

Algorytm Grovera

Kwantowe przeszukiwanie zbioru

Robert Nowotniak

Wydział Fizyki Technicznej, Informatyki i Matematyki Stosowanej
Politechnika Łódzka






Sekcja Informatyki Kwantowej, 24 października 2007

Plan prezentacji

- 1 Ogólny opis algorytmu Grovera
- 2 Intuicyjna interpretacja
- 3 Uzasadnienie matematyczne
- 4 Symulacja (w Numerical Python)

- 1 **Ogólny opis algorytmu Grovera**
- 2 Intuicyjna interpretacja
- 3 Uzasadnienie matematyczne
- 4 Symulacja (w Numerical Python)

Prezentacja oparta na publikacjach:

-  Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000
-  Krzysztof Giaro, Marcin Kamiński. *Wprowadzenie do algorytmów kwantowych*. Warszawa, 2003
-  Stefan Węgrzyn, Jerzy Klamka, Jarosław Adam Mischczak. *Kwantowe systemy informatyki*. 2002
-  Mika Hirvensalo. *Algorytmy Kwantowe*. WSiP, Warszawa, 2004
-  Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996

Prezentacja oparta na publikacjach:



Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000



Krzysztof Giaro, Marcin Kamiński. *Wprowadzenie do algorytmów kwantowych*. Warszawa, 2003



Stefan Węgrzyn, Jerzy Klamka, Jarosław Adam Miszczak. *Kwantowe systemy informatyki*. 2002



Mika Hirvensalo. *Algorytmy Kwantowe*. WSiP, Warszawa, 2004



Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996

Algorytm Grovera

- 1 Wyszukiwanie elementu w zbiorze
np. przeszukiwanie **nieposortowanej** bazy danych
- 2 Złożoność obliczeniowa: $O(\sqrt{N})$
- 3 Klasa złożoności **BQP** (ang. *bounded-error, quantum, polynomial*)
- 4 W rejestrze kwantowym przetwarzana jest **superpozycja** potencjalnych rozwiązań
- 5 Wykorzystuje iteracyjne „**wzmacnianie amplitudy** prawdopodobieństwa” (ang. *amplitude amplification*) poszukiwanego rozwiązania

Założenia algorytmu

- Dany jest N -elementowy zbiór Q oraz funkcja $f : Q \rightarrow \{0, 1\}$
- Dokładnie jeden element q zbioru Q posiada pewną poszukiwaną cechę,

$$\exists!_{q \in Q} f(q) = 1$$

- Celem algorytmu jest znalezienie tego elementu q , dla którego $f(q) = 1$

Algorytm Grovera

- 1 Przygotowanie stanu początkowego w superpozycji wszystkich stanów bazowych:

$$|\phi_0\rangle = \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} |\omega\rangle$$

Algorytm Grovera

- 1 Przygotowanie stanu początkowego w superpozycji wszystkich stanów bazowych:

$$|\phi_0\rangle = \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} |\omega\rangle$$

- 2 Wykonywanie (odpowiednią ilość razy) operacji na rejestrze:

$$|\phi_{n+1}\rangle = \mathcal{BA}|\phi_n\rangle$$

Algorytm Grovera

- 1 Przygotowanie stanu początkowego w superpozycji wszystkich stanów bazowych:

$$|\phi_0\rangle = \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} |\omega\rangle$$

- 2 Wykonywanie (odpowiednią ilość razy) operacji na rejestrze:

$$|\phi_{n+1}\rangle = \mathcal{B}\mathcal{A}|\phi_n\rangle$$

gdzie:

$$\mathcal{A} = \mathbb{I} - 2|\omega_0\rangle\langle\omega_0|$$

$$\mathcal{B} = 2|\phi_0\rangle\langle\phi_0| - \mathbb{I}$$

Algorytm Grovera

- 1 Przygotowanie stanu początkowego w superpozycji wszystkich stanów bazowych:

$$|\phi_0\rangle = \frac{1}{\sqrt{N}} \sum_{\omega=0}^{N-1} |\omega\rangle$$

- 2 Wykonywanie (**odpowiednią ilość razy**) operacji na rejestrze:

$$|\phi_{n+1}\rangle = \mathcal{B}\mathcal{A}|\phi_n\rangle$$

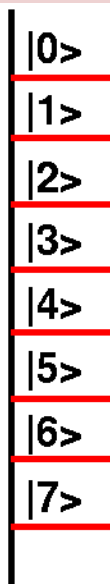
gdzie:

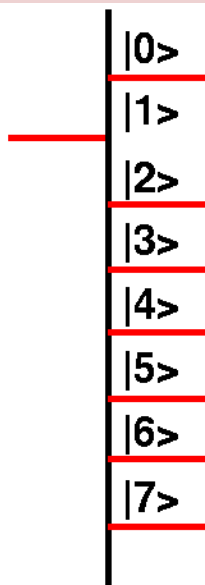
$$\mathcal{A} = \mathbb{I} - 2|\omega_0\rangle\langle\omega_0|$$

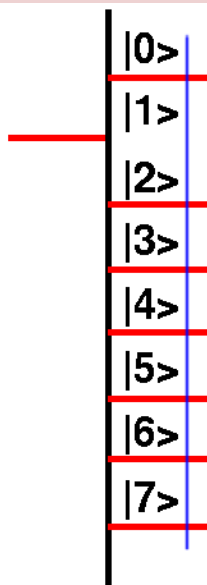
$$\mathcal{B} = 2|\phi_0\rangle\langle\phi_0| - \mathbb{I}$$

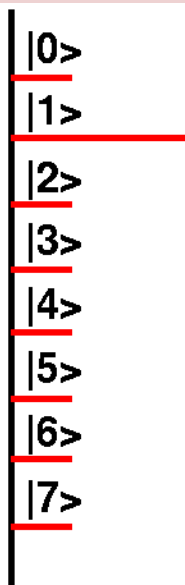
Plan prezentacji

- 1 Ogólny opis algorytmu Grovera
- 2 **Intuicyjna interpretacja**
- 3 Uzasadnienie matematyczne
- 4 Symulacja (w Numerical Python)

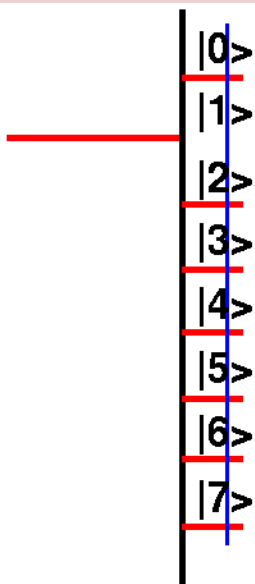


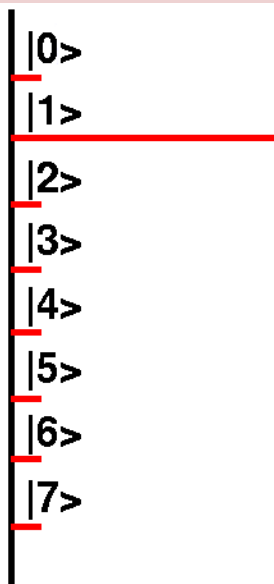












Prawdopodobieństwo sukcesu

Po k iteracjach prawdopodobieństwo odczytu stanu związanego z szukanym elementem wyraża się wzorem:

$$|\alpha_k|^2 = \sin^2 \left((2k + 1) \arcsin \left(\sqrt{\frac{1}{N}} \right) \right)$$

Prawdopodobieństwo sukcesu

Po k iteracjach prawdopodobieństwo odczytu stanu związanego z szukany elementem wyraża się wzorem:

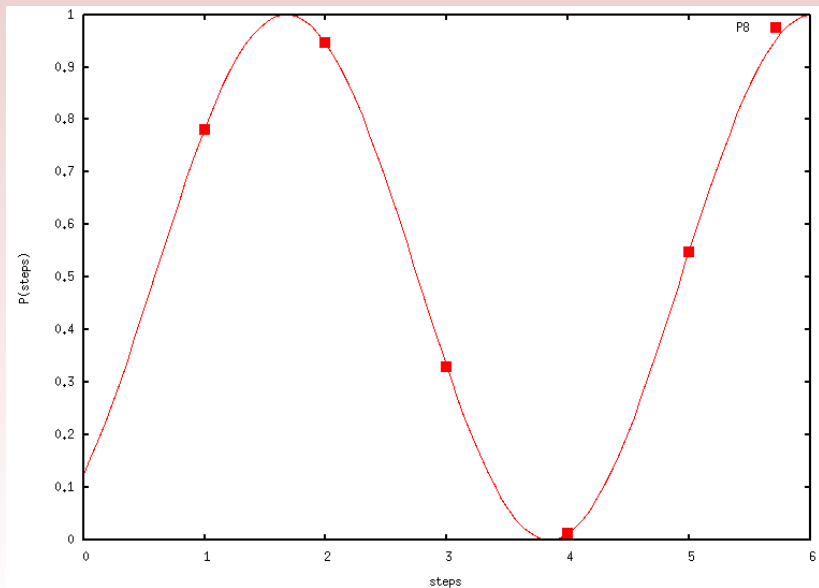
$$|\alpha_k|^2 = \sin^2 \left((2k + 1) \arcsin \left(\sqrt{\frac{1}{N}} \right) \right)$$

Optymalne prawdopodobieństwo odczytania pożądanego stanu $|\omega_0\rangle$ jest dla dla ilości kroków \bar{k} równej:

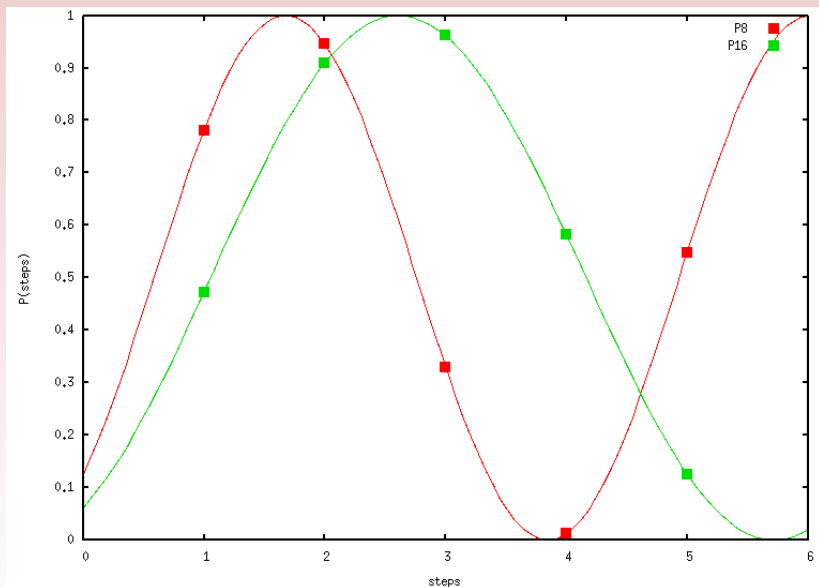
$$\bar{k} = \left\lceil \frac{\pi}{4} \left(\arcsin \left(\sqrt{\frac{1}{N}} \right) \right)^{-1} \right\rceil$$

Prawdopodobieństwo to jest większe od $1 - \frac{1}{N}$

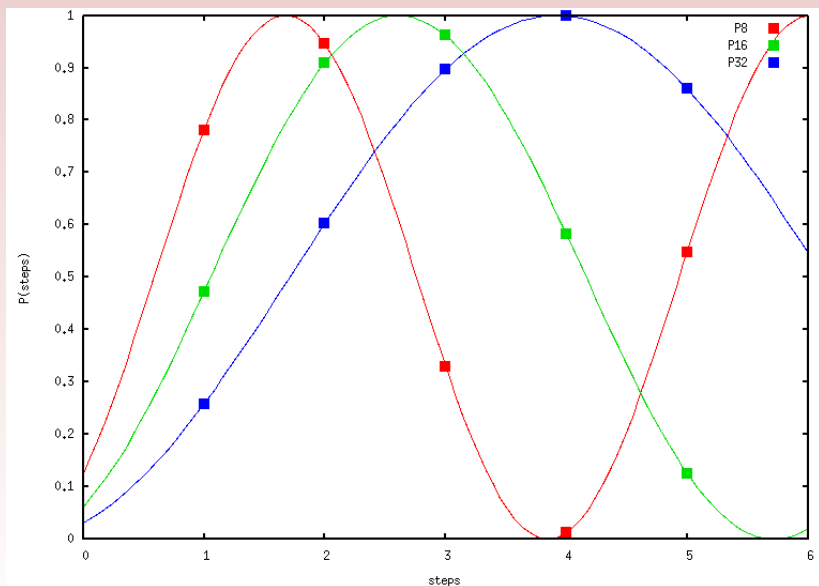
Prawdopodobieństwo sukcesu



Prawdopodobieństwo sukcesu



Prawdopodobieństwo sukcesu



- 1 Ogólny opis algorytmu Grovera
- 2 Intuicyjna interpretacja
- 3 **Uzasadnienie matematyczne**
- 4 Symulacja (w Numerical Python)

Jak działa operator \mathcal{A} na dowolny stan **bazowy** ω ?

Jak działa operator \mathcal{A} na dowolny stan **bazowy** ω ?

Działanie operatora \mathcal{A}

$$\mathcal{A}|\omega\rangle = (\mathbb{I} - 2|\omega_0\rangle\langle\omega_0|) \cdot |\omega\rangle$$

Jak działa operator \mathcal{A} na dowolny stan **bazowy** ω ?

Działanie operatora \mathcal{A}

$$\begin{aligned}\mathcal{A}|\omega\rangle &= (\mathbb{I} - 2|\omega_0\rangle\langle\omega_0|) \cdot |\omega\rangle \\ &= |\omega\rangle - 2|\omega_0\rangle \cdot \begin{cases} 1 & \text{gdy } \omega = \omega_0 \\ 0 & \text{gdy } \omega \neq \omega_0 \end{cases}\end{aligned}$$

Jak działa operator \mathcal{A} na dowolny stan **bazowy** ω ?

Działanie operatora \mathcal{A}

$$\begin{aligned}\mathcal{A}|\omega\rangle &= (\mathbb{I} - 2|\omega_0\rangle\langle\omega_0|) \cdot |\omega\rangle \\ &= |\omega\rangle - 2|\omega_0\rangle \cdot \begin{cases} 1 & \text{gdy } \omega = \omega_0 \\ 0 & \text{gdy } \omega \neq \omega_0 \end{cases} \\ &= \begin{cases} -|\omega\rangle & \text{gdy } \omega = \omega_0 \\ |\omega\rangle & \text{gdy } \omega \neq \omega_0 \end{cases} = (-1)^{f(\omega)}|\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na pewien stan **bazowy** ω ?

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na pewien stan **bazowy** ω ?

Działanie operatora \mathcal{B}

$$\mathcal{B}|\omega\rangle = (2|\phi_0\rangle\langle\phi_0| - \mathbb{I})|\omega\rangle$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na pewien stan **bazowy** $|\omega\rangle$?

Działanie operatora \mathcal{B}

$$\begin{aligned}\mathcal{B}|\omega\rangle &= (2|\phi_0\rangle\langle\phi_0| - \mathbb{I})|\omega\rangle \\ &= 2|\phi_0\rangle \underbrace{\langle\phi_0|\omega\rangle}_{=\frac{1}{\sqrt{N}}} - |\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na pewien stan **bazowy** $|\omega\rangle$?

Działanie operatora \mathcal{B}

$$\begin{aligned}\mathcal{B}|\omega\rangle &= (2|\phi_0\rangle\langle\phi_0| - \mathbb{I})|\omega\rangle \\ &= 2|\phi_0\rangle \underbrace{\langle\phi_0|\omega\rangle}_{=\frac{1}{\sqrt{N}}} - |\omega\rangle \\ &= \frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na pewien stan **bazowy** $|\omega\rangle$?

Działanie operatora \mathcal{B}

$$\begin{aligned}\mathcal{B}|\omega\rangle &= (2|\phi_0\rangle\langle\phi_0| - \mathbb{I})|\omega\rangle \\ &= 2|\phi_0\rangle \underbrace{\langle\phi_0|\omega\rangle}_{=\frac{1}{\sqrt{N}}} - |\omega\rangle \\ &= \frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\end{aligned}$$

Niech **dowolny stan** $|\phi\rangle$ będzie opisany w bazie standardowej przez współczynniki $\alpha_0, \dots, \alpha_{N-1}$:

$$|\phi\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega} |\omega\rangle$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na **dowolny** stan $|\phi\rangle$?

$$\begin{aligned}\mathcal{B}|\phi\rangle &= \mathcal{B}\left(\sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle\right) = \sum_{\omega=0}^{N-1} \alpha_{\omega}\mathcal{B}|\omega\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\right) \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}|\phi_0\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} |k\rangle\right) - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= 2 \cdot \underbrace{\frac{1}{N}\sum_{\omega=0}^{N-1} \alpha_{\omega}}_{\alpha} \sum_{k=0}^{N-1} |k\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \sum_{\omega=0}^{N-1} (2\alpha - \alpha_{\omega})|\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})]|\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na **dowolny** stan $|\phi\rangle$?

$$\begin{aligned}\mathcal{B}|\phi\rangle &= \mathcal{B}\left(\sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle\right) = \sum_{\omega=0}^{N-1} \alpha_{\omega}\mathcal{B}|\omega\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\right) \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}|\phi_0\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} |k\rangle\right) - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= 2 \cdot \underbrace{\frac{1}{N}\sum_{\omega=0}^{N-1} \alpha_{\omega}}_{\alpha} \sum_{k=0}^{N-1} |k\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \sum_{\omega=0}^{N-1} (2\alpha - \alpha_{\omega})|\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})]|\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na **dowolny** stan $|\phi\rangle$?

$$\begin{aligned}\mathcal{B}|\phi\rangle &= \mathcal{B}\left(\sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle\right) = \sum_{\omega=0}^{N-1} \alpha_{\omega}\mathcal{B}|\omega\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\right) \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}|\phi_0\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} |k\rangle\right) - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= 2 \cdot \underbrace{\frac{1}{N}\sum_{\omega=0}^{N-1} \alpha_{\omega}}_{\alpha} \sum_{k=0}^{N-1} |k\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \sum_{\omega=0}^{N-1} (2\alpha - \alpha_{\omega})|\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})]|\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na **dowolny** stan $|\phi\rangle$?

$$\begin{aligned}\mathcal{B}|\phi\rangle &= \mathcal{B}\left(\sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle\right) = \sum_{\omega=0}^{N-1} \alpha_{\omega}\mathcal{B}|\omega\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\right) \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}|\phi_0\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} |k\rangle\right) - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= 2 \cdot \underbrace{\frac{1}{N}\sum_{\omega=0}^{N-1} \alpha_{\omega}}_{\alpha} \sum_{k=0}^{N-1} |k\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \sum_{\omega=0}^{N-1} (2\alpha - \alpha_{\omega})|\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})]|\omega\rangle\end{aligned}$$

Uzasadnienie algorytmu

Jak działa operator \mathcal{B} na **dowolny** stan $|\phi\rangle$?

$$\begin{aligned}\mathcal{B}|\phi\rangle &= \mathcal{B}\left(\sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle\right) = \sum_{\omega=0}^{N-1} \alpha_{\omega}\mathcal{B}|\omega\rangle = \sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{2}{\sqrt{N}}|\phi_0\rangle - |\omega\rangle\right) \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}|\phi_0\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \frac{2}{\sqrt{N}}\sum_{\omega=0}^{N-1} \alpha_{\omega}\left(\frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} |k\rangle\right) - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= 2 \cdot \underbrace{\frac{1}{N}\sum_{\omega=0}^{N-1} \alpha_{\omega}}_{\alpha} \sum_{k=0}^{N-1} |k\rangle - \sum_{\omega=0}^{N-1} \alpha_{\omega}|\omega\rangle \\ &= \sum_{\omega=0}^{N-1} (2\alpha - \alpha_{\omega})|\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})]|\omega\rangle\end{aligned}$$

Działanie operatora \mathcal{B}

Ostatecznie mamy:

$$\mathcal{B}|\phi\rangle = \mathcal{B} \sum_{\omega=0}^{N-1} \alpha_{\omega} |\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})] |\omega\rangle$$

gdzie:

α - średnia arytmetyczna wszystkich amplitud

Działanie operatora \mathcal{B}

Ostatecznie mamy:

$$\mathcal{B}|\phi\rangle = \mathcal{B} \sum_{\omega=0}^{N-1} \alpha_{\omega} |\omega\rangle = \sum_{\omega=0}^{N-1} [\alpha + (\alpha - \alpha_{\omega})] |\omega\rangle$$

gdzie:

α - średnia arytmetyczna wszystkich amplitud

Dokonuje się obrót amplitudy każdego stanu wokół średniej wartości amplitud

Uzasadnienie algorytmu

- Operator \mathcal{A} – zmiana znaku amplitudy prawdopodobieństwa, związanej z poszukiwanym stanem $|\omega_0\rangle$
- Operator \mathcal{B} – obrót wszystkich amplitud prawdopodobieństwa wokół wartości średniej

Uzasadnienie algorytmu

- Operator \mathcal{A} – zmiana znaku amplitudy prawdopodobieństwa, związanej z poszukiwanym stanem $|\omega_0\rangle$
- Operator \mathcal{B} – obrót wszystkich amplitud prawdopodobieństwa wokół wartości średniej
- **Rezultat:** Zwiększanie amplitudy prawdopodobieństwa stanu $|\omega_0\rangle$ w kolejnych iteracjach

Plan prezentacji

- 1 Ogólny opis algorytmu Grovera
- 2 Intuicyjna interpretacja
- 3 Uzasadnienie matematyczne
- 4 Symulacja (w Numerical Python)

Implementacja algorytmu

```
# size of the search set
N = 2 ** 9

# index of the searched element (arbitrary value between 0 .. N - 1)
n = N - 3 # f(x) = 1 <=> x==n

# total number of steps (optimal success probability)
steps = int(floor(math.pi * 1.0/(math.asin(sqrt(1.0 / N))) / 4))

print 'Total numer of steps: ' + str(steps)
print

# identity matrix
I = identity(N)

w0 = transpose(array([zeros(N)]))
w0[n,0] = 1

# initial state of quantum register
phi0 = transpose([ones(N) / sqrt(N)])
```

Implementacja algorytmu

```
# computation matrices
A = I - 2 * outer(w0,w0)
B = 2 * outer(phi0, phi0) - I

phi = phi0
step = 0

while step <= steps:
    print 'Step number: ' + str(step)
    print '-----'
    print 'Probability of search success: ',
    print '%0.4f' % abs(float(phi[n]))**2
    print
    if step < steps:
        phi = dot(B, dot(A, phi))
    step += 1
```

Implementacja algorytmu

```
print 'Final quantum register |phi> state: '  
print phi  
print  
  
print 'Probability of search success: ',  
print '%0.4f' % abs(float(phi[n]))**2  
  
success = (random.random() < float(phi[n])**2)  
  
print 'MEASUREMENT! -> ',  
if success:  
    print 'success.'  
else:  
    print 'failure.'
```

Dziękuję.
Pytania lub wątpliwości?