu mare de căutare. Pentru probleme cu seturi de date mai restrânse, eficientizarea nu este destul de semnificativă pentru a justifica utilizarea acestuia.

La fel ca orice algoritm cuantic, algoritmul lui Grover este susceptibil la erori generate de noise– cauzat de interferenţe ale altor dispozitive, câmpuri magnetice şi interacţiuni între qubiţi; şi decoherence– colapsul superpoziţiei.

# Etapele Algoritmului Grover:

## 1. Starea inițială – Toate numerele sunt în suprapunere

* Algoritmul începe cu un registru cuantic reprezentând un șir de N elemente (în acest exemplu, avem 8 numere).
* Fiecare stare cuantică corespunde unui număr din șir.
* Aplicăm poarta Hadamard pe fiecare qubit, ceea ce creează o suprapunere uniformă a tuturor stărilor posibile.
* În această stare inițială, fiecare număr are aceeași amplitudine și, implicit, aceeași probabilitate de apariție la măsurare.

****

## 2. Inversarea fazei pentru numărul căutat

* Oracle-ul este o funcție cuantică care marchează numărul pe care dorim să îl găsim.
* Acest lucru se face prin inversarea fazei stării cuantice corespunzătoare numărului căutat.
* În termeni vizuali, dacă amplitudinea inițială a tuturor stărilor era pozitivă, după aplicarea oracle-ului, amplitudinea numărului căutat devine negativă, în timp ce celelalte rămân neschimbate**.**

****

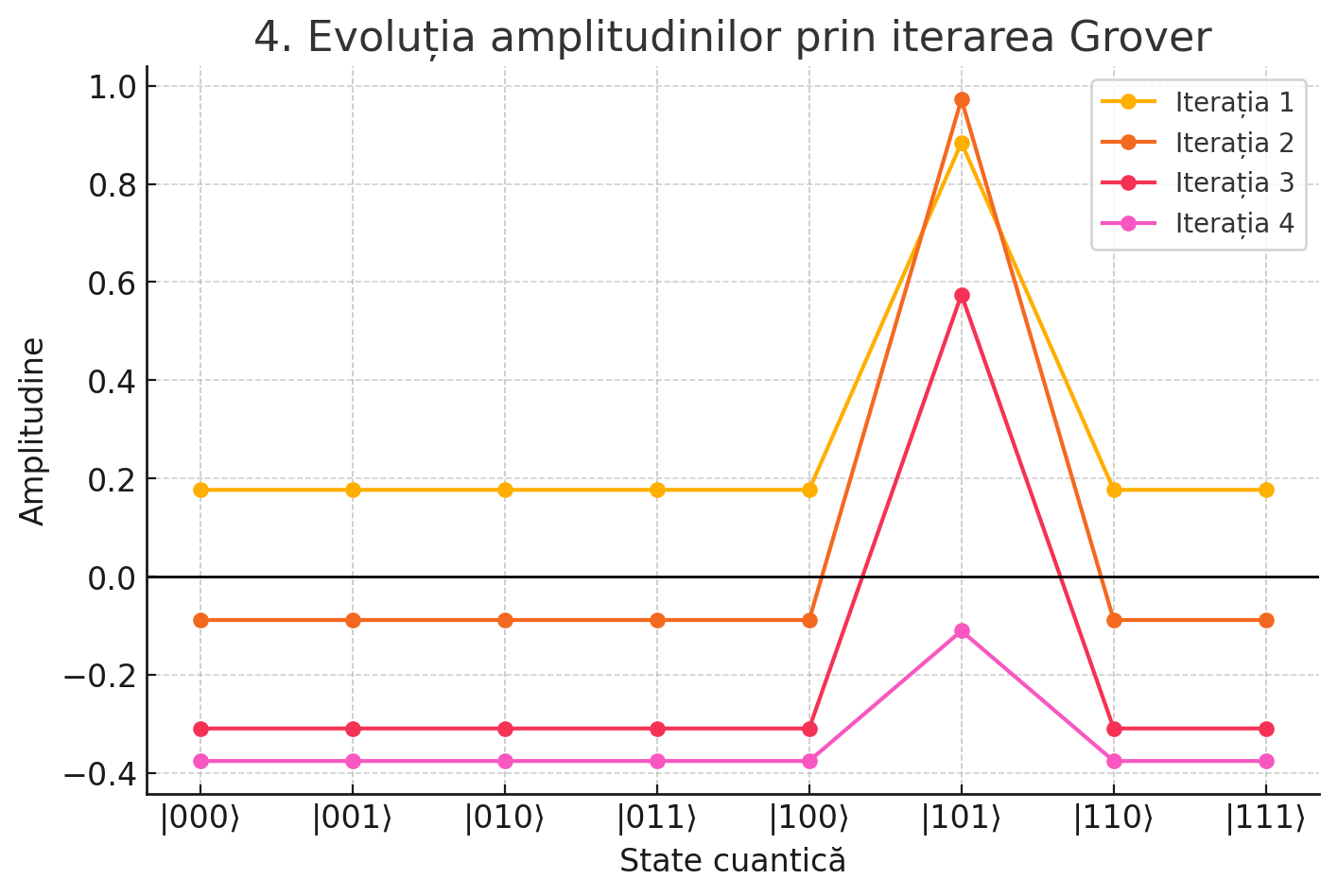
## 3. Operatorul Grover crește probabilitatea numărului căutat

* Se aplică operatorul de difuzie Grover, care este un fel de „reflecție față de medie”.
* Acesta funcționează astfel:
  1. Calculează media amplitudinilor tuturor stărilor.
  2. Ajustează amplitudinile astfel încât valorile deja mici scad și mai mult, iar valoarea numărului căutat crește.
* Aceasta este cheia algoritmului: în loc să căutăm numărul în mod clasic, creștem probabilitatea de a-l găsi prin amplificarea amplitudinilor sale.



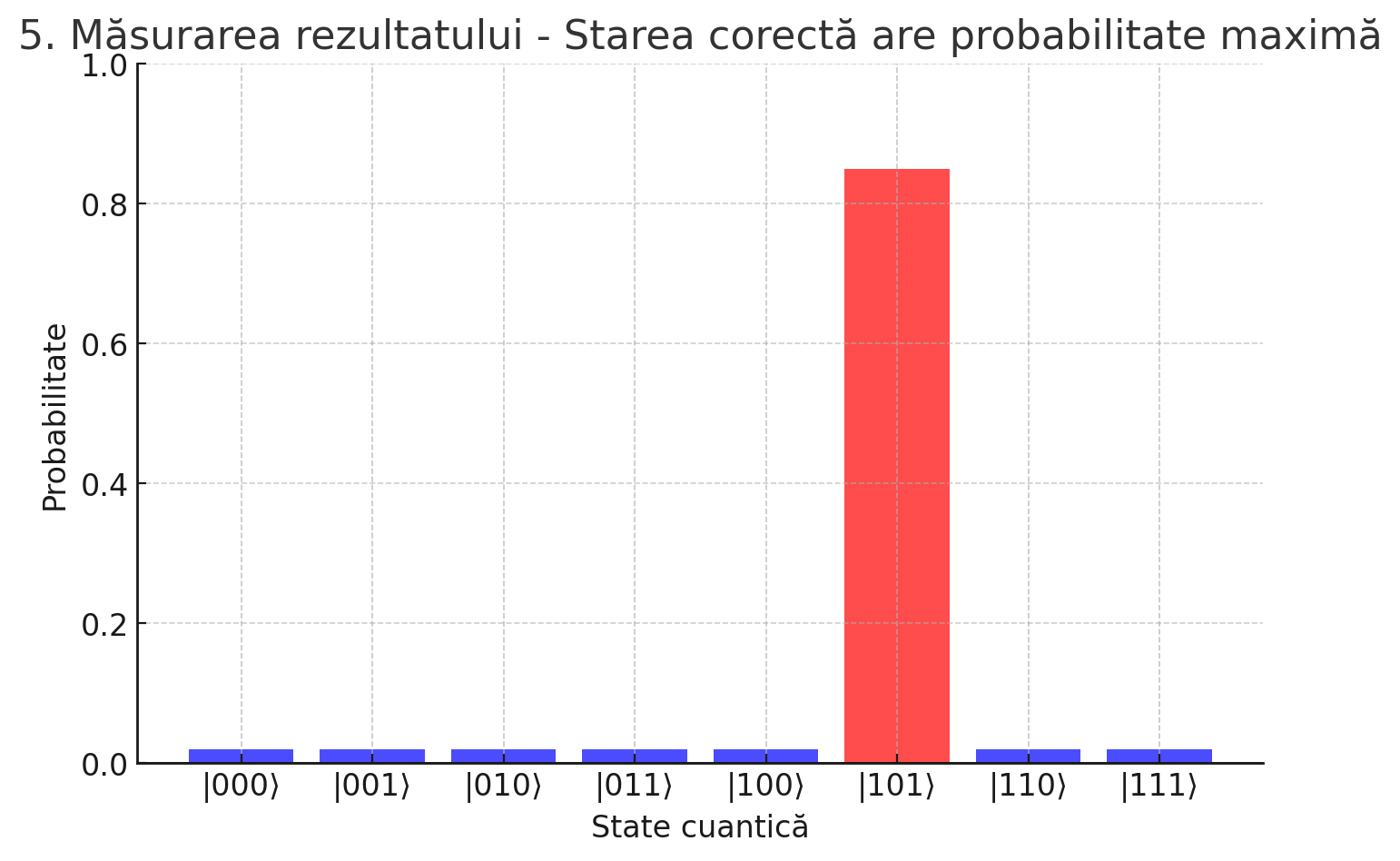
## 4. Creșterea progresivă a amplitudinii numărului căutat

* Pasul 2 (Oracle-ul) și pasul 3 (Amplificarea amplitudinilor) sunt repetate de un număr optim de ori, calculat ca O()
* De fiecare dată, amplitudinea numărului căutat crește progresiv, ceea ce înseamnă că probabilitatea de a-l măsura corect crește exponențial.



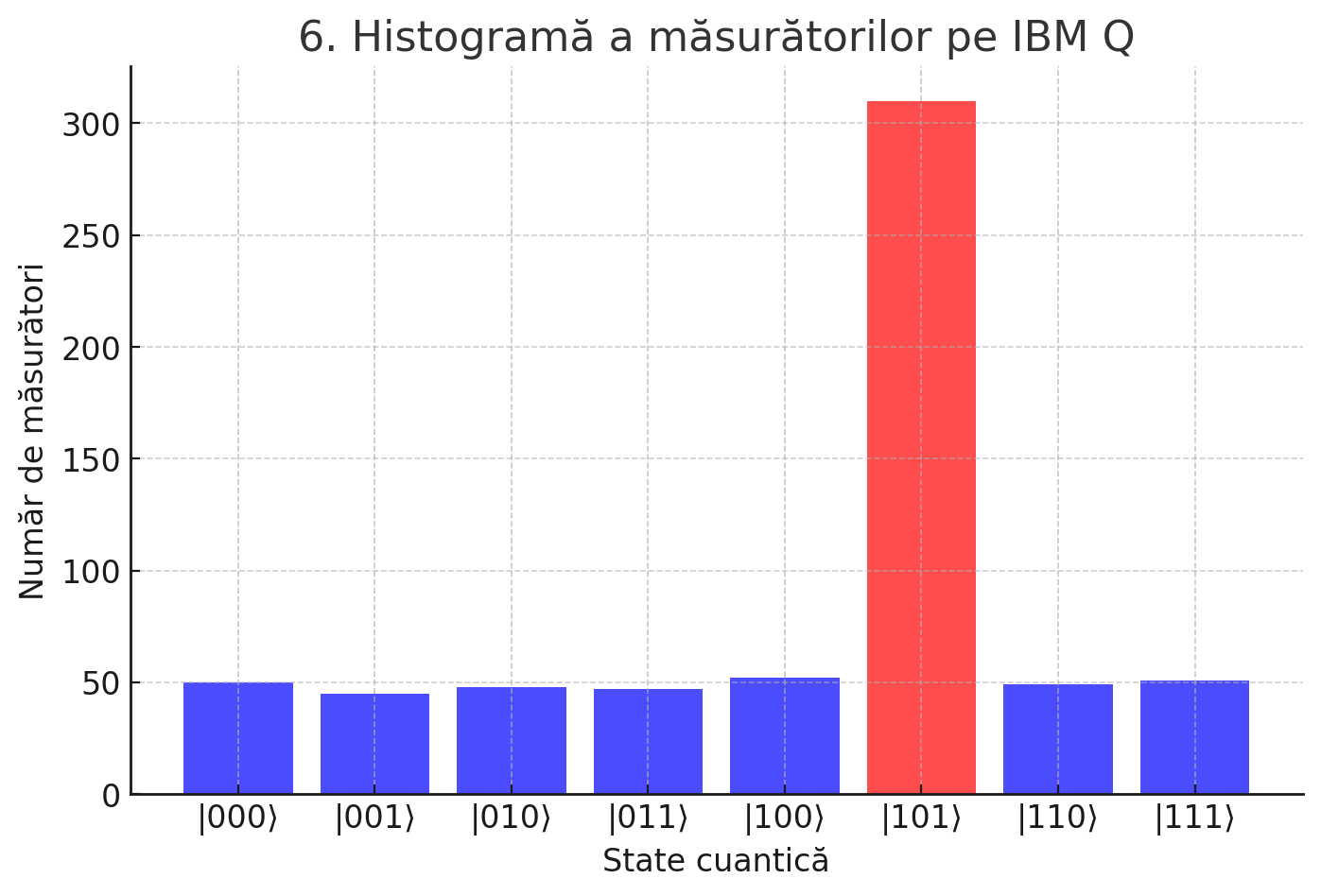
## 5. Probabilitățile finale după aplicarea Grover

* După suficiente iterații, amplitudinea numărului căutat este mult mai mare decât celelalte.
* Când măsurăm registrul cuantic, numărul căutat are cea mai mare probabilitate să fie observat.
* Practic, algoritmul Grover face ca numărul corect să fie selectat cu aproape 100% probabilitate.



## 6. Implementarea pe IBM Q – Histogramă a măsurătorilor

* Algoritmul este implementat pe un calculator cuantic real (IBM Quantum Computer).
* După rularea circuitului de mai multe ori (de exemplu, 1024 măsurători), se generează o histogramă a măsurătorilor.
* Într-un calculator cuantic real, există erori hardware, astfel că rezultatele pot să nu fie perfect precise, dar vor fi destul de apropiate de teorie.



# Implementarea Algoritmului Lui Grover pe IBM Quantum :

from qiskit import QuantumCircuit, transpile, assemble  
from qiskit.visualization import plot\_histogram  
from qiskit.quantum\_info import Statevector  
from qiskit\_ibm\_provider import IBMProvider  
import matplotlib.pyplot as plt  
import numpy as np  
  
  
# Oracle pentru starea |101⟩  
oracle = QuantumCircuit(3, name='oracle')  
oracle.x(1) # inversăm qubitul 1 (deoarece e 0 în ținta noastră)  
oracle.h(2)  
oracle.ccx(0, 1, 2)  
oracle.h(2)  
oracle.x(1) # readucem qubitul 1 la starea inițială  
  
oracle\_gate = oracle.to\_gate()  
  
print("Oracle (marchează starea |101⟩):")  
print(oracle.draw())  
  
# -------------------------------------------------  
# 2. Inițializare: superpoziție uniformă pe 3 qubiți  
# -------------------------------------------------  
init\_circ = QuantumCircuit(3)  
init\_circ.h([0, 1, 2])  
sv\_initial = Statevector.from\_instruction(init\_circ)  
  
print("\nStatevector inițial:")  
for i, amplitude in enumerate(sv\_initial.data):  
 print(f"Stare {format(i, '03b')}: {np.round(amplitude, 2)}")  
  
# -----------------------------------------------------  
# 3. Operatorul de difuzie (inversia în jurul mediei)  
# -----------------------------------------------------  
diffusion = QuantumCircuit(3, name='diffusion')  
diffusion.h([0, 1, 2])  
diffusion.x([0, 1, 2])  
diffusion.h(2)  
diffusion.ccx(0, 1, 2)  
diffusion.h(2)  
diffusion.x([0, 1, 2])  
diffusion.h([0, 1, 2])  
diffusion\_gate = diffusion.to\_gate()  
  
print("\nOperatorul de difuzie:")  
print(diffusion.draw())  
  
# --------------------------------------  
# 4. Circuitul complet Grover  
# --------------------------------------  
grover = QuantumCircuit(3)  
grover.h([0, 1, 2])  
grover.append(oracle\_gate, [0, 1, 2])  
grover.append(diffusion\_gate, [0, 1, 2])  
grover.measure\_all()  
  
print("\nCircuitul complet Grover:")  
print(grover.draw())  
  
# -----------------------------------------------------  
# 5. Simularea circuitului pe simulator (opțional)  
# -----------------------------------------------------  
from qiskit\_aer import Aer  
simulator = Aer.get\_backend('qasm\_simulator')  
job\_sim = simulator.run(transpile(grover, simulator), shots=1024)  
result\_sim = job\_sim.result()  
counts\_sim = result\_sim.get\_counts()  
  
print("\nRezultatele măsurătorii pe simulator:")  
print(counts\_sim)  
plot\_histogram(counts\_sim, title="Rezultatul Grover (Simulator)")  
plt.show()  
  
# -----------------------------------------------------  
# 6. Rularea circuitului pe hardware real IBM Q  
# -----------------------------------------------------  
# Asigură-te că tokenul tău este salvat: IBMProvider.save\_account(token='TOKENUL\_TAU')  
  
# Creează un provider pentru IBM Q  
provider = IBMProvider()  
  
# Selectează un backend real disponibil (ex: ibmq\_manila sau ibm\_oslo)  
backend = provider.get\_backend('ibmq\_manila')  
  
# Transpilează circuitul pentru backend-ul ales  
transpiled\_circuit = transpile(grover, backend=backend, optimization\_level=3)  
  
# Rulează circuitul pe hardware  
job\_hw = backend.run(transpiled\_circuit, shots=1024)  
print("\nJob-ul a fost trimis către hardware. Așteptăm rezultatele...")  
  
# Așteaptă rezultatul  
result\_hw = job\_hw.result()  
counts\_hw = result\_hw.get\_counts()  
  
print("\nRezultatele măsurătorii pe hardware IBM Q:")  
print(counts\_hw)  
plot\_histogram(counts\_hw, title="Rezultatul Grover (IBM Q Hardware)")  
plt.show()