Wyklad6

24 maja 2011
19:14

Obliczenia kwantowe

1. Wstęp

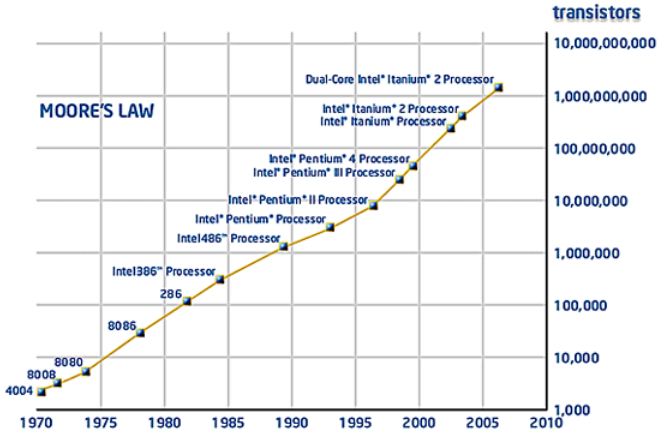
Obecne komputery też używają praw fizyki kwantowej:
stabilizatora półprzewodników, tranzystory, momenty magnetyczne atomów (spiny)

Ale ....

- 1 bit danych na dysku twardym: ($d \approx 0{,}5mm$)
  $250mm \times 250mm \times 25mm \approx 12{,}5$ mln atomów

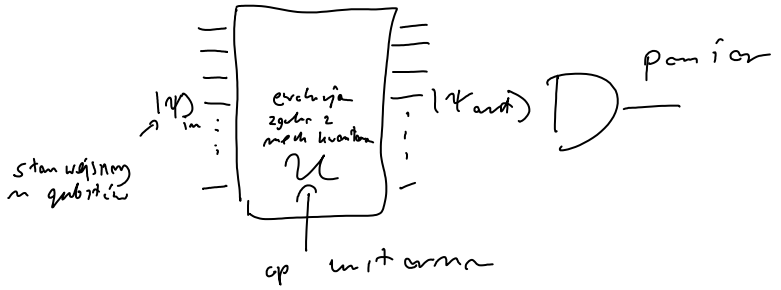
- 1 tranzystor w CPU
  $50nm \times 50nm \times 25nm \approx 500$ tys atomów

Wciąż dość dużo. Nie jesteśmy na etapie żeby używać pojedynczych atomów do obliczeń i wykorzystać pełne możliwości fizyki kwantowej
Kiedy zejdziemy do poziomu 1 atomu.



Rozmiar tranzystorów zmniejsza się dwa razy co 2 lata
Będzie rozmiar atomu ok roku 2030-2050?
Nawet jak to nastąpi nie oznacza to, że mamy już komputer kwantowy.
Musimy umieć utrzymać kwantowe superpozycje tak aby móc wykonywać potrzebne mech. kwantowej

2. Idea



- Klasyczne komputery budujemy zazwyczaj z pewnych elementarnych bramek: NOT, AND, OR, XOR ... Widzimy, że np. klasyczny układ logiczny można zbudować z bramki NAND .

- W klasycznych komputerach zazwyczaj używamy bramek nieodwracalnych (chociaż można równie dobrze inaczej) a, b → a XOR b — nie da się odtworzyć wejścia z bitu wyjściowego

- Moglibyśmy używać w klasycznych obliczeniach bramek odwracalnych, wystarczy np.

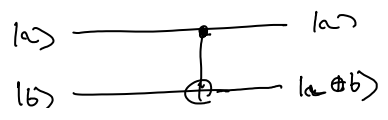
Kosztem komplikacji obwodów zazwyczaj się tego nie robi. Pamiętajmy jednak, że fizyka w zasadzie jest odwracalna. Nieodwracalność bierze się z tego że po prostu ignorujemy jakieś stopnie swobody

- Myśląc o komputerach kwantowych mamy operacje unitarne - które są odwracalne. Możemy oczywiście też zrobić nieodwracalne np. śledząc po pomiarze z wyjściowych qubitów, ale te zazwyczaj niszczą kwantowe superpozycje, więc nadają się tylko do operacji unitarnych.

Chcemy znaleźć elementarne bramki, z których można złożyć dowolny op. U. Bramki te muszą być odwracalne.

- Bramka CNOT

$|0\rangle|0\rangle \to |0\rangle|0\rangle$
$|0\rangle|1\rangle \to |0\rangle|1\rangle$



$|0\rangle|0\rangle \to |0\rangle|0\rangle$

$|0\rangle|1\rangle \to |0\rangle|1\rangle$

$|1\rangle|0\rangle \to |1\rangle|1\rangle$

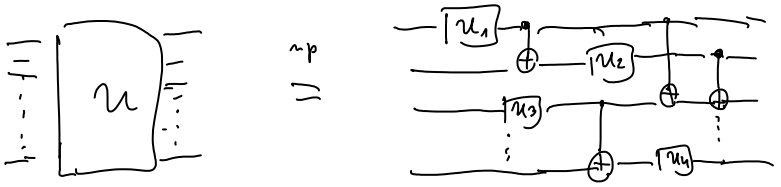
$|1\rangle|1\rangle \to |1\rangle|0\rangle$

$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

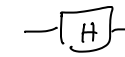
w bazie $|00\rangle, |01\rangle, |10\rangle, |11\rangle$.

Myśląc klasycznie to jest taki odwracalny XOR

<u>Fakt</u> Każda wielokubitowa U może być przedstawiona za pomocą jednokubitowych op. unitarnych i bramki CNOT



W ogólności potrzebujemy wielu bramek jednokubitowych (a więc obwodów spiny Blocha) można np. wybrać:



bramka Hadamarda $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ $\left\{ H^\dagger H = \mathbb{1} \right.$

operacja fazowa φ (bramka fazowa) $U_\varphi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\varphi} \end{bmatrix}$ ($\varphi = \frac{\pi}{2}$ ćwiartka okręgu)

3. Kwantowy Paralelizm - dlaczego komputer kwantowy ma szanse liczyć szybciej, ?

Idea: $f: \{0,1\} \to \{0,1\}$ jedno bitowa funkcja $f(0), f(1)$

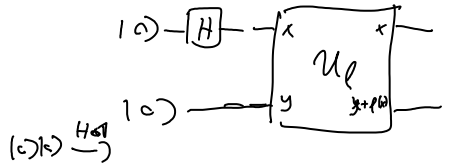
<u>Wyobraźmy sobie</u> że kodujemy funkcję f w bramce kwantowej U_f

$|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$

$|x\rangle$ — qubit z argumentem funkcji; $|y\rangle$ — qubit z wynikiem funkcji; $f(x)$ — dodawanie mod 2

U_f:
$|0\rangle|0\rangle \to |0\rangle|0 \oplus f(0)\rangle$
$|0\rangle|1\rangle \to |0\rangle|1 \oplus f(0)\rangle$
$|1\rangle|0\rangle \to |1\rangle|0 \oplus f(1)\rangle$
$|1\rangle|1\rangle \to |1\rangle|1 \oplus f(1)\rangle$

jest to op. unitarna



$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\right)$

$|0\rangle|0\rangle \xrightarrow{H\otimes\mathbb{1}} \to \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle|0\rangle+|1\rangle|0\rangle) \xrightarrow{U_f} \frac{1}{\sqrt{2}}\cdot\left(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\right)$

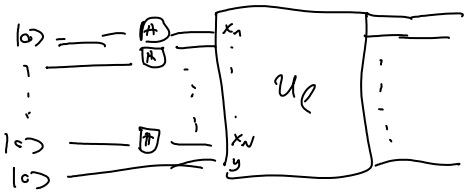
Uzyliśmy tylko raz obliczenia f a mamy stan $|\psi\rangle$ w którym pośrednio jest f zarówno dla $f(0)$ i $f(1)$.

„Liczy równocześnie" $f(0)$ i $f(1)$ dzięki temu, że

wymaga liczby superpozycja.

Ogólniej: $f: \{0,1\}^N \to \{0,1\}$ funkcja na N bitach

$|x_1, \dots, x_N, y\rangle \xrightarrow{U_f} |x_1 \dots x_N, y \oplus f(x_1, \dots, x_N)\rangle$



$|0\rangle^{\otimes N} |0\rangle \xrightarrow{H^{\otimes N} \otimes \mathbb{1}} \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes N} \otimes |0\rangle =$

$= \frac{1}{\sqrt{2^N}} \left(|0\dots0\rangle + |0\dots01\rangle + \dots + |1\dots1\rangle \right) \otimes |0\rangle \xrightarrow{U_f}$

wszystkie liczby N bitowe

$= \frac{1}{\sqrt{2^N}} \cdot \left(|0\dots0\rangle \otimes |f(0,\dots,0)\rangle + |0\dots01\rangle \otimes |f(0,\dots,1)\rangle + \dots + |1\dots1\rangle \otimes |f(1,\dots,1)\rangle \right)$

„Policzyliśmy" w jednym obliczeniu wartości funkcji f dla wszystkich 2^N możliwych danych wejściowych.
Natychmiastowe przyspieszenie obliczeń?
Ale nie tak szybko – nie istnieje pamięć pozwalająca jednocześnie odczytać wszystkie wartości f. Mierząc w bazie $|0\dots0\rangle, \dots, |1\dots1\rangle$. Stan zmienia się nam na jeden z wektorów superpozycji i poznamy tylko jedną wartość f ...

Ale może są problemy w których trzeba obliczenie wszystkich f jest elementem pośrednim a nie końcowym odczytujemy jakąś funkcję tych f i wystarczy jedno pomiar dający pożądany wynik.

Algorytm Deutscha

Najprostszy: całkowicie nie przydatny ale ceny dydaktycznie
Rozważmy funkcję $f: \{0,1\} \to \{0,1\}$.
Pytamy się czy funkcja jest niezmienna/stała?
Klasycznie funkcję musimy policzyć 2 razy.
A jeśli mamy pudełko kwantowe wystarczy raz

$U_f|x,y\rangle = |x, y \oplus f(x)\rangle$

U_f

$$\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \xrightarrow{U_f} \frac{1}{2}\Big(|0\rangle\big(|f(0)\rangle-|1\oplus f(0)\rangle\big) + |1\rangle\big(|f(1)\rangle-|1\oplus f(1)\rangle\big)\Big) =$$

$$= \frac{1}{2}\Big((-1)^{f(0)}|0\rangle(|0\rangle-|1\rangle) + (-1)^{f(1)}|1\rangle(|0\rangle-|1\rangle)\Big) =$$

$$= \frac{1}{2}\Big((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\Big)(|0\rangle-|1\rangle) \xrightarrow{H\otimes I}$$

$$= \frac{1}{2\sqrt{2}}\Big((-1)^{f(0)}(|0\rangle+|1\rangle) + (-1)^{f(1)}(|0\rangle-|1\rangle)\Big)\otimes(|0\rangle-|1\rangle) =$$

$$= \Big(\underbrace{\tfrac{1}{2}\big[(-1)^{f(0)}+(-1)^{f(1)}\big]}|0\rangle + \underbrace{\tfrac{1}{2}\big[(-1)^{f(0)}-(-1)^{f(1)}\big]}|1\rangle\Big)\otimes\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)$$

pierwszy nawias: 1 – f stała, 0 – f. zrównoważona; drugi nawias: 1 – f zrównoważona, 0 – f stała

Mierzymy pierwszy qubit, jeśli wynik $|0\rangle \Rightarrow$ f. stała

$|1\rangle \Rightarrow$ f zrównoważona

Dzięki temu że policzyliśmy jednocześnie dla obu wartości wejściowych wystarczyło tylko raz użyć pudełka.

4. Algorytm Shora (1993)

Rozkład liczby n bitowej na czynniki pierwsze w czasie n^3

Klasycznie problem trudny ale jednocześnie ważny (RSA).

Jeśli N liczba N bitowa $N < 2^n$

naiwna metoda:

- sprawdzamy czy dzieli się przez wszystkie liczby mniejsze od N (w zasadzie $\sqrt{N}$)

  mamy $\sim 2^{\frac{n}{2}}$ operacji

Czas obliczeń rośnie wykładniczo.

– Najlepsze algorytmy klasyczne mają skalowanie:

$$\sim 2^{\sqrt[3]{n}}$$ szybciej niż dowolny wielomian

Algorytm dość skomplikowany, ale podstawowy element to kwantowa transformata Fouriera

5. Kwantowa transformata Fouriera

• Klasyczna dyskretna trans Fouriera

$x_0, \dots, x_{N-1}$
$\downarrow F$
$y_0 \dots y_{N-1}$

$$y_k = \frac{1}{\sqrt{N}} \sum_j e^{\frac{2\pi i jk}{N}} x_j$$

Liczba operacji N^2, inaczej 2^{2n}
(jeśli N liczba n bitowa)
FFT może trochę przyspieszyć $N \log N \simeq 2^n n$
wciąż wykładniczo z liczbą bitów mierzących dane

• Kwantowa trans Fouriera
definiujemy op unitarny U_F na stanach n qubitowych

$$|j\rangle \xrightarrow{U_F} \frac{1}{\sqrt{N}} \sum_k e^{\frac{2\pi i jk}{N}} |k\rangle \qquad N = 2^n$$

Można sprawdzić że jest unitarna.

Jeśli zapiszemy nasze dane jako superpozycję:

$$\sum_j x_j |j\rangle \xrightarrow{U_F} \sum_j \frac{1}{\sqrt{N}} \sum_k x_j e^{\frac{2\pi i jk}{N}} |k\rangle =$$

$$= \sum_k y_k |k\rangle$$

gdzie $y_k = \frac{1}{\sqrt{N}} \sum_j x_j e^{\frac{2\pi i jk}{N}}$ to liczby

mamy na raz całą trans Fouriera.

mamy na raz całą trans Fouriera.

Uwagi!

– nie oznacza to że możemy odczytać wszystkie ??? na raz, czyli ten kw. paralelizm trzeba jeszcze sprytnie użyć dalej (okazuje się to możliwe właśnie w alg. Shora

– U_F trzeba zbudować z el. bramek kwantowych i pytanie jak to się skaluje z n, okazuje się że b. dobrze bo $\sim n^2$

Czyli mamy zysk w porównaniu z klasycznym FFT jak

n^2 do $n 2^n$!

Wykładniczy zysk.

6. Co ma trans Fouriera do rozkładu na czynniki pierwsze?

Transf. Fouriera pozwala znajdować okres funkcji!

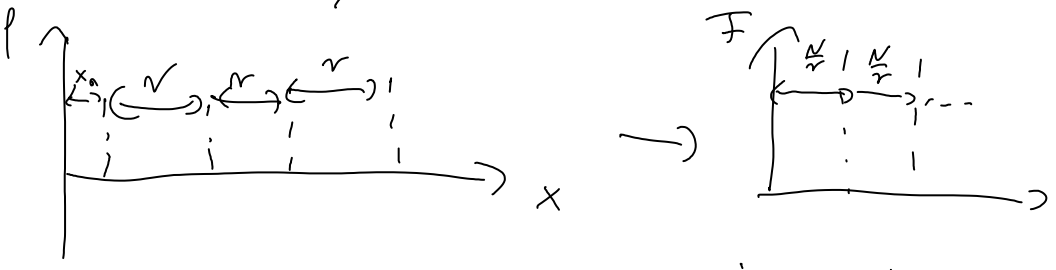
Rozważmy funkcję $f: Z_N \to Z$

↑ „liczby" mod N

Szukamy jej okresu $f(x+r)=f(x)$

liczby dla wszystkich argumentów

$$|f\rangle = \frac{1}{\sqrt{N}} \cdot \sum_{x=0}^{N-1} |x\rangle |f(x)\rangle$$



Mierzymy drugi rejestr jeśli wyjście y to x_0

$$\rightarrow \quad N = k \cdot r \quad , \quad \underbrace{\frac{1}{\sqrt{k}} \sum_{k=0}^{k-1} |x_0 + k \cdot r\rangle}_{|\psi_{x_0,r}\rangle} |y\rangle \quad , \quad f(x_0) = y_0$$

Mając stan $|\psi_{x_0,r}\rangle$ chcemy stwierdzić jakie jest r? Mierząc x nic nie da dostaniemy $x_0 + kr$ i nic nie wiemy, ale

możemy zrobić trans Fouriera

$$\mathcal{F}|\psi_{x_0,r}\rangle = \frac{1}{\sqrt{N}} \frac{1}{\sqrt{k}} \sum_{k=0}^{K-1} \sum_{c=0}^{N} e^{\frac{2\pi i \cdot l (x_0 + kr)}{N}} |l\rangle =$$

$$= \frac{1}{\sqrt{N}} \sum_{c=0}^{N} e^{\frac{2\pi i l x_0}{N}} |l\rangle \cdot \underbrace{\frac{1}{\sqrt{k}} \sum_{k=0}^{K-1} e^{\frac{2\pi i l \cdot r \cdot k}{N}}}_{\sqrt{k} \text{ jeśli } l \cdot r = s \cdot N} =$$

$$l = s \cdot \frac{N}{r}$$

$$= \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} e^{\frac{2\pi i s x_0}{r}} \left| s \cdot \frac{N}{r} \right\rangle$$

Teraz mierzymy i dostajemy

$\lambda = s \cdot \frac{N}{r}$ dla jakiegoś $s = 0 \ldots r-1$

Czy stąd możemy wyznaczyć r?

Jeśli $GCD(s, r) = 1$ to tak bo wie,

$\frac{\lambda}{N} = \frac{s}{r}$ więc sprowadzamy $\frac{\lambda}{N}$ do

nieskracalnej postaci mamy r.

Jeśli nie to niestety nie. Ale to jest

Jeśli nie to mieszczę się nie. ALE to jest mało prawdopodobne. Czyli mierzymy wymieniać r

Fakt z teorii liczb:
Liczba liczb pierwszych mniejszych od r równa jest $\frac{r}{\log r}$, czyli prawdopodobieństwo że wylosowane s jest $GCD(s,r) = 1$ jest co najmniej $\frac{1}{\log r}$.
Jeśli powtórzymy procedurę $\log N$ razy to mamy szanse $\sim 1 - \left(1 - \frac{1}{\log r}\right)^{\log N} \simeq 1 - \left(1 - \frac{1}{\log r}\right)^{\log r \frac{\log N}{\log r}} =$

$$= 1 - e^{-\frac{\log N}{\log r}} \longrightarrow 1$$

7 Co ma wyznaczenie okresu funkcji do rozkładu na czynniki pierwsze?

N – liczba, której czynników pierwszych szukamy
Losujemy liczbę $a < N$ i sprawdzamy $GCD(a, N)$ ⇧ Jeśli $GCD > 1$ to już mamy czynnik!
Jeśli $GCD(a,N) = 1$ (znacznie częściej) to:
Wtedy z Tw Eulera ⇧ istnieje r:

$$a^r = 1 \mod N \qquad (r - \text{mod } a)$$

Jeśli r jest parzyste:

$$a^r - 1 = 0 \mod N$$

$$\underbrace{\left(a^{\frac{r}{2}} - 1\right)}_{\alpha} \underbrace{\left(a^{\frac{r}{2}} + 1\right)}_{\beta} = k \cdot N \qquad \alpha \cdot \beta = k \cdot N$$

czyli α lub β muszą mieć wspólne dzielniki z N (chyba że α, β – wielokrotność N)

Fakt z teorii liczb: Dla $a < N$, t.że $GCD(a,N) = 1$ wybranego losowo prawd. że r ($a^r = 1 \mod N$) jest parzyste i $a^{\frac{r}{2}} \pm 1$ nie są wielokrotności N $\geq \frac{1}{2}$

Liczymy $GCD(\alpha, N)$ $GCD(\beta, N)$ dostajemy czynnik

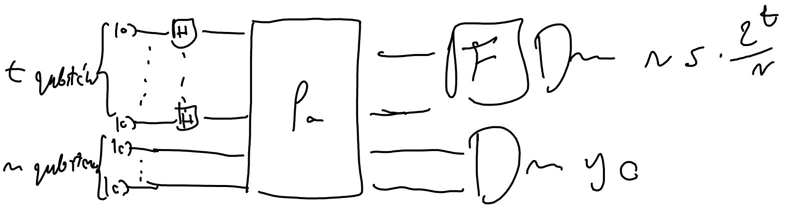
Liczby $GCD(\alpha, N)$, $GCD(\beta, N)$ dostajemy czynniki pierwsze N

Zauważmy, że najtrudniejsze jest wyznaczenie r.

Niech $f(x) = a^x \mod N$

Czym jest mod r? Okresem funkcji f!

$$f(x+r) = a^x \cdot a^r \mod N = a^x \mod N$$



$N < 2^m$

$$f_a : |x\rangle|0\rangle = |x\rangle|a^x \mod N\rangle$$

(liczba t bitów, liczba m bitów) — wymaga m^3 bramek

$$|0\rangle|0\rangle \xrightarrow{H,\ f\ (O(m^3))} \frac{1}{\sqrt{2^t}}\sum_{x=0}^{2^t-1}|x\rangle|a^x \mod N\rangle \longrightarrow$$

($|0\rangle = |0\dots0\rangle$ — t; $|0\rangle = |0\dots0\rangle$ — m)

mierzymy y_0

$$\longrightarrow \propto \sum_{\substack{x=0 \\ a^x \mod N = y_0}}^{2^t-1}|x\rangle|y_0\rangle \approx \frac{1}{\sqrt{K}}\sum_{k=0}^{K-1}|x_0+k\cdot r\rangle|y_0\rangle$$

($|y_0\rangle$ — nie istotne)

(może być tak (i zazwyczaj jest) że 2^t nie jest wielokrotnością r więc $2^t \approx r \cdot K$)

$$\xrightarrow{F\ (O(m^2))} \approx \frac{1}{\sqrt{r}}\sum_{s=0}^{r-1} e^{\frac{2\pi i x_0 s}{r}}\left|s\frac{2^t}{r}\right\rangle$$ otrzymujemy

przybliżenie t bitowe ułamka $\frac{s}{r}$, jeśli t odpowiednio duże (okazuje się że $t \approx 2m$) to wystarczy żeby uzyskać r. (continued fraction algorithm)

Uwaga: Ważna jest też implementacja funkcji f_a, poprzez elem. bramki kwantowe. Na szczęście to też jest $O(m^3)$

Czyli cała procedura jest $O(m^3)$!

Algorytm Grovera

Kwantowa przeszukiwanie nieuporządkowanej bazy danych. Niech baza składa się z N ($N=2^n$) elementów.

Niech f będzie funkcją identyfikującą poszukiwany element:

$f(x)=1$ gdy x jest poszukiwanym elementem

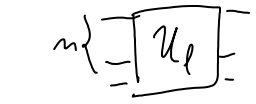
$f(x)=0$ w przeciwnym wypadku

Jeśli szukamy jednego elementu, musimy średnio obliczyć funkcję f $\sim \frac{N}{2}$ razy. Czy kwantowo da się lepiej?

Załóżmy że mamy kwantową wersję f

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle$$

(x: n qubitów, y: 1 qubit)

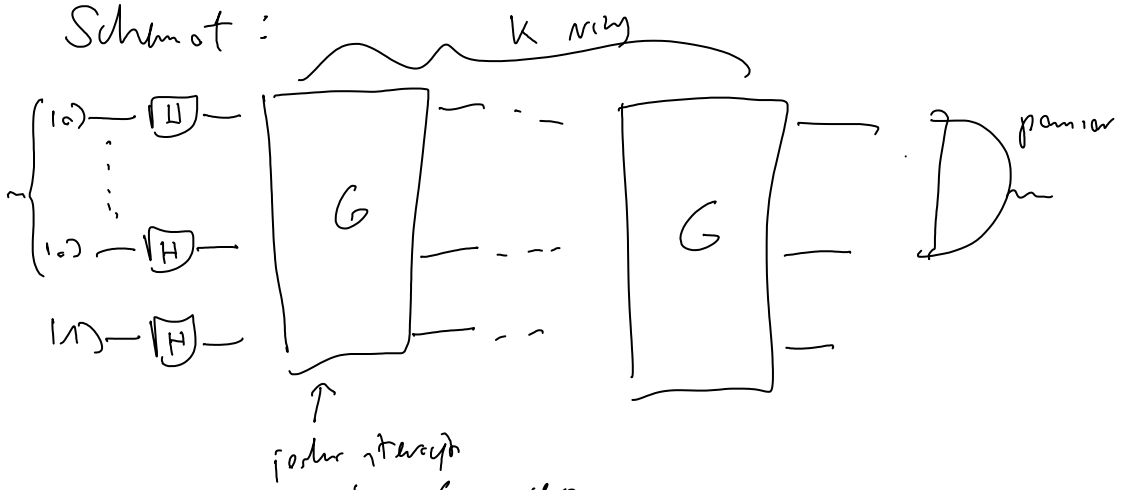


Zauważmy że:

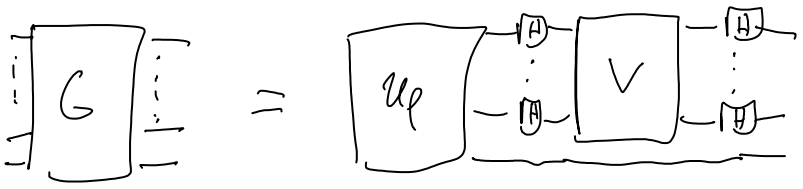
$$U_f |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = (-1)^{f(x)} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

W dolnym ciągu mamy ignorowaną ostatnią qubit on cały czas będzie pozostawał w stanie $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

Schemat:



w alg. Grovera



V – op unitarne t.ż $\quad V|0\rangle = |0\rangle$
$V|x\rangle = -|x\rangle, \quad x > 0$

$V = 2|0\rangle\langle 0| - \mathbb{1}$

$H^{\otimes N} V H^{\otimes N} = 2|\psi\rangle\langle\psi| - \mathbb{1}$, gdzie

$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_x |x\rangle$

Czyli: $\quad G = \left(2|\psi\rangle\langle\psi| - \mathbb{1}\right) U_f$

Mam $|x_0\rangle$ – bardzo odpowiedź szukanym stanem

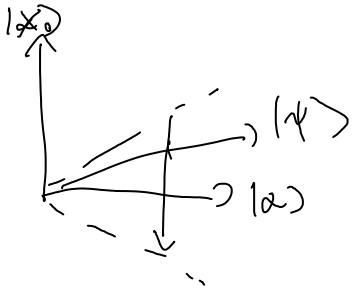
Mam $|\alpha\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x_0} |x\rangle$ bardzo superpozycja pozostałych stanów.

Rozważmy jak działa G na superpozycję $|x_0\rangle$ i $|\alpha\rangle$

Zauważmy że $\quad |\psi\rangle = \sqrt{\frac{N-1}{N}}|\alpha\rangle + \frac{1}{\sqrt{N}}|x_0\rangle$
podprzestrzeni.

$G(a|\alpha\rangle + b|x_0\rangle) = \left(2|\psi\rangle\langle\psi| - \mathbb{1}\right)(a|\alpha\rangle - b|x_0\rangle)$

odbicie względem kierunku $|\alpha\rangle$



$$= 2|\psi\rangle\left(\sqrt{\tfrac{N-1}{N}}\cdot a - \tfrac{1}{\sqrt{N}} b\right) - \left(a|\alpha\rangle - b|x_0\rangle\right) =$$

↑ odbicie względem $|\psi\rangle$

$$= 2\,\tfrac{N-1}{N}\, a|\alpha\rangle - 2\,\tfrac{1}{N}\, b|x_0\rangle + 2\tfrac{\sqrt{N-1}}{N}\, a|x_0\rangle - 2\tfrac{\sqrt{N-1}}{N} b|\alpha\rangle$$
$$- (a|\alpha\rangle - b|x_0\rangle) =$$

$$= \left(2a - \tfrac{2a}{N} - 2\tfrac{\sqrt{N-1}}{N} b - a\right)|\alpha\rangle + \left(b - \tfrac{2}{N} b + \tfrac{2\sqrt{N-1}}{N} a\right)|x_0\rangle$$

$$= \left(a\left(1 - \tfrac{2}{N}\right) - b\,\tfrac{2\sqrt{N-1}}{N}\right)|\alpha\rangle + \left(b\left(1 - \tfrac{2}{N}\right) + a\,\tfrac{2\sqrt{N-1}}{N}\right)|x_0\rangle$$

Czyli po prostu obrót o kąt θ:

$$\cos\theta = 1 - \tfrac{2}{N} \qquad \theta = \arccos\left(1 - \tfrac{2}{N}\right)$$

W Alg. Grovera startujemy ze stanu $|\psi\rangle$

$$|\psi\rangle = \sqrt{\tfrac{N-1}{N}}\,|\alpha\rangle + \tfrac{1}{\sqrt{N}}\,|x_0\rangle =$$

$$= \cos\tfrac{\theta}{2}|\alpha\rangle + \sin\tfrac{\theta}{2}|x_0\rangle$$

i w kolejnych krokach obracamy się o kąt θ

Czyli po k iteracjach:

$$G^k|\psi\rangle = \cos\left(\tfrac{2k+1}{2}\theta\right)|\alpha\rangle + \sin\left(\tfrac{2k+1}{2}\theta\right)|x_0\rangle$$

Jeśli N bardzo duże $\qquad \theta \approx \tfrac{2\sqrt{N-1}}{N} \approx \tfrac{2}{\sqrt{N}}$

Chcemy, żeby $\tfrac{2k+1}{2}\theta \approx \tfrac{\pi}{2}$

$$(2k+1)\cdot\tfrac{2}{\sqrt{N}} = \pi \qquad k \approx \sqrt{N}$$

Czyli kwadratowe przyspieszenie w porównaniu z algorytmem klasycznym.