Algoritmo di Shor

Algoritmi classici per fattorizzare interi - Migliore General Number Field Sieve $O(e^{c(\log N)^3 (\log\log N)^{2/3}})$

Shor Complessità polinomiale

$O((\log N)^3)$ con $O(\log N)$ spazio

- Algoritmo probabilistico
- Si comporta di una parte classica e una quantistica

Problema: Dato N trovare $p < N$ t.c. $p | N$

Procedura:

1. Riduzione al problema di trovare l'ordine di un elemento mod N (C)

2. Algoritmo quantistico per trovare l'ordine (Q)
   - QFFT
   - stima della fase
   - stima dell'autovalore

3. Ricostruzione razionale (Frazioni continue) (C)

Algoritmo

Input : N, composto

output : $1 < p < N$ t.c. $p \mid N$

(tempo : $O((\log N)^3)$ operazioni
Funziona con prob. $O(1)$)

C 1. se N è pari $\Rightarrow 2$

C 2. Determina se $\exists a > 1$ e $b \geq 1$ t.c. $N = a^b$
in caso $\Rightarrow a$

C 3. Scegli $1 < x \leq N-1$ random.
Se $\gcd(x, N) \neq 1 \Rightarrow x$

C 4. Determina $q = 2^{\cdot}$ minimo t.c. $N^2 \leq q < 2N^2$

Q 5 | Trova $r = O(x) \bmod N$
Misura $\rightarrow c$ t.c. $\frac{c}{q} \approx \frac{d}{r}$ $d \in \mathbb{N}$

C 6 Trova d, r (sviluppo in frazioni continue)

C 7.. Se $d \equiv 1 \ (2)$ o se $x^{r/2} \equiv -1 \ (N) \Rightarrow$ Vai a 3.
Ritorna $p, q = \gcd(x^{r/2} - 1, N)$.

Vediamo i vari passi.

Cerchiamo a ridurci al caso N dispari e compito.

2. Determinare se $N = a^b$, $\text{Log } N = L$

(per capire se ha 2 fattori distinti)

- Scelgo b ($b \leq L$ se $\exists$)
- se $b = \log_a N = \dfrac{\log_2 N}{\log_2 a}$ →

$\log_2 a = \dfrac{L}{b} = x \geq 1$

trovi $u_1, u_2 \in \mathbb{Z}$ $u_1 \leq 2^x \leq u_2$ } $O(L^2)$

- u_1^b e u_2^b (repeated squaring)

se u_1^b o $u_2^b = N$ → ok

- provo con altro b ($\text{Log } N = L$ volte)

$\Rightarrow \Theta(L^3)$.

Riduzione a OF

Se y t.c $y^2 \equiv 1 \ (N)$ e $y \not\equiv \pm 1 \ (N)$

$\Rightarrow \gcd(y^2 - 1, N)$ è un fattore non banale

Preso x random calcolare l'ordine di $x \bmod (N)$.

Sia $r = O(x)$. Se $r \equiv 1 \ (2)$ cambia x.

Se $r \equiv 0 \ (2)$, controlla $x^{r/2} \not\equiv -1 \ (N)$ e calcola $\gcd(x^{r/2} \pm 1, N)$.

Troviamo l'ordine di x

Definiamo l'operatore unitario $L = \log(N)$

$$U|y\rangle \to |xy \ (N)\rangle$$

$y \in \{0,1\}^L$.

Oss. per rendere unitario dobbiamo fare:

$$y \to xy \ (N) \quad 0 \le y < N \quad *$$

$$y \to y \qquad N \le y \le 2^L - 1$$

Quindi l'azione di U è non banale solo se $*$

Sia $\omega = e^{\frac{-2\pi i}{r}}$ radice primitiva r-ma di 1

se consideriamo

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega^{sk} |x^k(N)\rangle$$

allora

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \omega^{sk} \cdot |x^{k+1}(N)\rangle$$

$$= \frac{1}{\sqrt{r}} \omega^{-s} \cdot \sum_{h=1}^{r} \omega^{hs} |x^h(N)\rangle =$$

$$= \frac{1}{\sqrt{r}} \omega^{-s} \sum_{h=0}^{r-1} \omega^{hs} |x^h(N)\rangle =$$

$$\omega^{-s} \cdot |u_s\rangle$$

Quindi $|u_s\rangle$ è autovettore

con autovalore

$$\omega^{-s} = e^{\frac{2\pi i s}{r}}$$

Abbiamo ricondotto il problema alla determinazione di un autovalore - PHASE ESTIMATION

∴ Problema Come preparare $|u_s\rangle$? Dovremmo conoscere (r)

Però vale:

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = |1\rangle \quad !$$

Infatti:

$\omega = \frac{2\pi i}{r}$

a). $\sum_{s=0}^{r-1} \omega^{sk} = \frac{\omega^{kr} - 1}{\omega^k - 1} = 0$ se $k \neq 0$

$= r$ se $k = 0$

b). Valle avere $\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \omega^{-sk} |u_s\rangle =$

$$\frac{1}{r} \sum_{s=0}^{r-1} \omega^{-sk} \cdot \sum_{h=0}^{r-1} \omega^{sh} |x^h\rangle =$$

$$\frac{1}{r} \sum_{h=0}^{r-1} \cdot \sum_{s=0}^{r-1} \omega^{s(h-k)} |x^h\rangle = |x^k\rangle$$

e quindi applicando @

$$\frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} |u_s\rangle = \frac{1}{r} \sum_{s=0}^{r-1} \cdot \sum_{k=0}^{r-1} \omega^{sk} |x^k\rangle =$$

$$\frac{1}{r} \cdot \sum_{k=0}^{r-1} |x^k\rangle \cdot \sum_{s=0}^{r-1} \omega^{sk} =$$

$$\frac{1}{r} \sum_{k=0}^{r-1} r \cdot \delta_{k,0} |x^k\rangle = 1$$

$$=$$

$$U|1\rangle = \frac{1}{r} \cdot \sum_{s=0}^{r-1} U|u_s\rangle =$$

$$= \frac{1}{r} \cdot \sum_{s=0}^{r-1} \omega^{-s} \cdot |u_s\rangle \qquad \omega^{-s} = e^{\frac{2\pi i s}{r}}$$

$\forall s$ stima fase $\varphi \cong s/r$

FFT.

Caso discreto: Fourier

- Valutazione veloce (F)
- Interpolazione veloce (F^{-1})

Esempio Algoritmo di valutazione - interpolazione per il calcolo del prodotto di due polinomi. $a(x) = \sum_{i=0}^{n} a_i x^i$

$b(x) = \sum_{i=0}^{n} b_i x^i \in F[x]$ $\quad N = 2n+1$

$$(a(x), b(x)) \xleftrightarrow{\text{mult. polinomiale}} c(x) = u(x)$$

valutazione $\downarrow$ $\alpha_1 \dots \alpha_N$ $\qquad$ interpolazione $\uparrow$ $\deg u < N,\ u(\alpha_k) = c_k$

$$\{A_k = a(\alpha_k), B_k = b(\alpha_k)\} \longrightarrow \{C_k = A_k B_k\}$$

Problema P_N di dimensione N

Valutare un polinomio $a(x) = \sum_{0}^{N-1} a_i x^i$

in N punti distinti $\alpha_k \in F$

La soluzione: $\{A_k = a(\alpha_k)\}_k$

Chiamiamo $M(N)$ il numero

di moltiplicazioni sse F

Prop. P_N può essere risolto

con $M(N) = N^2 + O(N)$. $\forall \{\alpha_1 \dots \alpha_N\}$

Dim. Per ogni α_k calcoliamo

$$A(\alpha_k) = (((a_{N-1}\alpha_k + a_{N-2})\alpha_k + a_{N-3})\alpha_k +$$

$$\dots + a_1)\alpha_k + a_0 \quad \text{(Horner)}$$

Se però scegliamo i punti di valutazione $E_N = \{\alpha_k\}$ "meglio" possiamo migliorare

Def. Sia $N = 2n$. Diciamo che $E_N = \{\alpha_k\}$ ha la proprietà S se $E_N = \{\pm \alpha_k\}_0^{n-1}$

Se E_N soddisfa S, dal momento che $(-\beta)^2 = \beta^2$ ci sono solo $N/2$ quadrati distinti in E_N

Da questa osservazione otteniamo

Proposizione. Sia $N = 2n$ e $E_N = \{\alpha_k\}$ soddisfi (S), allora P_N può essere risolto in

$$M(N) = N^2/2 + O(N)$$

Dim. Decomponiamo $a(x) = \sum_0^{N-1} a_i x^i$

come • $a(x) = b(y) + x\, c(y)$ con $y = x^2$, $b(y) = \sum_{i=0}^{n-1} a_{2i}\, y^i$ $c(y) = \sum_0^{n-1} a_{2i+1}\, y^i$

allora prendiamo valutare

$A_k = a(\alpha_k)$ e $A_{-k} = a(-\alpha_k)$

nel seguente modo

Algoritmo (spettamento binario)

1. Calcola $\beta_k = \alpha_k^2$ $\quad k=0,\dots,n-1$

2. Usa Hner per calcolare

$$B_k = b(\beta_k)$$
$$C_k = c(\beta_k) \qquad k=0,\dots,n-1$$

3. Ritorna $A_k = B_k + \alpha_k C_k$

$$A_{-k} = B_k - \alpha_k C_k \qquad k=0,\dots,n-1$$

I passi 1 e 3 richiedono $O(N/2)$ moltiplicazioni mentre 2 con Hner $2\left(\frac{N}{2}\right)^2 + O(N/2)$ moltiplicazioni

$\Rightarrow M(N) = N^2/2 + O(N)$.

Migliorato ma sempre $O(N^2)$

Se scegliamo i punti di valutazione come:

$$[\omega] = \{1, \omega, \dots, \omega^{N-1}\}$$

con ω radice primitiva N-ma dell'unità anche meglio!

Ricordiamo che se $N = 2n$

1. $\omega^{k+n} = -\omega^k$ → prop. 5

2. ω^2 radice primitiva $N/2$-ma dell'unità.

⇒ se $N = 2^m$ otteniamo

$F_N = [\omega]$ risicano di Fourier

FFT (N, a, ω)

input : $N = 2^m$, $a(x) = \sum_{0}^{N-1} a_i x^i$, ω radice primitiva N-ma di 1

output : $A = (A_0, \ldots, A_{N-1})$ $A_k = a(\omega^k)$

if $N = 1$ then $A_0 = a_0$

else

$n = N/2$

$b(x) := \sum a_{2i} x^i$

$c(x) := \sum a_{2i+1} x^i$

$B := FFT(n, b, \omega^2)$

$C := FFT(n, c, \omega^2)$

for $k = 0 \ldots n-1$ repeat

$A_k := B_k + \omega^k C_k$

$A_{k+n} := B_k - \omega^k C_k$

return $A = (A_0, \ldots, A_{N-1})$

Vediamo che l'algoritmo è corretto per ogni n, $N = 2^n$

Se $n = 0$, allora $N = 1$ e $A = a(\omega^0) = a_0$

($a(x) = a_0$ se $N = 1$). ✓

Supponiamo l'algoritmo corretto per $n \geq 0$ e verifichiamo per $n+1$

$N = 2^{n+1}$. Quindi eseguiamo lo spezzamento binario e

costruiamo $b(x)$ e $c(x)$ t.c.

$$a(x) = b(x^2) + x\,c(x^2)$$

di grado $n-1$ e la chiamata ricorsiva rende

$$B_k = b(\omega^{2k}) \qquad C_k = c(\omega^{2k})$$

Combinando

$$A_k = B_k + \omega^k C_k$$

$$= b(\omega^{2k}) + \omega^k c(\omega^{2k})$$

$$= a(\omega^k)$$

$$A_{k+n} = B_k - \omega^k C_k =$$

$$= b(\omega^{2k}) - \omega^k C_k =$$

$$= b((\omega^{k+n})^2) + \omega^{k+n} c((\omega^{k+n})^2) =$$

$$= a(\omega^{k+n})$$

$$\Rightarrow FFT(2^{m+1}, a, \omega) \longrightarrow A = (A_0, \ldots, A_{N-1})$$

em $A_k = a(\omega^k)$ ✓

Th. la FFT richiede

$$M(N) = N/2 \log_2 N$$

moltiplicazioni per F per valutare per $F_N = [\omega]$ risiesime di Fourier.

Dim. La parte "else" della procedura richiede

$$M(N) = 2M(N/2) + N/2$$

($\overrightarrow{2\,FFT}$ / ricombinare)

$$\Rightarrow M(2^m) = 2M(2^{m-1}) + 2^{m-1}$$

iterando troviamo

$$M(2^m) = 2^m \underbrace{M(1)}_{=0} + m 2^{m-1}$$

$$\Rightarrow M(N) = N/2 \log_2(N) \quad \checkmark$$

Vediamo ora la teoria di interpolazione rispetto ai punti di Fourier.

∴ Dati $b_k = a(\alpha_k)$ trovare $a(x)$

Se introduciamo la matrice (di Vandermonde)

$$V = V(\alpha_0, \dots, \alpha_{N-1}) = \begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \dots & \alpha_0^{N-1} \\ 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{N-1} \\ \vdots & \vdots & & & \\ 1 & \alpha_{N-1} & \alpha_{N-1}^2 & & \alpha_{N-1}^{N-1} \end{bmatrix}$$

e $\underline{a} = (a_0, \dots, a_{N-1})$ $\underline{b} = (b_0, \dots, b_{N-1})$

Si ha

$$V\underline{a} = \underline{b} \iff b_k = a(\alpha_k)$$

Dal momento che per interpolare

$\forall i \neq \alpha_j \quad i \neq j$, la matrice

V è invertibile e quindi

$$V\underline{a} = \underline{b} \iff \underline{b} = V\underline{a} \iff b_k = a(\alpha_k)$$

Se consideriamo $[\omega] = \{1, \omega, \dots, \omega^{N-1}\}$

allora $V[\omega]^{-1} = \frac{1}{N} V[\bar{\omega}^1]$ ←

infatti vale:

$$\sum_{k=0}^{N-1} \omega^{ik} \cdot \omega^{-kj} = N\delta_{ij}$$

$$\sum_{k=0}^{N-1} \omega^{k(i-j)} = \begin{cases} N & i=j \\ \underbrace{\frac{1-(\omega^k)^N}{1-\omega^k}}_{=0} & \end{cases} =$$

Interpolazione -

FFI$(N, \omega, \underline{b})$

Input : $N = 2^m$, ω radice primitiva N-ma

$\underline{b} = (b_0, \dots, b_{N-1}) \in F^N$

output. $a(x) = \sum_0^{N-1} a_i x^i$, $a(\omega^k) = b_k$

$b(x) = \sum_0^{N-1} b_i x^i$

$A = FFT(N, b, \omega^{-1})$

$a(x) := \sum_{i=0}^{N-1} \frac{A_i}{N} x^i$

Return $a(x)$.

Per le varie applicazioni servirà

Dati $x_0, \dots, x_{N-1} \in \underline{\mathbb{C}}$ la FFT

calcola $y_0, \dots, y_{N-1} \in \mathbb{C}$ secondo

la legge $y_k = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} x_i \omega^{ik}$

con $\omega = e^{\frac{2\pi i}{N}}$

Quantum FFT

È l'operatore lineare che agisce su $|0\rangle, \ldots |N-1\rangle$ nel seguente modo:

$\omega = \frac{2\pi i}{N}$

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \omega^{jk} \cdot |k\rangle$$

e quindi su uno stato arbitrario

$$|u\rangle = \sum_{j=0}^{N-1} x_j |j\rangle \Rightarrow \sum_{k=0}^{N-1} y_k |k\rangle =$$

$$= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \left(\sum_{j=0}^{N-1} x_j \omega^{jk} \right) |k\rangle$$

Se considériamo la matrice

$U_\omega = \frac{1}{\sqrt{N}} V([\omega])$, Dato che $U_\omega^{-1} = U_\omega^{\dagger}$

otteniamo che è unitario.

Vogliamo costruire un circuito

che calcola la Q-FFT

Supponiamo $N = 2^m$

e consideriamo la base

$|0\rangle, \dots, |2^n - 1\rangle$ per un n-qbit

quantum computer.

Notazioni Dato lo stato $|j\rangle$ consideriamo

$j = j_1 j_2 \dots j_n$ la sua rappresentazione

binaria

$j = j_1 \cdot 2^{n-1} + j_2 2^{n-2} + \dots + j_n 2^0$.

È conveniente definire anche

$$0.j_\ell \cdots j_m = \frac{j_\ell}{2} + \cdots + \frac{j_m}{2^{m-\ell+1}}$$

oss.

$$0.j_\ell \cdots j_m \equiv \frac{j}{2^{m-\ell+1}} \quad (\mathbb{Z})$$

infatti

$$\frac{j_1 2^{n-1} + j_2 2^{n-2} + \cdots + j_\ell 2^{n-\ell} + \cdots j_m}{2^{\ell+1}} =$$

$$\underbrace{j_1 2^{n-\ell} + \cdots + j_{\ell-1}}_{c \in \mathbb{Z}} + \frac{j_\ell}{2} + \frac{j_{\ell+1}}{4} + \cdots \frac{j_m}{2^{n-\ell+1}}$$

e quindi in particolare

$$e^{\frac{2\pi i j}{2^{\ell+1}}} = e^{2\pi i \cdot c} \cdot e^{2\pi i \left(\frac{j_\ell}{2} + \cdots + \frac{j_m}{2^{n-\ell+1}}\right)} =$$

$$\| \qquad\qquad e^{2\pi i\, 0.j_\ell \cdots j_m}$$

$$1$$

Vediamo ora come scrivere la FFT di $|j\rangle$

$$|j\rangle = |j_1 \cdots j_n\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{2^n-1} \omega^{jk} |k\rangle$$

$$\frac{1}{\sqrt{N}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} e^{2\pi i j \left(\sum_{l=1}^{n} k_l 2^{-l}\right)} |k_1 \cdots k_n\rangle$$

$$= \frac{1}{\sqrt{N}} \sum_{k_1=0}^{1} \sum_{k_2=0}^{1} \cdots \sum_{k_n=0}^{1} \bigotimes_{l=1}^{n} e^{2\pi i j k_l 2^{-l}} |k_l\rangle$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \sum_{k_l=0}^{1} e^{2\pi i j k_l 2^{-l}} |k\rangle =$$

$$= \frac{1}{\sqrt{N}} \bigotimes_{l=1}^{n} \left(|0\rangle + e^{2\pi i j 2^{-l}} |1\rangle \right) =$$

$$= \frac{1}{\sqrt{N}} \left(|0\rangle + e^{2\pi i 0.j_1} |1\rangle \right) \cdots \left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n} |1\rangle \right)$$

Questa rappresentazione ci aiuta a scrivere un circuito

Sia $R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{bmatrix}$

allora $R_1 = \begin{bmatrix} 1 & 0 \\ 0 & e^{\pi i} \end{bmatrix} = H$

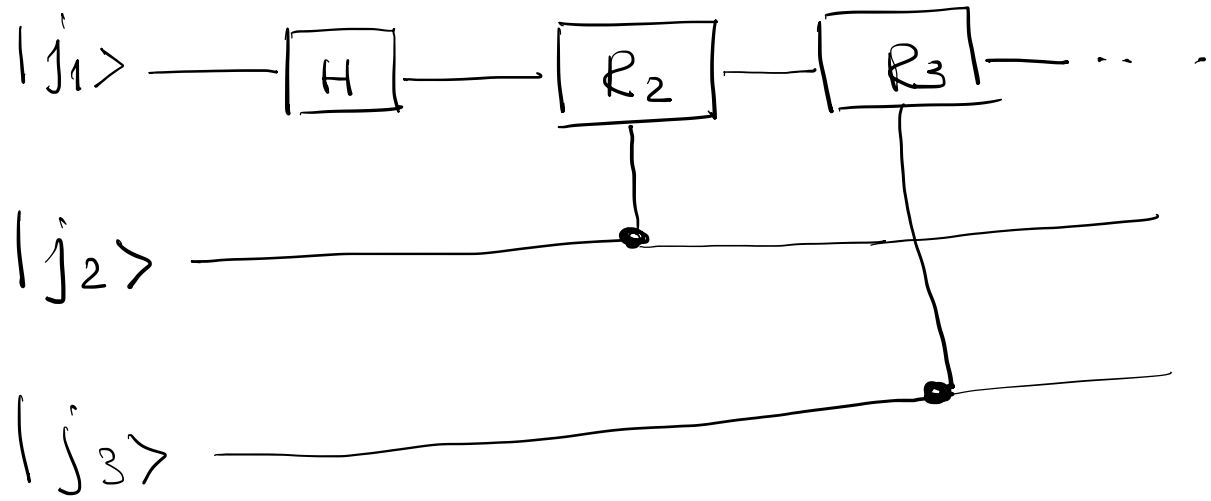
Dato $|j\rangle = |j_1\rangle|j_2\rangle \cdots |j_n\rangle$ applichiamo al primo bit $R_1 = H$ otteniamo (non consideriamo $|j_2 \cdots j_n\rangle$)

$$H|j_1\rangle = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) & |j_1\rangle = 0 \\ \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) & |j_1\rangle = 1 \end{cases}$$

ossia $H|j_1\rangle = \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)$

dato che $e^{2\pi i 0.j_1} = e^{\frac{2\pi i j_1}{2}} = \begin{cases} 1 & j_1 = 0 \\ -1 & j_1 = 1 \end{cases}$

Poi applichiamo CR_2



$$CR_2 \quad |j_2\rangle |H j_1\rangle = \begin{cases} |0\rangle |H j_1\rangle \\ |1\rangle |R_2 H j_1\rangle \end{cases}$$

Se $j_2 = 0$ $\quad |0\rangle + e^{2\pi i\, 0.j_1} |1\rangle$

se $j_2 = 1$ $\quad R_2|0\rangle + R_2 e^{2\pi i 0.j_1} |1\rangle =$

$$|0\rangle + e^{\frac{2\pi i}{4}} \cdot e^{2\pi i\, 0.j_1} |1\rangle =$$

$$|0\rangle + e^{2\pi i\, 0.j_1 j_2} |1\rangle .$$

$0.j_1 j_2 = \frac{j_1}{2} \times \frac{j_2}{4}$

Continuiamo applicando $CR_3, \ldots, CR_n$

Ognuno aggiunge un bit alla fase di $|1\rangle$

in conclusione

$$|j_1\rangle \longrightarrow \frac{1}{\sqrt{2}}|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle$$

Ricordiamo che in realtà

$$|j_1 \cdots j_n\rangle \longrightarrow \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle\right)|j_2 \cdots j_n\rangle$$

Poi facciamo una cosa simile sul secondo bit: H mette il sistema nello stato

$$\frac{1}{2}\left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2}|1\rangle\right)|j_3 \cdots j_n\rangle$$

e applicando i CR_k si ottiene

$$\frac{1}{2}\left(|0\rangle + e^{2\pi i 0.j_1 \cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2 \cdots j_n}|1\rangle\right)|j_3 \cdots j_n\rangle$$

Ottenuto lo stato si ottiene:

$$\frac{1}{\sqrt{N}}\left(|0\rangle + e^{2\pi i\, 0.j_1 \cdots j_n}|1\rangle\right)\cdot\left(|0\rangle + e^{2\pi i\, 0.j_2}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i\, 0.j_n}|1\rangle\right)$$

Dobbiamo poi invertire l'ordine dei qbits.

Quanti cancelli abbiamo usato?

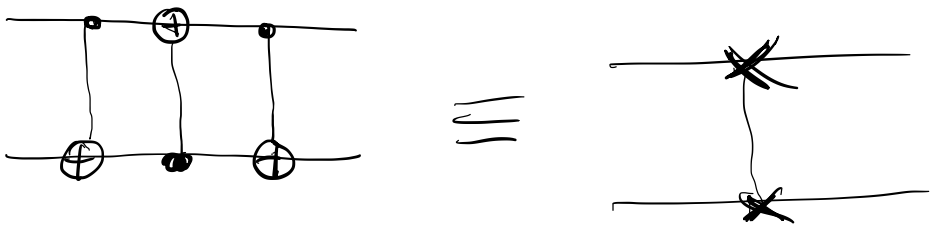
1° qbit	$H + (n-1)\ CR_k$	n
2° qbit	$H + (n-2)\ CR_k$	$n-1$
⋮		⋮

→ in totale $\frac{n(n+1)}{2}$ + inversione

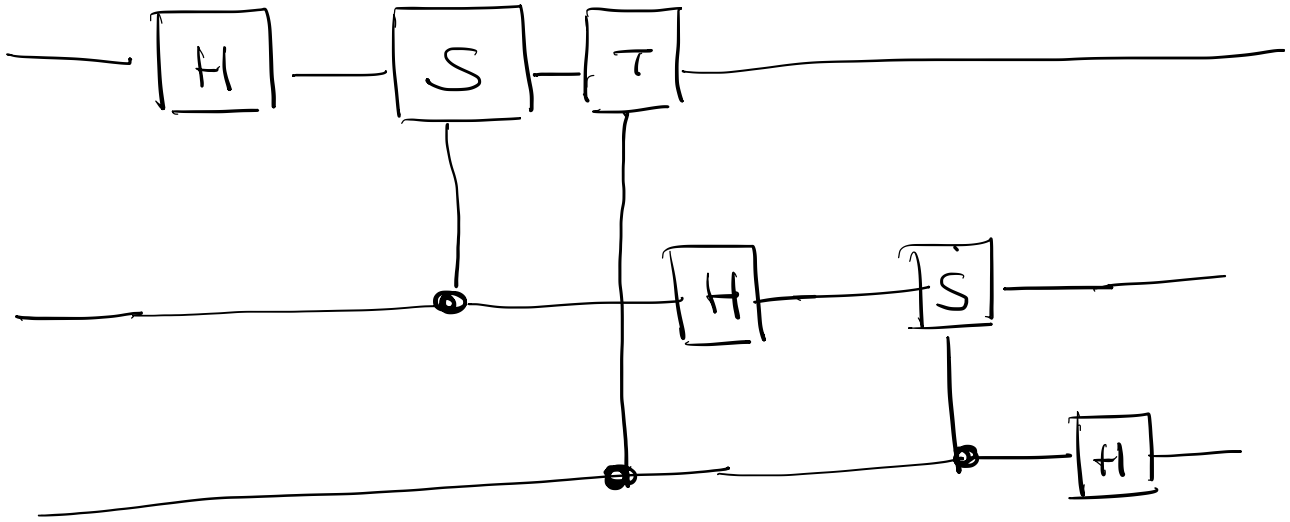
Servono al più $n/2$ scambi

e ognuno 3 CNOT $\Rightarrow \Theta(n^2)$

$$|a,b\rangle \to |a, a\oplus b\rangle \to |a\oplus(a\oplus b), a\oplus b\rangle = |b, a\oplus b\rangle$$
$$\to |b, (a\oplus b)\oplus b\rangle = |b, a\rangle$$



Esempio 3-qbits



dove $S = R_2 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

$T = R_4 = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$