SEMINARVORTRAG ZUM THEMA GROVER ALGORITHMUS

1 ZIELSETZUNG

Ziel des Vortrages ist es, einen Überblick über die Funktionsweise, Einsatzgebiete und Grenzen des Grover Algorithmus zu bieten. Zuerst soll der „Performancegewinn“ am Beispiel der Suche in einer unsortierten Datenbank aufgezeigt werden. Danach soll ein knapper Überblick über die Relevanz von Suchproblemen in der theoretischen Informatik gegeben werden und die besondere Stellung des Grover Algorithmus in dieser Problematik verdeutlicht werden. Als nächstes wird die detaillierte Funktionsweise des Algorithmus erklärt. Dies soll mit Bezugnahme auf die Vorhergehenden Vorträge geschehen. Als nächstes soll ein Beispiel, was auf einem klassischen Computer implementiert ist, die Funktionsweise des Algorithmus anschaulich darstellen. Falls am Ende noch Zeit bleibt, soll erklärt werden wie man die Anzahl der Lösungen a Priori bestimmen kann.

INHALTSVERZEICHNIS

2 SUCHPROBLEM

2.1 KLASSISCH

Das Finden von Einträgen in einer unsortierten Datenbank kann unter Umständen viel Zeit in Anspruch nehmen. Angenommen man hat einen einzelnen klassischen Computer und möchte ein nicht indiziertes Telefonbuch durchsuchen. Man stellt die Suchanfrage

```
SELECT * FROM `Adressen` WHERE `Telefonnummer`='$number' LIMIT 1
```

Die datenbankinterne Bearbeitung dieser Anfrage könnte in etwa so aussehen:

```
FOR EACH Item IN Adressen
   IF Item.Telefonnummer=$number THEN
      Return Item;
NEXT
```

Sei `N=SELECT COUNT(*) FROM `Adressen`` dann benötigt der oben beschrieben Algorithmus minimal 1 und Maximal N Durchläufe. Durchschnittlich also N/2 Durchläufe. Um Ausführungszeiten einzuordnen sind in der Informatik die sogenannten Landau-Symbole üblich. Das oben beschriebene System würde man die Kategorie $\mathcal{O}(N)$ einordnen. Dies bedeutet, dass die der Rechenaufwand linear (genauer affin) zu der Datenanzahl ist.

2.2 QUATENMECHANISCH

Quantenmechanisch kann man diese Suche, mittels einem im Jahre 1996 von Lov Grover entwickelten Algorithmus, in $\mathcal{O}\left(\sqrt{N}\right)$ Schritten ausführen.

A la Galileo bedeutet dies: Angenommen das erkennen einer Lösung dauert eine Sekunde. Falls ein klassischer Suchalgorithmus nun durchschnittlich eine halbe Stunde benötigt, um die richtige Lösung zu finden, braucht der quantenmechanische nur immer genau eine Minute.

Wie bei allen quantenmechanischen Algorithmen, liefert auch der Grover Algorithmus nur mit einer gewissen Wahrscheinlichkeit tatsächlich das gewünschte Ergebnis. Diese Wahrscheinlichkeit ist von N abhängig und nimmt für große N zu, so dass für $N=\infty$ eine Wahrscheinlichkeit von 1 erreicht würde. Die genaue Erklärung der Funktionsweise erfolgt später.

3 RELEVANZ VON SUCHPROBLEMEN

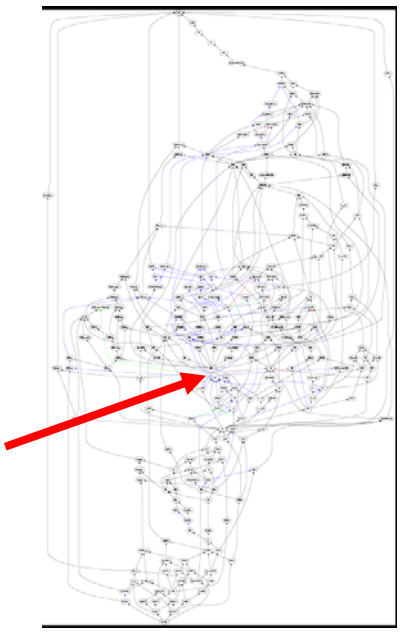


Abbildung 1 Komplexitätszoo

Genau wie es in der Physik einen Teilchenzoo gibt, so existiert in der theoretischen Informatik einen Komplexitätszoo der zurzeit 468 Komplexitätsklassen beheimatet. Betrachtet man den Stammbaum des Komplexitätszoos, wobei unten triviale Probleme und oben maximal Komplexe Probleme stehen, so liegt die Klasse NP, der **n**ichtdeterministisch aber in **p**olynomial zur Länge der Eingabe entscheidbaren Probleme, ungefähr auf halber Höhe der Komplexität. Liegt ein NP Problem vor kann man also in polynomialer Zeit testen, ob eine Antwort die Lösung zu dem Problem ist. Ein deterministischer Problemlösealgorithmus skaliert jedoch meist exponentiell zur Eingabelänge. Eine Untermenge der in NP liegenden Probleme sind die sogenannten NPC (auf Deutsch NP-Vollständigen, auf Englisch steht C für complete) Probleme. Diese zeichnen sich dadurch aus, dass sich alle Probleme aus NP mit maximal polonominalem Aufwand auf das Problem auf NPC reduzieren lassen. Also

$$L \in \mathrm{NP}C \Leftrightarrow L \in \mathrm{NP} \wedge \forall L' \in \mathrm{NP} : L' \preceq_p L \,.$$

Ein Beispiel für ein NPC Problem ist das Rucksackproblem. Bei dem Rucksacksackproblem hat man eine bestimmte Anzahl (N) von Gegenständen mit definiertem Gewicht und Nutzwert. Das Maximalgewicht des Rucksacks ist ebenfalls begrenzt. Nun soll der Nutzwert optimiert werden, ohne dass das Maximalgewicht überschritten wird. Erstaunlicher Weise gib es, abgesehen vom Aussortieren (*vgl. Algorithmus von Nemhauser und Ullmann*) derjenigen Lösungen, die mit höherem Gewicht geringere Nutzwerte bringen als andere Lösungen, nur die Möglichkeit alle Lösungen auszuprobieren. So lassen sich also alle NP Probleme auf Suchprobleme, die beispielweise und sogar optimal[1] mit dem Grover Algorithmus gelöst werden können zurückführen.

[1] Man kann zeigen, dass es keinen besseren Algorithmus als den Grover Algorithmus geben kann.

4 IMPLEMENTIERUNG DES GROVER ALGORITHMUS

Nach der Ausführlichen Beschreibung des Umfeldes des Grover Algorithmus folgt noch eine paar Vorbemerkung bis zur Vorstellung des eigentlichen Algorithmus.

Die Daten liegen nun, da es sich um einen Quantenalgorithmus handelt, nicht mehr in Form von elektrischen sondern von qBits vor. Mit n qBits kann man $N = 2^n$ Datenbankeinträge durch Basiszustände adressieren. Jeder möglicher Zustand kann nun als $|\psi\rangle = \sum_{i=0}^{N-1} a_i |i\rangle$ dargestellt werden. Zunächst nehmen wir an, dass wir eine Suchanfrage stellen, auf die genau eine Lösung hat. Diesen gesuchten Basiszustand nennen wir $|Z\rangle$. Des Weiteren, gehen wir davon aus, dass wir eine Funktion f von $|\psi\rangle$ die fast überall 0 ist und nur für $f(|Z\rangle) = 1$ ergibt, also die Lösung erkennt. Diese Funktion soll hinreichend schnell sein, da es sonst passieren kann, dass der Zeitvorteil, den man sich vom Grover Algorithmus verspricht durch das langsame f kompensiert oder sogar überkompensiert wird.

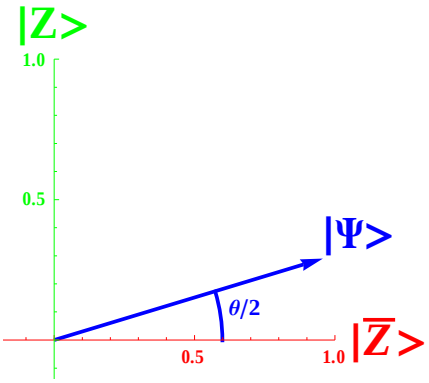
Als nächstes werden kurz die benötigten Operatoren vorgestellt, beziehungsweise es wird an sie erinnert.

4.1 EINFÜHRUNG DER BENÖTIGTEN OPERATOREN

4.1.1 HADAMARD GATTER

Das Hadamard-Gatter für ein einzelnes qBit sieht wie folgt aus: $\mathcal{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

Bei Anwendung auf ein qBit und erhält man $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} . (|1>) = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$; $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} . (|0>) = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$.



Bei der Messung ist jeder Basiszustand also gleich Wahrscheinlich. Für mehr qBit Systeme, erfolgt die Anwendung qBitweise ganz analog. Man schreibt aber $\mathcal{H}^{\otimes n}$ und bezeichnet dieses Gatter dann als Walsh Hadamard Gatter. Wendet man das Hadamard Gatter auf den Grundzustand an so erhält man eine gleichmäßige Superposition: $\mathcal{H}^{\otimes n}|0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle = \sqrt{\frac{N-1}{N}} |Z\rangle + \frac{1}{\sqrt{N}} |\bar{Z}\rangle$. Diesen Zustand wollen wir von nun an $|\Psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle$ nennen. Sei nun $|\bar{Z}\rangle = \frac{1}{\sqrt{N-1}} \sum_{\substack{i=0 \\ i \neq Z}}^{N-1} |i\rangle$ die Basis der nicht gesuchten Zustände. Man kann nun $|\Psi\rangle$ schreiben als $|\Psi\rangle = \frac{1}{\sqrt{N}} |Z\rangle + \sqrt{\frac{N-1}{N}} |\bar{Z}\rangle$. Mit der Definition $\cos\left(\frac{\theta}{2}\right) := \sqrt{\frac{N-1}{N}}$ folgt sofort aus $\sin^2 + \cos^2 = 1$, dass $\sin\left(\frac{\theta}{2}\right) = \frac{1}{\sqrt{N}}$. Dies ist in Abbildung 2 einfache Superposition veranschaulicht.

4.1.2 DAS ORAKEL

Der Operator U_f, das Orakel, ist wie folgt definiert:

$$U_f\left(\left|\xi\right\rangle\right) = (-1)^{f(\xi)}\left|\xi\right\rangle$$

f im Index steht hierbei für die in 4.1 eingeführte Funktion von $\left|\Psi\right\rangle$ die fast überall 0 ist und nur für $f\left(\left|Z\right\rangle\right) = 1$ ergibt.

Die Wirkungsweise des Orakels ist also wie folgt ist $\left|\xi\right\rangle = \left|Z\right\rangle$ wird die Phase um π verschoben.

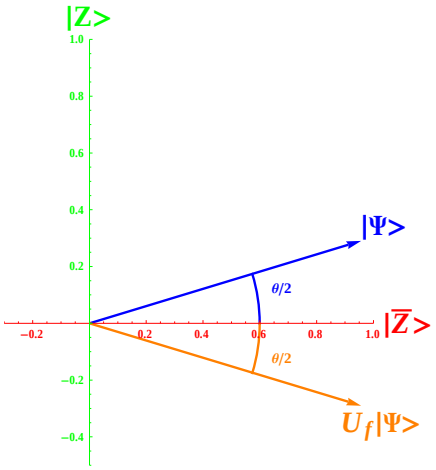


Abbildung 3 Zustand nach Anwenden des Orakels

Anschaulich kann man sich dies so vorstellen: Wendet man den Operator U_f auf den Zustand $\left|\Psi\right\rangle$ an so bewirkt dieser eine Spiegelung des Zustands $\left|\Psi\right\rangle$ am Vektor $\left|\bar{Z}\right\rangle$. Dies ist in Abbildung 3 veranschaulicht. Natürlich wird $\left|\Psi\right\rangle$ nicht um den Winkel π gedreht, weil nur die Amplitude bei einem qBit, bei qBit Z die Amplitude invertiert wurde. Dies ist grade eine Veränderung um $\frac{2}{\sqrt{N}}$, was nach Definition im vorherigen Abschnitt einer Drehung um Winkel θ entspricht. Ist

Beispielsweise n=2 und Z=3, dann ist $U_{f3}\left(\left|\xi\right\rangle\right) = \left(1 - 2\left|Z\right\rangle\left\langle Z\right|\right)\left|\xi\right\rangle$.

4.1.3 ROTATIONSOPERATOR

Der Rotationsoperator ist wie folgt definiert:

$$R = 2\left|0\right\rangle\left\langle 0\right| - 1$$

Alle Zustände außerdem dem Grundzustand werden Phasenverschoben, beziehungsweise um π gedreht. Um zum Bild (Abbildung 4) zurückzukehren, entspricht die Rotation angewandt auf $U_f\left|\Psi\right\rangle$ einer Spiegelung um den Vektor $\left|\Psi\right\rangle$. Zusammen mit 2 Hadamard Gates wird dieser Operator auch oft als Inversion um den Mittelwert bezeichnet, um dieses zu erklären wird aber eine Darstellungsform benötigt. Nämlich eine solche, die die Amplituden der Basiszustände darstellt. Darauf werden wir im nächsten Abschnitt zurückkommen. Zuerst wird der Ablauf des Grover Algorithmus beschrieben.

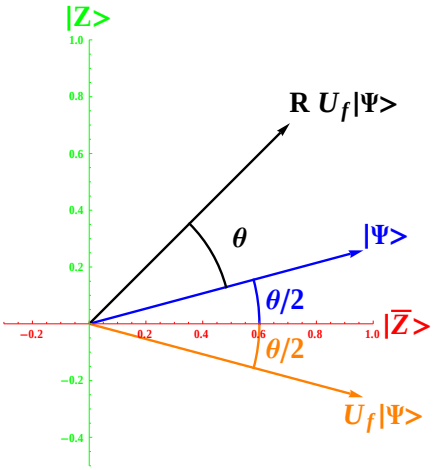


Abbildung 4 Graphische Darstellung einer Grover Iteration

4.2 ABLAUF

Jetzt folgt die genaue Beschreibung Grover Algorithmus. Dazu nehmen wir vorerst an, dass k Grover Iterationen notwendig sind.

4.2.1 ABLAUFPLAN

Der Ablaufplan sieht wie folgt aus:

```
i=0;
```

1. Anfangszustand $|0\rangle$ wird präpariert.
2. Gleichmäßige Superposition von $|\Psi\rangle$ durch Hadamard Transformation
3. For i < k
   a. Anwendung des Orakels → Vorzeichenwechsel bei der Amplitude des Lösungsanteils in $|\Psi\rangle$
   b. Hadamard Transformation
   c. Rotation
   d. Hadamard
   e. i++
4. Wahrscheinlichkeit für Messwert $|z\rangle$ bei Messung von $|\Psi\rangle$ hoch.

4.2.2 DISKUSSION

Man kann die Schritte 3 a bis d durch Einführung eines neuen Operators D signifikant Vereinfachen: $D := \mathcal{H}^{\otimes n} R \mathcal{H}^{\otimes n} = \mathcal{H}^{\otimes n}\left(2|0\rangle\langle 0| - 1\right)\mathcal{H}^{\otimes n} = \left(2|\Psi\rangle\langle\Psi| - 1\right)$. Als Übung wird kurz vorgerechnet wie D auf einen Allgemeinen Zustand $|\zeta\rangle = \sum_{k=1}^{N} \alpha_k |k\rangle$ wirkt:

$$D|\zeta\rangle = D\sum_{k=0}^{N-1} \alpha_k |k\rangle = 2|\Psi\rangle\langle\Psi|\sum_{k=0}^{N-1} \alpha_k |k\rangle - |\zeta\rangle$$

Setzt man nun $|\Psi\rangle = \frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i\rangle$ erhält man

$$2\frac{1}{\sqrt{N}}\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}\langle j|\sum_{i=0}^{N-1}|i\rangle\sum_{k=0}^{N-1}\alpha_k|k\rangle = \frac{2}{N}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}\alpha_k\delta_{jk} = \frac{2\sum_{l=0}^{N-1}\alpha_{k0}}{N}\sum_{i=0}^{N-1}|i\rangle$$

Mit der Abkürzung $\langle\alpha\rangle := \frac{1}{N}\sum_{l=0}^{N-1}\alpha_k$ folgt

$$D|\zeta\rangle = \sum_{k=0}^{N-1}\left(2\langle\alpha\rangle - \alpha_k\right)|k\rangle .$$

Schritt 3, im Allgemeinen auch Grover-Iteration genannt ist nun also als $G := DU_f$. Pro Grover Iteration wird die Amplitude für die $f(\Gamma) = 1$ gilt um $\frac{1}{\sqrt{N}}$ erhöht, beziehungsweise stellt die gesamte Grover Iteration eine Drehung um den Winkel θ dar.

4.2.3 BESTIMMUNG DER ANZAHL DER ITERATIONEN

Um der Frage nachzugehen wie viele Groer Iterationen notwendig sind, um den Zustand auf Z zu drehen stellt man folgende Überlegung an: $G^{\otimes k}|\Psi\rangle = \cos\left(\frac{2k+1}{2}\theta\right)|\bar{Z}\rangle + \sin\left(\frac{2k+1}{2}\theta\right)|Z\rangle$ Ziel ist es nun, dass $\sin\left(\frac{2k+1}{2}\theta\right)$ möglichst nahe an 1, das Argument an $\frac{\pi}{2}$, herankommt: $(2k+1)\theta = \pi \Rightarrow k = \frac{\pi-\theta}{2\theta} = \frac{1}{2}\left(\frac{\pi}{\theta}-1\right) = \frac{\pi}{4}\left(\frac{2}{\theta}-2\right) \approx \frac{\pi}{4}\left(\sqrt{N}-2\right)$ Dabei wurde die Näherung $\frac{1}{\sqrt{N}} = \sin\left(\frac{\theta}{2}\right) \approx \frac{\theta}{2} \Rightarrow \frac{2}{\theta} \approx \sqrt{N}$, die für große N sicherlich richtig ist durchgeführt. Der Algorithmus, muss wie oben motiviert das nahegelegenste ganzzahlige vielfache von $\frac{\pi}{4}\sqrt{N}$ Mal Angewandt werden, um die Wahrscheinlichkeit, bei einer Messung das richtige Ergebnis zu erhalten zu maximieren. Bei, relativ gesehen, großen Abweichungen zum nächsten Ganzzahligen vielfachen ist eine klassische Überprüfung des Messergebnisses sinnvoll.

4.3 DARSTELLUNG DER GROVER ITERATION ALS DREHMATRIX

Die obige Diskussion suggeriert, dass eine Grover Iteration als Drehung betrachtet und folglich der Grover Operator als Drehmatrix dargestellt werden kann. Dies verifiziert man mit einem einfachem nachrechnen.

```
G={{Cos[θ],-Sin[θ]},{Sin[θ],Cos[θ]}};Ψ={Cos[θ/2],Sin[θ/2]};

Print["G=",G];

Print["Ψ=",Ψ//MatrixForm];

Print["GΨ=",MatrixForm[G.Ψ]==MatrixForm[FullSimplify[G.Ψ]]];
```

$$G = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$\Psi = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$G\Psi = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right)\cos(\theta) - \sin\left(\frac{\theta}{2}\right)\sin(\theta) \\ \cos(\theta)\sin\left(\frac{\theta}{2}\right) + \cos\left(\frac{\theta}{2}\right)\sin(\theta) \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{3\theta}{2}\right) \\ \sin\left(\frac{3\theta}{2}\right) \end{pmatrix}$$

Hier wurde in Matrixdarstellung mit $\begin{pmatrix} |\bar{Z}\rangle \\ |Z\rangle \end{pmatrix}$ Darstellung gerechnet.

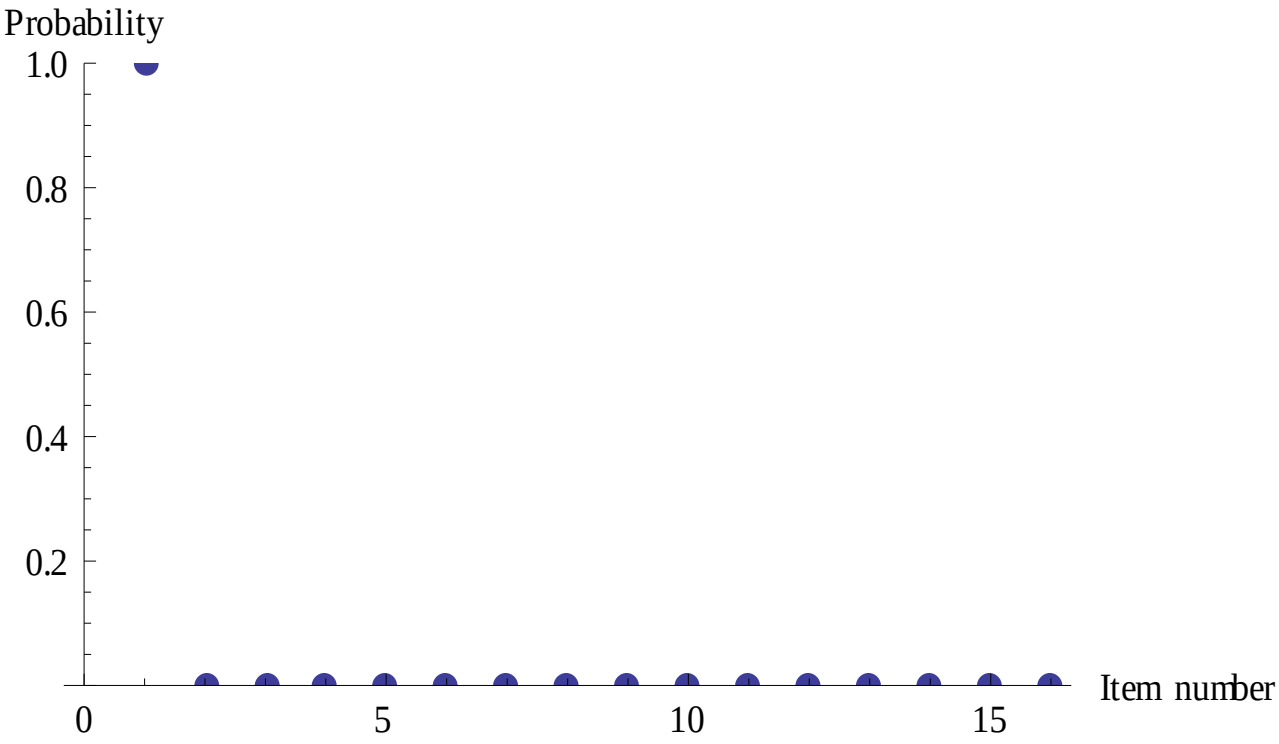
5 BEISPIELRECHNUNG

Der Algorithmus kann leicht mit Mathematika implementiert werden. Es folgt ein Beispiel (ohne Erklärung):

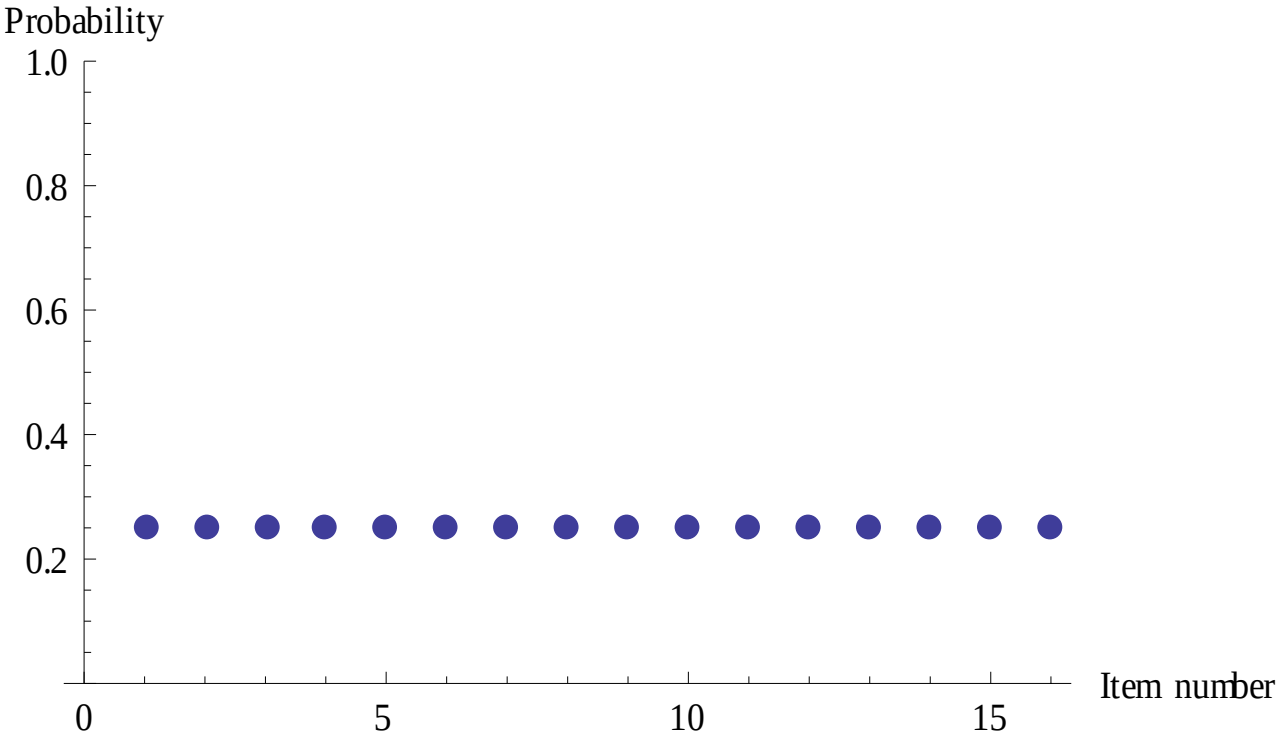
```
Needs["QDENSITY`Qdensity`"]

n=4;

  =2^n;

baseState[x_]:=ReplacePart[ZeroT[  ],1,x];

qtab:=Table[q_i,{i,1,N}];

  Ground=baseState[1];

H=HALL[n];

  =H.  Ground;

Z=RandomInteger[{1,  }];

f=baseState[Z];

U_f=DiagonalMatrix[(-1)^f];

R=2*DiagonalMatrix[baseState[1]]-IdentityMatrix[  ];

Dr=H.R.H;

G=Dr.U_f;

k=Max[IntegerPart[  /4 (Sqrt[  ]-1)],1]

Print["Z=",Z]
```

```
2

Z= 5
```

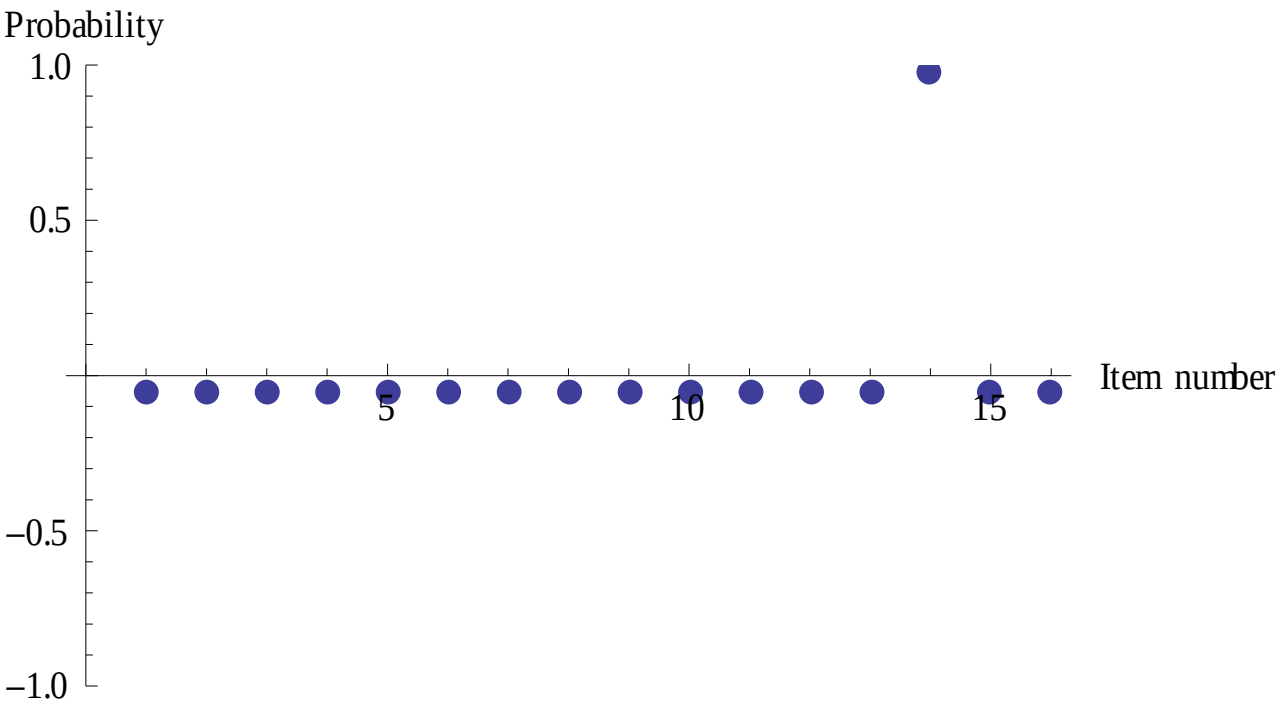
```
Plot1=ListPlot[  Ground,AxesLabel  {Item      number,Probability},PlotStyle
PointSize[0.025`],PlotRange  {0,1.`}]
```



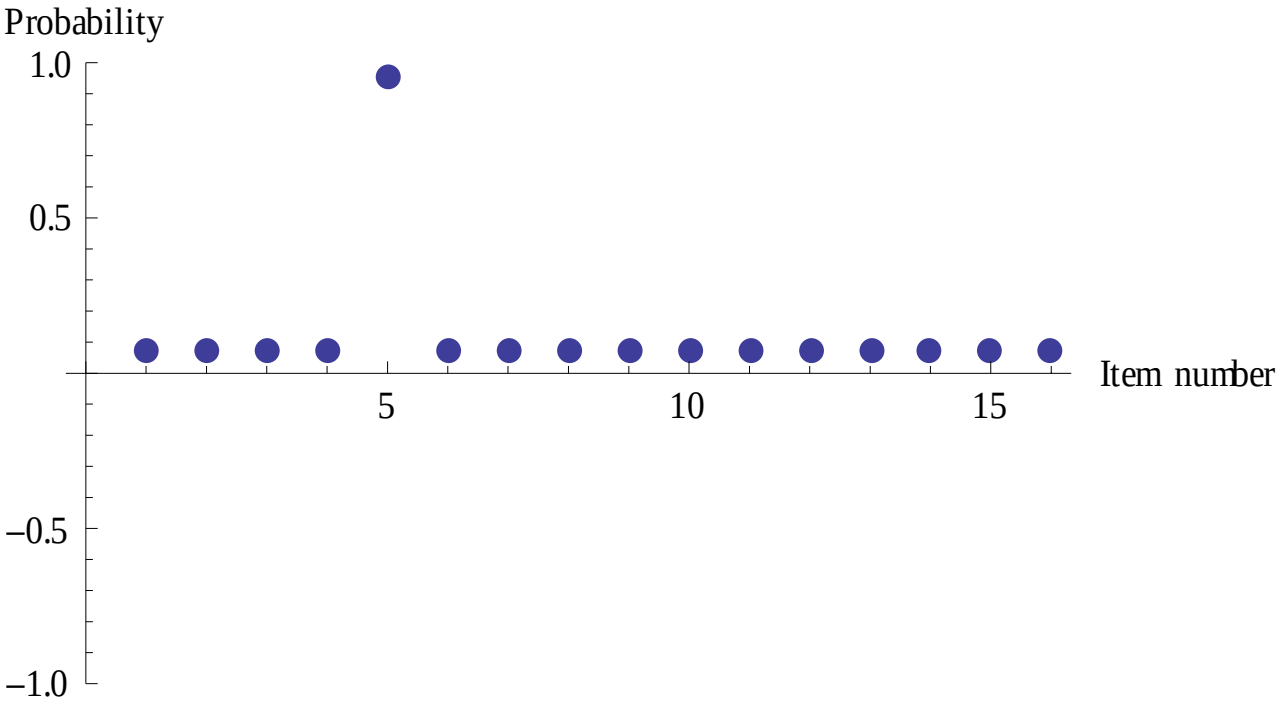

```
Plot1=ListPlot[  ,AxesLabel  {Item           number,Probability},PlotStyle
PointSize[0.025`],PlotRange  {0,1.`}]
```




```
Plot1=ListPlot[G.G.G.  ,AxesLabel  {Item     number,Probability},PlotStyle
PointSize[0.025`],PlotRange  {-1,1`}]
```




```
Plot1=ListPlot[MatrixPower[G,k].  ,AxesLabel  {Item
number,Probability},PlotStyle  PointSize[0.025`],PlotRange  {-1,1`}]
```



6 VERALLGEMEINERUNG AUF MEHRERE LÖSUNGEN

Die Verallgemeinerung des Grover-Algorithmus auf Suchprobleme mit mehreren Lösungen ist wie im Beispiel gezeigt relativ einfach. Grob gesprochen muss nur jedes N durch den Quotient aus N und M, wobei M für die Anzahl der Lösungen steht ersetzt werden. Die Schwierigkeit liegt allerding darin, M zu bestimmen. Dazu nutzt man die oben Angegebene geometrische Vorstellung aus. Mittels der Quanten Fourier Transformation, die im nächsten Seminar besprochen wird, kann die Periode der Lösungswiederholung und dadurch auch die Anzahl der Iterationen bestimmt werden. Alternativ könnte man dies auch dadurch realisieren, indem man von einer wahrscheinlichen Anzahl von Lösungen ausgeht, den Grover Algorithmus durchführt und später klassisch Prüft, ob die gefundene Lösung die richtige ist. Falls nicht muss man seine Annahme der Anzahl der Lösungen überprüfen. Dieses Verfahren ist allerdings sehr Zeitintensiv und es stellt nur einen geringen Vorteil zur klassischen Suche darstellt.

7 REFERENZEN

7.1 QUELLENVERZEICHNIS

Physical Review Letter Volume 79, Number 2 Quantum Mechanics Helps Searching for a Needle in a Haystack“ (Lov K. Grover)

http://www.theory.caltech.edu/people/preskill/ph219/#lecture

http://www.wikipedia.org

7.2 ABBILDUNGSVERZEICHNIS

Abbildung 2-4 wurden mit folgendem Mathematica-Code oder Teilen davon erzeugt:

```
Graphics[
  {
    {Thick,
     Blue,
     Arrow[{{0,    0},    {Cos[\[Theta]/2],
Sin[\[Theta]/2]}}],
     Text[
    Style["|\[CapitalPsi]>",  Large,  Bold],
{Cos[\[Theta]/2],
     Sin[\[Theta]/2]}, {-1, -1}],
     Text[
    Style["\[Theta]/2",    Bold,    Large],
{Cos[\[Theta]/2],
     Sin[\[Theta]/2]}, {4, 3}],
     Circle[{0, 0}, 0.6, {0, \[Theta]/2}]
     },
    {Thick,
    Orange,
     Arrow[{{0,   0},   {Cos[\[Theta]/2],   -
Sin[\[Theta]/2]}}],
     Text[
    Style["\!\(\*SubscriptBox[\"U\",
\"f\"]\)|\[CapitalPsi]>", Large,
     Bold],       {Cos[\[Theta]/2],       -
Sin[\[Theta]/2]}, {-0.5, 1}],
     Text[
    Style["\[Theta]/2", Bold,
     Large],       {Cos[\[Theta]/2],       -
Sin[\[Theta]/2]}, {4, -3}],
     Circle[{0, 0}, 0.6, {-\[Theta]/2, 0}]
   },
  {Thick,
    Black,
     Arrow[{{0,   0},   {Cos[3   \[Theta]/2],
Sin[3 \[Theta]/2]}}],
     Text[
    Style["R      \!\(\*SubscriptBox[\"U\",
\"f\"]\)|\[CapitalPsi]>",
     Large, Bold], {Cos[3 \[Theta]/2],
     Sin[3 \[Theta]/2]}, {-1, -1}],
     Text[
     Style["\[Theta]", Bold, Large], {Cos[3
\[Theta]/2],
      Sin[3 \[Theta]/2]}, {7, 7}],
      Circle[{0,  0},  0.5,  {\[Theta]/2,  3
\[Theta]/2}]
    }
   },
 Axes -> True,
 AxesLabel                                ->
{Style["|\!\(\*TagBox[OverscriptBox[\"Z\",
\"_\"],
Function[BoxForm`e$,
MatrixForm[BoxForm`e$]]]\)>",  Large,  Red,
Bold],
   Style["|Z>", Large, Green, Bold]},
 AxesStyle    ->    {Directive[Red,    Bold],
Directive[Green, Bold]},
 PlotRange -> {{-0.3, 1}, {-0.5, 1}}]
```