北京郵電大學

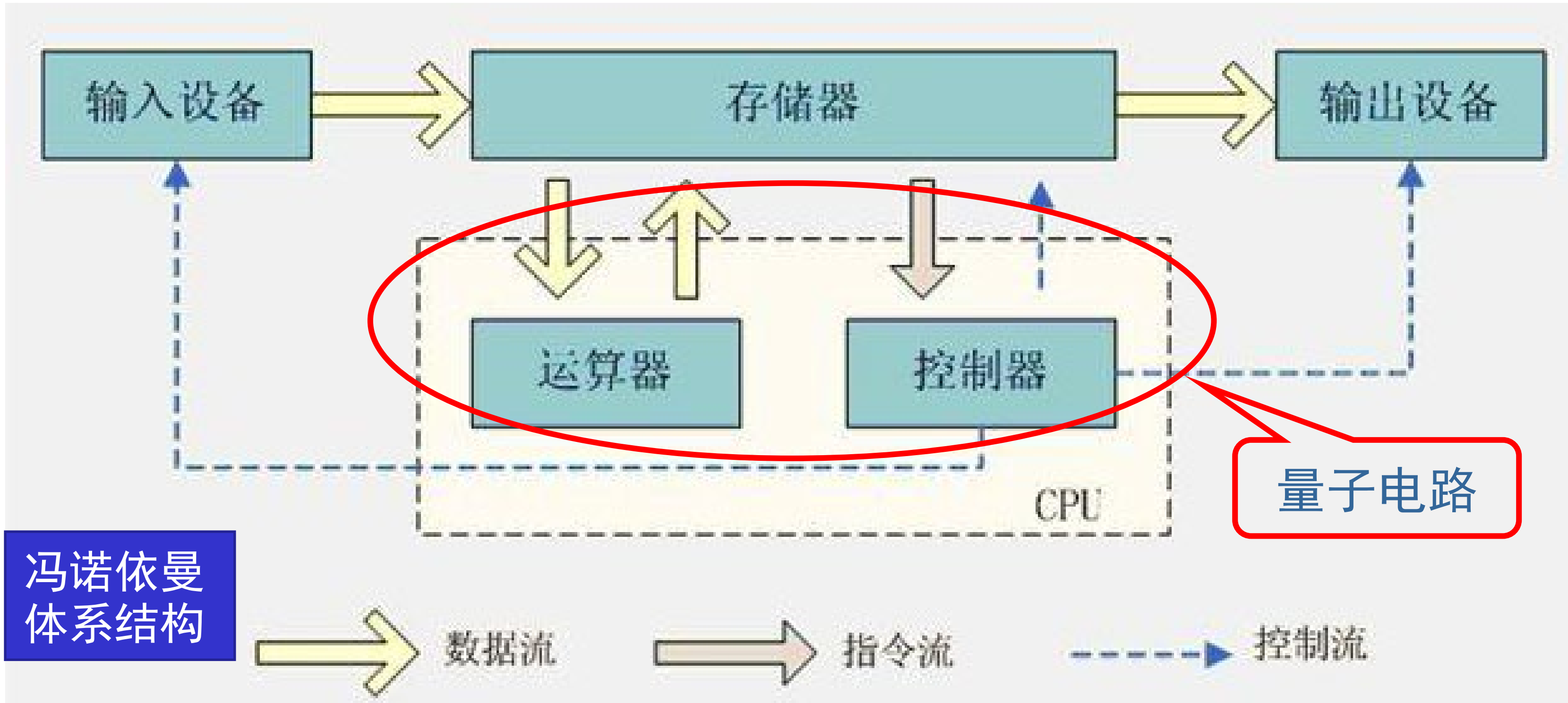
第3讲 计算复杂性/基础算法

高　飞
网络空间安全学院

通用门集合

目标：用几个简单门的组合，可以以任意精度近似实现任意幺正操作

用基本门实现通用量子计算



- 通用量子计算机：可对量子态实现任意的幺正操作
  - 电路模型下：找到一些“通用门”，用它们组合出任意的幺正操作

通用门集合的证明思路

2级（two-level）U门：仅对两个（或更少）的分量有实质操作

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \gamma \\ 0 & 0 & \beta & \delta \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \alpha & \gamma & 0 \\ 0 & \beta & \delta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha & 0 & 0 & 0 & 0 & 0 & 0 & \gamma \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \beta & 0 & 0 & 0 & 0 & 0 & 0 & \delta \end{bmatrix}$$

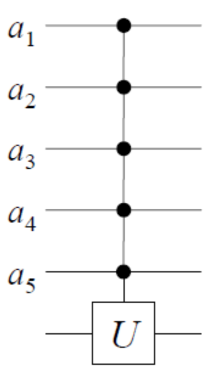
2级U门是通用的（即可实现任意U门）

通用门集合的证明思路

一位门和n控制U门
是通用的

用一位门和n控制U门可
实现任意2级U门

2级U门是通用的（
即可实现任意U门）

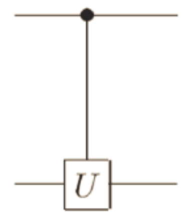


注：1控制U和n控制U
中的U是一位门

通用门集合的证明思路

一位门和CNOT门是通用的

用一位门和CNOT门可实现任意1控制U门

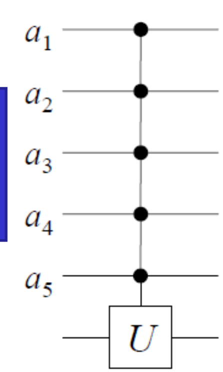


一位门和1控制U门是通用的

用1控制U门可实现任意n控制U门

一位门和n控制U门是通用的

用一位门和n控制U门可实现任意2级U门



2级U门是通用的（即可实现任意U门）

离散化

用 H 和 T 门可近似 (有效) 实现任意一位门

H，T和CNOT门是通用的

注：1控制U和n控制U中的U是一位门

结论

- 理论上，{H, T, CNOT}构成一个通用门集合（也称基本门），用它们可实现量子计算机
  - 考虑容错和实现因素（需要用到S门）： {H, T, S, CNOT}
  - 还存在其它通用门集合：{H, S, CNOT, Toffoli}
- 理论上，可将任意U按证明中的方法分解成基本门的组合
  - 例如：先分解成二级门的乘积，再将二级门可以分解成CNOT和一位门的组合，再将一位门分解成HT的组合
- 但并不是任意的U门都可以有效逼近（尽管一位门可以）
  - 可证明：某些U，不管用哪个基本门集合来分解，都需要指数多的基本门（见《QCQI》）
- 算法复杂度刻画：需要多少个基本门
  - 简化：用1控制U和一位门个数刻画（它们可由多项式规模的基本门实现）

计算复杂性

量化计算资源

- 算法复杂度一般由基本门的个数来刻画
  - 比如某算法计算两个n比特数的相加，需要门的数目

$$24n + 2\lceil \log n \rceil + 16$$

  - "该算法运算次数的规模大致为n"：在大规模计算时，一般只考虑主要的项（$24n$），并且忽略常数因子

- 渐进记号
  - 复杂度$f(n)$是$O(g(n))$：对于充分大的n，$f(n) \le cg(n)$，上界，即规模最大为$cg(n)$的级别
  - 复杂度$f(n)$是$\Omega(g(n))$：对于充分大的n，$f(n) \ge cg(n)$，下界，即规模最小为$cg(n)$的级别
  - 复杂度$f(n)$是$\Theta(g(n))$：既是$O(g(n))$的，又是$\Omega(g(n))$的，对于充分大的n，$c_1 g(n) \le f(n) \le c_2 g(n)$，渐进相等

量化计算资源

- 算

例子：

$2n$是$O(n^2)$的；　　　　2^n是$\Omega(n^3)$的；

$7n^2+\sqrt{n}\log(n)$是$\Theta(n^2)$的，因为对于充分大的n，它不小于$7n^2$，不大于$8n^2$

考虑

- 渐进记号

推广：n趋于某个值（比如n足够接近0）

  - 复杂度$f(n)$是$O(g(n))$：对于充分大的n，$f(n)\le cg(n)$，上界，即规模最大为$cg(n)$的级别
  - 复杂度$f(n)$是$\Omega(g(n))$：对于充分大的n，$f(n)\ge cg(n)$，下界，即规模最小为$cg(n)$的级别
  - 复杂度$f(n)$是$\Theta(g(n))$：既是$O(g(n))$的，又是$\Omega(g(n))$的，对于充分大的n，$c_1g(n)\le f(n)\le c_2g(n)$，渐进相等

举例：排序

- 将（乱序的）n个数字从小到大排列
  - 基本操作：比较-交换，即先比较，若次序不对就交换
  - 复杂度：调用比较-交换操作的次数

举例：排序

- 将（乱序的）n个数字从小到大排列
  - 基本操作：比较-交换，即先比较，若次序不对就交换
  - 复杂度：调用比较-交换操作的次数
- 一个初级算法

```
for j = 1 to n-1
    for k = j+1 to n
        compare-and-swap(j,k)
    end k
end j
```

基本思路

n个数中最小的，放第1

其余n-1个中最小的，放第2

其余n-2个中最小的，放第3

。。。

所需比较-交换的次数：

$$(n-1)+(n-2)+\cdots+1 = n(n-1)/2$$

所以该算法复杂度为$\Theta(n^2)$

举例：排序

- 将（乱序的）n个数字从小到大排列
  - 基本操作：比较-交换，即先比较，若次序不对就交换
  - 复杂度：调用比较-交换操作的次数
- 一个初级算法

```
for j = 1 to n-1
    for k = j+1 to n
        compare-and-swap(j,k)
    end k
end j
```

所需比较-交换的次数：

$$(n-1)+(n-2)+\cdots+1 = n(n-1)/2$$

所以该算法复杂度为$\Theta(n^2)$

可以证明下界：n个数的初始顺序有$n!$种。经过k次比较-交换操作，最多可将其中2^k种排好，要把所有可能的初始顺序都排好，最少需要

$$\begin{aligned}\log n! &= \log 1+\cdots+\log n \\ &\geq \log\frac{n}{2}+\log\left(\frac{n}{2}+1\right)+\cdots+\log n \\ &\geq \frac{n}{2}\log\frac{n}{2} \\ &\geq \frac{n}{2}\log n-\frac{n}{4}\log n(n\text{大时}) \geq \frac{n}{4}\log n\end{aligned}$$

目前已有高级算法可达到$O(n\log n)$，所以一般认为该问题复杂度是$\Theta(n\log n)$

问题的复杂性分类

- 人们常根据计算复杂度把问题分为“可解/不可解”两类
  - 可在多项式时间内求解：常称为容易、可解
    - 当α为常数，复杂度$O(n^{\alpha})$被称为n的多项式时间复杂度，记为 $\mathrm{poly}(n)$
  - 需用比多项式增长更快的时间求解：常称为指数增长、难、不可解
    - “指数增长”不一定准确，存在中间增速，如 $n^{\log n}$ （比多项式快，比指数慢）
  - 举例：求两整数乘积是可解的，因子分解是不可解的（尚未证明）
- 尽管多项式、指数的分类很有用，但也有缺点
  - 粗糙：复杂度为$2^{n/1000}$的算法，通常比复杂度为n^{1000}的要好（除非n特别大时）
  - 严格证明求解一个问题需要指数级运算是困难的

问题的复杂性分类

- 相关概念：P, NP, NPC, NPI, NP-hard 问题
  - P问题：可以找到一个能在多项式时间内解决它的算法
  - NP问题：可以在多项式时间内验证一个解的问题（不是非P）
    - 通常认为，NP问题才有可能是P问题（若验证不了就太难了）
    - 所有的P类问题都是NP问题
    - 终极问题：P是否等于NP？

问题的复杂性分类

- 相关概念：P, NP, NPC, NPI, NP-hard 问题
  - P问题：可以找到一个能在多项式时间内解决它的算法
  - NP问题：可以在多项式时间内验证一个解的问题（不是非P）
  - NPC问题：是NP问题，且所有NP问题都可规约成它（即，若能在多项式时间内解决该问题，则所有NP问题都能在多项式时间内解决）
    - 这种问题有很多：背包问题、旅行推销员问题、顶点覆盖、子图同构问题
    - NP问题不一定是难解的，NPC才是，它目前没有多项式算法
    - 如果能给NPC问题找一个多项式时间算法，则P=NP
    - 因为NPC问题太难，多数人觉得不可能找到多项式时间算法，倾向于P≠NP

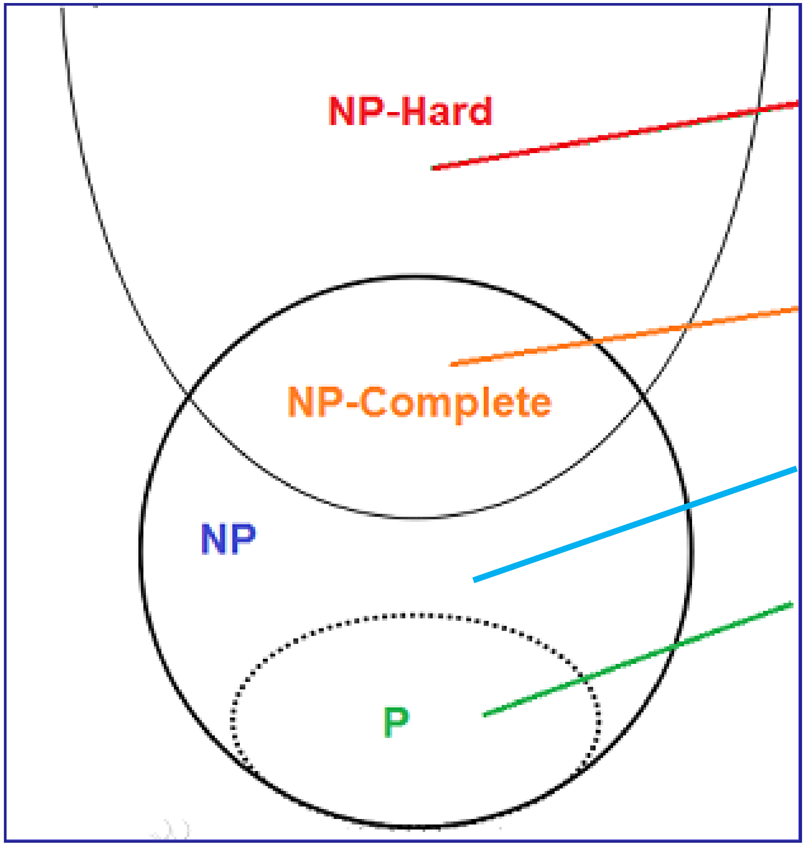
问题的复杂性分类

- 相关概念：P, NP, NPC, NPI, NP-hard 问题
  - P问题：可以找到一个能在多项式时间内解决它的算法
  - NP问题：可以在多项式时间内验证一个解的问题（不是非P）
  - NPC问题：是NP问题，且所有NP问题都可规约成它（即，若能在多项式时间内解决该问题，则所有NP问题都能在多项式时间内解决）
  - NPI问题：既不是多项式时间可解，又不是NPC的问题
    - 存在前提：假设P≠NP成立，则可证明存在NPI问题（证明者构造了一个问题）
    - NPI被认为是量子算法的研究目标：很多人怀疑量子计算也不能有效求解NPC
    - 人们猜测的NPI问题：因子分解、图同构等

问题的复杂性分类

- 相关概念：P, NP, NPC, NPI, NP-hard 问题
  - P问题：可以找到一个能在多项式时间内解决它的算法
  - NP问题：可以在多项式时间内验证一个解的问题（不是非P）
  - NPC问题：是NP问题，且所有NP问题都可规约成它（即，若能在多项式时间内解决该问题，则所有NP问题都能在多项式时间内解决）
  - NPI问题：既不是多项式时间可解，又不是NPC的问题
  - NP-Hard问题：所有NP问题都可规约成它
    - 不要求它是NP，所以NP-Hard问题要比 NPC问题的范围广
    - 即使NPC问题发现了多项式级的算法，NP-Hard问题有可能仍然无法有效求解

问题的复杂性分类



例如：图灵停机问题

例如：顶点覆盖问题

猜测：因子分解、图同构

例如：最短路径问题

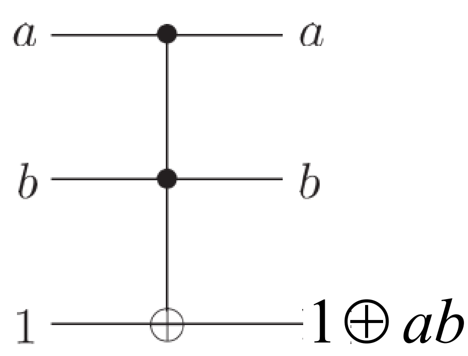
P≠NP假设下的关系图

基础量子算法

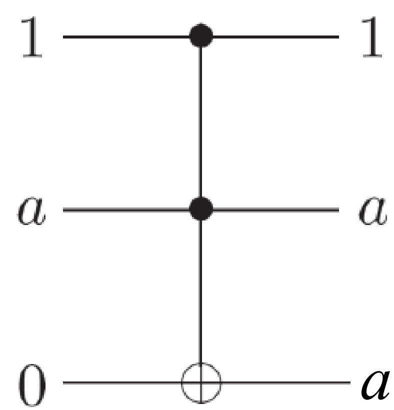
北京邮电大学 Beijing University of Posts and Telecommunications

模拟经典计算

- 大多经典电路是不可逆的，而量子电路是可逆的，所以不能用直接量子电路实现经典逻辑门
- 利用Toffoli门，量子计算机可以做经典计算机能做的所有确定性计算
  - 可以（以可逆方式）实现NAND和FANOUT门
  - 进而可以（以可逆方式）实现经典电路里的其他基本门



实现NAND门



实现FANOUT门（复制经典比特）

模拟经典计算

- 大多经典电路是不可逆的，而量子电路是可逆的，所以不能用直接量子电路实现经典逻辑门
- 利用Toffoli门，量子计算机可以做经典计算机能做的所有确定性计算
  - 可以（以可逆方式）实现NAND和FANOUT门
  - 进而可以（以可逆方式）实现经典电路里的其他基本门
- 加上量子随机数，量子计算机也可以做经典计算机可以做的所有非确定性计算
  - 非确定性计算的核心是产生随机数（比如用蒙特卡洛方法），量子计算机显然也能产生随机数

量子计算机可以完成所有经典计算任务

量子并行计算

- 如何实现一个经典函数$f(x):\{0,1\}\rightarrow\{0,1\}$?

经过适当的逻辑门（记为U_f），可将状态$|x,y\rangle=|x\rangle\otimes|y\rangle, x,y\in\{0,1\}$ 变为

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

注：可逆

其中$\oplus$代表模2加

推广：其它定义域、值域上的函数也可以

- 如果$y=0$，则第二qubit输出$f(x)$

- 特殊情况：如果$|y\rangle=\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)=|-\rangle$呢？

$$U_f|x\rangle(|0\rangle-|1\rangle)=U_f|x\rangle|0\rangle-U_f|x\rangle|1\rangle=|x\rangle|0\oplus f(x)\rangle-|x\rangle|1\oplus f(x)\rangle$$

$$=|x\rangle[|0\oplus f(x)\rangle-|1\oplus f(x)\rangle]=\begin{cases}|x\rangle(|0\rangle-|1\rangle),\text{当}f(x)=0\\-|x\rangle(|0\rangle-|1\rangle),\text{当}f(x)=1\end{cases}=(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

量子并行计算

- 如果第一寄存器（用一个符号来代表的存储单元，含1或多个qubit）是叠加态，则可实现并行计算
  - 通常用H门作用在|0>态上，得到均匀叠加态

$$H \otimes H|00\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$H^{\otimes n}|0^{\otimes n}\rangle = \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right]^{\otimes n} = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$$

$$N = 2^n$$

  - 此时执行量子操作，则可并行计算

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

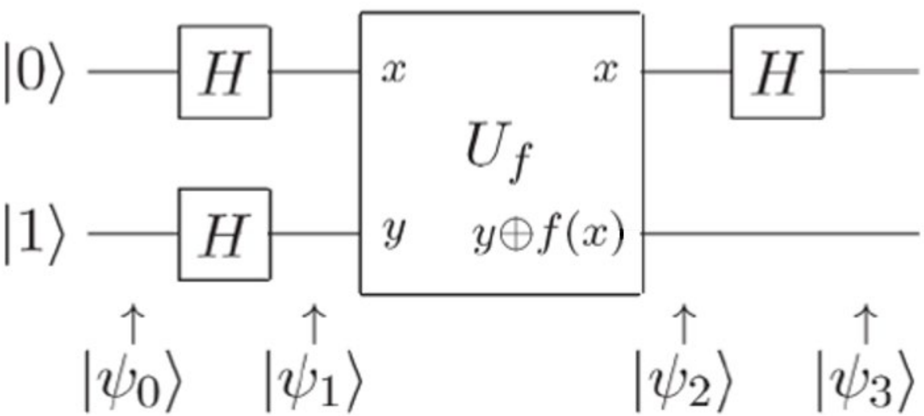
并非直接有用：如果此时进行测量，只能随机得到某一个$f(x)$，并不能得到全部

Deutsch’s algorithm

算法目的：得到（输入输出均为0或1）布尔函数$f(x)$的全局信息，即$f(0)\oplus f(1)$

经典算法：至少需要2次调用$f(x)$

量子算法：只需调用1次$f(x)$



1. 初态$|\psi_0\rangle = |01\rangle$，执行$H\otimes H$得

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|-\rangle$$

注：对函数$f(x)\in\{0,1\}$， $U_f: |x\rangle|y\rangle \to |x\rangle|y+f(x)\rangle$作用在$|x\rangle|-\rangle$上的效果：

$$U_f|x\rangle(|0\rangle - |1\rangle) = (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle)$$

2. 对$|\psi_1\rangle$执行U_f得

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}[(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle]|-\rangle$$

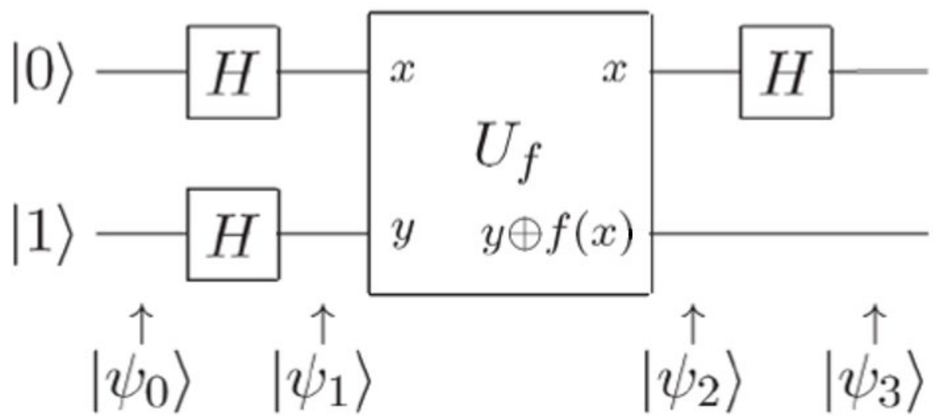
$$\begin{aligned} &U_f(|0\rangle + |1\rangle)|-\rangle \\ &= U_f|0\rangle|-\rangle + U_f|1\rangle|-\rangle \\ &= (-1)^{f(0)}|0\rangle|-\rangle + (-1)^{f(1)}|1\rangle|-\rangle \\ &= [(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle]|-\rangle \end{aligned}$$

Deutsch’s algorithm

算法目的：得到（输入输出均为0或1）布尔函数$f(x)$的全局信息，即$f(0)\oplus f(1)$

经典算法：至少需要2次调用$f(x)$

量子算法：只需调用1次$f(x)$



4.最后用$\{|0\rangle,|1\rangle\}$基测qubit 1，得|0>意味着$f(0)=f(1)$，得|1>则相反

1. 初态$|\psi_0\rangle=|01\rangle$，执行$H\otimes H$得

$$|\psi_1\rangle=\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|-\rangle$$

注：对函数$f(x)\in\{0,1\}$，$U_f:|x\rangle|y\rangle\rightarrow|x\rangle|y+f(x)\rangle$作用在$|x\rangle|-\rangle$上的效果：

$$U_f|x\rangle(|0\rangle-|1\rangle)=(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$$

2. 对$|\psi_1\rangle$执行U_f得

$$|\psi_2\rangle=\frac{1}{\sqrt{2}}[(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle]|-\rangle$$

3. 对qubit 1 执行H得

$$|\psi_3\rangle=s_+|0\rangle|-\rangle+s_-|1\rangle|-\rangle$$

其中 $s_\pm\equiv\frac{1}{2}[(-1)^{f(0)}\pm(-1)^{f(1)}]$

$f(0)\oplus f(1)=0$时，$s_-=0$；反之$s_+=0$

北京邮电大学 Beijing University of Posts and Telecommunications

Deutsch-Jozsa algorithm

有一个黑盒可以计算函数 $f(x): \{0,1,\cdots,2^n-1\} \to \{0,1\}$，且 $f(x)$ 有两种类型：对所有x是常数（constant）；一半x使函数取0，另一半得1（balance）

- 可访问黑盒：输入$x \in \{0,1,\cdots,2^n-1\}$，黑盒返回$f(x)=0$或1
- 问题：要确定f类型最少访问次数？

- 经典：一次访问只能获得一个x的函数值，最坏时需要访问$2^{n-1}+1$次
- 量子：若允许交换量子比特、执行量子计算，则只需访问1次

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{N}} \sum_{z_1,z_2,\cdots,z_n} (-1)^{x\cdot z} |z_1 z_2 \cdots z_n\rangle$$

1. 初态$|\psi_0\rangle = |0^{\otimes n}\rangle|1\rangle$，执行$H^{\oplus n+1}$得

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle|-\rangle$$

2. 执行$U_f: |x\rangle|y\rangle \to |x\rangle|y+f(x)\rangle$得

$$|\psi_2\rangle = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} (-1)^{f(x)}|x\rangle|-\rangle$$

3. 对前n个qb执行$H^{\otimes n}$得（忽略最后qb）

$$|\psi_3\rangle = \sum_{z_1,z_2,\cdots,z_n} s(z_1 z_2 \cdots z_n)|z_1 z_2 \cdots z_n\rangle$$

其中 $s(z_1 z_2 \cdots z_n) \equiv \frac{1}{N}\sum_x (-1)^{x\cdot z+f(x)}$

$$x \cdot z \equiv x_1 z_1 + x_2 z_2 + \cdots + x_n z_n$$

4. 测量末态看其是否全0即可判断。注：$s(00\ldots0) = \frac{1}{N}\sum_x(-1)^{f(x)}$；若函数是cons.，则$s(00\ldots0)=\pm1$；若函数是bal.，则$s(00\ldots0)=0$

算法小结

- 量子算法的核心：巧妙设计函数和变换，使得问题的解是函数的全局（global）信息，这样就可能通过并行性获得比经典算法更快的量子算法

操作常设为 $U_f : |x\rangle|y\rangle \to |x\rangle|y+f(x)\rangle$

作用在$|-\rangle$上 $U_f|x\rangle(|0\rangle-|1\rangle)=(-1)^{f(x)}|x\rangle(|0\rangle-|1\rangle)$

作用在均匀叠加态和$|-\rangle$上 $U_f\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}|x\rangle|-\rangle=\frac{1}{\sqrt{N}}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}|x\rangle|-\rangle$

对$|x\rangle$做$H^{\otimes n}$ $H^{\otimes n}|x\rangle=\frac{1}{\sqrt{N}}\sum_{z_1,z_2,\cdots,z_n}(-1)^{x\cdot z}|z_1z_2\cdots z_n\rangle$

Email: gaof@bupt.edu.cn
Tel: 86 -10 -62283192
Web: www.bupt.edu.cn

谢 谢!