OSNOVE KVANTNEGA RAČUNALNIŠTVA, 2. DEL

MATIJA PRETNAR

Fakulteta za matematiko in fiziko
Univerza v Ljubljani

Math. Subj. Class. (2010): 68Q12, 81P68

V drugem delu članka si ogledamo Deutschev algoritem, ki je bil prvi kvantni algoritem, ter najznamenitejša kvantna algoritma: Groverjev algoritem za iskanje v neurejeni tabeli in Shorov algoritem za razcep na praštevila.

THE BASICS OF QUANTUM COMPUTING, PART 2

The second part looks at Deutsch's algorithm, which was the first quantum algorithm, and at two most famous quantum algorithms: Grover's search algorithm and Shor's factorization algorithm.

Simulacija klasičnih vezij

Preden se začnemo navduševati nad učinkovitostjo kvantnih računalnikov, najprej preverimo, ali lahko z njimi res izračunamo vse, kar bi želeli. Torej, za vsako funkcijo $f\colon \{0,1\}^m \to \{0,1\}^n$, ki jo znamo izračunati na običajnem računalniku, želimo poiskati ustrezno unitarno preslikavo oziroma kvantno vezje U_f, ki bo dajala enake odgovore.

Kaj pomeni, da bodo odgovori enaki? Če nastavimo kubite na začetno bazno stanje $|x\rangle = |x_0 x_1 \cdots x_{m-1}\rangle$, kjer so x_i posamezni vhodni biti, potem želimo s pomočjo U_f izračunati stanje $|f(x)\rangle = |y_0 y_1 \cdots y_{n-1}\rangle$. Toda kako lahko z unitarnimi preslikavami izračunamo funkcijo, ki nima inverza? Še več: kaj, če funkcija f nima enakega števila vhodnih in izhodnih bitov?

Obema težavama se izognemo tako, da vezje poleg vhoda $|x\rangle$ sprejme še nekaj dodatnih kubitov, na katere bomo shranili izhod $|f(x)\rangle$. Za začetek si oglejmo primer, ko f izračuna en bit, torej ko je $n = 1$. Vezje U_f tedaj iz stanja $|x\rangle|0\rangle$ izračuna stanje $|x\rangle|f(x)\rangle$. Natančneje: iz stanja $|x\rangle|b\rangle$ bomo izračunali stanje $|x\rangle|b \oplus f(x)\rangle$, kjer je *ekskluzivni ali* $\oplus$ podan z:

$$0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 0 = 1 \quad 1 \oplus 1 = 0.$$

Iz enakosti $\textsc{cnot}|x\rangle|b\rangle = |x\rangle|b \oplus x\rangle$ izvira tudi oznaka za CNOT v kvantnih vezjih.

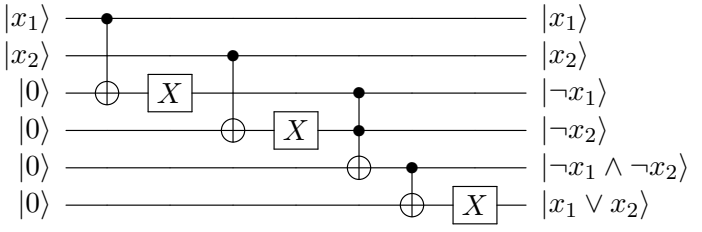
Če je izhodnih kubitov več, ravnamo podobno: iz vhoda $|x\rangle|b\rangle$, kjer je b zdaj zaporedje n kubitov, enako izračunamo $|x\rangle|b \oplus f(x)\rangle$, le da tokrat $\oplus$ deluje po posameznih kubitih.

Na splošni superpoziciji vezje U_f deluje linearno: stanje $\sum_{x,b} \alpha_{x,b}|x\rangle|b\rangle$, kjer $\sum_{x,b}$ označuje vsoto po vseh 2^m možnih baznih stanjih $|x\rangle$ in vseh 2^n možnih stanjih kubitov za odgovor $|b\rangle$, spremeni v $\sum_{x,b} \alpha_{x,b}|x\rangle|b \oplus f(x)\rangle$. Običajno pa bodo kubiti za odgovor nastavljeni na $|0\rangle$, zato bo odgovor kar superpozicija $\sum_x \alpha_x|x\rangle|f(x)\rangle$.

Ali lahko na tak način simuliramo vsa klasična vezja? Vemo, da lahko vsa taka vezja sestavimo iz negacije in konjunkcije. Negacijo simuliramo tako, da s kontrolirano negacijo stanje kontrolnega kubita presnamemo na ciljni kubit, nato pa ciljni kubit negiramo z vrati X. Zaradi izreka o nepodvajanju seveda ne moremo presneti splošnega stanja, le bazni stanji $|0\rangle$ in $|1\rangle$, a to je za simulacijo klasičnega vezja dovolj.

Za konjunkcijo pa uporabimo *Toffolijeva vrata* T. Ta delujejo podobno kot CNOT, le da imajo dva kontrolna in en ciljni kubit, ki ga negiramo takrat, ko sta oba kontrolna kubita enaka $|1\rangle$. Velja torej $T|110\rangle = |111\rangle$ in $T|111\rangle = |110\rangle$, vsa druga bazna stanja pa vrata T pustijo pri miru.

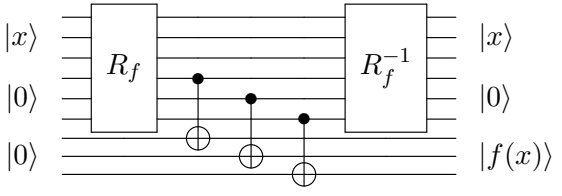
S pomočjo teh dveh vrat lahko izrazimo vsa preostala vezja, na primer disjunkcijo. Disjunkcijo lahko s pomočjo negacije in konjunkcije izrazimo kot $x_1 \vee x_2 = \neg(\neg x_1 \wedge \neg x_2)$, to pa implementiramo z naslednjim vezjem:



(1)

Pri tem smo zaradi sorodnega delovanja vrata T označili podobno kot CNOT.

Na tak način vsako klasično funkcijo f simuliramo s kvantnim vezjem R_f, ki poleg vhodnih podatkov dobi še nekaj kubitov delovnega prostora, poleg odgovora pa vrne še nekaj *smeti*, ki so posledica pomožnih računov. Teh smeti se hočemo znebiti, saj zaradi prepletenosti vplivajo na stanja vhodnih in izhodnih kubitov. K sreči tudi tu obstaja rešitev: po uporabi vezja R_f odgovor presnamemo na dodatne kubite, nato pa uporabimo inverz R_f^{-1}, ki kubite z odgovorom in smetmi povrne v prvotno prazno stanje, na dodatnih kubitih pa ostane kopija odgovora.

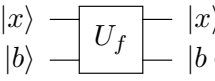


Ker so vsa vezja unitarna, vemo, da inverz obstaja. Toda kako ga izračunamo? Kot vemo, velja $(UV)^{-1} = V^{-1}U^{-1}$, zato inverz dobimo tako, da inverze posameznih vrat sestavimo v obratnem vrstnem redu. Ker so vrata X, Y, Z, H, CNOT in T sama svoj inverz, lahko vsako vezje, ki je sestavljeno iz njih, obrnemo kar tako, da ga izvedemo od desne proti levi (nekatere implementacije to enostavno dopuščajo).

Deutschev algoritem

Spodobi se, da si kot prvi kvantni algoritem ogledamo *Deutschev algoritem* [1], saj je bil to prvi kvantni algoritem, ki je deloval učinkoviteje od klasičnega. Recimo, da želimo za nam neznano funkcijo $f\colon \{0,1\} \to \{0,1\}$ ugotoviti, ali je konstantna, pri tem pa imamo na voljo le "črno škatlo", torej neko napravo, ki na nam neznan način računa vrednosti f.

Če je ta črna škatla klasično vezje, nam ne ostane drugega kot to, da jo uporabimo dvakrat, da izračunamo $f(0)$ in $f(1)$, nato pa odgovora primerjamo. Če pa imamo na voljo kvantno vezje U_f:



nam Deutschev algoritem odgovor izračuna z le eno uporabo U_f.

Glavni ideji sta dve. Prva je očitna: na vhodnem kubitu podamo superpozicijo $|+\rangle$, da hkrati izračunamo U_f na $|0\rangle$ in $|1\rangle$. Druga pa je bolj zvita: na izhodnem kubitu namesto $|0\rangle$ podamo $|-\rangle$. Po uporabi vezja na $|x\rangle|-\rangle$ je stanje namreč enako

$$U_f|x\rangle|-\rangle = \tfrac{1}{\sqrt{2}}U_f|x\rangle|0\rangle - \tfrac{1}{\sqrt{2}}U_f|x\rangle|1\rangle = \tfrac{1}{\sqrt{2}}|x\rangle|0 \oplus f(x)\rangle - \tfrac{1}{\sqrt{2}}|x\rangle|1 \oplus f(x)\rangle.$$

Če je $f(x) = 0$, je končno stanje enako $|x\rangle|-\rangle$, če pa je $f(x) = 1$, je izhodno stanje enako $-|x\rangle|-\rangle$. Torej v splošnem velja, da je izhodni kubit v stanju $(-1)^{f(x)}|x\rangle|-\rangle$.

Če za vhod podamo $|+\rangle$, dobimo

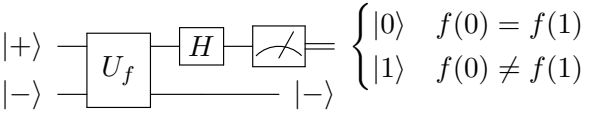
$$\begin{aligned} U_f|+\rangle|-\rangle &= \tfrac{1}{\sqrt{2}}U_f|0\rangle|-\rangle + \tfrac{1}{\sqrt{2}}U_f|1\rangle|-\rangle \\ &= \tfrac{1}{\sqrt{2}}(-1)^{f(0)}|0\rangle|-\rangle + \tfrac{1}{\sqrt{2}}(-1)^{f(1)}|1\rangle|-\rangle \\ &= (-1)^{f(0)}(\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{(-1)^{f(1)-f(0)}}{\sqrt{2}}|1\rangle)|-\rangle. \end{aligned}$$

Če je f konstantna in je $f(0) = f(1)$, je prvi kubit v stanju $|+\rangle$, sicer pa je v stanju $|-\rangle$. Kako ugotovimo, v katerem stanju je?

Po uporabi vezja na prvem kubitu uporabimo Hadamardova vrata H. Ta $|+\rangle$ slikajo v $|0\rangle$, $|-\rangle$ pa v $|1\rangle$, kar potem izmerimo na običajen način.

Podobno lahko stanje pomerimo v vsaki ortonormirani bazi, le namesto H moramo izbrati ustrezno unitarno preslikavo.

Celoten Deutschev algoritem izvedemo s sledečim vezjem:



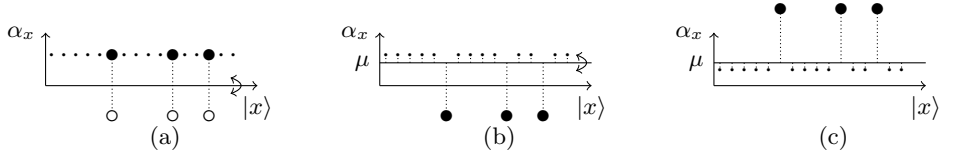
Če je funkcija f konstantna, izmerimo $|0\rangle$, če ni, pa $|1\rangle$.

Vprašanje, na katero odgovori Deutschev algoritem, ni ravno najbolj zanimivo, a v odgovoru nanj smo videli nekaj glavnih trikov, ki se uporabljajo v skoraj vseh kvantnih algoritmih.

Groverjev algoritem

Zanimivejša je naslednja naloga: za funkcijo $f\colon \{0, 1, \ldots, N-1\} \to \{0, 1\}$ želimo poiskati x, za katerega velja $f(x) = 1$. To je tako, kot bi v (papirnatem) telefonskem imeniku iskali osebo, ki živi na danem naslovu. Ker je imenik urejen po priimkih in ne po naslovih, nam ne ostane drugega, kot da ga preiščemo od začetka do konca. Če imamo srečo, osebo najdemo takoj, če ne, pa lahko šele čisto na koncu. Čas, ki ga potrebujemo za tako iskanje, narašča linearno z velikostjo imenika: za stokrat večji imenik potrebujemo stokrat več časa. Pravimo, da je časovna zahtevnost problema enaka $O(N)$.

Groverjev algoritem [3] pa isti problem na kvantnem računalniku reši s časovno zahtevnostjo $O(\sqrt{N})$, kar pomeni, da za stokrat večji imenik potrebujemo le desetkrat več časa. Ideja je sledeča: začnemo s superpozicijo vseh baznih stanj, nato pa amplitudi ustreznih stanj zamenjamo predznak, nazadnje pa vse amplitude »prezrcalimo« prek povprečja, s čimer se amplituda vseh ustreznih stanj poveča (slika 1).



Slika 1. Amplitude baznih stanj pri (a) začetnem stanju, (b) obratu faz ustreznih stanj in (c) zrcaljenju amplitud prek povprečja μ. Ustrezna stanja so označena z znakom •.

Ta dva koraka nato dovoljkrat ponovimo, na koncu pa izmerimo stanje. Z veliko verjetnostjo bo to izmerjeno stanje ravno eno od ustreznih. Ustreznost izmerjenega stanja nato preverimo še s prvotno funkcijo f, in če odgovor ni ustrezen, postopek ponovimo. Najprej moramo pokazati, da sta obrat faze in zrcaljenje prek povprečja unitarni preslikavi, nato pa še

ugotoviti, kolikokrat ju moramo ponoviti, da pridemo do odgovora. Zaradi enostavnosti bomo privzeli, da je $N = 2^n$, in stanja $|0\rangle, |1\rangle, \ldots, |N-1\rangle$ izenačili z vsemi baznimi stanji $|x\rangle$ sistema n kubitov.

Obrat faze F je seveda unitarna preslikava: vsa ustrezna bazna stanja imajo lastno vrednost -1, vsa preostala pa 1. Implementiramo jo tako, da izhodni kubit pri vezju U_f, ki računa funkcijo f, nastavimo na $|-\rangle$. Tako kot pri Deutschevem algoritmu vidimo, da je $U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$.

Zrcaljenje prek povprečja definiramo kot $D = 2|\psi\rangle\langle\psi| - I$, pri čemer je $|\psi\rangle = H^{\otimes n}|0^n\rangle$ superpozicija vseh 2^n baznih stanj, $|\psi\rangle\langle\psi|$ pa ortogonalni projektor na vektor $|\psi\rangle$, saj $|\varphi\rangle$ slika v $|\psi\rangle\langle\psi||\varphi\rangle = |\psi\rangle\langle\psi|\varphi\rangle = \langle\psi|\varphi\rangle|\psi\rangle$.

Preslikava D res zrcali prek povprečja μ, saj iz

$$\begin{aligned} D\sum_x \alpha_x|x\rangle &= \sum_x \alpha_x(2|\psi\rangle\langle\psi| - I)|x\rangle = \sum_x (2\alpha_x|\psi\rangle\langle\psi|x\rangle - \alpha_x|x\rangle) \\ &= 2\sum_{x'} \frac{\alpha_{x'}}{\sqrt{N}}|\psi\rangle - \sum_x \alpha_x|x\rangle = 2\sum_{x'} \frac{\alpha_{x'}}{\sqrt{N}} \sum_x \frac{1}{\sqrt{N}}|x\rangle - \sum_x \alpha_x|x\rangle \\ = 2\frac{\sum_{x'} \alpha_{x'}}{N} \sum_x |x\rangle - \sum_x \alpha_x|x\rangle &= 2\mu\sum_x |x\rangle - \sum_x \alpha_x|x\rangle = \sum_x (2\mu - \alpha_x)|x\rangle \end{aligned}$$

vidimo, da amplitudo α_x spremeni v $\mu - (\alpha_x - \mu) = 2\mu - \alpha_x$.

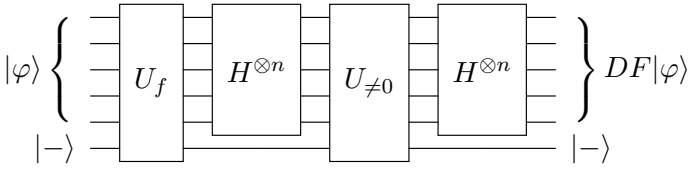
Ker je $|\psi\rangle = H^{\otimes n}|0^n\rangle$ in ker je $H^{\otimes n}$ sebi adjungirana, lahko pišemo:

$$D = 2|\psi\rangle\langle\psi| - I = 2H^{\otimes n}|0^n\rangle\langle 0^n|H^{\otimes n} - H^{\otimes n}H^{\otimes n} = H^{\otimes n}(2|0^n\rangle\langle 0^n| - I)H^{\otimes n}.$$

Preslikava $2|0^n\rangle\langle 0^n| - I$ bazni vektor $|0^n\rangle$ slika v $|0^n\rangle$, vse druge bazne vektorje $|x\rangle$ pa v $-|x\rangle$, zato je unitarna, torej je unitarna tudi preslikava D.

Poleg tega tudi vidimo, kako implementiramo D. Med dvoje Walsh-Hadamardovih vezij vrinemo vezje, ki obrne fazo vsem baznim stanjem, različnim od $|0^n\rangle$. To spet lahko dosežemo tako, da na izhodnem kubitu $|-\rangle$ uporabimo vezje, ki $|0^n\rangle$ slika v $|1\rangle$, preostala bazna stanja $|x\rangle$ pa v $|0\rangle$. Tako vezje lahko naredimo s konjunkcijo negacij vseh vhodnih kubitov.

Vezje, ki izvede celoten korak Groverjevega algoritma, je torej:



Kolikokrat moramo izvesti ta korak? Vidimo, da se vsem ustreznim stanjem amplituda spreminja hkrati. Podobno velja za vsa neustrezna stanja. Privzemimo, da vemo, da je vseh ustreznih stanj p. Tedaj lahko definiramo superpozicijo ustreznih stanj $|\varphi_1\rangle = \frac{1}{\sqrt{p}} \sum_{x,f(x)=1} |x\rangle$ in superpozicijo

neustreznih stanj $|\varphi_0\rangle = \frac{1}{\sqrt{N-p}} \sum_{x,f(x)=0} |x\rangle$. Tedaj je superpozicija vseh stanj enaka $|\psi\rangle = \sqrt{\frac{p}{N}}|\varphi_1\rangle + \sqrt{\frac{N-p}{N}}|\varphi_0\rangle$. Če označimo $\sqrt{\frac{p}{N}} = \sin\vartheta$, velja $|\psi\rangle = \sin\vartheta|\varphi_1\rangle + \cos\vartheta|\varphi_0\rangle$.

Očitno velja, da obrat faze F deluje kot $F|\varphi_1\rangle = -|\varphi_1\rangle$ in $F|\varphi_0\rangle = |\varphi_0\rangle$, saj vsem ustreznim stanjem obrne fazo, neustrezna pa pusti pri miru.

Za zrcaljenje prek povprečja D pa lahko pokažemo

$$\begin{aligned} D|\varphi_1\rangle &= 2|\psi\rangle\langle\psi|\varphi_1\rangle - |\varphi_1\rangle = 2\sin\vartheta|\psi\rangle - |\varphi_1\rangle \\ &= 2\sin\vartheta(\sin\vartheta|\varphi_1\rangle + \cos\vartheta|\varphi_0\rangle) - |\varphi_1\rangle \\ &= -\cos 2\vartheta|\varphi_1\rangle + \sin 2\vartheta|\varphi_0\rangle \end{aligned}$$

in podobno $D|\varphi_0\rangle = \sin 2\vartheta|\varphi_1\rangle + \cos 2\vartheta|\varphi_0\rangle$. Torej za korak $G = DF$ velja

$$\begin{aligned} G|\varphi_1\rangle &= -D|\varphi_1\rangle = \cos 2\vartheta|\varphi_1\rangle - \sin 2\vartheta|\varphi_0\rangle, \\ G|\varphi_0\rangle &= D|\varphi_0\rangle = \sin 2\vartheta|\varphi_1\rangle + \cos 2\vartheta|\varphi_1\rangle, \end{aligned}$$

zato je G rotacija za kot 2ϑ.

Iz začetnega stanja $|\psi\rangle = \sin\vartheta|\varphi_1\rangle + \cos\vartheta|\varphi_0\rangle$ po k rotacijah tako pridemo v stanje $G^k|\psi\rangle = \sin(2k+1)\vartheta|\varphi_1\rangle + \cos(2k+1)\vartheta|\varphi_0\rangle$. Če izberemo k, za katerega bo $(2k+1)\vartheta$ blizu $\frac{\pi}{2}$, bo v končnem stanju prevladovala superpozicija $|\varphi_1\rangle$ in z veliko verjetnostjo bomo izmerili eno od ustreznih stanj. Veljati mora torej $k \approx \frac{1}{2}(\frac{\pi}{2\vartheta} - 1)$, zato izberemo kar $k = \lfloor \frac{\pi}{4\vartheta} \rfloor$.

Če je ustreznih stanj malo (kar je klasično najtežje), je $\sin\vartheta = \sqrt{\frac{p}{N}}$ majhno število. Tedaj velja $\vartheta \approx \sin\vartheta$, zato moramo izvesti približno $\frac{\pi}{4}\sqrt{\frac{N}{p}}$ korakov, zaradi česar je časovna zahtevnost Groverjevega algoritma enaka $O(\sqrt{N})$.

Kaj pa, če ne vemo, koliko je p? Tedaj lahko najprej poskusimo primere, ko je $p = 1, 2, 4, 8, \ldots$, in z veliko verjetnostjo bomo v vsaj enem od poskusov izmerili ustrezen odgovor. Poleg tega pa obstajajo algoritmi, ki ocenijo p, njihova časovna zahtevnost pa je prav tako $O(\sqrt{N})$.

Shorov algoritem

Za konec si oglejmo še najznamenitejši kvantni algoritem: *Shorov algoritem* za praštevilski razcep [4]. V resnici bomo reševali osnovnejšo nalogo: namesto razcepa števila N bomo iskali neki njegov pravi delitelj, torej število $1 < d < N$, ki deli N. Kajti če najdemo tak d, lahko postopek ponovimo na d in N/d ter tako sčasoma pridemo do celotnega praštevilskega razcepa.

Število d najenostavneje poiščemo tako, da pregledamo vsa naravna števila, ki so večja od 2 in manjša od $\sqrt{N}$, ter se ustavimo, ko eno izmed njih

deli N. Časovna zahtevnost takega postopka je torej $O(\sqrt{N})$: za štirikrat večje število potrebujemo dvakrat več časa. Seveda obstajajo tudi učinkovitejši algoritmi, a vseeno iz praštevil p in q veliko laže izračunamo $p \cdot q$ kot pa iz zmnožka $p \cdot q$ dobimo nazaj razcep na p in q. Šifriranje RSA, ki je danes precej pogosto uporabljano, se zanaša ravno na težavnost praštevilskega razcepa. Časovna zahtevnost Shorovega algoritma pa je le $O((\log N)^3)$. Hitrost takega razcepa si najbolje predstavljamo s primerjavo časov, ki jih za razcep potrebujejo različni algoritmi (tabela 1).

velikost števila	$\approx 10^6$	$\approx 10^{10}$	$\approx 10^{20}$	$\approx 10^{30}$	$\approx 10^{40}$	$\approx 10^{50}$
naivni algoritem	1 ms	100 ms	3 h	30 let	$3 \cdot 10^6$ let	$3 \cdot 10^{11}$ let
algoritem GNFS	1 ms	75 ms	3 min	10 h	1 mesec	6 let
Shorov algoritem	1 ms	4 ms	40 ms	125 ms	300 ms	600 ms

Tabela 1. Časi, potrebni za razcep števila z naivnim algoritmom ($O(\sqrt{N})$), trenutno najučinkovitejšim klasičnim algoritmom GNFS ($O(e^{(64/9 \log N)^{1/3} (\log \log N)^{2/3}})$) in s Shorovim algoritmom ($O((\log N)^3)$). Za lažjo primerjavo predpostavimo, da vsi trije algoritmi za razcep števil okoli milijona potrebujejo eno milisekundo.

Shorov algoritem temelji na ideji, da poiščemo naravno število x, za katero velja $x^2 \equiv 1 \pmod{N}$. Tedaj je $x^2 - 1$ deljivo z N, zato ima vsaj eno od števil $x - 1$ in $x + 1$ skupni delitelj z N, ki pa ga lahko hitro izračunamo z Evklidovim algoritmom. Število x poiščemo v zaporedju

$$a \bmod N, a^2 \bmod N, a^3 \bmod N, \ldots,$$

kjer je a neko število, ki je tuje N.

Ker je vseh ostankov pri deljenju z N le končno mnogo, se mora neki člen v zaporedju ponoviti. Recimo, da za števili m in r velja $a^m \equiv a^{m+r} \pmod{N}$. Tedaj velja $a^m(a^r - 1) \equiv 0 \pmod{N}$, in ker je a in s tem tudi a^m tuje N, je $a^r \equiv 1 \pmod{N}$. Zato je r perioda zaporedja, in če je število r sodo, bo $a^{r/2}$ ravno iskani x.

Za primer vzemimo $N = 21$ in $a = 10$. Tedaj je zaporedje ostankov potenc enako $10, 16, 13, 4, 19, 1, 10, 16, \ldots$ Perioda zaporedja je enaka 6 in iskani x je enak $10^3 \equiv 13 \pmod{21}$. Torej je število $13^2 - 1$ deljivo z 21 in iskani delitelj najdemo tako z $D(12, 21) = 3$, kot z $D(14, 21) = 7$.

Če je število r liho, pa tak prijem ne deluje, zato postopek ponovimo za neki drug a. S tem in drugimi manjšimi zapleti se tu ne bomo ukvarjali, temveč se bomo posvetili ključnemu delu Shorovega algoritma: izračunu periode. Kogar tema bolj zanima, si lahko vse podrobnosti ogleda v [2].

Posamezne člene zaporedja $a^j \bmod N$ lahko izračunamo precej učinkovito. Če je j sodo, velja $a^j = (a^{j/2})^2$, sicer pa velja $a^j = a \cdot (a^{(j-1)/2})^2$. Tako postopoma zelo hitro izračunamo poljubno potenco. Na primer, a^{42}

izračunamo iz a^{21}, to iz a^{10}, to iz a^5, to iz a^2 in to iz a. Poleg tega ves čas računamo le z ostanki, kar nam stvari še olajša, saj nam ni treba delati s števili, večjimi od N.

A kljub temu s postopnim računanjem členov ne bomo prišli daleč, saj je perioda lahko skoraj tako velika kot N. Na tej točki prvič uporabimo prednosti kvantnih računalnikov: členov ne bomo računali drugega za drugim, temveč vse hkrati.

Klasično vezje, ki potence računa po zgornjem postopku, namreč znamo pretvoriti v kvantno vezje U, za katero velja $U|j\rangle|0\rangle = |j\rangle|a^j \bmod N\rangle$. Nato to vezje uporabimo na superpoziciji $\frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} |j\rangle|0\rangle$ za neko dovolj veliko število M, da dobimo superpozicijo

$$\tfrac{1}{\sqrt{M}}\big(|0\rangle|a^0 \bmod N\rangle + |1\rangle|a^1 \bmod N\rangle + \cdots + |M-1\rangle|a^{M-1} \bmod N\rangle\big).$$

Nato pomerimo izhodni register. Izmerili bomo $|a^m \bmod N\rangle$ za neki naključno izbran in nam neznan $m < r$. Ta vrednost nam sicer ne bo pomagala, vendar bomo s tem dosegli, da se bo na vhodu obdržala le superpozicija tistih stanj $|j\rangle$, za katera je $a^j \bmod N = a^m \bmod N$. Stanje vhoda bo torej

$$|\varphi\rangle = \tfrac{1}{\sqrt{n}}(|m\rangle + |m+r\rangle + |m+2r\rangle + \cdots + |m+(n-1)r\rangle),$$

pri čemer je $n = \lfloor \frac{M-m}{r} \rfloor$ število vseh stanj v superpoziciji. Ta superpozicija pa je sestavljena ravno iz baznih stanj, razmaknjenih za periodo r.

Na primer, za $N = 21$, $a = 10$ in $M = 64$ iz začetne superpozicije $\frac{1}{8} \sum_{j=0}^{63} |j\rangle|0\rangle$ z uporabo vezja U dobimo stanje

$$\begin{aligned}\tfrac{1}{8}\big(&|0\rangle|1\rangle + |1\rangle|10\rangle + |2\rangle|16\rangle + |3\rangle|13\rangle + |4\rangle|4\rangle + |5\rangle|19\rangle + |6\rangle|1\rangle + \\ &|7\rangle|10\rangle + |8\rangle|16\rangle + |9\rangle|13\rangle + \cdots + |61\rangle|10\rangle + |62\rangle|16\rangle + |63\rangle|13\rangle\big).\end{aligned}$$

Recimo, da pri meritvi izhoda izmerimo $|13\rangle$. Tedaj je stanje vhodnega registra enako $|\varphi\rangle = \frac{1}{\sqrt{11}}(|3\rangle + |9\rangle + |15\rangle + \cdots + |63\rangle)$. Če bi pri meritvi izmerili kak drug izhod, bi dobili drugačno stanje vhoda, a v vsakem primeru bi bila stanja v superpoziciji razmaknjena ravno za periodo r.

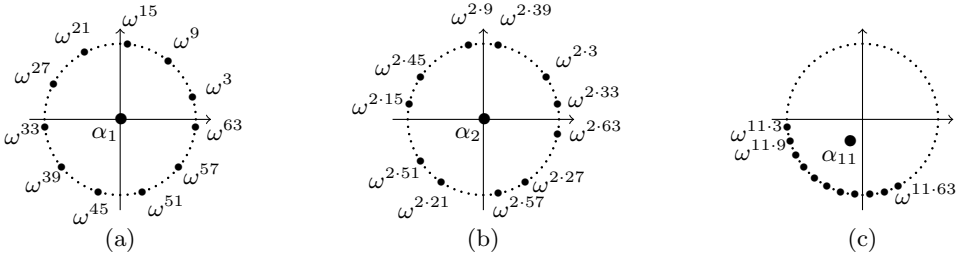
To periodo bomo izluščili tako, da bomo na $|\varphi\rangle$ uporabili *kvantno Fourierevo transformacijo*. To je unitarna preslikava, podana z matriko

$$Q = \frac{1}{\sqrt{M}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{M-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \cdots & \omega^{(M-1)(M-1)} \end{bmatrix},$$

kjer je $\omega = e^{2\pi i/M}$ kompleksni koren enote. Najprej si oglejmo, kako kvantna Fouriereva transformacija deluje, nato pa še to, kako jo učinkovito implementiramo s kvantnim vezjem. S tem bomo tudi preverili, da je unitarna.

Delovanje kvantne Fouriereve transformacije si najlaže predstavljamo, če si ogledamo amplitude baznih stanj v sliki $Q|\varphi\rangle$, torej skalarne produkte vrstic matrike Q s stanjem $|\varphi\rangle = \frac{1}{\sqrt{n}}(|m\rangle + |m+r\rangle + \cdots + |m+(n-1)r\rangle)$. Amplituda pri $|0\rangle$ je tako enaka $\alpha_0 = \sqrt{\frac{n}{M}}$, pri $|1\rangle$ pa $\alpha_1 = \frac{1}{\sqrt{nM}} \sum_{j=0}^{n-1} \omega^{m+jr}$.

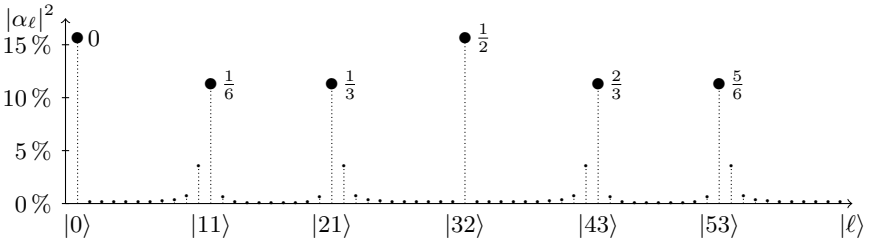
K tej vsoti torej prispevajo le potence, ki ustrezajo baznim stanjem, ki so v superpoziciji $|\varphi\rangle$. Predstavljamo si lahko, da hodimo po enotski krožnici in se v vsakem koraku premaknemo za eno potenco, v vsoto pa štejemo vsako r-to obiskano potenco (slika 2a). Potence so razporejene enakomerno, bazna stanja pa nastopajo periodično, zato se upoštevane potence medsebojno (skoraj) izničijo in amplituda je blizu 0.



Slika 2. Potence korena enote $\omega = e^{2\pi i/64}$, ki jih upoštevamo pri izračunu amplitud baznih stanj (a) $|1\rangle$, (b) $|2\rangle$ in (c) $|11\rangle$ v kvantni Fourierevi transformaciji superpozicije $|\varphi\rangle = \frac{1}{\sqrt{11}}(|3\rangle + |9\rangle + \cdots + |63\rangle)$. Amplituda je označena z znakom • na sredini kroga, zaradi berljivosti pa so nekatere oznake izpuščene.

Pri izračunu amplitude stanja $|2\rangle$ ravnamo podobno, le da se s koraki premikamo za dve potenci. Ker so upoštevane potence zopet razporejene enakomerno (vsaj za dovolj velike M), bo amplituda zopet zanemarljiva (slika 2b). Tudi pri nadaljnjih stanjih sklepamo enako — vse dokler ne pridemo do nekega stanja $|\ell\rangle$, pri katerem med dvema upoštevanima potencama približno obhodimo celotno krožnico. Tedaj namreč upoštevane potence niso več enakomerno razporejene po krožnici, temveč so si blizu, zato se njihovi prispevki ne izničijo (slika 2c).

V vsakem koraku se premaknemo za ℓ potenc, potenco pa upoštevamo na vsakih r korakov. Tako bomo z veliko verjetnostjo izmerili le stanja $|\ell\rangle$, za katera bo ℓr čim bliže polnega obhoda, torej števila M ali pa nekega njegovega večkratnika. Ko izmerimo stanje $|\ell\rangle$, prek razvoja v verižni ulomek izračunamo zaporedne približke ulomka $\frac{\ell}{M}$. Izkaže se [2], da je prvi približek, ki se od $\frac{\ell}{M}$ razlikuje za manj kot $\frac{1}{2M}$, oblike $\frac{k}{r}$ za neki k, s čimer lahko končno izračunamo periodo r.



Slika 3. Verjetnosti posameznih baznih stanj po uporabi kvantne Fouriereve transformacije na superpoziciji s periodo 6 pri $M = 64$. Ob vsakem vrhu je poleg ustreznega baznega stanja $|\ell\rangle$ zapisan tudi ulomek, ki ga dobimo z razvojem števila $\frac{\ell}{M}$ v verižni ulomek.

Iz števila r izračunamo $x = a^{r/2} \bmod N$, iz tega pa $D(x-1, N)$ in $D(x+1, N)$. Če smo našli pravi delitelj števila N, končamo. V primeru neuspeha poskusimo še z nekaj večkratniki dobljenega imenovalca, saj imata lahko k in r skupni delitelj, zato bo dobljeni ulomek okrajšan. Če tudi to ne uspe, poskusimo še s števili, ki so blizu ℓ, saj, kot vidimo na sliki 3, obstaja manjša verjetnost, da smo izmerili število blizu vrha. Če tudi v tem primeru ne najdemo števila r, celoten postopek ponovimo.

Nazadnje poglejmo še, kako kvantno Fourierevo transformacijo učinkovito implementiramo. Kot pri Groverjevem algoritmu bo tudi tu najenostavneje, če vzamemo M oblike 2^q in stanja $|0\rangle, |1\rangle, \ldots, |M-1\rangle$ izenačimo z vsemi baznimi stanji $|x\rangle$ sistema q kubitov. Stanje $|x_0 x_1 \cdots x_{q-1}\rangle$ torej predstavlja $|x_0 2^{q-1} + x_1 2^{q-2} + \cdots + x_{q-1}\rangle$. Poleg tega kvantno Fourierevo transformacijo pri $M = 2^q$ označimo s Q_q, koren enote $e^{2\pi i/2^q}$ pa z ω_q.

Poglejmo, kako lahko Q_q rekurzivno izrazimo s pomočjo Q_{q-1} in s tem njeno računanje prevedemo na enostavnejši primer. Če Q_q uporabimo na nekem sodem stanju $|2k\rangle$ in v dobljeni vsoti združimo člene, razmaknjene za 2^{q-1}, dobimo

$$Q_q|2k\rangle = \tfrac{1}{\sqrt{2^q}} \sum_{j=0}^{2^q-1} \omega_q^{2jk}|j\rangle = \tfrac{1}{\sqrt{2^q}} \sum_{j=0}^{2^{q-1}-1} \omega_q^{2kj}|j\rangle + \omega_q^{2k(j+2^{q-1})}|2^{q-1}+j\rangle.$$

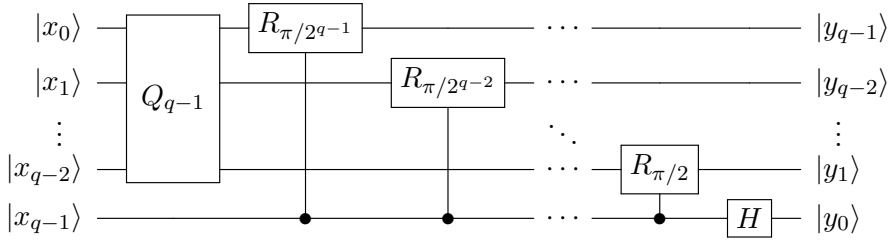
Če $|x_0 2^{q-1} + j\rangle$ pišemo kot $|x_0\rangle|j\rangle$ ter upoštevamo, da velja $\omega_q^2 = \omega_{q-1}$ in $\omega_q^{2^q} = 1$, lahko vsoto naprej zapišemo kot

$$\tfrac{1}{\sqrt{2^q}} \sum_{j=0}^{2^{q-1}-1} \omega_{q-1}^{kj}(|0\rangle|j\rangle + |1\rangle|j\rangle) = (\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle)\tfrac{1}{\sqrt{2^{q-1}}} \sum_{j=0}^{2^{q-1}-1} \omega_{q-1}^{kj}|j\rangle.$$

Zadnji kubit stanja $|2k\rangle$ je enak $|0\rangle$, prvih $q-1$ kubitov pa predstavlja $|k\rangle$, zato lahko pišemo $|2k\rangle = |k\rangle|0\rangle$ in tako dobimo $Q_q(|k\rangle|0\rangle) = |+\rangle(Q_{q-1}|k\rangle)$.

Z malce več truda za liha stanja dobimo $Q_q(|k\rangle|1\rangle) = |-\rangle(RQ_{q-1}|k\rangle)$, kjer je R rotacija, ki bazno stanje $|\ell\rangle$ slika v $\omega_q^\ell|\ell\rangle$. To izvedemo tako, da na j-tem izhodnem kubitu uporabimo fazno rotacijo $R_{\pi/2^j}$, ki stanje $|0\rangle$ ohranja, stanje $|1\rangle$ pa slika v $e^{i\pi/2^j}|1\rangle = \omega_q^{2^{q-j-1}}|1\rangle$, saj tako za $\ell = y_1 2^{q-2} + \cdots + y_{q-1}$ stanje $|\ell\rangle = |y_1 \cdots y_{q-1}\rangle$ slikamo v $\omega_q^{y_1 2^{q-2}} \cdots \omega_q^{y_{q-1}}|y_1 \cdots y_{q-1}\rangle = \omega_q^\ell|\ell\rangle$.

Tako za soda kot za liha stanja torej na prvih $q-1$ kubitih uporabimo Q_{q-1}, nato pa v odvisnosti od zadnjega kubita x_{q-1} uporabimo še rotacijo R ter prvi kubit izhoda y_0 nastavimo na $|+\rangle$ oziroma $|-\rangle$. To elegantno storimo z dogovorom, da bomo izhod brali v obratnem vrstnem redu. Tedaj moramo namreč v odvisnosti od zadnjega bita vhoda ustrezno nastaviti *zadnji* kubit izhoda, kar pa lahko storimo s Hadamardovimi vrati. V tem primeru tudi fazne rotacije izvedemo v obratnem vrstnem redu, zato kvantno Fourierevo transformacijo Q_q izvedemo z vezjem:



Za Q_q torej poleg vezja za Q_{q-1} potrebujemo še q dodatnih kvantnih vrat, torej vsega skupaj $\frac{q(q+1)}{2}$ vrat. Tako je zahtevnost kvantne Fourierevе transformacije enaka $O(q^2)$ oziroma $O((\log N)^2)$. Zahtevnost celotnega Shorovega algoritma, ki znaša $O((\log N)^3)$, pa je posledica računanja potenc in največjih skupnih deliteljev ter še nekaj drugih korakov preverjanja, ki jih tu nismo omenjali.

LITERATURA

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

[2] A. Ekert in R. Jozsa, *Quantum computation and Shor's factoring algorithm*, Rev. Mod. Phys. **68** (1996) 733–753.

[3] L. K. Grover, *Quantum Mechanics helps in searching for a needle in a haystack*, Phys. Rev. Lett. **79** (1997), 325–328.

[4] P. W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, SIAM J. Sci. Statist. Comput. **26** (1997), 1484–1509.