Die Überlegenheit von Quantenalgorithmen
Lektion 3.1 – Der Bernstein-Vazirani-Algorithmus

Mathematische Betrachtung – Arbeitsblatt 1

Der Bernstein-Vazirani-Algorithmus, entwickelt im Jahr 1993, baut auf dem Deutsch-Algorithmus auf und zeigt, wie Quantencomputer bestimmte Probleme effizienter lösen können als klassische Computer.

In Lektion 1 hast du bereits eine Aufgabe kennengelernt, bei der ein geheimer Bitstring mit Ja/Nein-Fragen Bit für Bit bestimmt werden musste. Ein klassischer Algorithmus benötigt dafür im ungünstigsten Fall eine Abfrage pro Bit. Für einen geheimen Bitstring mit *n* Bits sind also *n* Abfragen notwendig. Der Bernstein-Vazirani-Algorithmus schafft dies mit nur einer einzigen Quantenabfrage, unabhängig von der Länge des Bitstrings.

In dieser Lektion lernst du anhand eines Beispiels mit drei Bits, wie dieser Quantenvorteil entsteht. Anschließend überträgst du das Prinzip auf beliebige Bitlängen.

Das Problem

Stell dir eine Blackbox vor, die durch eine Funktion f_a beschrieben wird. Diese Box ist besonders: Wenn man ihr eine Binärzahl mit n Bits gibt, liefert sie als Ausgabe entweder 0 oder 1. Die Funktion f_a ist dabei über einen feste, geheimen Binärstring $a = (a_1, a_2, \ldots, a_n)$ definiert, wobei jedes a_i entweder 0 oder 1 ist.

Um den Funktionswert $f_a(x)$, zu berechnen, wird folgender Ablauf genutzt[1]:

- Man gibt eine n-stellige Binärzahl $x = (x_1, x_2, \ldots, x_n)$ ein, wobei jedes x_i ebenfalls 0 oder 1 ist.
- Multipliziere jedes Eingabebit mit dem entsprechenden Bit von a: $a_1 \cdot x_1 + a_2 \cdot x_2 + \ldots + a_n \cdot x_n$
- Anschließend prüft man die Summe:
  - Ist die Summe gerade, gibt die Funktion 0 zurück.
  - Ist die Summe ungerade, gibt die Funktion 1 zurück.

A1 – Verwende den geheimen Binärstring $\boldsymbol{a = (1,0,1)}$ für den Fall $\boldsymbol{n = 3}$ und vervollständige die folgende Tabelle, indem du $\boldsymbol{f_a(x)}$ berechnest.

Eingabe x	$a \cdot x = a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3$	Ungerade	Gerade	Ausgabe $f_a(x)$
$(0,0,0)$	$1 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 = 0$		x	0
$(0,0,1)$	$1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 = 1$	x		1
$(0,1,0)$				
$(0,1,1)$				
$(1,0,0)$				
$(1,0,1)$				
$(1,1,0)$				
$(1,1,1)$				

[1] Diese Art der Addition nennt man Addition modulo 2. Dabei gilt: Gibt es insgesamt eine gerade Anzahl von Einsen, ist das Ergebnis 0. Gibt es eine ungerade Anzahl von Einsen, ist das Ergebnis 1.

ENTNOMMEN AUS QUANTEN-COMPUTING IM MINT-UNTERRICHT
Ein Projekt von Science on Stage Deutschland e.V.

Verständnis des Problems

Unser Ziel ist es, die geheime Binärzahl a, die in der Funktion $f_a(x)$ verwendet wird, herauszufinden. Dazu können wir der Funktion Fragen stellen, indem wir ihr verschiedene Binärzahlen $x = (x_1, x_2, \ldots, x_n)$ übergeben und beobachten, welche Ausgabe wir erhalten.

Die Klassische Lösung

A2 – Bestimme die minimale Anzahl an Abfragen, die nötig ist, um den Binärstring $\mathrm{a} = (\mathrm{a}_1, \mathrm{a}_2, \mathrm{a}_3)$. vollständig zu ermitteln. Begründe deine Antwort.

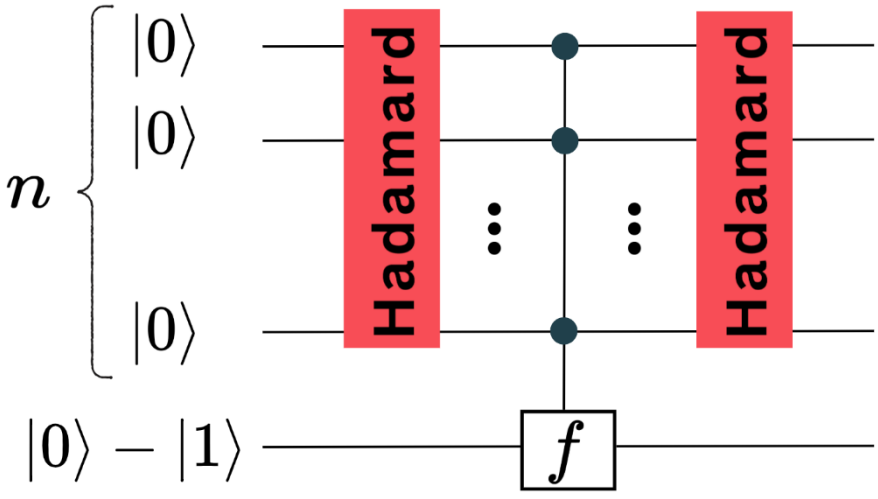
Hinweis: Denk an die Aufgabe zurück, bei der du eine versteckte Binärzahl durch Ja/Nein-Fragen herausfinden musstest. Wie hängt das mit dem aktuellen Problem zusammen?

A3 – Betrachte nun den Fall, dass der Binärstring n Bits hat: $a = (a_1, a_2, \ldots, a_n)$.

Bestimme, wie viele Abfragen in diesem allgemeinen Fall notwendig sind. Begründe deine Antwort.

Die Quantenlösung: Der Bernstein-Vazirani-Algorithmus

Anstatt die Funktion wie im klassischen Verfahren mehrfach abzufragen, bestimmt der Bernstein-Vazirani-Algorithmus den gesamten geheimen Bitstring mit nur einer einzigen Abfrage mithilfe des folgenden Quantenschaltkreises:



Er funktioniert in folgenden Schritten:

Schritt I: Erzeugung einer Superposition

Bereite im ersten Register n Qubits im Zustand $|0\rangle$ vor. Wende auf jedes dieser Qubits ein Hadamard-Gatter an. Dadurch entsteht eine Superposition aller möglichen Zustände der n Qubits:

$$H|0\rangle H|0\rangle \cdots H|0\rangle = \frac{1}{\sqrt{2^n}}(\ |00\cdots0\rangle + |00\cdots1\rangle + \cdots + |11\cdots1\rangle \ )$$

Schritt II: Anwenden der Funktion f_a und Einführen von Phasen

Wende die Funktion f_a auf den Zustand aus Schritt I an, indem du das zweite Register verwendest. Dabei wird jeder Basiszustand der Superposition mit einem Phasenfaktor $(-1)^0$ oder $(-1)^1$ multipliziert:

$$\frac{1}{\sqrt{2^n}}\left[(-1)^{f_a(00\cdots0)}|00\cdots0\rangle + (-1)^{f_a(00\cdots1)}|00\cdots1\rangle + \cdots + (-1)^{f_a(11\cdots1)}|11\cdots1\rangle \right]$$

Schritt III: Anwendung von Hadamard-Gattern

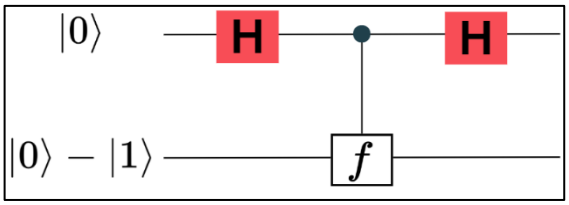
Wende erneut auf jedes Qubit im ersten Register ein Hadamard-Gatter an:

$$\frac{1}{\sqrt{2^n}}\left[(-1)^{f_a(00\cdots0)}H|0\rangle H|0\rangle \cdots H|0\rangle + \ (-1)^{f_a(00\cdots1)}H|0\rangle H|0\rangle \cdots H|1\rangle + \cdots \ (-1)^{f_a(11\cdots1)}H|1\rangle H|1\rangle \cdots H|1\rangle\right]$$

Hinweis: *Dass die Funktion diese Phasenverschiebungen im ersten Register einführt, ohne das zweite Register direkt zu verändern, ist ein zentrales Prinzip des Quantencomputings. Wer genauer verstehen möchte, warum und wie diese Phasen entstehen, kann das Arbeitsblatt zur Quantenauswertung von Funktionen (02.1_AB_QF) bearbeiten.*

Abschließende Messung: Eine Messung des ersten Registers liefert jetzt mit 100 % Wahrscheinlichkeit den geheimen Bitstring a. Im Gegensatz zur klassischen Methode, bei der mehrere Abfragen nötig sind, bestimmt der Quantenalgorithmus a mit nur einer Abfrage. Das Messergebnis ist dann der Zustand: $|a\rangle = |a_1 a_2 \cdots a_n\rangle$

A4 – Vergleiche den Schaltkreis des Deutsch-Algorithmus (siehe Abbildung rechts) mit dem Schaltkreis des Bernstein-Vazirani-Algorithmus und beantworte die folgenden Fragen:



Erläutere die Gemeinsamkeiten und Unterschiede zwischen den beiden Algorithmen. Wie hängt der Deutsch-Algorithmus mit dem Bernstein-Vazirani-Algorithmus zusammen?

Gemeinsamkeiten	Unterschiede