Inhaltsverzeichnis

III. Quantenalgorithmen

Was haben wir bis jetzt gelernt?

- Qbits
- Ein-Qbit Gatter
- Entanglement
- Zwei-Qbit Gatter

} universeller Satz an Gattern

- Physikalische Realisierungen

Was fehlt uns noch, um schwierige Rechnungen durchzuführen? Algorithmen

Und was heißt überhaupt schwierig?

Die Komplexitätsklasse eines Problems hängt vom Algorithmus ab, bzw. von den benötigten Ressourcen (Rechenoperationen, Speicher)

- P: Die Resourcen steigen polynomial in (Log x)=Anzahl Stellen der Zahl. Bsp: Multiplikation
- NP: kann in polynomialer Zeit *überprüft* werden, also schnell. Kann zum Rechnen aber auch sehr schwierig, z.B. exponentiell in Log x sein.
- NP-vollständiges Problem: Jedes Problem in NP muss in polynomialer Zeit darauf zurückgeführt werden können.
- Sicher ist: $P \subset NP$. Unbekannt: Ist P=NP?

Algorithmentypen

Was bringen da jetzt Quantencomputer? Neuartige Algorithmen möglich!

Bsp: Primzahlfaktorisierung ist klassisch mindestens NP (oder schwerer) und exponentiell, aber mit Shor‘s Algorithmus in polynomialer Zeit lösbar.

Numerische Rechnungen wie Multiplikation machen wenig Sinn, da

a) banale Multiplikation z.b. 1011x1100 ähnlich schwierig ist und Pentium schneller.

b) Rechnen mit Überlagerung, gibt zwar viele Ergebnisse , aber kann je nur eines messen. Rückschluss? $\left(\sum_{N=0}^{2^n-1}|N\rangle\right)\times\left(\sum_{M=0}^{2^m-1}|M\rangle\right)$

Quantenalgorithmen konzentrieren sich daher auf folgende Fragestellungen:

- Suche in einer Datenbank (Orakel)
- Erkennen einer globalen Eigenschaft einer Funktion, z.B. Periode, Mittelwert. Bsp.: Deutsch-Josza, Quanten-Fourier-Transformation ⇒ Faktorisierung, …
- Weitere: Zahlentheoretische Probleme (Finde eine Zahl, die …),
  - Berechnen des Gradienten in n-Dimensionen (Jordan)
  - http://www.its.caltech.edu/~sjordan/zoo.html

Übersicht über Algorithmen für QC: http://www.its.caltech.edu/~sjordan/zoo.html

III.1 Deutsch Josza-Algorithmus

Historisch erster Quantenalgorithmus, der schneller als klassischer ist.

Problem: Man sieht von einer Münze nur eine Seite. Ist die andere Seite gleich (=konstant) oder anders (=balanciert)?

Klassische Lösung: Münze umdrehen=2 Messungen. Geht's mit QC besser?

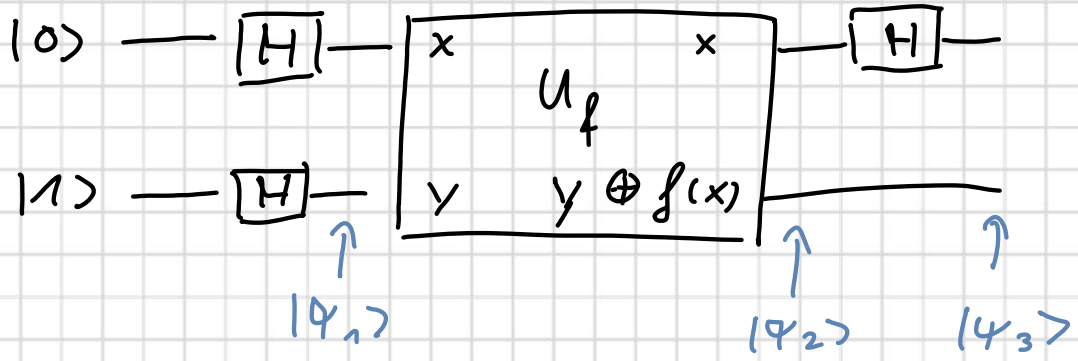
Mathematische Darstellung:

$$f(\underbrace{x \epsilon \{0,1\}}_{\text{Vorder/Rückseite}}) = \begin{cases} 0 & \text{Kopf} \\ 1 & \text{Zahl} \end{cases}$$

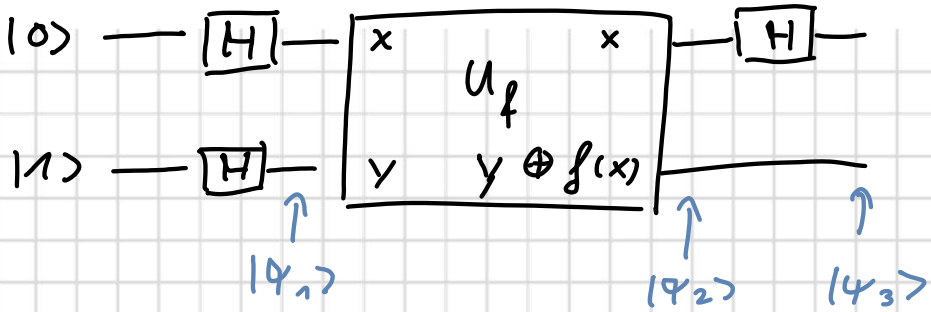
4 mögliche Funktionen: f(x)=x oder f(x)=NOT x „balanciert" (2 versch. Seiten)

f(x)=1, f(x)=0 „konstant"

Algorithmus:



Deutsch-Josza 2



$$|\psi_1\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \quad (|0\rangle - |1\rangle)$$

$$|\psi_2\rangle = \frac{1}{2}(|0, f(0)\rangle + |1, f(1)\rangle - |0, 1 \oplus f(0)\rangle - |1, 1 \oplus f(1)\rangle)$$

$$= \frac{1}{2}\begin{cases} (|0\rangle + |1\rangle) & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für konstant } f(0) = f(1) \\ (|0\rangle - |1\rangle) & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für balanciert } f(1) = 1 \oplus f(0) \end{cases}$$

$$|\psi_3\rangle = \frac{1}{2}\begin{cases} |0\rangle & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für konstant } f(0) = f(1) \\ |1\rangle & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für balanciert } f(1) = 1 \oplus f(0) \end{cases}$$

Eine einzige Messung von Qbit 1 ergibt, ob Münze 2 unterschiedliche Seiten hat („$|1\rangle$“) oder nicht! Funktion U_f wird nur einmal ausgeführt.

Das letzte Hadamard Gatter sorgt also für eine geschickte Interferenz der Ergebnisse von U_f, so dass man mit einer Messung eine globale Eigenschaft von f(x) erhält.

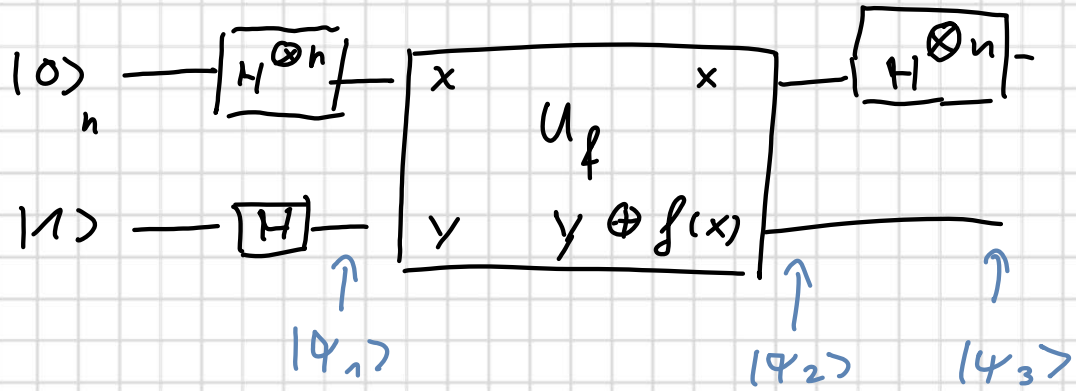
N+1 Qbit Deutsch-Josza Algorithmus

Alice schickt Bob eine N-bit Zahl x, und Bob schickt das 1-bit Ergebnis f(x) zurück.

Er hat ein f(x) gewählt, das entweder für alle x konstant ist, oder balanciert, d.h. für eine willkürliche Hälfte der x ist f(x)=0, für die andere 1.

Alice soll rausfinden, ob Bob‘s Funktion f(x) konstant oder balanciert ist. Dazu braucht sie im schlechtesten Fall x/2+1 Versuche (x hin, bekommt f(x) zurück).

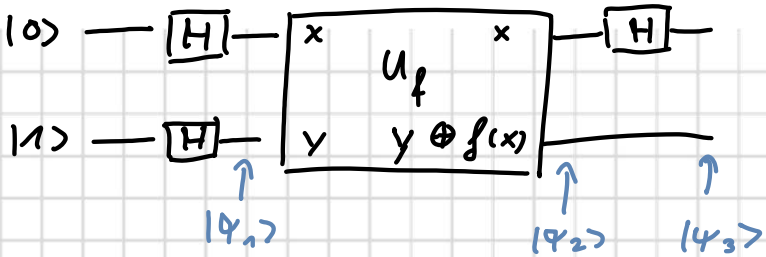
Der Deutsch-Josza-Algorithmus schafft es mit einem Versuch!



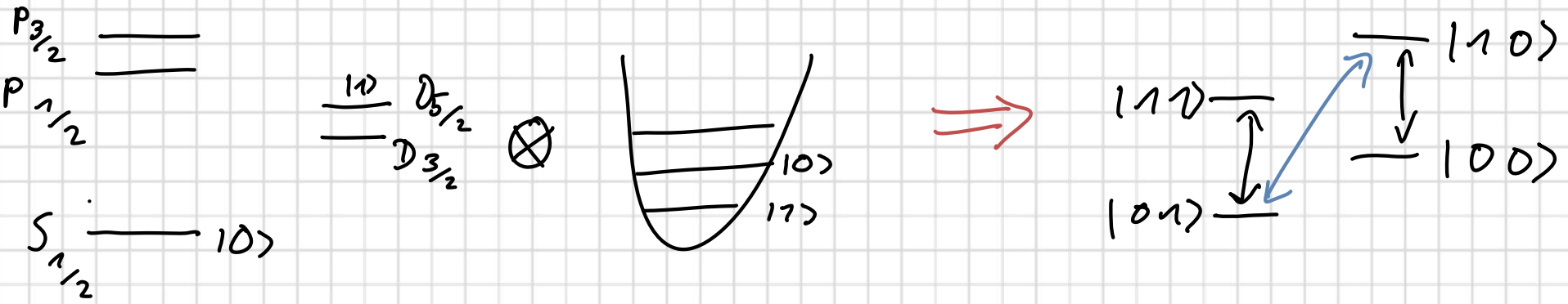
Algebra analog zu 2 Qbit, siehe Stolze & Suter 8.2.4 oder N&C 1.4.4

Ergebnis: Oberes Register=$|0\rangle^{\otimes n}$ genau dann, wenn konstante Funktion

Experiment:



Einzelnes Ca+ Ion:



1. Präparation im Grundzustand des internen Zustands mit 0 Phononen ⇒ $|01\rangle$
2. Hadamard Gatter mit Laserpulsen (für intern z.B. auf Trägerfrequenz)
3. Aber wie baut man jetzt U_f?

	f(x)=0	f(x)=1	f(x)=x	f(x)= NOT x
Qbit 2	y⊕f(x)=y⊕0	y⊕f(x)=y⊕1	y⊕f(x)=y⊕x	y⊕f(x)=y⊕NOT x
$\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$	$\|00\rangle$ $\|01\rangle$ $\|10\rangle$ $\|11\rangle$	$\|01\rangle$ $\|00\rangle$ $\|11\rangle$ $\|10\rangle$		
U_f=?	**1**	**NOT_2**	**CNOT**	**Z-CNOT**

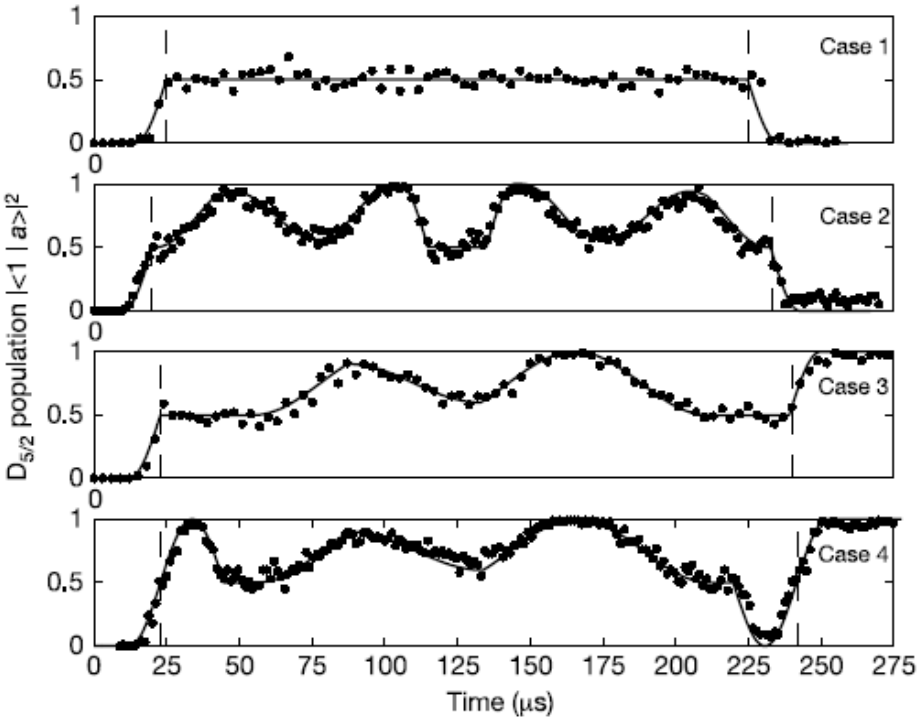
S. Gulde et al., Nature 421, 48 (2003)

Daten

D.h. man wählt eine der vier Funktionen f(x), und wendet – nach Initialisierung und Hadamard – das entsprechende Gatter U_f (z.B. CNOT im Fall f(x)=x) an.

Dann sollte man im internen Qbit je nach f(x) mit Fluoresenz messen:

$$|\psi_3\rangle = \frac{1}{2}\begin{cases} |\mathbf{0}\rangle & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für konstant } f(0) = f(1) \\ |\mathbf{1}\rangle & (|f(0)\rangle - |1 \oplus f(0)\rangle) \quad \text{für balanciert } f(1) = 1 \oplus f(0) \end{cases}$$



f(x) = 0	**U_f=1**
f(x) = 1	**U_f=NOT$_2$**
f(x) = x	**U_f=CNOT**
f(x) = NOT x	**U_f=Z-CNOT**

S. Gulde et al., Nature 421, 48 (2003)

III.2 Grover Algorithmus

Problem: Suche in einer unstrukturierten Datenbank, Beispiel Telefonbuch, wie heisst der Besitzer des Anschlusses 1234567?

Klassisch: weitverbreitetes Problem. Durch Ausprobieren finden, im besten Fall 1 Versuch, im schlechtesten N ⇒ im Mittel N/2

Grover: $O(N^{1/2})$ Operationen

Hilfsmittel: **Orakel-Funktion**

- Wir wollen $N=2^n$ Einträge überprüfen
- M mögliche Lösungen, mit Orakel-Funktion $f(x) = \begin{cases} 1 & \text{falls x eine Lösung ist} \\ 0 & \text{sonst} \end{cases}$
- Ziel: Die Anzahl der Abfagen f(x) zu minimieren.
- Im QC wird f(x) durch einen unitären Operator O beschrieben:

$$O|x\rangle_{\text{Datenbank}}|q\rangle_{\text{Orakel}} = |x\rangle|q \oplus f(x)\rangle$$

wähle Orakel Qbit,

$$|q_0\rangle = (|0\rangle - |1\rangle)\,1/\sqrt{2}$$

$$|q_0 \oplus 1\rangle = -|q_0\rangle$$

$$O|x\rangle|q_0\rangle = (-1)^{f(x)}|x\rangle|q_0\rangle$$

da sich $|q_0\rangle$ nicht ändert, können wir es i.F. weglassen

Rezept

Ziel des Grover-Algorithmus ist es, dass die Wahrscheinlichkeitsamplitude der vom Orakel mit „-“ gekennzeichneten Lösungen zunimmt

1) Initialisierung: $|\psi_1\rangle = |0\rangle^{\otimes n}$ (n-Qbits)

2) Hadamard-Superposition: $|\psi_2\rangle = H^{\otimes n}\,|\psi_1\rangle = 1/\sqrt{N}\ \Sigma_{x=0}^{N-1}\ |x\rangle$ wobei z.B. bei 3 Qbits $|x\rangle=|3\rangle:= |011\rangle$. Dabei heißt $|3\rangle$ Dezimal und $|011\rangle$ Binär-Schreibweise

3) Iteration in vier Schritten:

a) $|\psi_{k+1/4}\rangle = O\ \ |\psi_k\rangle$

b) $|\psi_{k+1/2}\rangle = H^{\otimes n}\,|\psi_{k+1/4}\rangle$

c) $|\psi_{k+3/4}\rangle = C_\pi\ \ |\psi_{k+1/2}\rangle$ mit $C_\pi = \begin{pmatrix} 1 & 0 & 0 & .. \\ 0 & -1 & 0 & .. \\ 0 & 0 & -1 & .. \\ .. & .. & .. & .. \end{pmatrix} = -\mathbf{1} + 2|0\rangle\langle 0|$

d.h. -1 für $|x\rangle=|1\rangle$

a) $|\psi_{k+1}\rangle = H^{\otimes n}\,|\psi_{k+3/4}\rangle$

Zusammenfassung:

$$\begin{aligned} G &= H^{\otimes n} C_\pi H^{\otimes n}\, O \quad \text{mit } C_\pi = -\mathbf{1} + 2|0\rangle\langle 0| \\ &= (-\underbrace{H^{\otimes n}\mathbf{1}H^{\otimes n}}_{-\mathbf{1}} + 2\,\underbrace{H^{\otimes n}|0\rangle}_{|\psi\rangle_2}\,\underbrace{\langle 0|H^{\otimes n}}_{\langle\psi|_2})\, O \\ &= (2|\psi_2\rangle\langle\psi_2| - \mathbf{1})\, O \end{aligned}$$

Geometrische Erklärung des Algorithmus

Definiere orthonormale Zustände:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_x (1 - f(x))|x\rangle$$ Überlagerung der Nicht-Lösungen

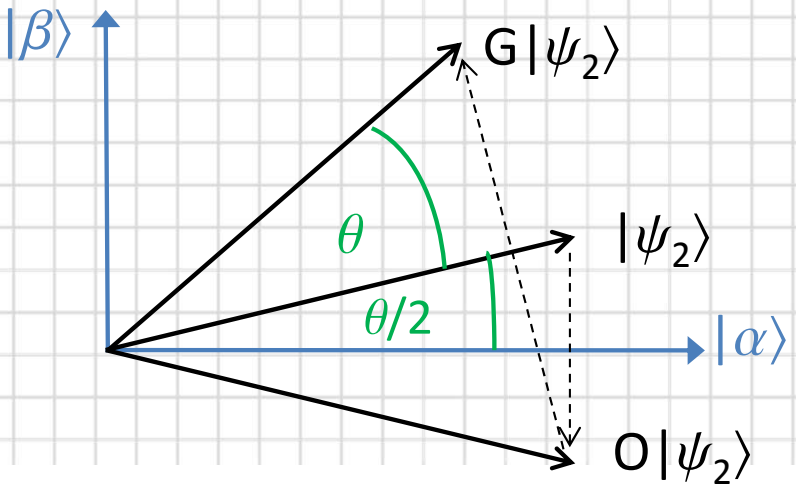
$$|\beta\rangle = \frac{1}{\sqrt{M}} \sum_x f(x)|x\rangle$$ Überlagerung der Lösungen

$$\Rightarrow |\psi_2\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}\,|\beta\rangle = \cos\left(\frac{\theta}{2}\right)|\alpha\rangle + \sin\left(\frac{\theta}{2}\right)|\beta\rangle$$

$$O|\psi_2\rangle = \cos\left(\frac{\theta}{2}\right)|\alpha\rangle - \sin\left(\frac{\theta}{2}\right)|\beta\rangle$$ Spiegelung an der $|\alpha\rangle$ Achse

$$H^{\otimes n} C_\pi H^{\otimes N} = (2|\psi_2\rangle\langle\psi_2| - \mathbf{1}) = \underbrace{|\psi_2\rangle\langle\psi_2|}_{\text{Projektion auf } |\psi_2\rangle} - \underbrace{(\mathbf{1} - |\psi_2\rangle\langle\psi_2|)}_{\text{Projektion auf orthogonal } |\psi_2\rangle}$$

Spiegelung an $|\psi_2\rangle$ - Achse



Anzahl der Iterationen?

G $|\psi_2\rangle$= cos (3θ/2) $|\alpha\rangle$ + sin (3θ/2) $|\beta\rangle$, nach k Anwendungen

G^k$|\psi_2\rangle$= cos ((2k+1) θ/2) $|\alpha\rangle$ + sin ((2k+1)θ/2) $|\beta\rangle$

Für wenige Lösungen M≪N $\Rightarrow$ $\theta \ll \pi$ kann man ungefähr erreichen, dass $\frac{2k+1}{2}\theta = m\frac{\pi}{2}$ und die Wahrscheinlichkeit, $|\beta\rangle$ zu messen ist maximal.

Wir müssen M kennen, um das ideale k zu kennen:

$$k = \frac{\pi}{2\theta} - \frac{1}{2} = \frac{\pi}{4\arcsin\left(\sqrt{\frac{M}{N}}\right)} - \frac{1}{2} \leq \frac{\pi}{4}\sqrt{\frac{N}{M}}, \qquad \text{da } \arcsin x > x$$

Mit Hilfe der von verbesserten Algorithmen kann man aber auch M über die QFT finden.

III.3 Der Shor-Algorithmus

Durchbruch: Primzahlzerlegung einer Zahl N in polynomialer Zeit.

Bedeutung: Eines der wenigen Probleme, an dem Mathematiker, Bankiers und Militärs gleichzeitig Interesse haben. Zahlentheorie / Kryptographie.

Beispiel: Bank macht eine große Zahl N öffentlich, die der Internetkunde zum Verschlüsseln seiner Nachrichten benutzt. Die Nachricht kann nur entschlüsselt werden, wenn man die Primfaktoren (privater Schlüssel) der großen Zahl kennt. Genauer: RSA-Verschlüsselung, braucht einige Zahlentheorie, siehe Mermin 3.2-4

Klassisch: Es existiert (noch?) kein effizienter Algorithmus zur Primzahlzerlegung. Naiv: Zahlen bis $N^{½}$ ausprobieren $\Rightarrow$ $T \sim \mathrm{Exp}(\mathrm{Log}[N]/2)$. Der beste ist das allgemeine Zahlkörpersieb. $T \sim \mathrm{Exp}(\mathrm{Log}[N]^{1/3})$. (Erklärung: „rational sieve“ in wikipedia)

Schwieriges Problem – der Quantenalgorithmus auch nicht trivial… An die Arbeit!

Der Kern des Algorithmus besteht aus 2 Elementen:

1. Zahlentheorie: Primzahlzerlegung auf Finden der Periodizität bestimmter Funktionen zurückführen
2. Periode finden = Fourier-Transformation. Es existiert ein sehr effizienter Algorithmus für Quantencomputer, die „Quanten-Fourier-Transformation“

Siehe Stolze & Suter, 8.3, Ekert & Josza, Rev. Mod. Phys. 68, 733 (1996)

Hilfsmittel I: Euklidscher Algorithmus (ca 300 v. Chr)

N > y sind ganze Zahlen und wir suchen g, den größten gemeinsamen Teiler ggt (N, y)

$\Rightarrow N - k\,y, \quad k \in \mathbf{N}$ sind Vielfache von g

Es sei $r = N - k\,y < y$ der Rest der Division N/y

- Falls r = 0: Problem gelöst, ggt = y
- Falls $r \neq 0$: <u>ggt (y, r) = ggt (N, y) = g !</u> Denn $N / g = (k\,y + r) / g = k\,y/g + r/g \in$ □

  wiederholen bis gelöst.

Beispiel: N = 100, y = 35, wir suchen g = ggt (100, 35)

1. $r = 30 \Rightarrow N' = 35, y' = 30$, suche g = ggt (35, 30)
2. $r' = 5 \Rightarrow N' = 30, y' = 5$, suche g = ggt (30, 5)
3. $r' = 0 \Rightarrow g' = y' = 5$, d.h. der ggt (100, 35) = 5.

Der Euklidsche Algorithmus ist auch heute noch eine effiziente Methode, um den ggt zu finden.

Hilfsmittel II: Modulares Exponentieren

$a^x \bmod N$ (d.h. wir suchen den Rest der Division a^x/N)

Die Ordnung r von a mod N ist definiert als die kleinste Zahl r, für die $a^r \bmod N = 1$ gilt.

Beispiele

Ordnung von $11^r \bmod 15 = 1 \Rightarrow r=2$

$\Rightarrow a^r = k\,N + 1$ und $a^{r+1} \bmod N = (k\,N\,a + a) \bmod N = a$

d.h. $F_N(x) = a^x \bmod N$ ist periodisch mit Periode r ($F_N(r+1) = F_N(1)$).

$r \leq N$, weil $F_N(x)$ der Rest einer Division durch N ist.

Die Periode r werden wir mit der Quanten-Fourier-Transformation finden. Dies ist der schwierige Teil des Shor-Algorithmus

Zahlentheoretische Grundlage für Shor

1. Wir wollen N zerlegen.
2. Wir wählen eine Zahl a<N.
   - Wenn ggt(a,N)>1 prima, dies ist Primteiler. Das wäre verdammt viel Glück.
   - Es sei ggt(a,N)=1. Wir finden die Ordnung $a^r = 1 \bmod N$ (mit QC)

Mögliche Ergebnisse für r:

a. *r ist ungerade* *(uninteressant)*

b. *r ist gerade und $a^{r/2} \bmod N = -1$* *(uninteressant)*

c. r ist gerade und $a^{r/2} \bmod N \neq -1$, *(Bsp: $11^1 \bmod 15 = -4$ oder $+11$)*

mit $a^r = 1 \bmod N \quad \Rightarrow \quad (a^r - 1) \bmod N = 0$

$(a^{r/2} + 1)(a^{r/2} - 1) \bmod N = 0$

$\Rightarrow$ $a^{r/2} \pm 1$ hat einen gemeinsamen Teiler mit N und den können wir mit Euklid ja einfach finden $\rightarrow$ Primteiler.

Der Teiler ist $\neq N$, denn aus c. folgt: $(a^{r/2} + 1) \bmod N \neq 0$. Und $(a^{r/2} - 1) \bmod N \neq 0$, denn sonst wäre $a^{r/2} \bmod N = 1 \quad \rightarrow \quad r/2$ wäre Ordnung. Widerspruch.

Beispiel: N=15, Wähle a=11. ggt(15,11)=1. r=2, da $11^2=121=8 \times 15+1$.

Suche ggt($11^{r/2} \pm 1$, 15): ggt(10,15)=5, ggt(12,15)=3 ⇒ 15=3x5

Es hätte auch für andere a={2,4,7,8,11,13,14} mit Ordnung {4,2,4,4,2,4,2} geklappt, außer für 14, wo mit r=2 14^1 mod 15=-1 -> Fall b.

Probabilistischer Algorithmus

In den Fällen a. und b. ist keine Aussage möglich! Wie oft tritt c. auf, wenn wir a zufällig wählen?

1. Falls $N=p^s$ (p prim, $s \geq 2$) kann p einfach bestimmt werden: Man versucht für alle $s \leq$ Log[N]/Log[2] *(von $N =^? 2^s$)*, ob Log[N]/Log[p]=s ϵ ℕ
2. Falls $N=p_1^{c1} \ldots p_m^{cm}$ ($m \geq 2$) und a zufällig aus dem Intervall 1<a<N-1 gewählt ist mit ggt (N, a)=1 und a^r mod N = 1 folgt :

   Prop (r gerade und $a^{r/2}$ mod N ≠ -1) $\geq 1-2^{-m} \geq$ ¾ (Beweis N&C, App. 4, Ekert& Josza App. B)

⇒ Die Chance, einen nichttrivialen Primfaktor zu finden, ist größer als 75%.

Aber: Der Shor-Algorithmus ist probabilistisch. Er findet das Ergebnis nur mit einer gewissen Wahrscheinlichkeit. Dann ist es aber korrekt.

Zusammenfassung von Shors Strategie

Gesucht: Primfaktoren einer Zahl N

1. Überprüfe „triviale“ Fälle: Ist N gerade? Ist $N = p^s$, $p \geq 1$, $s \geq 2$?
2. Wähle zufällige Zahl a mit $1 < a \leq N-1$ und bestimme ggt (N, a)
3. Falls ggt (N, a) = 1 bestimme die Ordnung r von a^r mod N.
4. Falls r geradzahlig und $a^{r/2}$ mod $N \neq -1$ berechne ggt ($a^{r/2} \pm 1$, N) und überprüfe, ob das Ergebnis ein Faktor von N ist.

Ggf. wiederholen von 2. bis 4.

Die Aufgabe des Quantencomputers liegt im effizienten Auffinden der Periode r von $F_N(x) = a^x$ mod N mit Hilfe der Quanten-Fourier-Transformation.

III.4 Quanten-Fourier-Transformation (QFT)

Klassische, diskrete Fourier-Transformation: $$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} e^{2\pi i k \cdot j/N} \cdot x_j$$

Inverse Operation: $$x_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{-2\pi i j \cdot k/N} \cdot y_k$$

In Matrix-Darstellung: $\vec{y} = M \cdot \vec{x}$ mit den Matrix-Einträgen: $M_{kj} = e^{2\pi i k \cdot j/N}$

⇒ Rechenaufwand: N^2 Multiplikationen und Additionen.

Das geht noch schneller mit der Fast-Fourier-Transformation (FFT) nach Gauß, indem wir die Summe nach geraden und ungeraden j aufteilen und etwas umsortieren . Das ergibt 2 Fourier-Transformationen mit je (N/2) Matrix-Einträgen ⇒ Rechenaufwand $2(N/2)^2$. Durch Wiederholen dieser Halbierung (dafür muss N eine Zweierpotenz sein) erniedrigt sich der Rechenaufwand von N^2 auf N log(N), wodurch die FFT von hohem praktischem Nutzen ist.

FFT: wiki, QFT: Stolze & Suter, 8.3, Ekert & Josza, Rev. Mod. Phys. 68, 733 (1996)

Noch schneller: QFT

Betrachte einen $N=2^n$- dimensionalen Hilbert-Raum (n Qbits) und die unitäre Transformation, definiert als:

$$|j\rangle \overset{\text{QFT}}{\mapsto} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i\, j\cdot k/N} \cdot |k\rangle$$

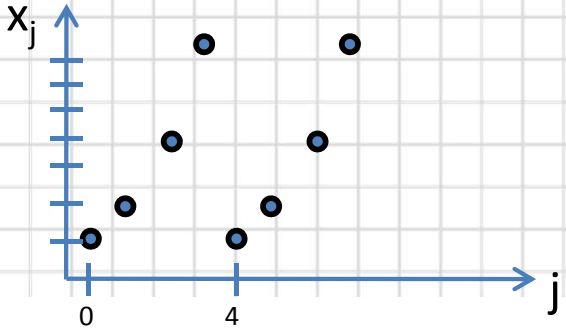
Mit $|j\rangle$, $|k\rangle$ dem j.ten bzw. k. Basisvektor des H-Raums

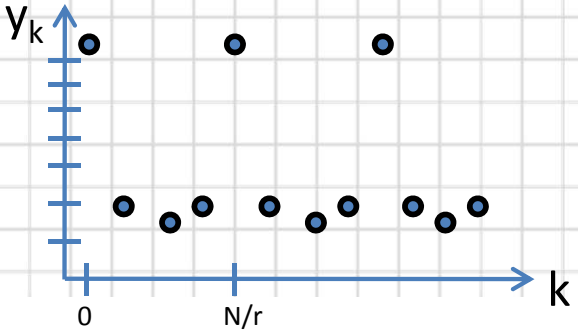
Diese Definition ist nützlich, denn ein beliebiger Quantenzustand mit Koeffizienten x_j geht über in

$$\sum_{j=0}^{N-1} x_j |j\rangle \quad \mapsto \quad \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \sum_{k=0}^{N-1} e^{2\pi i\, j\cdot k/N} \cdot |k\rangle$$

$$= \sum_{k=0}^{N-1} \underbrace{\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i\, j\cdot k/N}}_{y_k} \cdot |k\rangle$$

D.h. wenn wir die Wahrscheinlichkeit für ein gewisses $|k\rangle$ messen, erhalten wir y_k, die FT von x_j.
Praktischerweise wählt man für $x_j=F_N(j)=a^j \bmod N$.





Beispiel: N=15,a=2

j	x_j
0	1
1	2
2	4
3	8
4	1

$\Rightarrow$ r=4

Implementierung der QFT

Binärdarstellung von $|k\rangle = |k_1\rangle \otimes |k_v\rangle \ldots \otimes |k_n\rangle$ mit $|k_v\rangle = \{|0\rangle$ oder $|1\rangle\}$, falls

$k = \sum_{l=1}^{n} k_l 2^{n-l}$, wie angenommen, nur klassische Werte annimmt. Analog $|j\rangle$.

$$|j\rangle \mapsto \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i\, j\cdot k/N} \cdot |k\rangle = 2^{-\frac{n}{2}} \sum_{k_1=0}^{1} \ldots \sum_{k_n=0}^{1} e^{2\pi i\, j \sum_{l=1}^{n} k_l\, 2^{n-l}/2^n} \cdot |k_1\rangle \ldots |k_n\rangle$$

$$= 2^{-\frac{n}{2}} \sum_{k_1=0}^{1} \ldots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i\, j\, k_l\, 2^{-l}} \cdot |k_l\rangle$$

$$= 2^{-\frac{n}{2}} \bigotimes_{l=1}^{n} \left(\sum_{k_l=0}^{1} e^{2\pi i\, j\, k_l\, 2^{-l}} \cdot |k_l\rangle \right)$$

$$= 2^{-\frac{n}{2}} \bigotimes_{l=1}^{n} \left(|0\rangle_l + e^{2\pi i\, j\, 2^{-l}} |1\rangle_l \right)$$

Bsp: n=4-Qbit, j=3, *l*=2

Binär: j=0011,

$j \cdot 2^{-2} = 0 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2}$

$= 3/4 = 00.11$

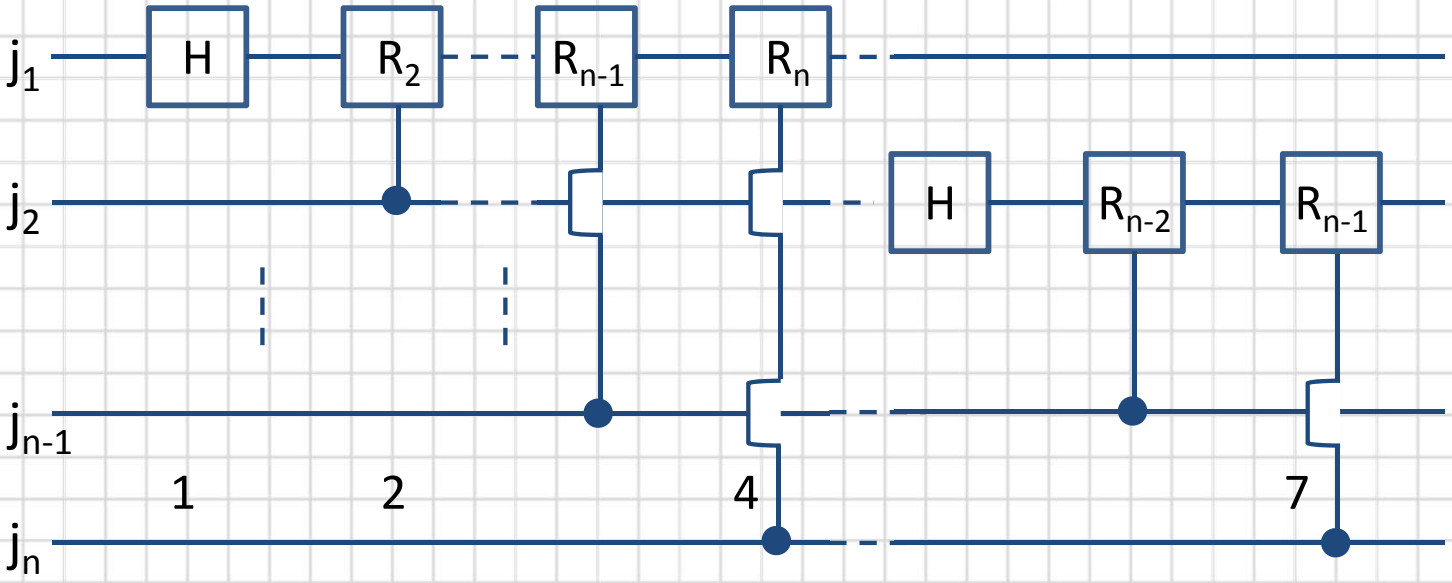
mit $j \cdot 2^{-l} = \sum_{\nu=1}^{n} j_\nu 2^{n-\nu-l} =: \underbrace{j_1\, j_2 \ldots j_{n-l}}_{\text{ganze Zahl}} \cdot \underbrace{j_{n-l-1} \ldots j_n}_{\text{rationale Zahl}}$ und $e^{2\pi i \cdot \text{ganze Zahl}} = 1$

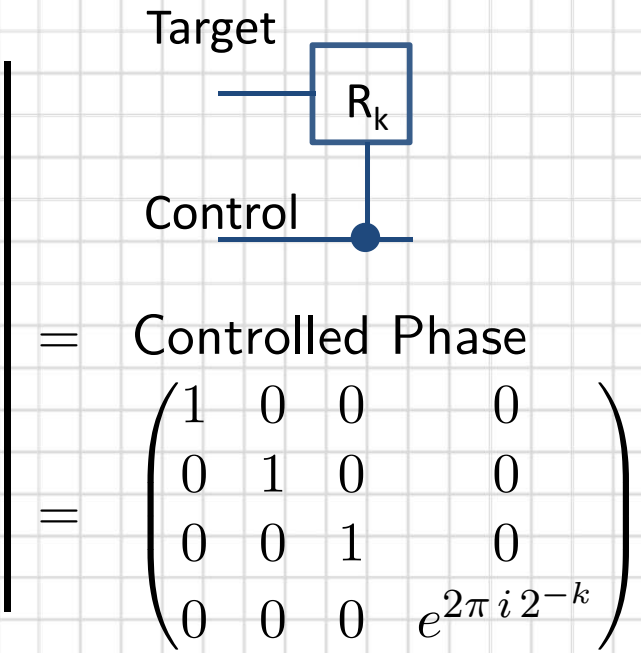
$$= 2^{-\frac{n}{2}} \left(|0\rangle_1 + e^{2\pi i\, 0.j_n} |1\rangle_1 \right) \left(|0\rangle_2 + e^{2\pi i\, 0.j_{n-1} j_n} |1\rangle_2 \right) \ldots \left(|0\rangle_n + e^{2\pi i\, 0.j_1\, j_2 \ldots j_n} |1\rangle_n \right)$$

Implementierung QFT

$$|j\rangle \;\mapsto\; 2^{-\frac{n}{2}} \left(|0\rangle_1 + e^{2\pi i\, 0.j_n}|1\rangle_1\right) \left(|0\rangle_2 + e^{2\pi i\, 0.j_{n-1}j_n}|1\rangle_2\right) \ldots \left(|0\rangle_n + e^{2\pi i\, 0.j_1\, j_2 \ldots j_n}|1\rangle_n\right)$$

Der Schaltkreis, der diese Operation erzeugt, muss Qbit ×1 die bedingte Phase $e^{2\pi i\, 0.j_n}$ geben (bedingt auf Zustand von Qbit n), Qbit zwei hat schon 2 bedingte Phasen $e^{2\pi i\, 0.j_{n-1}j_n}$,Qbit n $e^{2\pi i\, 0.j_1 j_2 \ldots j_n}$. Der folgende Schaltkreis macht das fast, er gibt allerdings noch Qbit 1 die für Qbit n bestimmte Phase, 2 für n-1 etc. Man braucht also noch viele Swaps.





1	$\left(\|0\rangle_1 + e^{2\pi i\, 0.j_1}\|1\rangle_1\right) \quad \|j_2 \ldots j_n\rangle \quad \text{da } e^{2\pi i\, 0.j_1} = (-1)^{j_1}$
2	$\left(\|0\rangle_1 + e^{2\pi i\, 0.j_1 j_2}\|1\rangle_1\right) \quad \|j_2 \ldots j_n\rangle$
4	$\left(\|0\rangle_1 + e^{2\pi i\, 0.j_1 \ldots j_n}\|1\rangle_1\right) \|j_2 \ldots j_n\rangle$
7	$\left(\|0\rangle_1 + e^{2\pi i\, 0.j_1 \ldots j_n}\|1\rangle_1\right) \left(\|0\rangle_2 + e^{2\pi i\, 0.j_2 \ldots j_n}\|1\rangle_2\right) \|j_3 \ldots j_n\rangle$

Anzahl Schritte QFT?
$=n+(n-1)+\ldots+1=n*(n+1)/2 \sim O(n^2)$
<<FFT=$O(N\log(N))=O(2^n * n)$

Nebenbemerkungen

Auf welchem Ausgangzustand führen wir denn nun die QFT aus?

$$|j\rangle \overset{\text{QFT}}{\mapsto} \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i\, j\cdot k/N} \cdot |k\rangle$$

Wir nehmen ein Targetregister mit n Qbits dazu und initialisieren

$$|0\rangle|0\rangle_T \overset{Hadamard}{=} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|0\rangle_T$$

Dann wird $F_N(j) = a^j \bmod N$ auf das Target angewendet

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|0\rangle_T \overset{F_N(j)}{=} \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|a^j \bmod N\rangle_T$$

Jetzt QFT

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle|a^j \bmod N\rangle_T \overset{QFT}{\mapsto} \frac{1}{N} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} e^{2\pi i\, j\cdot k/N} |k\rangle|a^j \bmod N\rangle_T$$

Wenn wir jetzt $|k\rangle$ messen, bekommen wir mit hoher Wahrscheinlichkeit ein Vielfaches von N/r, denn der Koeffizient von $|k\rangle$ ist dann besonders groß:

$$\text{Koeffizient} = \sum_{j=0}^{N-1} e^{2\pi i\, j\cdot k/N} |a^j \bmod N\rangle_T \overset{k=N/r}{\longrightarrow} \sum_{j=0}^{N-1} \boxed{e^{2\pi i\, j/r} |a^j \bmod N\rangle_T}$$

$\square_j$

$\square_j$ Periodisch in r!

Stolze 8.3.3, ab (8.39)