Freie Universität Berlin
Seminar über Algorithmen für Quantencomputer
Sommersemester 2002

Quantensuchalgorithmen

Reinhardt Karnapke
karnapke@inf.fu-berlin.de

Simon Rieche
rieche@inf.fu-berlin.de

Inhaltsverzeichnis

1 Klassische Suchalgorithmen

Die Laufzeit von klassischen Suchalgorithmen hängt davon ab, ob die Daten sortiert sind, oder nicht.
Bei unsortierten Daten hat man im schlimmsten Fall n Vergleiche, im Durchschnitt n/2.
Deshalb arbeitet man meistens mit sortierten Daten.

1.1 Laufzeiten

Beim nachträglichen sortieren hat man unterschiedliche Laufzeiten, ja nach Algorithmus.
z.B.

- Bubblesort O(n2),
- Quicksort O(n log n)
- etc...

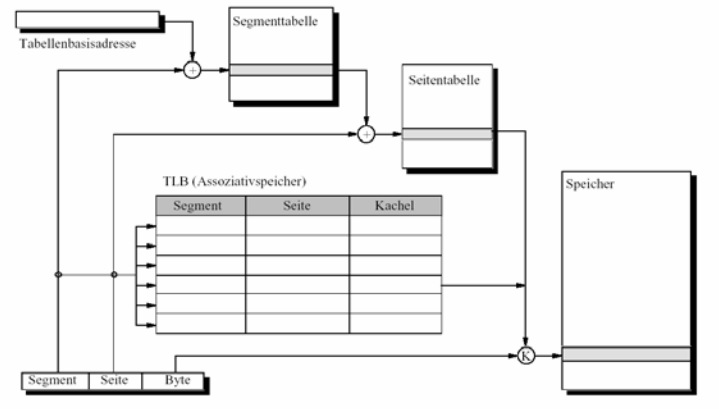
Also brauchen klassische Suchalgorithmen in Software lange, entweder beim Suchen in unsortierten Listen oder beim sortierten einfügen in eine Liste. .

2 Assoziativspeicher

Assoziativspeicher „realisieren“ Suchalgorithmen mittels Hardware. In Assoziativspeicher werden Daten unsortiert eingetragen, und unsortiert belassen. Sobald man nun Daten sucht, wird der Schlüssel angelegt, und **gleichzeitig** mit allen Einträgen verglichen. Wird das gesuchte Datum identifiziert, wird es sofort am Ausgang angelegt.
Suchen im Assoziativspeicher benötigt zwar n Vergleiche, diese geschehen jedoch gleichzeitig. Damit diese Vergleiche gleichzeitig stattfinden können benötigen wir jedoch n Vergleicher. Dies ist sehr teuer, und wird heutzutage nur in extrem geschwindigkeitskritischen Systemen wie z.B. Routern eingesetzt.
Die Vorteile und Nachteile sind dabei:

- Vorteil
  Ein Datum kann an beliebiger Stelle stehen
- Nachteil
  Hoher Hardwareaufwand
  (jede Zeile ein Vergleicher)
  → nur für kleine Datenmengen realisierbar

Die folgende Abbildung zeigt einen Cache als Assoziativspeicher.



3 Fourier Transformation

Die Fourier Transformation wird hauptsächlich zur Entstörung von Signalen verwendet. Sie beruht auf der Tatsache, das man fast jede Funktion in eine Summe von Sinus- und Kosinusfunktionen zerlegen kann. Im Fourierraum können klassische Computer wesentlich schneller rechnen.

Jede 2π-periodische Funktion ist darstellbar als:

$$f(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos kt + b_k \sin kt)$$

Die Umwandlung ist zwar aufwendig, der Geschwindigkeitsgewinn im Fourierraum ist jedoch so groß, dass man den Aufwand in kauf nimmt.
Die Hadamard-Transformation ist der Fourier-Transformation sehr ähnlich. Sie ist genau genommen eine abgewandelte Form der Fouriertransformation.

4 Hadamard

Als Erinnerung an die letzten Vorträge noch mal die Hadamard Funktion

Hadamard Transformation:

$$\hat{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Anwendung:

$$\hat{H}|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)$$

$$\hat{H}|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)$$

5 Lov K. Grover

Er erfand in den Bell Labs von Lucent Technologies 1996 den Algorithmus „A fast quantum mechanical algorithm for database search“ auf den alle weiteren Varianten aufbauen.

6 Groveralgorithmus

Der Grover Suchalgorithmus beschreibt die Suche nach einem Element in einer Domäne binärer Funktionen, unter der Vorraussetzung, dass das gesuchte Element existiert und eindeutig ist. Die klassische Suche nach einem speziellen Element von n Elementen erfordert im Mittel n/2 Rechenoperationen.
Der Grover Suchalgorithmus auf einem Quantencomputer erfordert lediglich √n Suchschritte

6.1 Quanten-Orakel

„Quanten-Orakel“ durch Quanten-Gatter $U_{f\omega}$ (unitäre Transformation) implementieren:
$f_\omega(x) := 0$ (falls $f(x)\neq a$, also $x\neq\omega$)
$f_\omega(x) := 1$ (falls $f(x)=a$, also $x=\omega$)
$U_{f\omega}$: $|x\rangle|y\rangle \rightarrow |x\rangle|y\ f\omega(x)\rangle$
$U_{f\omega}$: $|x\rangle|0\rangle \rightarrow |x\rangle|f\omega(x)\rangle$
$|x\rangle$ ist ein n-Qubit Register (n so gewählt, daß $2^n \geq N$),
$|y\rangle$ ein einzelnes Qubit;

6.2 Grovers Idee

$U_{f\omega}$ auf Superposition aller 2n möglichen Eingaben anwenden

$$\rightarrow \quad \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left| x, f_\omega(x) \right\rangle$$

Amplitude des Vektors $|\omega,1\rangle$ erheblich verstärken, alle anderen Amplituden deutlich reduzieren (→ Grover-Iteration)
Messung des letzten Qubits:
→ liefert mit großer Wahrscheinlichkeit $|1\rangle$ als Ergebnis;
→ projiziert obigen Zustand auf $|\omega,1\rangle$

6.3 Der Algorithmus

1. Hadamard Transformation $\quad |\phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$

2. Vorzeichen verändern $\quad |\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle$

3. Um den Durchschnitt spiegeln
   $D_n = -H_n R_n^{-1} H_n$

4. Durchlaufe (π/4) √2n mal Stufe 2 und 3,

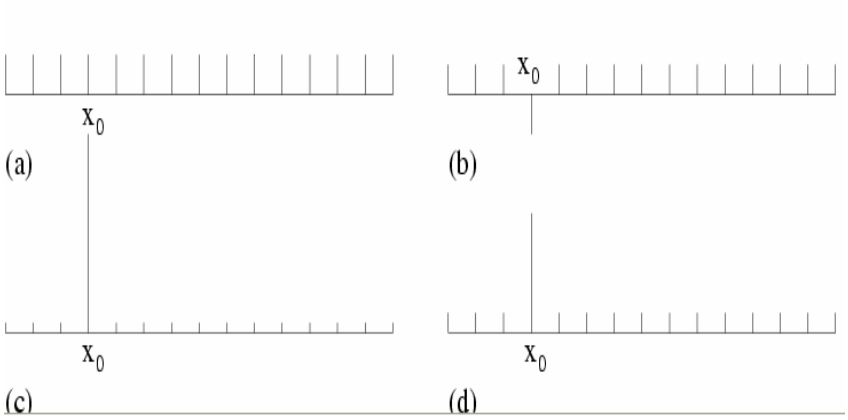
dass heißt $G_n=-H_nR_n^{1}H_nV_f$

5. Messe x, wenn $f(x_0)\neq 1$ von vorne beginnen

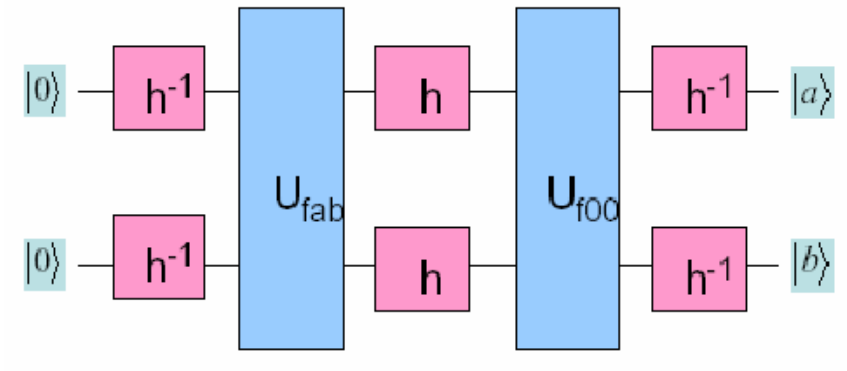
6.4 Beschreibung

Grovers Algorithmus beruht auf der Idee, den Quantencomputer wie einen Backofen zu benutzen (Stean).
Wir starten in einer Superposition mit gleicher Amplitude für alle Strings. Danach erhöhen wir die Wahrscheinlichkeit des korrekten Ergebnisses und verringern die der falschen Ergebnisse.

Nach $\frac{\pi}{4}\sqrt{n}$ Schritten ist die Wahrscheinlichkeit des gesuchten Strings genau 1. In diesem Moment muss man messen, denn danach sinkt die Wahrscheinlichkeit wieder.
Zum Glück gibt es einen Weg, den genauen Moment zu berechnen, in dem die Messung durchgeführt werden muss.
Der Beweis, dass der Algorithmus funktioniert ist simpel, zu verstehen warum er funktioniert ist es nicht.
Die folgende Abbildung veranschaulicht den Algorithmus:



6.5 Implementierung mit Quantengatter



Man kann anstelle der Hadamard Transformation auch jeden anderen unitären Operator benutzen

6.6 Grover-Laufzeit

Der Grover Algorithmus benötigt O(√n) Schritte zum finden des Ergebnisses (klassisch O(n/2)).
Wenn mehr Einträge das Kriterium erfüllen als nicht, findet der G-Algorithmus das Ergebnis in einem Schritt.
Damit ist Grovers Algorithmus potentiell eine Methode, RSA oder DES etc. zu knacken.

Die folgende Tabelle Zeit die Laufzeiten:

problem	classic	quantum
decision	Θ(N/t)	Θ(√(N/t))
search	Θ(N/t)	Θ(√(N/t))
counting	Θ(N)	Θ(√(t(N-t)))
approximation	Θ(N/(ε2t))	Θ((1/ε) √(N/t))

N = Anzahl der Einträge
ε = die Genauigkeit,
t= die Anzahl der richtigen Lösungen

6.7 Beispiel Grover

Beispiel: Gegeben sei eine Telefonnummer aus
einem Telefonbuch mit 2 Millionen Teilnehmern.

Eine klassische Suche würde im Mittel etwa **1 Millionen Rechenoperationen** erfordern.

Ein Quantencomputer benötigt mit Hilfe des Grover Algorithmus nur etwa **1400 Rechenschritte** !

7 Variationen

Es gibt viele variierte Grover Algorithmen, die alle auf dem Original aufbauen.

- G-BBHT Algorithmus
- Quantensuchalgorithmus II
- Minimumsuche
- etc.

7.1 G-BBHT

Der G-BBHT Algorithmus wurde von Bøyer, Brassard, Høyer und Tapp entwickelt.
Er ist für Mengen, wo das Suchergebnis mindestens einmal vorhanden sein muss.

1. n-qubits durch Hadamard in Superposition bringen

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$$

2. (π/4) √(2n/t) mal die Iteration $G=-H_n R_n^{1} H_n V_f$
3. Register x messen
4.Wenn $f(x_0)=1$ fertig, sonst von vorne

7.2 Quantensuche II

Der Algorithmus ist für Daten mit $t \leq 3/4 * 2^n$, wobei t die Anzahl der Ergebnisse ist.

Der Algorithmus funktioniert nach folgendem Schema
0. Nimm m=1 und λ= 6/5
1. Wähle j_0 aus [1,...m] zufällig
2. j_0 Iterationen durchführen zum Zustand

$$\frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

3. x_0 messen
4. Wenn $f(x_0)=1$ dann Problem gelöst und Ende der Berechnung
5. Nimm m = min{λm,√2n} und beginne bei Schritt 1

Der Quantensuche II Algorithmus läuft in

$$O(\sqrt{\frac{2^n}{t}})$$

Die Erfolgswahrscheinlichkeit liegt bei 84%.

7.3 Suche in sortierten Listen

Die Suche in sortierten Listen wird durch Quantenalgorithmen nur unwesentlich beschleunigt. Also arbeiten wir entweder mit sortierten Listen klassisch oder mit unsortierten Listen auf Quantencomputern

7.4 Minimumsuche

Bei sortierten Listen ist die Minimalsuche trivial.
Bei unsortierten Listen wird beim Quantencomputer der Algorithmus nach durch eine Abwandlung des G-BBHT Algorithmus

$22{,}5\sqrt{n} + 1{,}4\log_2(n)$ Laufzeit gestoppt.