Computação Quântica
Aula 08

Murilo V. G. da Silva

DINF/UFPR

O Algoritmo de Simon

Entrada: Uma função 2-para-1 $f : \{0,1\}^n \to \{0,1\}^n$ com $s \in \{0,1\}^n$ tal que se $x \neq y$ e $f(x) = f(y)$, então $x \oplus s = y$ (i.e., $f(x \oplus s) = f(x)$).

Saida: Retornar a string s (obs: assumimos $s \neq 0^n$)

Nota: novamente estamos no "modelo *black box*"
Nota: $N = 2^n$.

Pergunta:

- Com um algoritmo clássico, quantas *queries* precisamos fazer?
- Podemos fazer melhor usando um algoritmo quântico?

Algoritmos clássicos:

- Determinístico: $\mathcal{O}(N) = \mathcal{O}(2^n)$
- Aleatorizado: $\mathcal{O}(\sqrt{N})$, lembrando que $\sqrt{N} = 2^{n/2}$

O Algoritmo de Simon

Entrada: Uma função 2-para-1 $f : \{0,1\}^n \to \{0,1\}^n$ com $s \in \{0,1\}^n$ tal que se $x \neq y$ e $f(x) = f(y)$, então $x \oplus s = y$ (i.e., $f(x \oplus s) = f(x)$).

Saida: Retornar a string s (obs: assumimos $s \neq 0^n$)

Passos do Algoritmo

- Para algum r aleatório, criar o estado $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$
- A partir do estado acima, obter string y aleatória tal que $y \cdot s = 0 \pmod 2$
- Repetir isso para obter $n-1$ strings y com tal propriedade

Ideia central do algoritmo:

- cada string y nos dá a equação $y_1 s_1 + y_2 s_2 + ... + y_n s_n = 0 \pmod 2$ (incógnitas $s_1, ..., s_n$ e coeficientes $y_1, ..., y_n$)
- Se obtivermos $n-1$ equações, obtemos os valores $s_1, ..., s_n$ (ou seja os bits de s)

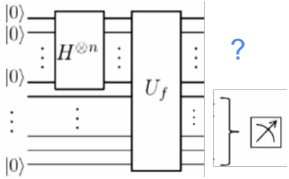
Nota: como estamos trabalhando (MOD 2), há duas soluções: a solução trivial 0^n e a solução s, cujos bits são das variáveis $s_1, ..., s_n$ e que pode ser obtida usando eliminação gaussiana módulo 2 que roda em tempo $\mathcal{O}(n^3)$

- Cuidado extra: as equações obtidas devem ser linearmente independentes

(1) Como obter o estado $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$?

O Algoritmo de Simon

Considere o circuito abaixo:



Qual a saída dos n qubits da parte de cima do circuito?

- Estado dos $2n$ qubits saindo de U_f antes da medida: $\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}} |x\rangle |f(x)\rangle$
- Observe que a parte de cima do circuito contém x e a parte de baixo $f(x)$
- Medindo os n qubits da parte de baixo, estes qubits colapsam em algum valor z
- Observe que $|z\rangle = |f(r)\rangle$, para algum valor aleatório r
- Na realidade, pela definição de f, dado $z = f(r) = f(r \oplus s)$, para um r aleatório
- Depois da medida parcial o estado será condizente com os bits medidos, e portanto os $2n$ qubits serão $\frac{1}{\sqrt{2}} \sum_{x \in \{r, r \oplus s\}} |x\rangle |z\rangle = \frac{1}{\sqrt{2}} |r\rangle |z\rangle + \frac{1}{\sqrt{2}} |r \oplus s\rangle |z\rangle$
- Com isso o estado dos n qubits de cima é $\boldsymbol{\frac{1}{\sqrt{2}} |r\rangle + \frac{1}{\sqrt{2}} |r \oplus s\rangle}$

O Algoritmo de Simon

Entrada: Uma função 2-para-1 $f : \{0,1\}^n \to \{0,1\}^n$ com $s \in \{0,1\}^n$ tal que se $x \neq y$ e $f(x) = f(y)$, então $x \oplus s = y$ (i.e., $f(x \oplus s) = f(x)$).

Saida: Retornar a string s (obs: assumimos $s \neq 0^n$)

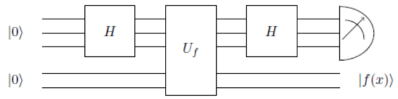
Passos do Algoritmo

- Para algum r aleatório, criar o estado $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$
  (Resolvido!!)
- A partir do estado acima, obter string y aleatória tal que $y \cdot s = 0 \pmod 2$
  (Próximo passo)
- Repetir isso para obter $n-1$ strings y com tal propriedade
  (Depois)

(2) Como obter string aleatória y, tal que $y \cdot s = 0 \pmod 2$?

O Algoritmo de Simon

Obtendo string aleatória y com a propriedade $y \cdot s = 0 \pmod 2$:



- Basta fazer uma amostragem de Fourier em $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$
- Nos qubits de cima, depois de $H^{\otimes n}$ e antes da medida, temos uma superposição de estados $|y\rangle$, onde cada estado $|y\rangle$ tem amplitude β_y
- Ideia: mostrar que os estados $|y\rangle$ indesejados tem amplitude $\beta_y = 0$
  (lembrando que os estados indesejados são os que tem $y \cdot s \neq 0 \pmod 2$)
- Dado um y, temos $\beta_y = \frac{1}{\sqrt{2}} \frac{(-1)^{r \cdot y}}{2^{n/2}} + \frac{1}{\sqrt{2}} \frac{(-1)^{(r \oplus s) \cdot y}}{2^{n/2}} = \frac{(-1)^{r \cdot y}}{2^{(n+1)/2}} [1 + (-1)^{s \cdot y}]$
  - Caso 1: $y \cdot s = 0 \pmod 2 \quad \longrightarrow \quad \beta_y = \frac{(-1)^{r \cdot y}}{2^{(n-1)/2}}$
  - Caso 2: $y \cdot s = 1 \pmod 2 \quad \longrightarrow \quad \beta_y = 0$

O Algoritmo de Simon

Entrada: Uma função 2-para-1 $f : \{0,1\}^n \to \{0,1\}^n$ com $s \in \{0,1\}^n$ tal que se $x \neq y$ e $f(x) = f(y)$, então $x \oplus s = y$ (i.e., $f(x \oplus s) = f(x)$).

Saida: Retornar a string s (obs: assumimos $s \neq 0^n$)

Passos do Algoritmo

- Para algum r aleatório, criar o estado $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$
  (Resolvido)
- A partir do estado acima, obter string y aleatória tal que $y \cdot s = 0 \pmod 2$
  (Resolvido)
- Repetir isso para obter $n-1$ strings y com tal propriedade
  (Agora)

Ideia central do algoritmo:

- cada string y nos dá a equação $y_1 s_1 + y_2 s_2 + ... + y_n s_n = 0 \pmod 2$
  (incógnitas $s_1, ..., s_n$ e coeficientes $y_1, ..., y_n$)
- Se obtivermos $n-1$ equações, obtemos os valores $s_1, ..., s_n$ (ou seja os bits de s)

Nota: como estamos trabalhando (MOD 2), há duas soluções: a solução trivial 0^n e a solução s, cujos bits são das variáveis $s_1, \ldots, s_n$ e que pode ser obtida usando eliminação gaussiana módulo 2 que roda em tempo $\mathcal{O}(n^3)$

- Cuidado extra: as equações obtidas devem ser linearmente independentes

O Algoritmo de Simon

Entrada: Uma função 2-para-1 $f : \{0,1\}^n \to \{0,1\}^n$ com $s \in \{0,1\}^n$ tal que se $x \neq y$ e $f(x) = f(y)$, então $x \oplus s = y$ (i.e., $f(x \oplus s) = f(x)$).

Saida: Retornar a string s (obs: assumimos $s \neq 0^n$)

Ideia do Algoritmo

- Para algum r aleatório, obter o estado $\frac{1}{\sqrt{2}}|r\rangle + \frac{1}{\sqrt{2}}|r \oplus s\rangle$ (OK)
- Obter string aleatória y com a propriedade $y \cdot s = 0 \pmod 2$ (OK)
- Repetir isso para obter $n-1$ strings y com tal propriedade Agora!

Lembrando que cada $y \cdot s = 0$ nos dá uma equação $y_1 s_1 + y_2 s_2 + ... + y_n s_n = 0$ para resolvermos um sistema linear (mod 2)

- Problema: queremos que as $n-1$ equações sejam linearmente independentes
- Veremos que a probabilidade de que o algoritmo gere $n-1$ equações linearmente independentes é $\geq 1/4$, portanto basta usar tentativas repetidas:
  - Resolvendo o sistema e obtendo s, apenas testamos se $f(x) = f(x \oplus s)$
  - Caso o teste falhe, rodamos novamente, digamos 100 vezes.
  - A Probabilidade de falhar 100 vezes é $\leq 0.75^{100}$, ou seja, nula para efeitos práticos.

O Algoritmo de Simon

Probabilidade de que as equações obtidas sejam L.I é $\geq 1/4$:

- A primeira equação $y_1 s_1 + y_2 s_2 + \ldots + y_n s_n = 0$ (qualquer uma serve, exceto $y_i = 0$, para todo i)
  Probabilidade de "dar problema" $(y = y_1 y_2 \ldots y_n = 0)$: $\frac{1}{2^{n-1}}$
- A segunda equação deve ser diferente da primeira e $y \neq 0$
  Probabilidade de "dar problema": $\frac{2}{2^{n-1}}$
- A terceira não pode ser combinação linear duas primeiras equações
  Probabilidade de "dar problema": $\frac{4}{2^{n-1}}$ (são 4 c.l., pois estamos em $\mathbb{Z}_2$)
  (tem que ser $\neq 0$, $\neq$ primeira equação, $\neq$ segunda equação, $\neq$ soma das duas)

  $\vdots$
- A $n-2$-ésima equação não pode combinação linear das equações anteriores
  Probabilidade de "dar problema": $\frac{2^{n-3}}{2^{n-1}} = \frac{1}{4}$
- A $(n-1)$-ésima equação não pode ser combinação linear das equações anteriores
  Probabilidade de "dar problema": $\frac{2^{n-2}}{2^{n-1}} = \frac{1}{2}$

Prob. de "dar problema" até penúltima equação é uma soma geométrica $\leq 1/2$
Prob. de "dar problema" na última equação é $1/2$

Probabilidade geral de dar certo (todas eq. L.I.) é $\geq \frac{1}{4}$