# Quantum Computing 5 – Simple quantum algorithms

Angelo Bassi

## Algorithms with oracle *f*<sup>3</sup> : 0 "→ 0*,* 1 "→ 1*, f*<sup>4</sup> : 0 "→ 1*,* 1 "→ 0*.*

Suppose we are supplied with a quantum  $oracle - a$  black box whose internal workings are not important at this stage – with the ability to recognize solutions to a given problem by computing a suitable function f. More precisely, the oracle is a unitary operator,  $\mathsf{U}_\mathsf{f}$ , defined by its action on the computational basis: *Dependence of an and 2, respectively, and considerable of an and 2, respectively, and considerable of an and considerable of an and considerable of an and considerable of an and considerable of a* The first two cases, *f*<sup>1</sup> and *f*2, are called *constant*, while the rest, *f*<sup>3</sup> and *f*4, Suppose we are supplied with a quantum **oracle** – a black box whose

$$
U_f\,:\,|x,y\rangle\,\mapsto\,|x,y\oplus f(x)\rangle
$$

 $\mathsf{where}\ \oplus \mathsf{is}\ \mathsf{addition}\ \mathsf{mod}\ 2.$  $\sigma$  and  $\sigma$  and  $\sigma$ Given an input *x*, a typical quantum computer computes *f*(*x*) in such a way

 $\mathcal{L}_{\mathcal{A}}$  $\frac{1}{2}$  acts on the input which is a **superposition of many state** Since  $\mathsf{U}_\mathsf{f}$  is a linear operator, it acts simultaneously on all the vectors that 2 (*|*0*, f*(0)# − *|*0*, ¬f*(0)# + *|*1*, f*(1)# − *|*1*, ¬f*(1)#)*,* of all the results *U<sup><i>f* **is a linear operator, it acts simultaneously on all the vectors that constitute on all the**</sup> Suppose  $\mathsf{U}_\mathsf{f}$  acts on the input which is a **superposition of many states**. constitute the superposition. Thus the output is also a superposition *U<sup>f</sup>* : *|x*!*|*0! "→ *|x*!*|f*(*x*)!*,* (4.61) where  $U_f$  is a mixed operator, it acts simulated stay on an Iftute the superposition. Th**us the output is also a superpositio**n

$$
U_f: \sum_x |x\rangle|0\rangle \mapsto \sum_x |x\rangle|f(x)\rangle.
$$

All values of the function computed at once. Very easy!! an computer computer computer an enormously computer and computer and computer and computer and computer and co<br>But... measurements will make the wave function collapse giving only one output. No advantage. is antended to a compared the wave runction conduct giving<br>.... in that it makes will make the wave runction conduct

most quantum algorithms. This superposition is prepared by the action The goal of a quantum algorithm is to operate in such a way that the *|*00 *...* 0! = *|*0! ⊗ *|*0! ⊗ *...* ⊗ *|*0! resulting in particular outcome we want to observe has a larger probability to be measured than the other outcomes.

#### Deutsch Algorithm is one of the first quantum is one of the first quantum of the first quantum  $\mathbb{R}^n$ 5.1 Deutsch Algorithm  $q$  algorithms may be more efficient than than the more effects than the interpretational counterparts.  $\mathcal{L}_{\mathcal{M}}$  algorithms may be more efficient than than the more efficient than the interparts. The interparts is the interpart of  $\mathcal{L}_{\mathcal{M}}$  $\sum_{n=1}^{\infty}$  readers to  $\sum_{n=1}^{\infty}$  derstand from and  $\sum_{n=1}^{\infty}$ superior to classical algorithms. We follow closely Meglicki [1]. Boutcch Algorithm  $r$ ithms we introduce a few simple  $\mathcal{L}$ rithms, we introduce a few simple quantum algorithms which will be of help

Let  $f: \{0,1\} \to \{0,1\}$  be a binary function. Note that there are only four possible  $f$ , namely The Deutsch algorithm is one of the first quantum algorithm is one of the first quantum algorithms which showed

$$
\frac{f_1: 0 \mapsto 0, 1 \mapsto 0}{f_3: 0 \mapsto 0, 1 \mapsto 1,} \frac{f_2: 0 \mapsto 1, 1 \mapsto 1}{f_4: 0 \mapsto 1, 1 \mapsto 0.}
$$

The first two cases,  $f_1$  and  $f_2$ , are called *constant*, while the rest,  $f_3$  and  $f_4$ , The first two cases,  $f_1$  and  $f_2$ , are cannot constant, while the rest,  $f_3$  and  $f_4$ ,<br>are <u>balanced</u>. If we only have classical resources, we need to evaluate f twice to tell if f is constant or balanced. There is a quantum algorithm, however, to ten if f is constant of balanced. There is a quantum algorithm, however,<br>with which it is possible to tell if f is constant or balanced with a single evaluation of  $f$ , as was shown by Deutsch [2].  $f_3: 0 \mapsto 0, 1 \mapsto 1, \quad f_4: 0 \mapsto 1, 1 \mapsto 0.$ <br>The first two cases,  $f_1$  and  $f_2$ , are called <u>constant</u>, while the rest,  $f_3$  and  $f_4$ , to tell if *f* is constant or balanced. There is a quantum algorithm, however, evaluation of  $f$ , as was shown by Deutsch [2]. *f*<sub>1</sub>  $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$   $\frac{1}{2}$ 

the state *<sup>|</sup>*ψ0# <sup>=</sup> <sup>1</sup> <sup>2</sup> (*|*00#−*|*01#+*|*10#−*|*11#). We apply *f* on this state in terms First we need to turn the classical function f(x) into a quantum one. To this purpose we define the quantum oracle with which is possible to tell if it is constant or balanced with a single *f* irst we need to turn the classical function f(x) into a quantum  $\frac{1}{2}$  assical function f(x) into a quantum one possibal function *f*<sub>1</sub>, classical function *f*<sup>1</sup> : 0 "→ 0*,* 1 "→ 0*, f*<sup>2</sup> : 0 "→ 1*,* 1 "→ 1*,* First we need to turn the classical function f(x) into a quantum one. are *balanced*. If we only have classical resources, we need to evaluate *f* twice the quantum or back *f*<sup>1</sup> : 0 "→ 0*,* 1 "→ 0*, f*<sup>2</sup> : 0 "→ 1*,* 1 "→ 1*,*

$$
\boxed{U_f\,:\,\ket{x,y}\,\mapsto\,\ket{x,y\oplus f(x)}
$$

<sup>2</sup> (*|*00# −*|*01#+*|*10# −*|*11#). We apply *f* on this state in terms

The algorithm is structured as follows. evaluation of *f* as well be as well by Deutsch  $\alpha$ . are *balanced*. If we only have classical resources, we need to evaluate *f* twice to tell if  $f$  is constant or balanced. There is a quantum algorithm, however,  $\frac{1}{2}$  and  $\frac$ the state *<sup>|</sup>*ψ0# <sup>=</sup> <sup>1</sup> of the unitary operator *U<sup>f</sup>* : *|x, y*# "→ *|x, y* ⊕ *f*(*x*)#, where ⊕ is an addition to tell if *f* is constant or balanced. There is a quantum algorithm, however, The algorithm is structured as follows.

*<u>|</u>*  $\frac{1}{2}$  **f**(1)  $\frac{1}{2}$  *l*  $\frac{1}{2}$   $\frac{1}{2}$  1. Start with the state  $|01\rangle$ . Let *|*0# and *|*1# correspond to classical bits 0 and 1, respectively, and consider  $\frac{101}{2}$  $\frac{1}{2}$  and  $\frac{1}{2}$  state  $\frac{1}{2}$  and  $\frac{1}{2}$ the state **1** 

*|*ψ1# = *U<sup>f</sup> |*ψ0#

<sup>=</sup> <sup>1</sup> 2

 $\overline{\phantom{a}}$ 

Apply an Hadamard on both qubits:  $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$ 2. Apply an Hadamard on both qubits:  $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$ Let *|*0# and *|*1# correspond to classical bits 0 and 1, respectively, and consider  $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$  $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle)$  $\frac{1}{100}$  (and with the state  $\frac{1}{100}$ ,  $\frac{1}{100}$ ,  $\frac{1}{100}$ ,  $\frac{1}{100}$ ,  $\frac{1}{100}$ 2. Apply an Hadamard on both qubits:  $\frac{1}{2}(|00\rangle - |01\rangle + |10\rangle - |11\rangle$ 

*|*ψ1# = *U<sup>f</sup> |*ψ0#

3. The *state operator* of implementing the randtion 3. Apply the operator  $U_f$  implementing the function  $\mathcal{L}$  and  $\mathcal{L}$  to be explicit, we obtain the explicit substitution of  $\mathcal{L}$ 3. Apply the operator  $U_f$  implementing the function (*|*0*, f*(0)# − *|*0*, ¬f*(0)# + *|*1*, f*(1)# − *|*1*, ¬f*(1)#)*,*

$$
\frac{1}{2}(|0, f(0)\rangle - |0, 1 \oplus f(0)\rangle + |1, f(1)\rangle - |1, 1 \oplus f(1)\rangle)
$$
  
= 
$$
\frac{1}{2}(|0, f(0)\rangle - |0, \neg f(0)\rangle + |1, f(1)\rangle - |1, \neg f(1)\rangle),
$$

where  $\sim$  stands for negative therefore the standard for negative this operation. Therefore this notation is no

## where  $\mathbf{r}$  stands for negative therefore the standard but the sta Quantum parallelism: all values computed at once

## Deutsch Algorithm

4. Apply an Hadamard to the first qubit 1 Apply 2p Lladamard to the first qubit

 $\frac{1}{\sqrt{2}}$  $\frac{1}{2\sqrt{2}}\left[ (|0\rangle + |1\rangle)(|f(0)\rangle - |\neg f(0)\rangle) + (|0\rangle - |1\rangle)(|f(1)\rangle - |\neg f(1)\rangle) \right]$  $=$   $\sqrt{2}$ 100 *QUANTUM COMPUTING f*(*x*) = 1 and left unchanged otherwise. Subsequently we apply a Hadamard of  $\overline{X}$ 

If the function is constant, the two blue terms are runction is constant, the<br>are equal, the state reduces to The wave function reduces to the wave function reduces to the wave function  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{2}}|0\rangle (|f(0)\rangle-|\neg f(0)\rangle)
$$

if *f* is balanced, for which *|¬f*(0)! = *|f*(1)!. Therefore the measurement of the function is balanced, the factor  $f$  is  $f$ Let us consider a quantum circuit which implements the Deutsch algorithm. *<sup>|</sup>*ψ2! <sup>=</sup> <sup>1</sup> √ *|*1!(*|f*(0)! − *|¬f*(0)!) (5.2) terms are opposite, the state reduces to in case *f* is constant, for which *|f*(0)! = *|f*(1)!, and *If the function is balanced, the two blue* in case *f* is constant, for which *|f*(0)! = *|f*(1)!, and

$$
\frac{1}{\sqrt{2}}|1\rangle (|f(0)\rangle-|\neg f(0)\rangle)
$$

Therefore **the measurement of the first qubit tells us whether f is** obtain *|*ψ0!. We need to introduce a conditional gate *U<sup>f</sup>* , i.e., the controlled- $X^2$ , whose action is *f(x)*, whose action is  $Y^2$  :  $Y^2$  : Therefore the measurement of the first qubit tells us whether f is Let us constant or balanced. The Deutsch algorithm in plantum constant or balanced.

5. Measure the first qubit to determine f. The full algorithm is: 5. Measure the first qubit to determine f. The full algorithm is:



#### Deutsch-Jozsa Algorithm utsch  $\sim$  the Deutsch-Jozsa algorithm  $\sim$ Let us first define the Deutsch-Jozsa problem. Suppose there is a binary 5.2 Deutsch-Jozsa Algorithm and Bernstein-Vazirani Deutsch-Jozsa Algorithm LULJUI Doutcah lozen Algorithm DUULSUI JULSU MIGUITUIIII

Let us first define the **Deutsch-Jozsa problem**. Suppose there is a binary function function  $\alpha$  constant value of the input value  $\alpha$ . When it is is input value  $\alpha$ . When it is is input value  $\alpha$ . When it is input value  $\alpha$ . When it is input value  $\alpha$ . When it is input value of  $\alpha$  is input value of  $\alpha$  Let us function the Deutsch-Jozsa problem. Suppose the Deutsch-Jozsa problem. Supp

$$
f: S_n \equiv \{0, 1, \ldots, 2^n - 1\} \to \{0, 1\}.
$$

We require that  $f$  be either *constant* or *balanced* as before. When  $f$  is constant, it takes a constant value 0 or 1 irrespetive of the input value *x*. When it is balanced the value  $f(x)$  for the half of  $x \in S_n$  is 0, while it is 1 for the rest of x. In other words,  $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$ , where |A| denotes the number  $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ it takes a constant value 0 or 1 irrespetive of the input value x. When it is<br>below and the sphere  $f(x)$  for the helf of  $x \in S$  is 0 mbile it is 1 for the vect of balanaced the value  $f(x)$  for the half of  $x \in S_n$  is 0, while it is 1 for the rest of <br>*x* In other words  $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$  where | A| denotes the number x. In other words,  $|f^{-1}(0)| = |f^{-1}(1)| = 2^{n-1}$ , where |A| denotes the number  $\frac{1}{2}$  at *f* be either *constant* or *balanced* as before. When *f* is constant.

of elements in a set *A*, known as the cardinality of *A*. Although there are It is clear that we need at least  $2^{n-1} + 1$  steps, in the worst case with classical manipulations, to make sure if  $f(x)$  is constant or balanced with  $\frac{100\% \text{ confidence. It will be shown below that the number of steps reduces to }$ a single step if we are allowed to use a quantum algorithm. It is clear that we need at least 2*<sup>n</sup>*−<sup>1</sup> + 1 steps, in the worst case with a single step if we are allowed to use a qualitum algorithm. 100% confidence. It will be shown below that the number of steps reduces to

The **oracle** for the Deutsch-Jozsa algorithm is always the sam  $\mathcal{L}_{\mathcal{S}}$  step if we are allowed to use a quantum algorithm. 1. Prepare an (*<sup>n</sup>* + 1)-qubit register in the state *<sup>|</sup>*ψ0% <sup>=</sup> *<sup>|</sup>*0%<sup>⊗</sup>*<sup>n</sup>* <sup>⊗</sup> *<sup>|</sup>*1%. First The **aracle** for the Doutsch Jazes algorithm is always the same 1110 **chacte** for the Deutsch sozsa algorithmins always the same It is clear that we need at least 2*<sup>n</sup>*−<sup>1</sup> + 1 steps, in the worst case with The oracle for the Deutsch-Jozsa algorithm is always the same the state  $\frac{1}{2}$ <sup>2</sup> (*|*00# −*|*01#+*|*10# −*|*11#). We apply *f* on this state in terms

$$
U_f\,:\,|x,y\rangle\,\mapsto\,|x,y\oplus f(x)\rangle
$$

qubits, but work as a scratch pad to store temporary information are

- 1. Prepare an (*<sup>n</sup>* + 1)-qubit register in the state *<sup>|</sup>*ψ0% <sup>=</sup> *<sup>|</sup>*0%<sup>⊗</sup>*<sup>n</sup>* <sup>⊗</sup> *<sup>|</sup>*1%. First 1. Prepare an  $(n + 1)$ -qubit register in the state  $|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$ . First *n* qubits work as input qubits, while the  $(n + 1)$ st qubit serves as a "scratch pad." Such qubits, which are neither input qubits nor output "scratch pad." Such qubits, which are neither input qubits nor output<br>qubits, but work as a scratch pad to store temporary information are<br>called ancillas or ancillary qubits called ancillas or ancillary qubits.
- have the state 2. Apply the Walsh-Hadamard transforamtion to the register. Then we have the state (*|*0% − *|*1%)*.* (5.4) have the state  $\overline{a}$

$$
U_{\mathrm{H}}^{\otimes n+1}|\psi_{0}\rangle = \frac{1}{\sqrt{2^{n}}} (|0\rangle + |1\rangle)^{\otimes n} \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle)
$$

$$
= \frac{1}{\sqrt{2^{n}}} \sum_{x=0}^{2^{n}-1} |x\rangle \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).
$$

mehy the oracle. The state changes into *U*<sub>f</sub>  $\mu$ <sub>*p*</sub>  $\mu$ <sub>*y*</sub>  $\mu$ <sub>*c*</sub>  $\mu$ <sup>*c*</sup> 3. Apply the *f*(*x*)-controlled-NOT gate on the register, which flips the 3. Apply the oracle. The state changes into

$$
=\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle\frac{1}{\sqrt{2}}(|f(x)\rangle-|\neg f(x)\rangle)
$$

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

Although the gate  $U_f$  is applied once for all, it is applied to *all* the *n*-qubit states  $|x\rangle$  simultaneously.

### Deutsch-Jozsa Algorithm  $\bigcap$ *x* (−1)*<sup>f</sup>*(*x*) *<sup>|</sup>x*! <sup>1</sup> √2 (*|*0! − *|*1!)*.* (5.5) 102 *QUANTUM COMPUTING*  $\mathcal{L}_{\text{max}}$ tcch Lozca Algorithn

4. The Walsh-Hadamard transformation is applied on the first  $n$ qubits next. We obtain  ${\rm n}$ *sformati* <sup>√</sup>2*<sup>n</sup> x*=0

the following form,

 $\mathcal{L}_{\text{max}}$  and  $\mathcal{L}_{\text{max}}$  and  $\mathcal{L}_{\text{max}}$  and  $\mathcal{L}_{\text{max}}$ 

 $\int$  1s ap (*|f*(*x*)! − *|¬f*(*x*)!)

$$
\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n - 1} (-1)^{f(x)} U_{\rm H}^{\otimes n} |x\rangle \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle
$$

4. The Walsh-Hadamard transformation (4.11) is applied on the first *n*

### *<sup>U</sup>*H*|x*! <sup>=</sup> <sup>1</sup> *On the Hadamard ga*  $\lambda$  On the Hadamard gate **On the Hadamard gate** *<sup>|</sup>*ψ3! = (*W<sup>n</sup>* <sup>⊗</sup> *<sup>I</sup>*)*|*ψ2! <sup>=</sup> <sup>1</sup> 2 !*n*−1

↑ It is instructive to write the action of the one-qubit Hadamard gate in the following form.  $\frac{1}{1}$  **1** *1* the following form, **501€**<br>−011bit Hadamard øate i (*|*0! − *|*1!)*.* (5.6) It is instructive to write the action of the action of the one-qubit Hadamard gate in  $\mathcal{A}$ 

$$
U_{\rm H}|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-1)^x|1\rangle) = \frac{1}{\sqrt{2}} \sum_{y \in \{0,1\}} (-1)^{xy}|y\rangle,
$$

 $x \in \{0,$  $\frac{1}{2}$ <sup>1</sup> where  $x \in \{0,1\}$ , to find the resulting state. The action of the Walsh-<br>Hadamard transformation on  $|x\rangle = |x| + |x| \cos \theta$  yields *Hadamard transformation on*  $|x\rangle = |x_{n-1} \dots x_1 x_0\rangle$  *yields*  $\blacksquare$ 

$$
W_n|x\rangle = (U_{\text{H}}|x_{n-1}\rangle)(U_{\text{H}}|x_{n-2}\rangle)\dots(U_{\text{H}}|x_0\rangle)
$$
  
=  $\frac{1}{\sqrt{2^n}} \sum_{y_{n-1}, y_{n-2}, \dots, y_0 \in \{0,1\}}$   

$$
\times |y_{n-1}y_{n-2}\dots y_0\rangle
$$
  
=  $\frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle$ ,  
where  $x \cdot y = x_{n-1}y_{n-1} \oplus x_{n-2}y_{n-2} \oplus \dots \oplus x_0y_0$ .

 $5.1$  The first pose  $f(x)$  is constant. Then  $\frac{1}{2}$ We then have  $\frac{1}{2}$   $\frac{1}{2}$  is constant. Suppose *f*(*x*) is constant. Then **i** where *x·y* = *x<sup>n</sup>*−<sup>1</sup>*y<sup>n</sup>*−<sup>1</sup> ⊕*x<sup>n</sup>*−<sup>2</sup>*y<sup>n</sup>*−<sup>2</sup> ⊕*...*⊕*x*0*y*0. Substituting this result

2*n*

<sup>=</sup> <sup>1</sup>

!*n*−1

$$
= \frac{1}{2^n} \left( \sum_{x,y=0}^{2^n - 1} (-1)^{f(x)} (-1)^{x \cdot y} |y\rangle \right) \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).
$$

√2

(−1)*<sup>x</sup>·<sup>y</sup>|y*!*,* (5.7)

(*|*0! − *|*1!)*.* (5.8)

## Deutsch-Jozsa Algorithm ieutsch-

5. The first *n* qubits are measured. Suppose  $f(x)$  is constant. Then

$$
\frac{1}{2^n}\sum_{x,y}(-1)^{x\cdot y}|y\rangle\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)
$$

up to an overall phase. Now let us consider the summation

$$
\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{x\cdot y}
$$

with a fixed  $y \in S_n$ . Clearly it vanishes since  $x \cdot y$  is 0 for half of *x* and 1 for the other half of *x* unless  $y = 0$ . Therefore the summation yields  $\delta_{y0}$ . Now the state reduces to <sup>it</sup> vonic

$$
|0\rangle^{\otimes n}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),
$$

and the measurement outcome of the first *n* qubits is always 00 *...* 0. *i* -<br>outcome of the first *n* out

Suppose  $f(x)$  is balanced next. The probability amplitude of  $|y = 0\rangle$  in  $|\psi_3\rangle$  is proportional to

$$
\sum_{x=0}^{2^{n}-1} (-1)^{f(x)} (-1)^{x \cdot 0} = \sum_{x=0}^{2^{n}-1} (-1)^{f(x)} = 0.
$$

Therefore the probability of obtaining measurement outcome  $00...0$  for the first *n* qubits vanishes. In conclusion, the function  $f$  is constant if we obtain  $00...0$  upon the meaurement of the first *n* qubits in the state  $|\psi_3\rangle$ , and it is balanced otherwise.





## Bernstein-Vazirani Algorithm nor balanced *|*ψ3# is a superposition of several states including *|*00#, which is

The Bernstein-Vazirani algorithm is a special case of the Deutsch-Jozsa algorithm, in which  $f(x)$  is given by  $f(x) = c \cdot x$ , where  $c = c_{n-1}c_{n-2} \ldots c_0$ is an *n*-bit binary number [4]. Our aim is to find *c* with the smallest number of evaluations of  $f$ . If we apply the Deutsch-Jozsa algorithm with this  $f$ , we obtain  $\Gamma^{2^n-1}$  **c**  $\Gamma^{2^n-1}$  and  $\Gamma^{2^n-1}$ 

$$
|\psi_3\rangle = \frac{1}{2^n} \left[ \sum_{x,y=0}^{2^n - 1} (-1)^{c \cdot x} (-1)^{x \cdot y} |y\rangle \right] \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).
$$

Let us fix *y* first. If we take  $y = c$ , we obtain (−1)*<sup>x</sup>·<sup>y</sup>*

$$
\sum_{x} (-1)^{c \cdot x} (-1)^{x \cdot c} = \sum_{x} (-1)^{2c \cdot x} = 2^n.
$$

If  $y \neq c$ , on the other hand, there will be the same number of x such that  $c \cdot x = 0$  and *x* such that  $c \cdot x = 1$  in the summation over *x* and, as a result, the probability amplitude of  $|y + c|$  vanishes. By using these results, we end the probability amplitude of  $|y \neq c\rangle$  vanishes. By using these results, we end<br>up with up with up with  $c \cdot x = 0$  and *x* such that  $c \cdot x = 1$  in the summation over *x* and, as a result, and the measurement outcome of the first *n* qubits is always 00 *...* 0.

$$
|\psi_3\rangle = |c\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).
$$

We are able to tell what  $c$  is by measuring the first  $n$  qubits.  $\frac{1}{2}$  is by mooguring the *x*=0  $\alpha$  *z* is by measuring the first *n* qubits. (−1)*<sup>c</sup>·<sup>x</sup>*(−1)*<sup>x</sup>·<sup>c</sup>* <sup>=</sup> ! (−1)<sup>2</sup>*c·<sup>x</sup>* = 2*<sup>n</sup>.*

### EXERCISE 5.2 Consider the Bernstein-Vazirani algorithm with *n* = 3 and Exercise the first *n* qubits vanishes. In conclusion, the function *f* is constant if Fxercise  $\frac{1}{2}$  , and it is balanced out in the internal conduction of  $\frac{1}{2}$ the probability amplitude of *|y* "= *c*# vanishes. By using these results, we end

**EXERCISE 5.1** Let us take  $n = 2$  for definiteness. Consider the following **EXERCISE 3.1** Let us take  $n = 2$  for definiteness. Consider the following cases and find the final wave function  $|\psi_3\rangle$  and evaluate the measurement outcomes and their probabilities for each case outcomes and their probabilities for each case.

(1)  $f(x) = 1 \ \forall x \in S_2$ .

 $(2) f(00) = f(01) = 1, f(10) = f(11) = 0.$ 

(3)  $f(00) = 0, f(01) = f(10) = f(11) = 1.$  (This function is neither constant nor balanced.)

1 if *f* is constant and with probability 0 if balanced. If *f* is neither constant **EXERCISE 5.2** Consider the Bernstein-Vazirani algorithm with  $n = 3$  and  $q = 101$ . Work out the guarantum circuit depicted in Fig. 5.2 to show that the  $c = 101$ . Work out the quantum circuit depicted in Fig. 5.2 to show that the measurement outcome of the first three qubits is  $c = 101$ .

### Grover's search algorithm telephone directory is an easy structured datebase search problem, while to unstructured database seach problem, with which we are concerned in this

Suppose there is a stack of  $N = 2<sup>n</sup>$  files, randomly placed, that are numbered by  $x \in S_n \equiv \{0, 1, \ldots, N-1\}$ . Our task is to find an algorithm which picks out a particular file which satisfies a certain condition.

In mathematical language, this is expressed as follows. Let  $f: S_n \to \{0,1\}$ be a function defined by

$$
f(x) = \begin{cases} 1 & (x = z) \\ 0 & (x \neq z), \end{cases}
$$

where  $z$  is the address of the file we are looking for. It is assumed that  $f(x)$  is *instantaneously* calculable, such that this process does not require any  $f(x)$  is *instantaneously* calculable, such that this process does not require any computational steps. A function of this sort is often called an oracle as noted  $\frac{1}{2}$  in Chapter 5. Thus, the problem is to find *z* such that  $f(z) = 1$ , given a function  $f: S_n \to \{0,1\}$  which assumes the value 1 only at a single point. In mathematical language,  $\frac{1}{2}$  is  $\frac{1}{2}$   $\$ 



function *f* : *S<sup>n</sup>* → *{*0*,* 1*}* which assumes the value 1 only at a single point. Clearly we have to check one file after another in a classical algorithm, and it will take  $O(N)$  steps on average. It is shown below that it takes only  $O(\sqrt{N})$  steps with Grover's algorithm. This is accomplished by *amplifying* the amplitude of the vector  $|z\rangle$  while cancelling that of the vectors  $|x\rangle$   $(x \neq z)$ .

We first needs to implement the function f(x) quantum mechanically. The oracle  $U_f$  is defined in the usual way

$$
U_f|x, y\rangle = |x, y \oplus f(x)\rangle \quad \text{with } f(x) = \text{either } 0 \text{ or } 1
$$

In particular we have

We describe the algorithm in several steps. The algorithm in several steps.

$$
U_f|x\rangle \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle] = (-1)^{f(x)}|x\rangle \frac{1}{\sqrt{2}}[|0\rangle - |1\rangle]
$$

### *Grover's search algorithm* (In other words, *R<sup>f</sup>* changes the sign of *w<sup>z</sup>* while leaving all other coefficients *|*ϕ# =  $\overline{1}$ *xch* alg  $\alpha$ *|*

Therefore, without loss of generality, we can neglect the last qubit and assume a unitary matrix  $\mathbb{R}^n$ *R*<sup>*R*</sup>  $\frac{1}{2}$   $\$ 

$$
U_f|x\rangle=(-1)^{f(x)}|x\rangle
$$

on the computational basis. We see that if  $x$  is an unmarked item, the *n*acle does nothing to the state. It flips the phase for the marked item. It is easy to see that *K*<sup>*i*</sup> see that <sup>δ</sup>*xy* = (−1)*<sup>f</sup>*(*x*)

$$
U_f=I-2|z\rangle\langle z|
$$

Step 1: Create an initially equal weighted superposition of all states (this is done with N Hadamard gates):<br> Step 1: Create an initially equal weighted superposition of all states (t



**Step 2: Apply the oracle U<sub>f</sub>**. Geometrically this corresponds to a *n x* = (*x*) *n x* = *x x* = (*x*) *x* = (*x*) *x* = ( that the average amplitude has been lowered. amplitude in front of the (z) state becomes negative, which in turn means (In other words, *R<sup>f</sup>* changes the sign of *w<sup>z</sup>* while leaving all other coefficients Define the kernel of the selective phase rotation transform *R<sup>f</sup>* by *K<sup>f</sup>* (*x, y*) = *e<sup>i</sup>*π*f*(*x*) **Step 2: Apply the oracle**  $U_f$ . Geome



## $G$ rover's search algorithm

Define the kernel of the selective phase rotation transform *R<sup>f</sup>* by

Step 3: Apply the gate

$$
D = -I + 2|\varphi_0\rangle\langle\varphi_0|.
$$

The action of the gate is the following *D|*ϕ# = *N* !−1 *i* **he gate is the following** average and  $\frac{1}{2}$  average and  $\frac{1}{2}$  **average** 2011 **/**/ *x* te is the following

Step 3: Apply the gate  
\n
$$
D = -I + 2|\varphi_0\rangle\langle\varphi_0|.
$$
\n
$$
\begin{bmatrix}\n\omega_x - \bar{\omega} & \omega_x \\
\omega_x & \bar{\omega} & \bar{\omega} \\
\omega_x & \bar{\omega}
$$

$$
|\varphi\rangle = \sum_{x=0}^{N-1} \omega_x |x\rangle \rightarrow D|\varphi\rangle = \left[\frac{2}{N} \sum_{x,y=0}^{N-1} |x\rangle\langle y|\right] \sum_{z=0}^{N-1} \omega_z |z\rangle - \sum_{x=0}^{N-1} \omega_x |x\rangle
$$

$$
= \frac{2}{N} \left[\sum_{x=0}^{N-1} |x\rangle\right] \left[\sum_{y=0}^{N-1} \omega_y\right] - \sum_{x=0}^{N-1} \omega_x |x\rangle = \sum_{x=0}^{N-1} (2\bar{\omega} - \omega_x) |x\rangle
$$

with 
$$
\bar{\omega} = \frac{1}{N} \sum_{x=0}^{N-1} \omega_x
$$
 average

*In our case we get* 



*<sup>N</sup>* (−1)*<sup>x</sup>·<sup>y</sup>,* (*x, y* <sup>∈</sup> *<sup>S</sup>n*) (7.7) Since the average amplitude has been lowered by the first reflection, this transformation boosts the negative amplitude times its original value, while it decreases the other amplitudes.

=<br>/km

#### Grover's search algorithm *N x*=0*,x*"=*z w (7.16)* **(7.16) (7.16)** is the average value of the coefficients of the state *R<sup>f</sup> |*ϕ". *Grover's Search Algorithm* 129 where !*<sup>N</sup>*−<sup>1</sup> *<sup>x</sup>*=0 *|wx|* <sup>2</sup> = 1 and  $G$ rover's search alg  $\overline{\phantom{0}}$ *x*=0*,x*"=*z rer's search all*

Step 3: go to step 2 an repeat the application of *U<sub>f</sub>* and D a sufficient number of times. Let us call  $G_f = D U_f$ .  $\frac{1}{2}$  suppose that the state is observed with probability construction of  $\frac{1}{2}$  when  $\frac{1}{2}$ າ ລ**.** *x*=0*,x*"=*z*  $\frac{1}{2}$   $\frac{1}{2}$  This result shows that the amplitude of *|z*" has increased upon the operation Step 3: go to step 2 an repeat the application of  $U_f$  and [  $\frac{1}{2}$  are provided applications of  $\frac{1}{2}$  increase the amplitude of  $\frac{1}{2}$  increase the amplitude of  $\frac{1}{2}$ **where**  $\theta$  is the and  $\theta$  in the applications of  $\theta$  $\overline{a}$  number of times. Let us call  $\overline{a}_f$  = D  $\overline{b}_f$ 

**PROPOSITION 7.2** Let us write **Propagate is the use with probability in the 1 when**  $\frac{1}{2}$  **when**  $\frac{1}{2}$  **when**  $\frac{1}{2}$  **when**  $\frac{1}{2}$  **when**  $\frac{1}{2}$ **EXCOPOSITION 7.2** Let us write times on the initial state *|*ϕ0". times on the initial state *|*ϕ0".

$$
G_f^k|\varphi_0\rangle=a_k|z\rangle+b_k\sum_{x\neq z}|x\rangle
$$

*|z*" so that this particular state is observed with probability close to 1 when

with the initial condition

$$
a_0 = b_0 = \frac{1}{\sqrt{N}}.
$$

Then the coefficients  $\{a_k, b_k\}$  for  $k \ge 1$  satisfy the recursion relations  $\sum_{k=1}^{N} a_k (M - 1)$  $\alpha$ <sup>*k*</sup>  $\geq$  1 satis *h*en the coefficie

of *U<sup>f</sup>* while that of *|x*" (*x* #= *z*) has decreased, assuming that all the weights

$$
a_k = \frac{N-2}{N}a_{k-1} + \frac{2(N-1)}{N}b_{k-1},
$$
\n(7.18)

$$
b_k = -\frac{2}{N}a_{k-1} + \frac{N-2}{N}b_{k-1}
$$
\n(7.19)

for  $k = 1, 2, \ldots$ **b**<sup>*k*</sup>  $\frac{1}{2}$   $\$ *<sup>N</sup> <sup>b</sup><sup>k</sup>*−<sup>1</sup> (7.19)

*Proof* It is easy to see the recursion relations are satified for  $k = 1$ *a*<sup>*k*</sup> + *b ad* + 1*l*<sub>*l*</sub> + *b<sub>1</sub></sub>* + *b<sub>1</sub></sup>*  $\equiv$  $\overline{a}$  $\langle G_f^k|\varphi_0\rangle = G_f\left(\left. a_{k-1}|z\rangle + b_{k-1}\sum_i|x\rangle\right.\right)$ Let  $G_f^{k-1}|\varphi_0\rangle = a_{k-1}|z\rangle + b_{k-1}\sum_{x\neq z}|x\rangle$ . Then  $= (-I + 2|\varphi_0\rangle\langle\varphi_0|)\left(-a_{k-1}|z\rangle + b_{k-1}\sum |x\rangle\right)$  $\frac{2}{\pi}$  $-b_{k-1}\sum_{i}|x\rangle + a_{k-1}|z\rangle + \frac{2}{\sqrt{N}}(N = -b_{k-1}\sum_{x\neq z}|x\rangle + a_{k-1}|z\rangle + \frac{2}{\sqrt{N}}(N-1)b_{k-1}|\varphi_0\rangle - \frac{2a_{k-1}}{\sqrt{N}}|\varphi_0\rangle$  $= -b_{k-1} \sum_{x \neq z} |x\rangle + a_{k-1}|z\rangle + \frac{2}{N}(N-1)b_{k-1} \sum_{x} |x\rangle - \frac{2a_{k-1}}{N} \sum_{x} |x\rangle$  $2(N-1)$ ,  $\begin{bmatrix} x \\ 2 \end{bmatrix}$ ,  $\begin{bmatrix} 2 \\ N-2 \end{bmatrix}$  $\left[\frac{N}{N} - a_{k-1} + \frac{N}{N} - b_{k-1}\right] |z| +$ ' ( *|z*" *Proof.* It is easy to see the recursion relations are satified for  $k = 1$  $\sqrt{ }$  $a_{k-1}|z\rangle + b_{k-1}\sum$  $x \neq z$  $|x\rangle$  $\setminus$  $\mathcal{F}_f^2 |\varphi_0\rangle = G_f \left( \frac{a_{k-1|z\rangle} + b_{k-1} \sum_{x \neq z} |x\rangle}{\varphi(x)} \right).$  $\sqrt{ }$  $\Big(-a_{k-1}|z\rangle + b_{k-1}\sum$  $x \neq z$  $|x\rangle$  $\setminus$  $\overline{1}$  $x \neq z$  $|x\rangle + a_{k-1}|z\rangle + \frac{2}{\sqrt{2}}$  $\frac{2}{\sqrt{N}}(N-1)b_{k-1}|\varphi_0\rangle - \frac{2a_{k-1}}{\sqrt{N}}|\varphi_0\rangle$  $x \neq z$  $|x\rangle + a_{k-1}|z\rangle + \frac{2}{N}(N-1)b_{k-1}\sum_{x}$  $\sum_{x} |x\rangle - \frac{2a_{k-1}}{N}$  $\overline{\phantom{0}}$  $\hat{z} = -b_{k-1} \sum_{x \neq z} |x\rangle + a_{k-1}|z\rangle + \frac{1}{N}(N-1)b_{k-1} \sum_x |x\rangle - \frac{2\alpha_{k-1}}{N} \sum_x |x\rangle$  $=\left[\frac{N-2}{N}a_{k-1} + \frac{2(N-1)}{N}b_{k-1}\right]$  $\sqrt{\phantom{a}}$ (  $|z\rangle + \left[ -\frac{2}{N}a_{k-1} + \frac{N-2}{N}b_{k-1} \right]$  $\overline{\nabla}$  $x \neq z$  $\left| \frac{1}{N} a_{k-1} + \frac{1}{N} b_{k-1} \right| |z\rangle + \left| -\frac{1}{N} a_{k-1} + \frac{1}{N} b_{k-1} \right| \sum |x\rangle,$  $\frac{N-2}{N}a_{k-1} + \frac{2(N-1)}{N}b_k$  $\vert z \rangle +$  $\sigma_f |\varphi_0\rangle = G_f \left( \frac{a_{k-1|z\rangle} + b_{k-1}}{x \neq z} \right)$ = (−*I* + 2*|*ϕ0"&ϕ0*|*) −*a<sup>k</sup>*−<sup>1</sup>*|z*" + *b<sup>k</sup>*−<sup>1</sup>  $a_k$  $\left. -a_{k-1} | z \rangle + b_{k-1} \sum_{x \neq z} | x \rangle \right)$  $\frac{1}{2}$ *|x*" = −*b<sup>k</sup>*−<sup>1</sup>  $\mathbb{Z}$  $\langle x \rangle$  $\sqrt{N}$  (*N* + *N* (*N* + *N* +  $\sqrt{N}$  +  $\lceil N-2 \rceil$  $k-1$  $\frac{2(N-1)}{b}$ <sub>*b*k−1</sub> | |z<sup>*i*</sup> +  $\left[-\frac{2}{a}$ <sub>*k*−1</sub> +  $\frac{N}{b}$ *x*  $\left[\frac{x}{x}\right]$   $\frac{1}{2}$   $\left[\frac{1}{2}\right]$ *N* − 2 (  $G_f^k |\varphi_0\rangle = G_f$  $\sum |x|$  $x \rightarrow$   $\top$ *|x*" *|x*" + *a<sup>k</sup>*−<sup>1</sup>*|z*" +  $-b_{k-1}\sum_{x \neq z} |x\rangle + a_{k-1}|z\rangle + \frac{2}{N}(N-1)b_{k-1}$ *|x*" + *a<sup>k</sup>*−<sup>1</sup>*|z*" +  $\left[\frac{N-2}{N}a_{k-1} + \frac{2(N-1)}{N}\right]$  $\left[\frac{1}{2}b_{k-1}\right]|z\rangle +$  $\frac{2}{5}$ *|x*"

**and proposition is proved.**  $\overline{a}$  $\frac{1}{2}$  and propositie  $\overline{r}$ *|z*"

*x*

### Grover's search algorithm + *<u>A</u> <i>n* 8 sear Grover's search algorithm **UPPERSONS COMPUTER**

*<sup>N</sup> <sup>a</sup><sup>k</sup>*−<sup>1</sup> <sup>+</sup>*<sup>N</sup>* <sup>−</sup> <sup>2</sup>

**PROPOSITION 7.3** The solutions of the recursion relations in Proposition 7.2 are explicitly given by  $\mathbf{P}$ 

$$
a_k = \sin[(2k+1)\theta], \quad b_k = \frac{1}{\sqrt{N-1}}\cos[(2k+1)\theta], \tag{7.20}
$$

for  $k = 0, 1, 2, \ldots$ , where  $\dots,$  where

$$
\sin \theta = \sqrt{\frac{1}{N}}, \quad \cos \theta = \sqrt{1 - \frac{1}{N}}.
$$
 (7.21)

written in a matrix form,  $\sqrt{2\pi}$  $\overline{\text{written}}$  $\lim$ a $\lim$  fc *written* in a matrix form, % (*<sup>N</sup>* <sup>−</sup> 2)*/N* <sup>2</sup> <sup>√</sup>*<sup>N</sup>* <sup>−</sup> <sup>1</sup>*/N* <sup>√</sup>*<sup>N</sup>* <sup>−</sup> <sup>1</sup>*/N* (*<sup>N</sup>* <sup>−</sup> 2)*/N* &  $\ddot{\phantom{0}}$ *Proof.* Let  $c_k = \sqrt{N-1}b_k$ . The recursion relations (7.18) and (7.19) are

$$
\begin{pmatrix} a_k \\ c_k \end{pmatrix} = M \begin{pmatrix} a_{k-1} \\ c_{k-1} \end{pmatrix}, M = \begin{pmatrix} (N-2)/N & 2\sqrt{N-1}/N \\ -2\sqrt{N-1}/N & (N-2)/N \end{pmatrix} = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ -\sin 2\theta & \cos 2\theta \end{pmatrix}.
$$
  
Note that *M* is a rotation matrix in  $\mathbb{R}^2$ , and its *k*th power is another rota-

% *a<sup>k</sup>* & = *M<sup>k</sup>* % *a*<sup>0</sup> & relation is easily solved to yield tion matrix corresponding to a rotation angle  $2k\theta$ . Thus the above recursion

$$
\begin{pmatrix} a_k \\ c_k \end{pmatrix} = M^k \begin{pmatrix} a_0 \\ c_0 \end{pmatrix} = \begin{pmatrix} \cos 2k\theta & \sin 2k\theta \\ -\sin 2k\theta & \cos 2k\theta \end{pmatrix} \begin{pmatrix} \sin \theta \\ \cos \theta \end{pmatrix} = \begin{pmatrix} \sin[(2k+1)\theta] \\ \cos[(2k+1)\theta] \end{pmatrix}.
$$

Replacing  $c_k$  by  $b_k$  proves the proposition.

state We have proved that the application of *U<sup>f</sup> k* times on *|*ϕ0" results in the Replacing *c<sup>k</sup>* by *b<sup>k</sup>* proves the proposition.

We have proved that the application of  $G_f$  k times on  $|\varphi_0\rangle$  results in the state state

$$
G_f^k|\varphi_0\rangle = \sin[(2k+1)\theta]|z\rangle + \frac{1}{\sqrt{N-1}}\cos[(2k+1)\theta]\sum_{x \neq z}|x\rangle. \tag{7.22}
$$

Measurement of the state  $U_f^k|\varphi_0\rangle$  yields  $|z\rangle$  with the probability

$$
P_{z,k} = \sin^2[(2k+1)\theta].
$$
 (7.23)

<sup>16</sup> <sup>−</sup> <sup>1</sup> *,* **STEP 4** Our final task is to find the k that maximizes  $P_{z,k}$ . A rough estimate for the provincing k is obtained by putting  $f(x)$  are maximizing *k* is obtained by putting  $f(x)$  $P_{z,k}$ . A roug  $\mathbf{Y}$ **STEP 4** Our final task is to find the *k* that maximizes  $P_{z,k}$ . A rough estimate

$$
(2k+1)\theta = \frac{\pi}{2} \to k = \frac{1}{2} \left( \frac{\pi}{2\theta} - 1 \right). \tag{7.24}
$$

## Grover's search algorithm

### **PROPOSITION 7.4** Let  $N \gg 1$  and let

$$
m = \left\lfloor \frac{\pi}{4\theta} \right\rfloor,\tag{7.25}
$$

where  $|x|$  stands for the floor of x. The file we are searching for will be obtained in  $U^m_f |\varphi_0\rangle$  with the probability  $G_f^{mf}$ 

$$
P_{z,m} \ge 1 - \frac{1}{N} \tag{7.26}
$$

and  $m = O(\sqrt{N}).$  (7.27) and **computing computing computi** 



*N*)*.* (7.27)

2 define  $\tilde{m}$  by *Proof.* Equation (7.25) leads to the inequality  $\pi/4\theta - 1 < m \leq \pi/4\theta$ . Let us

This is the number of times we<br>repeat the algorithm, which

*r* is the number of times we repeat the algorithm, which

grows with the square root of N

*f*  $\frac{1}{\sqrt{2}}$  with the probability of  $\frac{1}{\sqrt{2}}$ 

*<sup>f</sup> |*ϕ0# with the probability

$$
(2\tilde{m}+1)\theta = \frac{\pi}{2} \to \tilde{m} = \frac{\pi}{4\theta} - \frac{1}{2}.
$$

Observe that  $m$  and  $\tilde{m}$  satisfy

$$
|m - \tilde{m}| \le \frac{1}{2},\tag{7.28}
$$

from which it follows that from which it follows that

$$
|(2m+1)\theta - (2\tilde{m}+1)\theta| = |(2m+1)\theta - \frac{\pi}{2}| \le \theta.
$$
 (7.29)

Considering that  $\theta \sim 1/\sqrt{N}$  is a small number when  $N \gg 1$  and sin *x* is *P*<sub>m</sub>, *P*<sup>*m*</sup> = singlet  $P$ <sup>*n*</sup> + 1)*e*<sub>m</sub> monotonically increasing in the neighborhood of  $x = 0$ , we obtain

$$
0 < \sin\left[\left(2m+1\right)\theta - \pi/2\right] < \sin\theta
$$

*N* that

√

 $=\cos[(2m+1)\theta]$ 

 $\blacksquare$ 

or or

$$
\cos^2[(2m+1)\theta] \le \sin^2\theta = \frac{1}{N}.\tag{7.30}
$$

Thus it has been shown that Thus it has been shown that

$$
P_{m,z} = \sin^2[(2m+1)\theta] = 1 - \cos^2[(2m+1)\theta] \ge 1 - \frac{1}{N}.
$$
 (7.31)

It also follows from  $\theta > \sin \theta = 1/\sqrt{N}$  that

$$
m = \left\lfloor \frac{\pi}{4\theta} \right\rfloor \le \frac{\pi}{4\theta} \le \frac{\pi}{4} \sqrt{N}.
$$
 (7.32)

#### Grover's search algorithm  $A^2$  $\mathsf{H}$ # ≤  $\overline{\phantom{a}}$  $\overline{\phantom{a}}$  and  $\overline{\phantom{a}}$ π 4 ∣ø∩rithm *y*=0

It is important to note that this quantum algorithm takes only  $O(\sqrt{N})$  $K$ <sup>*K*</sup>*K***<sup>***x***</sup> is important to note that this quantum argorithm takes only**  $O(VN)$  **steps and this is much faster than the classical counterpart which requires** *O*(*N*) steps.

We now show how to implement the gates  $\mathbf{r}$ *z*=0 "*N y*=0 (*x, y*)*K*(*y, z*)

## Selective Phase Rotation Transform Let *U* be an *N* × *N* unitary matrix which acts on the *n*-qubit space *H* = (C<sup>2</sup>)⊗*<sup>n</sup>*. Let *{|x*# <sup>=</sup> *<sup>|</sup>x<sup>n</sup>*−<sup>1</sup>*, x<sup>n</sup>*−<sup>2</sup> *...,x*0#*}* (*x<sup>k</sup>* <sup>∈</sup> *{*0*,* <sup>1</sup>*}*) be the standard binary basis of *<sup>H</sup>*, where *<sup>x</sup>* <sup>=</sup> *<sup>x</sup><sup>n</sup>*−<sup>1</sup>2*<sup>n</sup>*−<sup>1</sup> <sup>+</sup> *<sup>x</sup><sup>n</sup>*−<sup>2</sup>2*<sup>n</sup>*−<sup>2</sup> <sup>+</sup> *...* <sup>+</sup> *<sup>x</sup>*02<sup>0</sup>. Then

DEFINITION 6.2 (Selective Phase Rotation Transform) Let us de-<br>fine a kernal fine a kernel fine a kernel  $e$ lec *y*=0 *|y*#%*y|U|x*# = .<br>Rot *y*=0 **DEFINITION 6.2** (Selective Phase Rotation Transform) Let us de-<br>fine a kernel

$$
K_n(x, y) = e^{i\theta_x} \delta_{xy}, \quad \forall x, y \in S_n,
$$
\n(6.43)

˜*f*(*y*) = *n* rotation is then *Selective phase rotation is then defined by the following unitary and transformation, and <i>Aperstor* (C<sup>2</sup>)⊗*<sup>n</sup>*. Suppose *<sup>U</sup>* acts on a basis vector *<sup>|</sup>x*# as operator $r$  *n*  $\alpha$  *k*)  $\alpha$ Selective phase rotation is then defined by the following unitary *e*<br> *e*<br> *ei f*(*x*) *m f*(*x*) *m f*(*x*) *k*(*x*) *k*) *f*(*x*) *f*(*x*) *k*) *f*(*x*) *k*) *f*(*x*) *k*) *f*(*x*) *k*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*) *f*(*x*)

$$
U|x\rangle = \sum_{y=0}^{N-1} K(y,x)|y\rangle.
$$

The matrix representations for  $K_1$  and  $K_2$  are sense that The matrix representations for  $K_1$  and  $K_2$  are

$$
K_1 = \left( \begin{array}{cc} e^{i \theta_0} & 0 \\ 0 & e^{i \theta_1} \end{array} \right), \quad K_2 = \left( \begin{array}{cccc} e^{i \theta_0} & 0 & 0 & 0 \\ 0 & e^{i \theta_1} & 0 & 0 \\ 0 & 0 & e^{i \theta_2} & 0 \\ 0 & 0 & 0 & e^{i \theta_3} \end{array} \right).
$$

### Selective Phase Rotation Transform  $m$ 0 *e<sup>i</sup>*θ<sup>1</sup> inverse transformation *K*−<sup>1</sup> *<sup>n</sup>* . The matrix representations for *K*<sup>1</sup> and *K*<sup>2</sup> are

The implementation of  $K_n$  is achieved with the universal set of gates as follows. Take  $n = 2$ , for example. The kernel  $K_2$  has been given above. This is decomposed as a product of two two-level unitary matrices as  $\frac{1}{2}$ with the unive *.* follows. Take *n* = 2, for example. The kernel *K*<sup>2</sup> has been given above. This

$$
K_2 = A_0 A_1,\t\t(6.45)
$$

where

$$
A_0 = \begin{pmatrix} e^{i\theta_0} & 0 & 0 & 0 \\ 0 & e^{i\theta_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, A_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\theta_2} & 0 \\ 0 & 0 & 0 & e^{i\theta_3} \end{pmatrix}.
$$
 (6.46)

Note that

$$
A_0 = |0\rangle\langle 0| \otimes U_0 + |1\rangle\langle 1| \otimes I, \quad U_0 = \begin{pmatrix} e^{i\theta_0} & 0 \\ 0 & e^{i\theta_1} \end{pmatrix},
$$
  

$$
A_1 = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_1, \quad U_1 = \begin{pmatrix} e^{i\theta_2} & 0 \\ 0 & e^{i\theta_3} \end{pmatrix}.
$$

Thus  $A_1$  is realized as an ordinary controlled- $U_1$  gate while the control bit is negated in  $A_0$ . Then what we have to do for  $A_0$  is to negate the control bit first and then to apply ordinary controlled- $U_0$  gate and finally to negate the control bit back to its input state. In summary,  $A_0$  is implemented as in

Fig. 6.5. In fact, it can be readily verified that the gate in Fig. 6.5 is written as

$$
(X \otimes I)(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes U_0)(X \otimes I)
$$
  
=  $X|0\rangle\langle 0|X \otimes I + X|1\rangle\langle 1|X \otimes U_0 = |1\rangle\langle 1| \otimes I + |0\rangle\langle 0| \otimes U_0 = A_0.$ 

Thus these gates are implemented with the set of universal gates. In fact, the order of  $A_i$  does not matter since  $[A_0, A_1] = 0$ .



### $Back$  on Grover's search algorithm *|*ϕ# = *x*=0  $\mathbf{C}$

We need to prove that the D gate used to perform the quantum search can be implemented efficiently. We now show that is the *B* gate abea to perform the quantum bearen.<br>(In other words, *R* 

$$
D = W_n R_0 W_n,\t\t(7.6)
$$

where  $W_n$  is the Walsh-Hadamard transform,

$$
W_n(x, y) = \frac{1}{\sqrt{N}} (-1)^{x \cdot y}, \quad (x, y \in S_n)
$$
 (7.7)

and  $R_0$  is the selective phase rotation transform defined by **p**  $\frac{1}{2}$  are rotation transform defined by

$$
R_0(x, y) = e^{i\pi (1 - \delta_{x0})} \delta_{xy} = (-1)^{1 - \delta_{x0}} \delta_{xy}.
$$
 (7.8)

## **Proof**

$$
\langle x|D|y\rangle = \langle x| [-I + 2|\varphi_o\rangle\langle\varphi_0| ]\, |y\rangle = -\delta_{xy} + \frac{2}{N} \qquad |\varphi_0\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle
$$

$$
\langle x|W_n R_0 W_n|y\rangle = \sum_{u,v} \langle x|W_n|u\rangle \langle u|R_0|v\rangle \langle v|W_n|y\rangle = \frac{1}{N} \sum_{u,v} (-1)^{x \cdot u} (-1)^{1-\delta_{u0}} \delta_{uv} (-1)^{v \cdot y}.
$$

$$
= \frac{1}{N} \sum_u (-1)^{x \cdot u} (-1)^{y \cdot u} (-1)^{1-\delta_{u0}}
$$

$$
= \frac{1}{N} \left[ 1 - \sum_{u \neq 0} (-1)^{x \cdot u} (-1)^{y \cdot u} \right] = A
$$

$$
\mathbf{x} = \mathbf{y}; \qquad A = \frac{1}{N} \left[ 1 - \sum_{u \neq 0} \right] = \frac{1}{N} \left[ 1 - (N - 1) \right] = -1 + \frac{2}{N}
$$

*N* !−1 *u*=0 l to the Deuts<mark>c</mark> **x ≠ y**. As discussed in relation to the Deutsch-Jozsa algorithm

$$
\sum_{u=0}^{N-1} (-1)^{x \cdot u} = 0 \rightarrow \sum_{u \neq 0}^{N-1} (-1)^{x \cdot u} = -1
$$

 $\overline{11}$ (−1)*<sup>x</sup>*0*u*<sup>0</sup> <sup>δ</sup>*<sup>u</sup>*0*v*<sup>0</sup> *.* '  $\tau = \frac{1}{N} [1 - (-1)] = \frac{1}{N}$  $\tilde{\mathcal{L}}$ #*x|D|y*" <sup>=</sup> <sup>1</sup> Therefore:  $A = \frac{1}{N} [1 - (-1)] = \frac{2}{N}$ 

## Back on Grover's search algorithm

Therefore the D gate can be implemented efficiently. The overall circuit is



We are not interested on how to implement the oracle  $U_f$  since this is supposed to be given

**Optimality of Grover's algorithm**. We have shown that a quantum computer can search N items, consulting the search oracle only O(√N) times and can prove that no quantum algorith times. One can prove that no quantum algorithm can perform this task 0 (*<sup>x</sup>* #∈ *<sup>A</sup>*)*.* (7.33) using fewer than

O(√N) accesses to the search oracle, and thus the algorithm we have demonstrated is ontimal demonstrated is optimal.

*R<sup>f</sup>* = *I* − 2 " *z*∈*A |z*%&*z|.* (7.34) For a proof, see e.g. Nielsen – Chuang p. 269.