Quantum Factorization

Chapter 1

Introduction

1.1 First Section

1.1.1 First Subsection

Chapter 2

Quantum Computing Foundation

2.1 First Section

Chapter 3

Arithmetische Operation auf Qubits ausführen - Die QInteger Library

3.1 Überblick

3.2 Addition

Chapter 4

Der Weg zu Shor

4.1 Überblick

In diesem Kapitel werden wir uns die notwendigen Konzepte und Ideen hinter dem quantenbasierten Teil von Shor's Algorithmus anschauen. Dabei starten wir beim simplen Konzept des "Phase Kickback"s, schauen uns dann die darauf basierende Phase Estimation an, bevor wir dann deren Anwendung in Period Finding anschauen. Zum Schluss werden wir uns dann die komplette Implementation vom quantenbasierten Teil von Shors Algorithmus anschauen und überprüfen.

4.2 Phase-Kickback

Beginnen wir den Abschnitt mit einer Frage: Wenn wir eie kontrollierte Operation ausführen, sollte sich das Control-Qubit eigentlich nicht ändern, oder? In diesem Abschnitt werden wir sehen, dass dies überraschender Weise nicht so ist. Dafür schauen wir uns das CNOT-Gate an. Was passirt, wenn wir CNOT auf zwei Qubits im State $|+-\rangle$ anwenden, mit dem ersten Qubit als Control-Qubit? Zuerst haben wir $|+-\rangle = |00\rangle - |01\rangle + |10\rangle - |11\rangle$, nachdem wir das CNOT anwenden bekommen wir den State $|00\rangle - |01\rangle - |10\rangle + |11\rangle = |--\rangle$. Überraschenderweise stellen wir fest, dass sich das Control-Qubit verändert hat, während das Ziel-Qubit gleich blieb. Was ist passiert? Nehmen wir das CNOT-Gate auseinander: Das CNOT-Gate ist eigentlich nichts anderes als eine kontrollierte Version vom X-Gate. Was passiert wenn wir das X-Gate auf den $|-\rangle$-State anwenden? $X|-\rangle = -|0\rangle + |1\rangle = -|-\rangle = (-1) * |-\rangle$. Hier können wir sehen, dass $|-\rangle$ ein Eigenvektor des X-Gates mit Eigenwert -1. Das heisst, der State des Qubits ändert sich nicht, es wird nur die Phase mit dem Eigenwert multipliziert. Da wir nur ein einzelnes Qubit anschauen, hat das keine Auswirkung, da die Phase global ist und wir dehalb keinen Unterschied feststellen können. Wenn wir aber die Operation kontrolliert durchführen, wird diese Phase nur in den States sichtbar, in der die Operation durchgeführt wird,

sprich in den States, wo das Control-Qubit im State $|1\rangle$ ist. Dies konnten wir vorher beim CNOT-Gate beobachten. Schauen wir uns nun mal ein generelleres Gate an. Sagen wir, wir nehmen das Gate U mit einem Eigenvektor $|\psi\rangle$ und dem Eigenwert λ. Nehmen wir jetzt ein Qubit q_c im State $\alpha|0\rangle + \beta|1\rangle$, n Qubits $q_0...q_{n-1}$ im State $|\psi\rangle$, und führen ein kontrolliertes U auf die Qubits $q_0...q_{n-1}$ mit Kontroll-Qubit q_c durch:

$$(\alpha|0\rangle + \beta|1\rangle)|\psi\rangle \xrightarrow{\text{C-U}} \alpha|0\psi\rangle + \beta|1\rangle * U|\psi\rangle = (\alpha|0\rangle + \lambda\beta|1\rangle)|\psi\rangle$$

Das Ziel-Qubit verändert sich nicht, es ist ja ein Eigenvektor, dafür sehen wir, dass der Eigenwert in die Phase des Kontroll-Qubit gekickt wird. Daher kommt der Name "Phase Kickback". Wir werden in der nächsten Sektion sehen, wie dieser Effekt ausgenutz werden kann, um den Eigenwert eines Operators abzuschätzen.

4.3 Phase Estimation

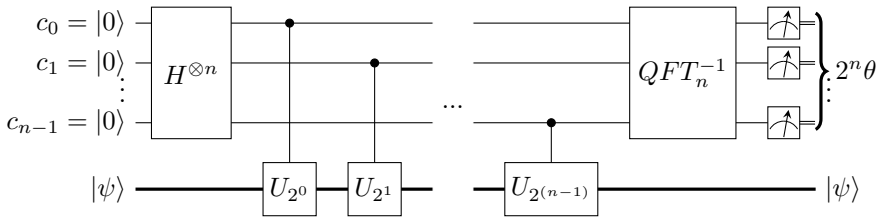
Verschiedene Quanten-Algorithmen basieren darauf, den Eigenwert eines Operators zu einem Eigenvektor abzuschätzen. Dazu benutzen wir Phase-Kickbacks, um den Eigenwert in ein Quantum-Register in der Fourier-Basis zu schreiben, welches wir dann mit der inversen Quanten-Fouriertransformation in die binäre Basis zurückrechnen. Dazu können wir die Anzahl Qubits variieren, um die Präzision der Approximation festlegen. Besser gesagt gibt der Algorithmus zum Eigenwert $\lambda = e^{2i\pi\theta}$ die Zahl $2^n\theta$ zurück, wobei n die Anzahl Qubits des Zählerregisters ist, die für bessere Präzision erhöht werden kann.
Um zu verstehen, wie dieser Algorithmus funktioniert, erinnern wir uns zuerst nochmals, wie eine Zahl in der Fourierbasis aussieht. Dafür benutzen wir nochmals die Bloch-Kugel. Wir erinnern uns, dass für die Zahl x in der Fourierbasis mit n Qubits das k-te Qubit um $\frac{2^k x}{2^n}$ um die Z-Achse gedreht wird. Das heisst, es befindet sich im Zustand $\frac{1}{\sqrt{2}}(|0\rangle + e^{2i\pi\frac{2^k x}{2^n}}|1\rangle)$. Wir machen jetzt die Beobachtung, dass wir mit Hilfe von Phase-Kickback das gesuchte θ in der Fourierbasis in die Kontrollqubits schreiben können, da der Phase-Kickback nichts anderes macht, als das Kontrollqubit auf die selbe Art und Weise zu rotieren. Schauen wir uns mal an, was passert, wenn wir das kontrollierte U 2^k mal anwenden:

$$\begin{aligned}
\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|\psi\rangle \xrightarrow{(C-U)^{2^k}} \frac{1}{\sqrt{2}}(|0\psi\rangle + |1\rangle * U^{2^k}|\psi\rangle) \\
= \frac{1}{\sqrt{2}}(|0\rangle + (e^{2i\pi\theta})^{2^k}|1\rangle)|\psi\rangle \\
= \frac{1}{\sqrt{2}}(|0\rangle + e^{2i\pi 2^k\theta}|1\rangle)|\psi\rangle \\
= \frac{1}{\sqrt{2}}(|0\rangle + e^{2i\pi\frac{2^k(2^n\theta)}{2^n}}|1\rangle)|\psi\rangle
\end{aligned}$$

Dies entspricht genau dem k-ten Qubit der Repräsentation von $2^n\theta$ in der Fourierbasis. Das heisst, wenn wir für jedes Qubit im Zählerregister mehrmals ein kontrolliertes U anwenden, können wir einen Zustand kreieren, welcher der Zahl $2^n\theta$ in der Fourierbasis entspricht. Wenden wir dann die inverse Fouriertransformation an, können wir die Zahl $2^n\theta$ im Zählerregister ablesen. Falls $2^n\theta$ keine ganze Zahl ist, dann bekommen wir im Zählerregister eine Superposition, wobei eine Zahl wahrscheinlicher ist, je näher sie am echten Wert ist.

Algorithmus

1. Initialisiere zwei Quantenregister, das Zählerregister und das Eigenstate-Register, und setze das Eigenstate-Register auf den gewünschten Eigenstate ψ.
2. Wende $H^{\otimes n}$ auf das Zähler-Register an, um es auf $|+\rangle^{\otimes n}$ zu setzen.
3. Für das i-te Bit im Zählerregister, wende das kontrollierte U mit c_i als Kontroll-Qubit 2^i Mal auf den Eigenstate an.
4. Wende die inverse Quantenfouriertransformation auf das Zählerregister an, um die Approximation in die binäre Basis umzurechnen.
5. Miss das Zählerregister, um die Abschätzung abzulesen.



4.4 Period Finding

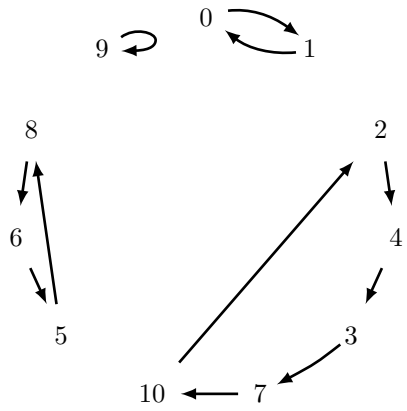
Gegeben sei eine Funktion $f : S \to S$ mit $S \subset \mathbb{Z}$, welche sich auf einem Quantencomputer implementieren lässt, und ein Wert $x \in S$. Wir versuchen nun, die kleinste Zahl $r \in \mathbb{N}$ zu berechnen, so dass $f^r(x) = x$ gilt. In anderen Worten: Sei $f_x(i) = f^i(x)$. Wir wollen nun die Periode von f_x zu berechnen.

Wir haben gesagt, unsere Funktion soll auf einem Quantencomputer implementierbar sein. Daraus folgt bereits, dass f bijektiv ist: Falls es ein a und ein b mit $f(a) = f(b) = c$ gibt, dann lässt sich $f^{-1}(c)$ nicht berechnen, was im Wiederspruch zur Reversibilität steht. Daraus folgt, dass f injektiv ist. Gleichzeitig müssen deshalb $|S|$ verschiedene Bilder von f existieren, damit jeder Wert ein eigenes Bild hat. Unsere Funktion permutiert die Elemente in S. Schaut man sich diese Permutation als Graph an, so hat jeder Knoten einen

Eingangs- und einen Ausgangsgrad von 1. Dies ist jedoch nur möglich, wenn der Graph eine Vereinigung disjunkter Zyklen ist. Dies bedeutet auch, dass man S in verschiedene Teilmengen $S_0, S_1, ...$ aufteilen kann, so dass jede dieser Teilmengen ein einzelner Zyklus des Graphen bildet. Sei nun $x \in S_i$. Da S_i ein Zyklus bildet, gilt $f^{|S_i|}(x) = x$. Gleichzeitig kann kein $r \in \mathbb{N}$ mit $r < |S_i|$ existieren, so dass $f^r(x) = x$ gilt, denn sonst hätte unser Zyklus nur $r < |S_i|$ Elemente. Wir wollen nun also für ein $x \in S_i$ die Grösse $|S_i|$ finden.

Als Beispiel nehmen wir mal $g : A \to A$ mit $A = \mathbb{Z}/11\mathbb{Z}$, $g(x) = -x^3 + 1$. Man kann zeigen, dass $x^3 \pmod p$ bijektiv ist, falls $p \equiv 2 \pmod 3$. Somit ist auch f bijektiv. Wenn wir den Graphen anschauen, dann sehen wir die einzelnen Zyklen: $A_0 = \{0, 1\}$, $A_1 = \{2, 3, 4, 7, 10\}$, $A_2 = \{5, 6, 8\}$ und $A_3 = \{9\}$. Wir sehen nun, dass $f^1(9) = 9$, $f^3(8) = 8$, $f^5(2) = 2$ etc.

TODO Beispiel $f : \mathbb{Z}/11\mathbb{Z} \to \mathbb{Z}/11\mathbb{Z}$, $f(x) = -x^3 + 1$



Die Frage ist nun, wie können wir effizient die Grösse der Teilmenge finden, in der x sich befindet. Dafür müssen wir den Operator f genauer betrachten. Was passiert, wenn wir dem Operator eine Superposition der Zahlen in S_i übergeben? Seien $r = |S_i|$, $x_0, x_1, ..., x_{r-1}$ die Zahlen in S_i, so dass $f(x_j) = x_{(j+1)\%r}$, und U_f die Quantenoperation, die f implementiert. Schauen wir mal, was passiert, wenn wir U_f auf den Zustand $\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |x_j\rangle$ anwenden? Wir bekommen:

$$U_f(\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |x_j\rangle) = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |f(x_j)\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |x_{(j+1)\%r}\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |x_j\rangle)$$

Daraus schliessen wir, dass $\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} |x_j\rangle)$ ein Eigenstate von U_f mit Eigenwert 1 ist. Dieser Eigenwert ist nicht wirklich interessant. Wir können ihn aber interessanter machen, indem wir den einzelnen Summanden eine Phase mitgeben. Dazu konstruieren wir die Superposition $\frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} (e^{-2i\pi \frac{kj}{r}} |x_j\rangle)$ für ein $k < r$.

Was passiert, wenn wie U_f darauf anwenden?

$$U_f(\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}(e^{-2i\pi\frac{kj}{r}}\left|x_j\right\rangle)) = \frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}(e^{-2i\pi\frac{kj}{r}}\left|x_{(j+1)\%r}\right\rangle) =$$
$$\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}(e^{-2i\pi\frac{k(j-1)}{r}}\left|x_j\right\rangle) = e^{2i\pi\frac{k}{r}}(\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}(e^{-2i\pi\frac{kj}{r}}\left|x_j\right\rangle))$$

Auch hier haben wir wieder einn Eigenvektor, aber mit einem interessanterem Eigenwert, nämlich $e^{2i\pi\frac{k}{r}}$, denn r ist im Eigenwert enthalten. Wir machen auch die Beobachtung, dass unser Eigenstate von vorher $(\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}\left|x_j\right\rangle)$ auch von der Form ist, die wir gerade analysiert haben, einfach mit $k = 0$. Falls wir jetzt irgendwie einen State von der Form $\frac{1}{\sqrt{r}}\sum_{j=0}^{r-1}(e^{-2i\pi\frac{kj}{r}}\left|x_j\right\rangle)$ erzeugen können, könnten wir mit Hilfe der Phase Estimation der Quotient $\frac{k}{r}$ abschätzen. Die Frage ist, wie können wir solch einen State generieren?