

Uniwersytet Zielonogórski	Wykonali:	Grupa:	Nr ćwiczenia: 6	Ocena:
Laboratorium				
Temat: Wprowadzenie do algorytmu Grovera		Prowadzący:	Data wyk. ćw.	Data odd. spr.

Zadanie 1

Przeprowadzić symulację alg. Grovera dla zbioru $N=4$. Po ilu iteracjach prawdopodobieństwo znalezienia szukanego elementu jest największe?

Funkcja:

```
def zad1():
    q=qcs.QubitReg(2) ; q.Reset()
    q.Had()
    m1=q.GroverChangeSignGen(2);
    m2=q.GroverTurnAroundMean(2);
    q.Pr() ; print
    for i in range(1,10):
        print i
        q.MatrixApply(m1)
        q.MatrixApply(m2)
        q.Pr()
        print

    del m2 ; del m1 ; del q
```

WYNIK:

```
QubitReg(int i) new 2 qubits
0.500000 + 0.000000i |00>
0.500000 + 0.000000i |01>
0.500000 + 0.000000i |10>
0.500000 + 0.000000i |11>

1
0.000000 + 0.000000i |00>
0.000000 + 0.000000i |01>
-1.000001 + 0.000000i |10>
0.000000 + 0.000000i |11>

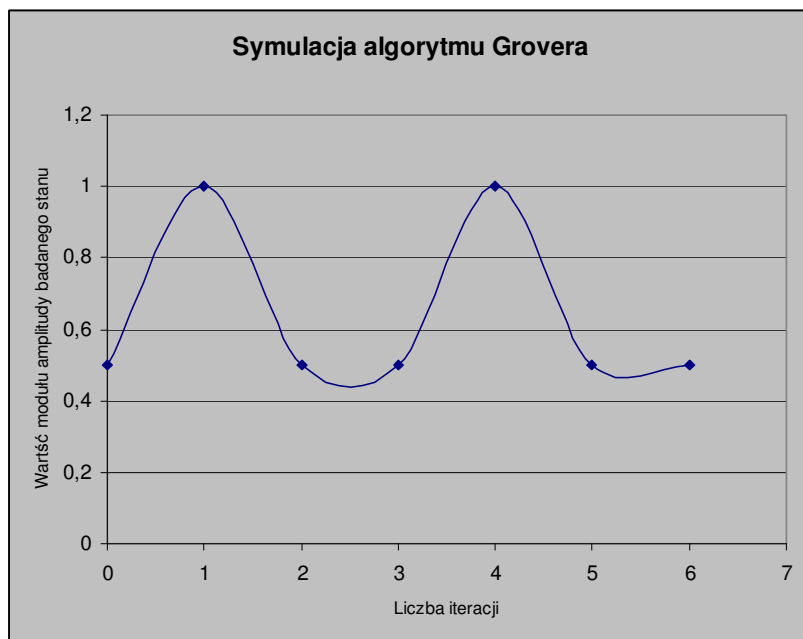
2
-0.500000 + 0.000000i |00>
-0.500000 + 0.000000i |01>
0.500000 + 0.000000i |10>
-0.500000 + 0.000000i |11>

3
0.500000 + 0.000000i |00>
0.500000 + 0.000000i |01>
0.500000 + 0.000000i |10>
0.500000 + 0.000000i |11>

4
-0.000000 + 0.000000i |00>
-0.000000 + 0.000000i |01>
-1.000001 + 0.000000i |10>
0.000000 + 0.000000i |11>

5
-0.500000 + 0.000000i |00>
-0.500000 + 0.000000i |01>
0.500000 + 0.000000i |10>
-0.500000 + 0.000000i |11>

6
0.500000 + 0.000000i |00>
0.500000 + 0.000000i |01>
0.500000 + 0.000000i |10>
0.500000 + 0.000000i |11>
```



Rys. 1 Wykres symulacji algorytmu Grovera

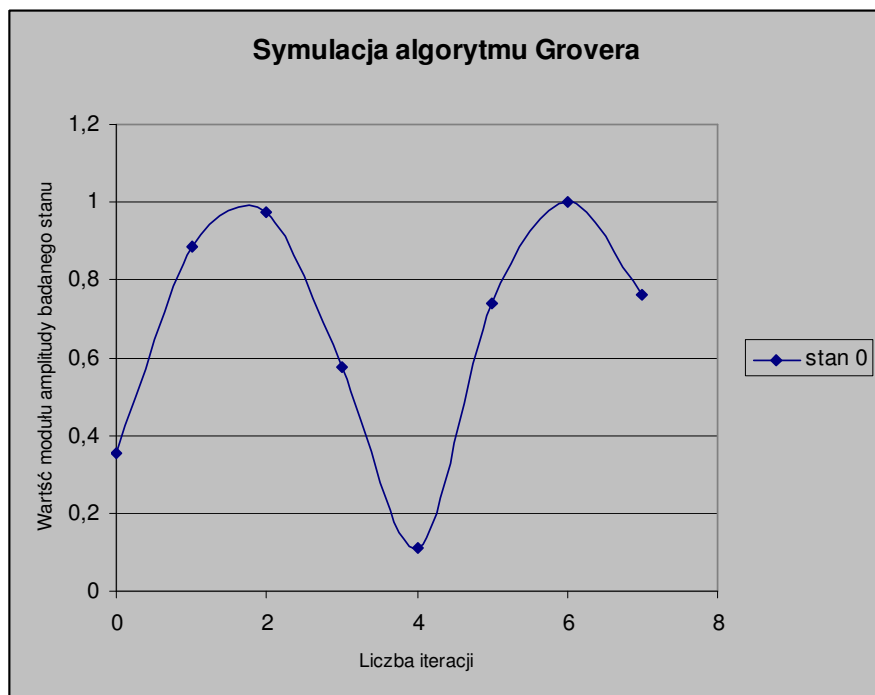
Zadanie 2

Przeprowadzić symulację alg. Grovera dla zbioru $N=8$, dla wszystkich ośmiu możliwości.
Po ilu iteracjach prawdopodobieństwo znalezienia szukanego elementu jest największe?

Funkcja:

```
def zad2():
    q=qcs.QubitReg(3) ; q.Reset()
    q.Had()
    m1=q.GroverChangeSignGen(0);
    m2=q.GroverTurnAroundMean(3);
    q.Pr() ; print
    for i in range(1,10):
        print i
        q.MatrixApply(m1)
        q.MatrixApply(m2)
        q.Pr()
        print

    del m2 ; del m1 ; del q
```



Rys. 2 Wykres symulacji algorytmu Grovera

WYNIK dla stanu 0:

```
0.353554 + 0.000000i |000>
0.353554 + 0.000000i |001>
0.353554 + 0.000000i |010>
0.353554 + 0.000000i |011>
0.353554 + 0.000000i |100>
0.353554 + 0.000000i |101>
0.353554 + 0.000000i |110>
0.353554 + 0.000000i |111>

1
-0.883884 + 0.000000i |000>
-0.176777 + 0.000000i |001>
-0.176777 + 0.000000i |010>
-0.176777 + 0.000000i |011>
-0.176777 + 0.000000i |100>
-0.176777 + 0.000000i |101>
-0.176777 + 0.000000i |110>
-0.176777 + 0.000000i |111>

2
0.972273 + 0.000000i |000>
-0.088388 + 0.000000i |001>
-0.088388 + 0.000000i |010>
-0.088388 + 0.000000i |011>
-0.088388 + 0.000000i |100>
-0.088388 + 0.000000i |101>
-0.088388 + 0.000000i |110>
-0.088388 + 0.000000i |111>

3
-0.574525 + 0.000000i |000>
0.309360 + 0.000000i |001>
0.309360 + 0.000000i |010>
0.309360 + 0.000000i |011>
0.309360 + 0.000000i |100>
0.309360 + 0.000000i |101>
0.309360 + 0.000000i |110>
0.309360 + 0.000000i |111>

4
-0.110486 + 0.000000i |000>
-0.375651 + 0.000000i |001>
-0.375651 + 0.000000i |010>
-0.375651 + 0.000000i |011>
-0.375651 + 0.000000i |100>
-0.375651 + 0.000000i |101>
-0.375651 + 0.000000i |110>
-0.375651 + 0.000000i |111>

5
0.740253 + 0.000000i |000>
0.254117 + 0.000000i |001>
0.254117 + 0.000000i |010>
0.254117 + 0.000000i |011>
0.254117 + 0.000000i |100>
0.254117 + 0.000000i |101>
0.254117 + 0.000000i |110>
0.254117 + 0.000000i |111>

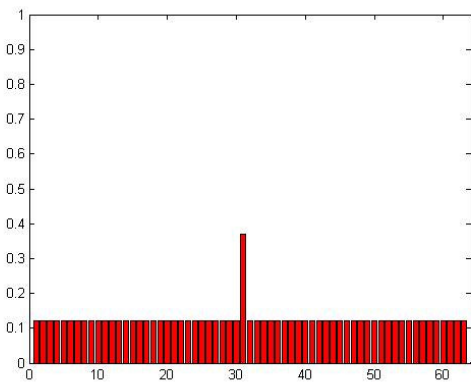
6
-0.999894 + 0.000000i |000>
-0.005524 + 0.000000i |001>
-0.005524 + 0.000000i |010>
-0.005524 + 0.000000i |011>
-0.005524 + 0.000000i |100>
-0.005524 + 0.000000i |101>
-0.005524 + 0.000000i |110>
-0.005524 + 0.000000i |111>
```

Zadanie 4

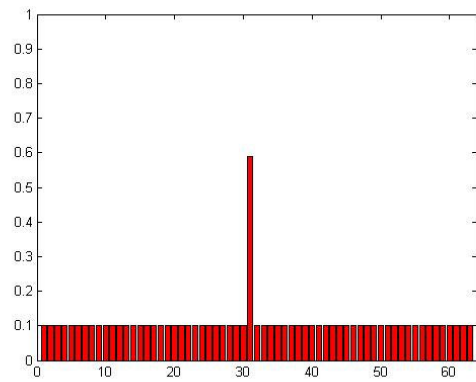
Dla zbioru $N=64$, utworzyć odpowiednio przeskalowany wykresu rozkładu amplitud dla poszczególnych iteracji

Funkcja:

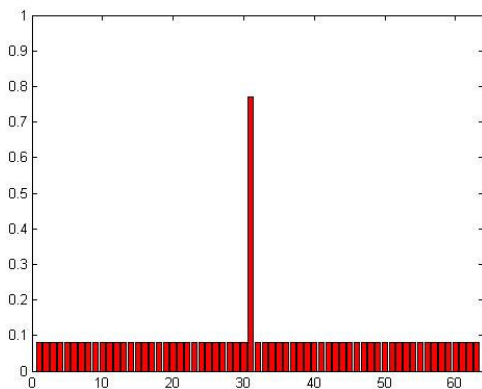
```
def zad4():  
    q=qcs.QubitReg(6) ; q.Reset()  
    q.Had()  
    m1=q.GroverChangeSignGen(30);  
    m2=q.GroverTurnAroundMean(6);  
    M=qcs.Matrix(9,64);  
    q.Pr(); print  
    for i in range(1,9):  
        print i  
        q.MatrixApply(m1)  
        q.MatrixApply(m2)  
        print  
        for j in range(0,63):  
            a=q.GetVecStateN(j)  
            M.AtDirect(i,j,a.re,0)  
  
del m2 ; del m1 ; del q  
  
M.PrMatlab()
```



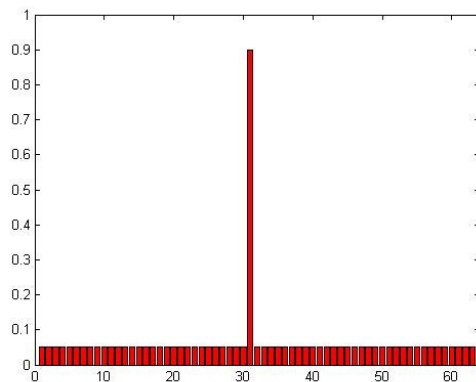
Rys. 3 Wykres rozkładu amplitud dla 1 iteracji



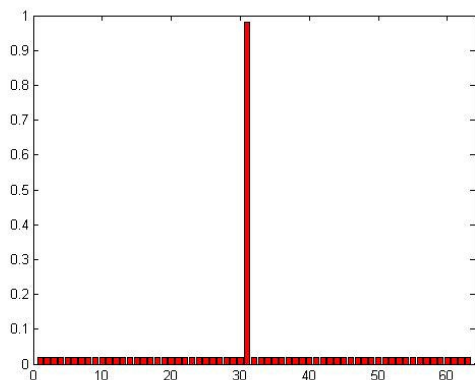
Rys. 4 Wykres rozkładu amplitud dla 2 iteracji



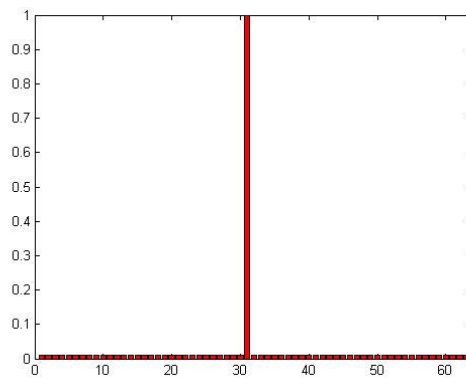
Rys. 5 Wykres rozkładu amplitud dla 3 iteracji



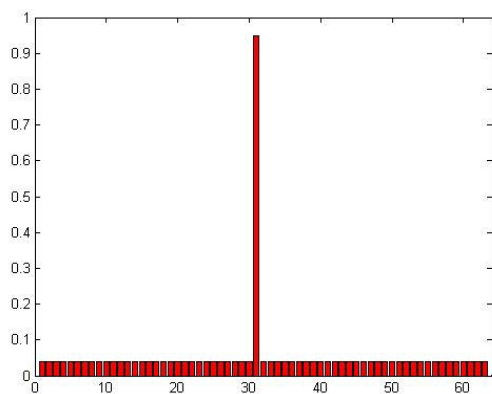
Rys. 6 Wykres rozkładu amplitud dla 4 iteracji



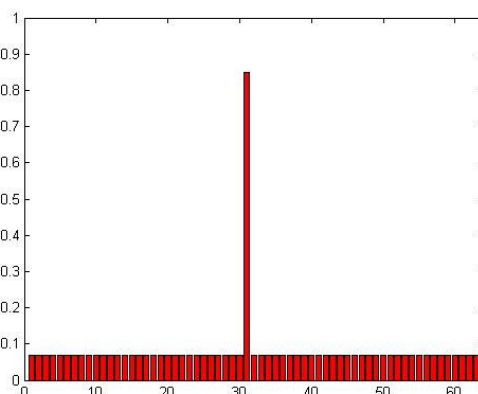
Rys. 7 Wykres rozkładu amplitud dla 5 iteracji



Rys.8 Wykres rozkładu amplitud dla 6 iteracji



Rys.9 Wykres rozkładu amplitud dla 7 iteracji



Rys.10 Wykres rozkładu amplitud dla 8 iteracji

Zadanie 5

Dysponując zbiorem $N=16$ sprawdzić w jaki sposób poszczególne iteracje alg. Grovera w sensie miary Fidelity przybliżają aktualny stan do stanu w którym operacja pomiaru z największym prawdopodobieństwem wskaże poprawny wynik. Pomiary wykonać dla znaczącej ilości iteracji np.: 32 bądź 64 iteracjach alg. Grovera.

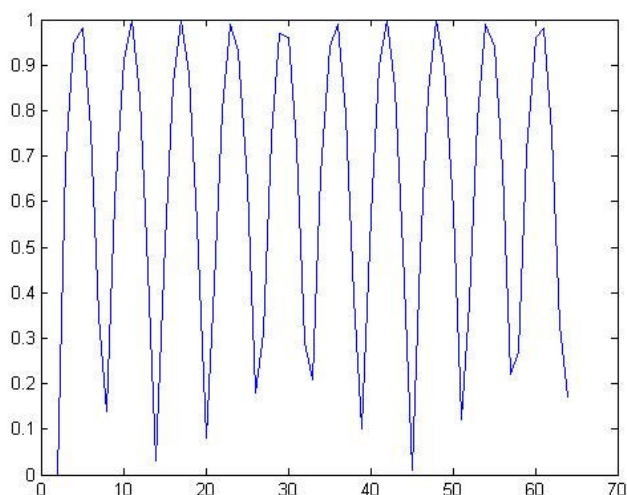
Funkcja:

```
def zad5():
    q=qcs.QubitReg(4)
    q.Reset()
    q.Had()
    m1=q.GroverChangeSignGen(2);
    m2=q.GroverTurnAroundMean(4);
    q.Pr(); print

    r=qcs.QubitReg(4)
    r.Reset()
    r.SetN(2)
    dwzorcowe=r.GenDenMat()

    Macierz=qcs.Matrix(64,1);
    for i in range(1,64):
        q.MatrixApply(m1)
        q.MatrixApply(m2)

        d=q.GenDenMat()
        fi=qcs.Fidelity(d,dwzorcowe)
        Macierz.AtDirect(i,1,fi.re,0);
    Macierz.PrMatlab()
    del m2; del m1; del q
```



Rys. 11 Wykres miary Fidelity dla poszczególnych iteracji

Wnioski:

W ćwiczeniu tym zajęliśmy się badaniem algorytmu Grovera.

W zadaniu pierwszym przeprowadzaliśmy symulacje algorytmu Grovera dla zbioru $N=4$. Zgodnie ze wzorem $N = 2^n$, gdzie n oznacza rejestr kwantowy o n bitach, wykorzystaliśmy rejestr kwantowy o 2 bitach. Algorytm ma charakter iteracyjny. Kolejne kroki modyfikują stan układu, zwiększając prawdopodobieństwo związane z badanym stanem (w naszym przypadku był to stan $|10\rangle$), zmniejszając natomiast co do modułu amplitudy pozostałych stanów. Przed rozpoczęciem pracy algorytmu ustawiliśmy nasz rejestr w stan superpozycji wszystkich stanów z równymi amplitudami zgodnie ze wzorem:

$|\phi_0\rangle = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} |k\rangle$ Do tego celu wykorzystaliśmy bramkę

Hadamarda. Na rysunku 1 możemy zobaczyć, że już po pierwszej iteracji prawdopodobieństwo znalezienia szukanego elementu jest największe, ponieważ moduł amplitudy stanu przyjmuje wartość 1. Na wykresie widać także, że prawdopodobieństwo znalezienia szukanego elementu jest największe także po 4 iteracji, co oznacza, że algorytm ma charakter cykliczny.

W zadaniu drugim przeprowadzaliśmy symulacje algorytmu Grovera dla zbioru $N=8$, dla wszystkich ośmiu możliwości. Po przeprowadzeniu badań okazało się, że wartości modułów amplitud dla wszystkich stanów były identyczne. Prawdopodobieństwo znalezienia szukanego elementu jest największe po 6 iteracji.

W zadaniu czwartym badaliśmy rozkładu amplitud dla poszczególnych iteracji. Badaliśmy rejestr 6 bitowy ($2^6 = 64$). Maksymalna wartość amplitudy otrzymaliśmy po 6 iteracji. Badaliśmy stan 30, co widać na otrzymanym wykresie.

W zadaniu szóstym zajęliśmy się badaniem metryki Fidelity. Metryką nazywamy funkcję $d(\cdot, \cdot) : \mathcal{C} \times \mathcal{C} \rightarrow \mathbb{R}^+ \cup \{0\}$ spełniającą dla dowolnych elementów $\psi, \phi, \varphi \in \mathcal{C}$ trzy warunki

1. $d(\psi, \phi) = 0 \Leftrightarrow \psi = \phi$
2. $d(\psi, \phi) = 0 \Leftrightarrow d(\phi, \psi)$
3. $d(\psi, \phi) = 0 \leq d(\psi, \varphi) + d(\varphi, \phi)$

Norma $\|\cdot\|$ indukuje w przestrzeni metrykę $d(\psi, \phi) = \|\psi - \phi\| = \sqrt{\langle \psi - \phi | \psi - \phi \rangle}$

Miara Fidelity jest użyteczną miarą odległości pomiędzy stanami. Wykorzystuje się do badania wpływu błędów kwantowych na stan. Definiowana jest w następujący sposób:

$F(\rho, \sigma) = \text{tr} \sqrt{\rho^{1/2} \sigma \rho^{1/2}}$, gdzie ρ i σ są macierzami gęstości.