Chapter 17

Quantum Algorithm

Clearly, this dramatic change in the hardware's principles calls for an equally dramatic change in software.

17.1 The Deutsch algorithm

David Deutsch, in his attempt to support the many world interpretation of quantum mechanics, devised a simple decision problem which elucidates the power of quantum parallelism in quantum computation.[1] The problem is this: given a black box which computes a binary function $f : \{0, 1\} \to \{0, 1\}$ – decide whether f is constant or balanced. On a classical computer, one needs to call f twice in order to make the

[1] David Deutsch *Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer*, Proc. R. Soc. Lond. A**400** (1985), 97–117.

decision. On a quantum computer, one call proves to be enough.[2] The generalization of the Deutsch algorithm is the Deutsch-Josza algorithm[3]

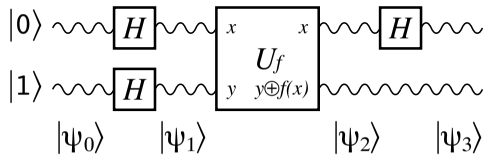

Figure 17.1: Quantum circuit for Deutsch's algorithm.

To wit:

$$|\psi_0\rangle = |01\rangle \tag{17.1}$$

$$\rightarrow |\psi_1\rangle = \left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \tag{17.2}$$

$$\rightarrow |\psi_2\rangle = \begin{cases} \pm\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) & \text{if } f(0)=f(1) \\ \pm\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) & \text{if } f(0)\neq f(1) \end{cases} \tag{17.3}$$

$$\rightarrow |\psi_3\rangle = \begin{cases} \pm|0\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) & \text{if } f(0)=f(1) \\ \pm|1\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) & \text{if } f(0)\neq f(1) \end{cases} \equiv \pm|f(0)\oplus f(1)\rangle\left(\frac{|0\rangle-|1\rangle}{\sqrt{2}}\right) \tag{17.4}$$

Evidently, the measurement of the first qubit determines unambigously whether f

[2] In the original formulation of Deutsch's algorithm, the successrate was 3/4. In the improved version, which was later found by Cleve, Ekert, Macchiavello and Mosca in *Quantum algorithms revisited*, Proc. R. Soc. Lond. A**454** (1998), 339-354, the successrate goes to one-hundred percent. We here present the improved version.

[3] David Deutsch and Richard Josza *Rapid solutions of problems by quantum computation*, Proc. R. Soc. Lond. A**439** (1992), 553

is constant or balanced. The quantum circuit determines a global property of $f(x)$, that is $f(0) \oplus f(1)$, requiring only one single "call" of f!

17.2 Shor's factoring algorithm

Given a large composite positive integer N, the problem is to find a non-trivial factor p, i.e. to find p in $N = pq$.

The most simple algorithm would be to try numbers 2 through $\sqrt{N}$. For N a n-bit number, this requires $O(\sqrt{N}) \sim 2^{n/2}$ steps, i.e. the algorithm is exponential.

With the fastest algorithm known to date – the number field sieve – the number of steps is still exponential, albeit of some reduced order $\exp[n^{1/3}(\log n)^{2/3}]$. Still one would need a few years to factorize a 200-digit number ...

With the Shor algorithm, the number of steps scales $O(n^3)$, and hence factorization becomes tractable. The algorithmus is nicely reviewed in "Quantum computation and Shor's factoring algorithm" by A. Ekert and R. Josza, Rev. Mod. Phys. **68** (1996), 733.

The Shor algorithm has a classical part, essentially a theorem from number theory, and a quantum part, essentially the determination of a period of a periodic function (which is a *global* property).

The classical part is contained in the following

Theorem For natural numbers N and a coprime, with $1 < a < N$, let r denote the period of the function

$$f_{a,N}(x) := a^x \bmod N\,, \tag{17.5}$$

i.e. $a^r \equiv 1 \mod N$. If r is even and $a^{r/2} \neq -1 \bmod N$, at least one of the numbers

$$p, q = \gcd(a^{r/2} \pm 1, N) \tag{17.6}$$

is a non-trivial factor of N.

Clearly, the congruence $a^r \equiv 1 \mod N$ is equivalent $a^r - 1 \equiv \mod N$, that is $a^r - 1 = k \cdot N$, where k is some integer. But $a^r - 1 = (a^{r/2} + 1) \cdot (a^{r/2} - 1)$. Hence if p divides N, and p is prime, it must be a factor of either $a^{r/2} + 1$ or $a^{r/2} - 1$.

The Shor algorithm reads

1. given N choose a randomly
2. if $\gcd(a, N) \neq 1$ return factor $p = \gcd(a, N)$ and stop.
3. else: find the order r of a modulo N
4. if r is even and $a^{r/2} \neq -1 \bmod N$ compute $\gcd(a^{r/2} \pm 1, N)$, test which of these is a factor of N, return the result and stop.
5. else: goto step 1.

and has the following properties

- all steps are easy, except the order finding (step 3), which is classically hard
- the algorithm is **probabilistic**. For unbiased choice of a, the probability to obtain r even and $a^{r/2} \neq -1 \bmod N$ is at least one-half.

The **quantum part** goes as follows. For the factorization of a n-bit number N, you need approx. $3n$ qubits. Divide the qubits into two sets, the x-register with approx. $2n$ qubits, and the f-register with n qubits.

Let M be a natural number $N^2 \leq M < 2N^2$ with $M = 2^L$, where L ist the number of qubits of the x-Register.

1. Prepare x register in state

$$\hat{H} \otimes \hat{H} \otimes \cdots \otimes \hat{H} |00\cdots0\rangle = \tfrac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle \,, \tag{17.7}$$

2. Prepare f-register in state

$$|0\rangle = |\underbrace{00\cdots0}_{n \text{ qubits}}\rangle \tag{17.8}$$

3. Call $f_{a,N}$

$$\hat{U}_f |\Psi_0\rangle = \tfrac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle |f_{a,N}(x)\rangle \tag{17.9}$$

4. Measure on f-register only, reading l, say (all readings are equally likely). The x-register is left in post-measurement state

$$|\psi_l\rangle = \tfrac{1}{\sqrt{A}} \sum_{j=0}^{A-1} |jr + l\rangle \,, \tag{17.10}$$

where r only depends on the initial choice of a, the off-set l being the value obtained in the f-measurement, and $M - r \leq l + (A-1)r < M$, or

$$A - 1 \leq \frac{M}{r} < A + 1 \,. \tag{17.11}$$

Note that, upon repetition with the same a, the value of l – and hence the final state $|\psi_l\rangle$ – will vary, but r remains the same.

5. Extract r by applying the quantum discrete Fourier transform (see below).

Applying QDFT to the x-Register, the x-Register state $|\psi_l\rangle$ is mapped

$$\hat{F}_M|\psi_l\rangle = \tfrac{1}{\sqrt{MA}} \sum_{y=0}^{M-1} \sum_{j=0}^{A-1} \exp\left\{2\pi i \frac{(jr+l)y}{M}\right\} |y\rangle := \sum \tilde{\psi}_y |y\rangle \tag{17.12}$$

with amplitudes

$$\tilde{\psi}_y = \frac{e^{2\pi i l y/M}}{\sqrt{MA}} \sum_{j=0}^{A-1} \exp\left\{2\pi i \frac{jry}{M}\right\} \tag{17.13}$$

Consider the case that r divides M exactly, such that $s := M/r$ is an integer (and hence $A = s+1$). Then

$$\tilde{\psi}_y = \frac{e^{2\pi i l y/M}}{\sqrt{MA}} \sum_{j=0}^{s} \exp\left\{2\pi i \frac{jy}{s}\right\} \tag{17.14}$$

$$= \frac{e^{2\pi i l y/M}}{\sqrt{MA}} \begin{cases} s & \text{if } y \text{ is a multiple of } s \\ 0 & \text{otherwise} \end{cases} \tag{17.15}$$

i.e. the Fouriertrafo of a state with period r is a state with period $s = M/r$,

$$= \tfrac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left\{2\pi i \frac{lk}{r}\right\} \left| k\frac{M}{r} \right\rangle \tag{17.16}$$

You may wonder whether $f_{a,N}(x)$ – for given $x < 2^m$ – can be computed efficiently. The answer is yes. Just expand x in binary notation,

$$x = x_{m-1}2^{m-1} + x_{m-2}2^{m-2} + \ldots + x_0 2^0 \tag{17.17}$$

you face

$$a^x = (a^{2^{m-1}})^{x_{m-1}}(a^{2^{m-2}})^{x_{m-2}} \cdots (a)^{x_0} \tag{17.18}$$

Each term in brackets has a large Exponent 2^j – but remember that you need this only $\mod N$. So you really only need a table of m numbers $a, a^2, a^4, \ldots, a^{2^{m-1}}$ (all mod N), in order to obtain $a^x \mod N$. By virtue

$$a^{2^j} \mod N = (a^{2^{j-1}} \mod N)^2 \mod N \tag{17.19}$$

you need only $m-1$ multiplications $\mod N$ to build that table. The computation of $y = a^x \mod N$ then proceeds by executing the algorithm

```
y=1
for j=1 to m-1
   if x_j=1 then f=a^{2^{j}}*f mod N
   else continue
return y
```

which requires at most $m \mod N$-multiplications, where each multiplication requires of order $(\log N)^2$ elementar operations. With m of order $\log(N)$, the complexity of computing $f_{a,N}(x)$ for goven x is $O((\log N)^3)$

17.3 Quantum discrete Fourier transform

The quantum discrete Fourier transform modulo M is a unitary transformation which maps basis vectors $|0\rangle, \ldots, |M-1\rangle$ as follows:

$$|x\rangle \mapsto \hat{F}_M|x\rangle = \tfrac{1}{\sqrt{M}} \sum_{y=0}^{M-1} \exp\left\{2\pi i \frac{xy}{M}\right\} |y\rangle \,. \tag{17.20}$$

A general state $|\psi\rangle = \sum_{x=0}^{M-1} \psi_x |x\rangle$ is mapped

$$\hat{F}_M|\psi\rangle = \sum_{y=0}^{M-1} \tilde{\psi}_y |y\rangle\,, \tag{17.21}$$

where

$$\tilde{\psi}_y = \tfrac{1}{\sqrt{M}} \sum_{x=0}^{M-1} \exp\left\{2\pi i \frac{xy}{M}\right\} \psi_x\,. \tag{17.22}$$

In the computational basis, the state $|x\rangle$ of a L-qubit register is represented[4]

$$|x\rangle = |x_L x_{L-1} \cdots x_i \cdots x_1\rangle \tag{17.23}$$

where the label $x_i = 0, 1$ of the ith register cell derives from the binary expansion $x = \sum_{i=1}^{L} x_i 2^{i-1}$. With analog notation for the $|y\rangle$, the QDFT () is reformulated

$$\begin{aligned}
|x\rangle \;\mapsto\; & \sum_{y_L=0}^{1} \sum_{y_{L-1}=0}^{1} \cdots \sum_{y_1=0}^{1} \exp\left\{2\pi i \frac{x}{2^L} \sum_{j=1}^{L} y_j 2^{j-1}\right\} |y_L y_{L-1} \cdots y_1\rangle \\
= \; & \sum_{y_L=0}^{1} \sum_{y_{L-1}=0}^{1} \cdots \sum_{y_1=0}^{1} \bigotimes_{j=1}^{L} \exp\left\{2\pi i \frac{x}{2^L} y_j 2^{j-1}\right\} |y_j\rangle \\
= \; & \bigotimes_{j=1}^{L} \left(\sum_{y_j=0}^{1} \exp\left\{2\pi i \frac{x}{2^L} 2^{j-1} y_j\right\} |y_j\rangle \right) \\
= \; & \bigotimes_{j=1}^{L} \left(|0\rangle + e^{2\pi i [0.x_{L-(j-1)} \cdots x_1]} |1\rangle\right) \\
= \; & \left(|0\rangle + e^{2\pi i [0.x_1]}|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i [0.x_{L-(j-1)} \cdots x_1]}|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i [0.x_L \cdots x_1]}|1\rangle\right)
\end{aligned} \tag{17.24}$$

[4] In pedantic notation $|x\rangle = |x_L\rangle \otimes |x_{L-1}\rangle \otimes \cdots \otimes |x_1\rangle$.

where we have introduced the notation $[0.b_j b_{j-1} \cdots b_1] = b_j/2^1 + b_{j-1}/2^2 + \ldots + b_1/2^j$.
Implementation in the circuit model via Hadamard

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{17.25}$$

and "controlled rotation"

$$R_k = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}}_{\text{trgt}} \otimes \underbrace{|1\rangle\langle 1|}_{\text{ctrl}} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes |0\rangle\langle 0| \tag{17.26}$$

using matrix representation for the computational basis $|0\rangle \mapsto \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $|1\rangle \mapsto \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

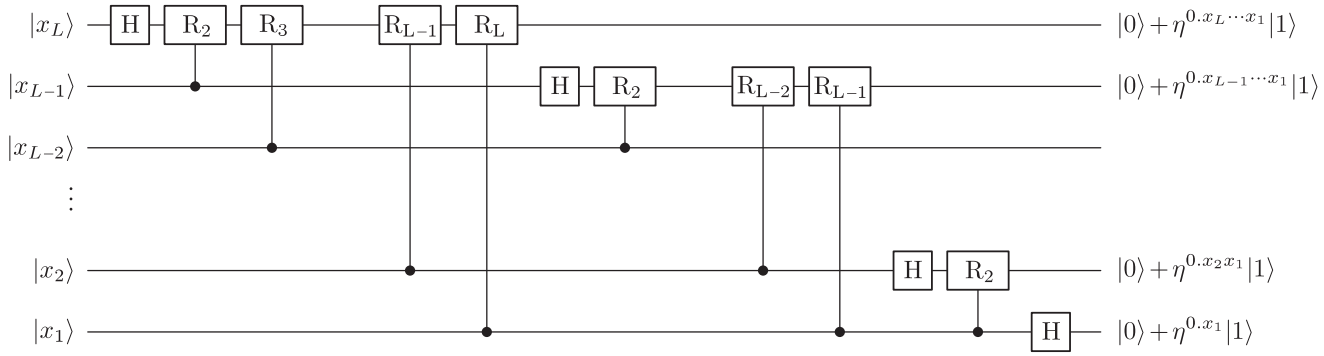

Figure 17.2: Non-trivial part of the quantum discrete Fourier transform, which is realized by augmenting with $L/2$ swaps (not shown).

Abbreviating $\eta^{0.x_j x_{i-1} \ldots x_1} = e^{2\pi i x}$ with $x = x_j/2^1 + x_{j-1}/2^2 + \ldots + x_1/2^j$, the state on on the top line evolves

$$|x_L\rangle \overset{H}{\to} \frac{1}{\sqrt{2}}\left[|0\rangle + \eta^{0.x_L}|1\rangle\right] \overset{R_2}{\to} \frac{1}{\sqrt{2}}\left[|0\rangle + \eta^{0.x_L x_{L-1}}|1\rangle\right] \overset{R_3}{\to} \ldots \overset{R_L}{\to} \frac{1}{\sqrt{2}}\left[|0\rangle + \eta^{0.x_L x_{L-1} \ldots x_1}|1\rangle\right] \tag{17.27}$$

Comparing the output of the circuit model

$$\left(|0\rangle + e^{2\pi i[0.x_L \cdots x_1]}|1\rangle\right) \cdots \left(|0\rangle + e^{2\pi i[0.x_1]}|1\rangle\right) \tag{17.28}$$

with Eq. (17.24), we realize that the circuit model 17.2 must only be augmented by $L/2$ swaps in order to realize the QDFT. The overall complexity is then given by $\frac{L(L+1)}{2}$ gates plus $L/2$ swaps, which makes the hole implementation $O(L^2)$.

17.4 Grover Algorithm

Given a phone book with N entries, each entry consisting of (1) a name of a person, and (2) the person's phone number. To make things simple, names are numbers – i.e. each person is uniquely identified by a certain number $x \in X$, with $X = \{0, 1, 2, \ldots, N-1\}$ the set of all names. The phone number of x is a number $T_x \in \mathcal{T}$, where $\mathcal{T}$ is the set of all phone numbers.

In a phone book the names are listed in lexicographic order, which for numbers translates in "ascending order", i.e. for every pair of individuals $\{a, b\} \in X \times X$, the person named a ist listed before b whenever the number a is smaller than the number b. Due to this ordering, finding the phone number of person x, say, is easy (proof in excercise session). But what if you have some phone number $\sigma \in \mathcal{T}$, and you want to know the owner of that number, i.e. you ask for $s \in X$ with $T_s = \sigma$? Keep in mind, that the phone numbers appear in no particular order.

You may just scan the entries sequentially, starting at the first entry. Upon reading x, your brain asks you to also read the phone number T_x. It then checks whether $T_x = \sigma$, and if "Yes!" returns the just read x as s with a "Heureka!", and if "No!" your brain asks you to continue reading the next entry $x + 1$. With this strategy you will certainly be successful within N queries, on average you'll need $N/2$ queries. With the Grover-Algorithm you'll need $\sqrt{N}$ queries [Lov K. Grover *Quantum Mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett. **78**, 325-328 (1997)].[5]

In more abstract terms, your brain is a subroutine that quickly checks a proposed solution x to a decision problem "does $T_x = \sigma$ hold or not". In information theory such device a is called an **oracle**. Our oracle is a simple, two-valued function on the name space

$$f_s(x) = \begin{cases} 1\,, & T_x = \sigma \\ 0\,, & T_x \neq \sigma \end{cases} \tag{17.29}$$

Don't get confused here. The function name comes with "the solution" (here: s) as a subscript. It doesn't mean that "you have to know s in order to implement f". The subscript just serves as a mnemonic that f returns "Match!" if for query x the phone number T_x matches σ, which – by virtue $\sigma = T_s$ – implies $x = s$.

The oracle (17.29) is promoted a **quantum oracle** via the unitary

$$\hat{U}_{f_s} : |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f_s(x)\rangle \tag{17.30}$$

with $|x\rangle$ a n-qubit basis state, $x \in X$, and $|y\rangle$ a single-qubit state.

With the single-qubit y-register in state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, the quantum oracle produces $(-1)^{f_s(x)}|x\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. With the y-register unchanged, it may safely be ignored,

[5] That does'nt sound much of an improvement, but watch this: to find the satisfying assignment to a SAT-formula with 80 variables, the improvement, using a computer with Gigahertz CPU, would be from 38 million years using brute-force to 18 minutes using Grover.

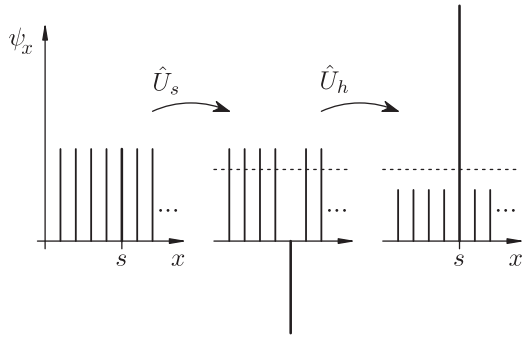

Fig. 17.1 *Amplification-Effect of the Grover $\hat{G}$. The dashed line indicates the mean.*

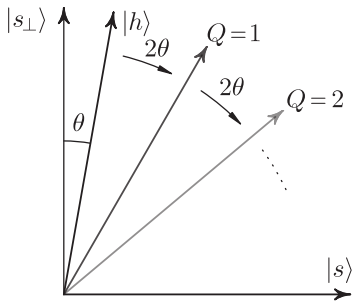

Fig. 17.2 *The action of the Grover-Rotation $\hat{G}$.*

i.e. the oracle is reduced to a unitary map on the x-register, $\hat{U}_s : |x\rangle \mapsto (-1)^{f_s(x)}|x\rangle$, i.e. it flips the sign if the x-register is in state $|s\rangle$, but acts trivially on any state orthogonal to $|s\rangle$,

$$\hat{U}_s = \mathrm{id} - 2|s\rangle\langle s| \tag{17.31}$$

Geometrically, $\hat{U}_s$ reflects the vector about the hyperplane orthogonal to $|s\rangle$, i.e. it preserves the component in the hyperplane, and flips the component along $|s\rangle$. In Fig. 17.2, the action of $\hat{U}_s$ is illustrated for the Hadamard-state $|h\rangle$ – see Eq. (17.32) below.

The Grover algorithm is defined by a successive application of a unitary, which rotates a particular initial state (h mnemonic for "Hadamard-State")

$$|h\rangle = \frac{1}{\sqrt{N}} \sum_{x \in X} |x\rangle \tag{17.32}$$

as close as possible to the solution $|s\rangle$. The unitary in question reads

$$\hat{G} := \hat{U}_h \hat{U}_s \tag{17.33}$$

with $\hat{U}_s$ as in Eq. (17.31), and

$$\hat{U}_h = 2|h\rangle\langle h| - \mathrm{id}\,, \tag{17.34}$$

Acting on some given state $|\psi\rangle = \sum_{x\in X} \psi_x |x\rangle$, the unitary $\hat{U}_h$ causes a "reflection about the mean", $\hat{U}_h|\psi\rangle = \sum_{x\in X}(-\psi_x + 2\bar{\psi})|x\rangle$ – see Fig. 17.2. Here $\bar{\psi}$ is the mean amplitude of $|\psi\rangle$, i.e. $\bar{\psi} = \frac{1}{N}\sum_{x\in X}\psi_x$. Evidently the sequential application of $\hat{U}_s$ followed by $\hat{U}_h$ effectively causes an amplification of the sought state's amplitude.

Geometrically, $\hat{U}_h$ preserves the component along $|h\rangle$ and flips the the component in the hyperplane orthogonal to $|h\rangle$. Both, the unitaries $\hat{U}_s$ and $\hat{U}_h$, leave the subspace

spanned by the two vectors $|s\rangle$ and $|h\rangle$ invariant. As an orthonormal basis on that space we may chose $|s\rangle$ and

$$|s_\perp\rangle := \frac{1}{\sqrt{N-1}}\left(\sqrt{N}|h\rangle - |s\rangle\right) \tag{17.35}$$

On an arbitrary vector $|\psi\rangle = \alpha|s\rangle + \beta|s_\perp\rangle$, the Grover acts

$$\hat{G}|\psi\rangle = \left(\frac{N-2}{N}\alpha + 2\frac{\sqrt{N-1}}{N}\beta\right)|s\rangle + \left(-2\frac{\sqrt{N-1}}{N}\alpha + \frac{N-2}{N}\beta\right)|s_\perp\rangle \tag{17.36}$$

which – switching to a matrix-representation $|s\rangle \mapsto \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $|s_\perp\rangle \mapsto \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ – reads

$$\hat{G} \mapsto \begin{bmatrix} \frac{N-2}{N} & 2\frac{\sqrt{N-1}}{N} \\ -2\frac{\sqrt{N-1}}{N} & \frac{N-2}{N} \end{bmatrix}. \tag{17.37}$$

This is nothing but a orthogonal matrix which describes a rotation by an angle ϕ, the angle being defined by $\cos(\phi) = \frac{N-2}{N}$, or $\sin(\phi) = 2\frac{\sqrt{N-1}}{N}$. With $\langle h|s\rangle = \frac{1}{\sqrt{N}} := \sin(\theta)$, and $\langle h|s_\perp\rangle = \sqrt{\frac{N-1}{N}} := \cos(\theta)$ we have $\sin(\phi) = 2\sin(\theta)\cos(\theta) = sin(2\theta)$. In summary, (**??**) (or **??**) for that matter) describes a rotation by an angle 2θ, with $\sin(\theta)$ given by $\sin(\theta) = \frac{1}{\sqrt{N}}$.

Starting with $|h\rangle$, which is tilted by θ against $|s_\perp\rangle$, the number of queries Q (application of $\hat{G}$) needed to transform $|h\rangle$ to $|s\rangle$, is given by $\theta + 2Q\theta = \pi/2$, or

$$(2Q+1)\arcsin(1/\sqrt{N}) = \pi/2 \tag{17.38}$$

For large N, we have $\arcsin(1/\sqrt{N}) \approx 1/\sqrt{N}$, in which case the number of queries

$$Q \approx \frac{\pi}{4}\sqrt{N} \tag{17.39}$$

i.e. $Q \sim \sqrt{N}$ as promised.