Nagy számok faktorizációja a Shor-algoritmussal

Koronka Gábor

May 31, 2012

Tartalomjegyzék

1 Bevezetés

A számítógépek által megoldható problémákkal és ezen megoldások futásidejével, bonyolultságával a számítástudomány foglalkozik.

Ez a tudományág foglalkozik azzal is, hogy mely problémákat tudunk megoldani és melyeket nem különböző számítógépmodellek segítségével. Ezzel a témával kapcsolatos első jelentősebb eredmények a huszadik század elején jelentek meg, melyek többek közt Church, Turing, és Post nevéhez fűződik.

Az eredményeik között szerepeltek Turing-géppel megoldható és nem megoldható problémák. Turing többek között megmutatta, hogy létezik nem felismerhető nyelv, vagyis léteznek olyan szavai ennek a nyelvnek, amiről nem tudjuk eldönteni Turing-géppel, hogy ez a bizonyos szó eleme-e a nyelvnek. Tehát ennek eldöntése nem megoldható Turing-géppel.

A legfontosabb tézis Church nevéhez fűződik, amelyben azt állítja, hogy akármilyen számítógépre írt algoritmus szimulálható Turing-géppel. Ennek azért van nagy jelentősége, mert ekkor minden, amit nem tudunk megoldani a Turing-géppel, azt egyéb más számítógépmodellekkel sem fogjuk tudni megoldani. Church nevéhez fűződik az előbbi tézis egy erősebb verziója is, amely azt mondja, hogy bármely fizikai gép lépései szimulálhatóak Turing-géppel, mégpedig annak polinomiálisan sok lépésével. Ez igen erős állítás, mert ha P≠NP, akkor semmilyen számítógéppel nem lesznek megoldhatóak gyorsan az NP nehéz problémák.

A fenti egyenlőtlenségben a P a determinisztikusan polinomiális lépésszámmal megoldható problémák halmaza, mely probléma megoldásának lépésszáma az input méretének olyan függvénye, amit felülről tudunk becsülni egy polinom függvénnyel. A másik halmaz az NP-beli problémák halmaza, melynek elemeit nem determinisztikusan meg tudjuk polinom időben oldani, vagyis a lépésszám az inputtól polinomiálisan függ. Ez a két halmaz lehet ugyanaz a két halmaz, de jelenleg nem ismert bizonyítás arra, hogy megegyeznének, és cáfolni sem tudták még.

Így jelenleg azoknak a problémáknak, amikről tudják, hogy NP-beliek, de amelyekről nem ismert, hogy P-beliek lennének-e, a megoldásához determinisztikus algoritmussal exponenciális lépésre van szükség, ami elég nagy input esetén lefuttathatatlan. Ezért fontos kérdés, hogy bizonyos problémák P-beliek-e, és így polinom lépésben futó algoritmussal megoldhatóak-e. Megjegyezhetjük, hogy természetesen nagyon magas fokú polinom esetén, nagy inputra megint csak túl hosszú időre lenne szükség egy ilyen lépésszámú algoritmus lefuttatásához. Így a gyakorlatban olyan, minél

gyorsabb algoritmusokat keresnek, amelyek lépésszámának az inputtól való függése nagyon kisfokú polinommal felülről becsülhető.

A fenti tézist cáfolandó csináltak olyan elméleti számítógépeket, amelyekkel meg lehet oldani polinom időben NP nehéz problémákat. Viszont ezek a cáfolatok azért nem voltak sikeresek, mert nem feleltek meg a fizika szabályainak, így később nem is lehetett őket a valóságban megépíteni. Annak ellenére, hogy sokan dolgoztak különféle egyéb alternatív számítógépek kitalálásával és megcsinálásával, még nem sikerült olyan számítógépet építeni, amellyel meg lehet oldani NP nehéz problémát.

Így jött az ötlet, amelyet 1982-ben Richard Feynmann, Nobel-díjas fizikus vetett fel, hogy lehetne építeni olyan számítógépet, amely a kvantumfizika, a számítástechnikában különlegesnek számító tulajdonságait használná ki. Később több nagy matematikus is foglalkozott azzal a témával, hogy erre a számítógépre algoritmust írjon, eddig viszont csak két nagyobb eredmény született ebben a témában, ami Grover és Shor nevéhez fűződik.

Shor-algoritmus nagy számokat faktorizál. E probléma történelme régebbre nyúlik vissza, és elég nagy jelentőségűvé vált a huszadik században.

A nagy számok faktorizálásánál ezen nagy számok prímtényezős felbontását készítjük el. A faktorizálás jelentős művelet, és a számelméletben is nagy szerepe van. Egy adott szám sok különböző számelméleti függvényeknél felvett értéke és tulajdonságai a prímtényezős felbontásáról leolvashatóak, és abból kiszámíthatóak.

Maga a faktorizálás azért is kapott még nagyobb figyelmet, mert később olyan titkosításokat találtak ki, amelyek számelméleti tulajdonságokat használnak ki. Így a faktorizálás segítségével ki tudjuk számítani a szükséges számelméleti függvények értékét is, ami a titkosított információk visszafejtésének lehetőségét adja a kezünkbe.

Az egyik legfontosabb ilyen kód az RSA. Ezt a kódolást 1976-ban Ron Rivest, Adi Shamir és Len Adleman fejlesztette ki, és ezzel megjelent az első nyílt kulcsú titkosítás. Ennek azért van nagy jelentősége, mert nem kell a kódnyelven kommunikáló feleknek titkos találkozón információkat megosztaniuk.

Egy nyilvános és egy titkos kulcson alapul az eljárás. Egy nyílt kulcs segítségével bárki bekódolhat üzenetet, de csak az tudja a kódot elolvasni, akinek a titkos kulcs is megvan. Ez a fél küldi szét a nyilvános kulcsot is. Ez az eljárással fordított irányban is működik, a titkos kulccsal bekódolt információkat szétküldi a folyamat, azaz mindenki el tudja olvasni a nyilvános kulcs segítségével a titkosított információt, de senki más nem tud üzenetet írni, mert ahhoz szüksége lenne a titkos kulcsra is, ilyen módon készítenek digitális aláírásokat is.

A titkosításban és a titkosítással foglalkozó elméletben ez nagy előrelépés volt, és nagyon nagy hatással volt a világra. Ilyen, és ehhez hasonló kódokat alkalmaznak az internet kódolásakor. Például egy internetes oldalt csak a titkosító fél tudjon szerkeszteni, de mindenki képes legyen megnyitni, olvasni és böngészni. Hasonlóan a bankoknál is gyakran használnak ilyen kódolásokat. Mikor különböző tranzakciókat végzünk, ezek a kódok teszik lehetővé, hogy az adott tranzakció sikeres legyen, ahogyan azt is, hogy a bankszámla védett legyen.

Egyszóval, a mai világunkban nagyon megnőtt a szerepe a nyíltkulcsú kódoknak. Ezzel természetesen együtt jár, hogy próbálnak különféle módszereket találni ezen titkosítások feltöréséhez, visszafejtéséhez. A fent említett RSA kód matematikailag visszafejthető, de olyan hosszú ideig tart egy-egy titkosított információ visszafejtése, hogy mire dekódolnak egy üzenetet, már régen elvesztette a lényegét a dekódolt információ. Ráadásul a kulcsokat, amin a titkosítás alapul, viszonylag sűrű időközönként frissítik, így teljesen felesleges a dekódolást elkezdeni is. Tehát, ha valaki szeretne feltörni egy ilyen kódot viszonylag gyors algoritmusra van szüksége. Ez sok matematikus és informatikus érdeklődését is felkeltette, de mindeddig a hagyományos számítógépre nem sikerült ilyen algoritmust írni.

Később, amikor megjelentek új számítógép modellek, és ezekre algoritmusokat írtak, sikerült Shornak kvantum polinomiális időben megoldani ezt a problémát, vagyis olyan algoritmust írnia, amelynek lépésszáma az input méretétől polinomiálisan függ és faktorizál egy számot. Ennek segítségével már kényelmesen visszafejthető bármilyen titkosított információ.

A probléma ezekkel a számítógépekkel, hogy megépíteni nem tudták őket, csak az elmélet volt meg hozzá. Így volt ez a kvantumszámítógéppel és a Shor által írt algoritmussal is. Jelenleg sincsen olyan kvantumszámítógép, ami elég nagy inputot tudna kezelni egy mai kód feltöréséhez. Azonban 1998-ban az IBM, MIT, a Californiai Egyetem és az Oxford Egyetem kutatói közösen kifejlesztettek egy kvantumszámítógépet, aminek a processzora hidrogén és klóratomokból állt. Ezzel a géppel eredményesen tudtak demonstrálni néhány alapműveletet, és futtatni néhányat a legegyszerűbb algorimusokból, de a számítógép kapacitása túl kicsi volt ahhoz, hogy komolyabb algoritmusokat futtassanak rajta. Mindeddig nem sikerült elég kapacitással rendelkező kvantumszámítógépet építeni.

2 A Shor-algoritmus vázlata

2.1 Az algoritmus lépései

Kezdetben bemutatjuk az algoritmus lépéseit, és csak majd ezt követően részletezzük, és magyarázzuk meg azokat.

Adott egész szám inputra annak prímtényezős felbontását adja meg az algoritmus.

Elsőként az N inputot, ha lehetséges, nem-triviális módon felbontjuk két egész szám szorzatára, majd a szorzat tagjait ilyen módon tovább bontjuk, amíg a szorzat egyik tagja sem bontható tovább. Ekkor a szorzat minden tagja már prím. A következőkben egy szám nem triviális felbontását részletezzük.

1. Lefutattunk egy prímtesztet N-re. Ha prím, nem bontható tovább, és készen vagyunk. Egyéb esetben futtatjuk a 2. lépést.

2. Ha N páros, akkor ismerjük egy nem-triviális felbontását N-nek. Ha páratlan, akkor futtatjuk a 3. lépést.

3. Megkeressük, hogy mely s egésznek a hatványa N. Ha találtunk, akkor ezzel N-et szorzatra bontottuk. Ha viszont nem találunk ilyet, akkor N nem lehet prímhatvány, az 1. lépés miatt nem lehet prím, ezért legalább két prímtényezője van, ekkor futtatjuk a 4. lépést.

4. Választunk véletlenszerűen egy m számot. Legyen $(m, N) = d$, ha $d \neq 1$, akkor $d \times \frac{N}{d}$ egy nem-triviális felbontás. Ha $d = 1$ akkor futtatjuk az 5. lépést.

5. Keressük P-t, ahol $O_N(m) = P$. Ebben a lépésben használjuk ki a kvantum számítógép tulajdonságait, így kvantum polinomiális időben kiszámítható m rendje, azaz P. Ezután futtatjuk a 6. lépést.

6. Ha P páros, és $(N, m^{\frac{P}{2}} - 1) \neq 1$, $(N, m^{\frac{P}{2}} + 1) \neq 1$, akkor $(N, m^{\frac{P}{2}} - 1) \times (N, m^{\frac{P}{2}} + 1)$ egy nem-triváilis felbontás. A feltétel nem teljesülése esetén futtatjuk a 3. lépést.

2.2 Hatvány tulajdonság ellenőrzése

Megnézzük minden $1 \leq k \leq \log N$ esetén $\sqrt[k]{N}$ egész-e. Ezt úgy csináljuk, hogy N k-adik gyökét olyan pontossággal közelítjük, hogy csak néhány egész szám essen a hibahatárok közé. Ezeket az egész számokat k-dikra emelve ellenőrizhetjük, hogy N k-adik gyökei-e, és így polinomiális időben megtaláljuk az összes egész számot, amelynek hatványa N. Mivel a gyökvonást ilyen pontossággal polinomiális időben el tudjuk végezni. Hasonlóan a visszaszorzásokat is konstansszor polinomiális időben végig tudjuk számolni. Ezeket a műveleteket pedig annyiszor kell elvégezni, ahány féle lehet k, vagyis $\log N$-szer, amivel együtt még így is polinomiális időben fut az algoritmus ezen része.

2.3 Nem prímhatvány esetén a nem triviális felbontás

Egy véletlen m választása után az euklideszi-algoritmussal polinomiális időben megkapjuk $(N, m) = d$-t. Ha $d \neq 1$ készen vagyunk. Ha $d = 1$ kiszámítjuk P-t, hogy $O_N(m) = P$. Ezt később részletezzük.

Tegyük fel, hogy P páros, és $(N, m^{\frac{P}{2}} - 1) \neq 1$, $(N, m^{\frac{P}{2}} + 1) \neq 1$, ekkor

$$m^P \equiv 1 \pmod{N},$$

vagyis

$$m^P - 1 \equiv 0 \pmod{N},$$

és

$$m^P - 1 = (m^{\frac{P}{2}} - 1)(m^{\frac{P}{2}} + 1),$$

tehát

$$N \mid (m^{\frac{P}{2}} - 1)(m^{\frac{P}{2}} + 1),$$

így $(N, m^{\frac{P}{2}} - 1) \times (N, m^{\frac{P}{2}} + 1)$ nem triviális felbontása N-nek.

Állítás: $\mathbb{P}((P \text{ páratlan}) vagy ((N, m^{\frac{P}{2}} - 1) = 1) vagy ((N, m^{\frac{P}{2}} + 1) = 1)) \leq \frac{1}{4}$. Vagyis annak a valószínűsége, hogy a választott m számunk segítségével nem tudjuk felbontani N-et, és új véletlen számot kell választanunk.

Bizonyítás: Legyen $\prod_{i=1}^{k} p_i^{\alpha_i} = N$ prímtényezős felbontása. Ekkor a kínai maradéktételből következően

$$\mathbb{Z}_N^* \text{ izomorf } \mathbb{Z}_{{p_1}^{\alpha_1}}^* \times \mathbb{Z}_{{p_2}^{\alpha_2}}^* \times \mathbb{Z}_{{p_3}^{\alpha_3}}^* \times \cdots \times \mathbb{Z}_{{p_k}^{\alpha_k}}^* \text{-el},$$

az $a \leftrightarrow (a_1, a_2, a_3, \ldots, a_k)$ kölcsönösen egyértelmű relációtartó hozzárendelésre nézve, ahol $\mathbb{Z}_N^*$-ben $\bmod N$, míg $\mathbb{Z}_{{p_i}^{\alpha_i}}^*$-ben pedig $\bmod {p_i}^{\alpha_i}$ relációkat tekintjük. Itt a relációtartást a következőképpen definiáljuk:

$$a \bmod N \Leftrightarrow a_1 \bmod p_1^{\alpha_1}, a_2 \bmod p_2^{\alpha_2}, a_3 \bmod p_3^{\alpha_3}, \ldots, a_k \bmod p_k^{\alpha_k}.$$

Tehát, a véletlenszerűen választott m helyett választhatunk véletlenszerűen egy az $m_i \in \mathbb{Z}_{{p_i}^{\alpha_i}}^*$ elemekből álló $(m_1, m_2, \ldots, m_k)$ vektort amivel egyértelműen meghatározzuk az m számot.

Legyen m az a szám amelyre teljesül, hogy $1 \leq m \leq N$, és m_i-k által meghatározott kongruencia-rendszer megoldása m, vagyis

$$m \equiv m_1 \pmod{p_1^{\alpha_1}}, m \equiv m_2 \pmod{p_2^{\alpha_2}}, \ldots, m \equiv m_k \pmod{p_k^{\alpha_k}}$$

Így ha az $(m_1, m_2, \ldots, m_k)$ vektort egyenletesen választjuk, akkor ez a fentiek alapján meghatározott m-nek is egy egyenletes eloszlású választását adja.

Legyen g_i a $\mathbb{Z}_{{p_i}^{\alpha_i}}^*$ multiplikatív csoport generátoreleme, x_i olyan, hogy $g_i^{x_i} \equiv m_i \pmod{p_i^{\alpha_i}}$, $r_i = O_{{p_i}^{\alpha_i}}(g_i^{x_i})$, és $X_i = \{x : 1 \leq x \leq r_i\}$.

Ekkor $x_i \leftrightarrow m_i = g_i^{x_i}$ egy kölcsönösen egyértelmű hozzárendelés X_i elemei és $\mathbb{Z}_{{p_i}^{\alpha_i}}^*$ elemei közt. Így $(x_1 \in X_1, x_2 \in X_2, \ldots x_k \in X_k) \leftrightarrow a \in \{1, 2, \ldots, N\}$ egy kongruenciatartó kölcsönösen egyértelmű hozzárendelés $X_1 \times X_2 \times \cdots \times X_k$ elemei és $\{1, 2, \ldots, N\}$ elemei közt.

Ezért a fentiek alapján véletlenszerűen választott m-et helyettesíthetjük egy véletlen $(x_1, x_2, \ldots, x_k)$ vektorral, amely a g_i-k kitevőjeként egyértelműen meghatározza m_i-ket, és így m-et is.

Legyen s az az index, melyre $2^l \mid r_s$-t, viszont minden i-re $2^{l+1} \nmid r_i$-t. Ha $2 \nmid P$, akkor $l = 0$, és mivel $2 \mid O_{p_i^{\alpha_i}}(g_i)$, ekkor $2 \mid x_i$ szükségszerűen, viszont $\mathbb{P}(2 \mid x_i) \leq \frac{1}{2}$, ebből pedig már tisztán következik, hogy $\mathbb{P}(2 \nmid P) \leq \frac{1}{2}$.

Legyen $O_{p_i^{\alpha_i}}(g_i)$ legnagyobb kettő-hatvány osztója 2^{h_i}. Ha minden $1 \leq i \leq k$-ra $2^l \mid r_i$-t, ekkor $2^{h_i - l} \mid x_i$, de $2^{h_i - l + 1} \nmid x_i$. Tetszőleges ismert, a fenti

módon definiált, r_s esetén, ha $i \neq s$ $\mathbb{P}(2^{h_i-l} \mid x_i, 2^{h_i-l+1} \nmid x_i) \leq \frac{1}{2}$. Ekkor tehát bármilyen is r_s, $i \neq s$ esetén az x_i-knek maximum a fele teljesíti a feltételt.
Vagyis

$$\mathbb{P}(2 \nmid P, \exists l : \forall i(2^l \mid r_i, 2^{l+1} \nmid r_i)) \leq (\frac{1}{2})^k + (\frac{1}{2})^{k-1}$$

Ha nem prímhatvány N, akkor $k \geq 2$, és így legalább $\frac{1}{4}$ valószínűséggel P páros, és létezik i, hogy $2^l \nmid r_i$. Erre az i-re

$$(g_i^{x_i})^{\frac{P}{2}} \equiv 1 \pmod{p_i^{\alpha_i}},$$

vagyis

$$m^{\frac{P}{2}} \equiv 1 \pmod{p_i^{\alpha_i}},$$

ekkor viszont $p_i^{\alpha_i} \mid m^{\frac{P}{2}} - 1$.

Másrészt P az r_i-k legkisebb közös többszöröse, így P-nek is 2^l-en a legnagyobb kettő-hatvány osztója. Ezért, és mert ciklikus $\mathbb{Z}^*_{{p_s}^{\alpha_s}}$

$$(g_s^{x_s})^{\frac{P}{2}} \equiv -1 \pmod{p_s^{\alpha_s}},$$

$$m^{\frac{P}{2}} \equiv -1 \pmod{p_s^{\alpha_s}},$$

vagyis $p_s^{\alpha_s} \mid m^{\frac{P}{2}} + 1$. Tehát ekkor $(N, m^{\frac{P}{2}} - 1) \neq 1$, $(N, m^{\frac{P}{2}} + 1) \neq 1$.

2.4 Példa számítás

Legyen az $N = 21$, erre az egész inputra futtassuk az algoritmust. A következőben az algoritmus lépésein haladunk végig erre az inputra.

1. Lefuttatunk egy prímtesztet a 21-re. A tesztelés után kiderül, hogy nem prím, és az algoritmus futtatja a következő lépést.

2. Leellenőrzi, hogy 21 páros-e, de 21 nem osztható kettővel, így az algoritmus futtatja a 3. lépést.

3. Ebben a részben keres egy számot, amely valahanyadik gyöke 21-nek, de ilyet nem talál, mert 21 nem hatvány. Ezért futtatja a 4. lépést.

4. Válasszuk véletlenszerűen a 2-t. Megnézzük a 2 és a 21 legnagyobb közös osztóját. Mivel relatív prímek, ezért a legnagyobb közös osztójuk az 1. Tehát futtatjuk az 5. lépést.

5. Ebben a lépésben a kvantumszámítógép kiszámolja a rendjét 2-nek 21 modulus mellett. Ezt hagyományos számítógéppel jelenleg nem tudjuk polinomiális időben megtenni, ezért fontos itt a kvantumszámítógép használata. A rendmeghatározó algoritmust később taglaljuk, és a példával be is mutatjuk a működését. A 2 rendje 6 lesz 21 modulus mellett.

6. Mivel a rend a 6 páros, így megnézhetjük 21 és a $2^{\frac{6}{2}} - 1 = 7$ legnagyobb közös osztóját, ami a 7. Illetve megnézhetjük a 21 és $2^{\frac{6}{2}} + 1 = 9$ legnagyobb közös osztóját, ami a 3. Ilyen módon a 21-et faktorizáltuk $21 = 7 \times 3$. Az algoritmus befejeződik.

Megjegyezzük, hogy a 3 és a 7 a prímtesztelésnél kiderül, hogy prímek, és így ott az egész algoritmus is megáll.

A 2 választása szerencsés volt, mivel ezzel sikerül felbontani a 21-et. Egy kis számolással kijön, hogy mely számok azok amelyekkel felbomlik a 21. Ezek a számok a 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19. Ami tehát rossz választás lett volna az a 4, 16, 17, 20. Tehát a 19 számból összesen 4 rossz volt, így $\frac{4}{5}$ eséllyel jót választottunk.

3 Kvantumszámítógépek bevezetése

A kvantumszámítógépek előnye abból származik, hogy adott méretű input a méretéhez képest exponenciális információt hordoz. De ezek a méréseknél csak bizonyos valószínűségekkel kaphatóak meg.

3.1 A kvantum bit

A kvantumszámítógép egy bitje alapesetben $|0\rangle$ vagy $|1\rangle$ lehet. Viszont szuperpozícióban lévő kvantumbit bizonyos valószínűségekkel mindkét értéket felveheti. Általános esetben a kvantumbitet egy $\mathbb{C}^2$ komplex euklideszi tér egy egységvektorával modellezhetjük, így egy q kvantumbit, vagy más néven qubit felírható $q = a|0\rangle + b|1\rangle$ alakban, ahol $a, b \in \mathbb{C}$, és $|a|^2 + |b|^2 = 1$. Ez azt jelenti, hogy a kvantumbit mérésekor $|0\rangle$-t kapunk $|a|^2$ és $|1\rangle$-et kapunk $|b|^2$ valószínűséggel.

Hasonlóan működik több bitre is, így például egy szuperpozícióban lévő két kvantumbitből álló rendszert a következőképpen írhatunk le: $a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle$, ahol $a, b, c, d \in \mathbb{C}$, $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$. Ekkor annak a valószínűsége, hogy $|00\rangle$-t, $|01\rangle$-t, $|10\rangle$-t, $|11\rangle$-t mérünk, rendre $|a|^2$, $|b|^2$, $|c|^2$, $|d|^2$.

Általánosabban, egy n kvantumbites rendszer modellje egy $\varphi : \{0, 1, \ldots, 2^{n-1}\} \mapsto \mathbb{C}^2$ hullámfüggvény, ahol $\varphi = \sum_{i=0}^{2^{n-1}} a_i|i\rangle$, és $\sum_{i=0}^{2^{n-1}} |a_i|^2 = 1$, $|i\rangle$ jelentse az i kettes számrendszerbeli alakja által meghatározott n-bites bázisvektort.

Másrészről ezt a φ-t tekinthetjük a $\mathbb{C}^{2^n}$ euklideszi tér egy egységvektorának is. Ekkor a hagyományos módon jelölve a bázisvektorokat, bizonyos szempontból könnyebben kezelhető modellt kapunk. Ezek a 2^n dimenziós standard bázisvektorok.

$|j\rangle = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$, ahol csak a j-edik tag 1, a többi 0. Vagyis például egy bit

esetén $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, és $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Ekkor pedig $\sum_{i=0}^{2^{n-1}} a_i|i\rangle = \sum_{i=0}^{2^{n-1}} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_i \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

3.2 A kvantum számítások

A kvantumszámítógépek a kvantum regisztert módosítva jutnak el az inputból az outputig. A kvantumszámítógép minden lépésben csak unitér transzformációkat tud alkalmazni a kvantumregiszterre, mivel a kvantumszámítógép minden állapotában csak egységvektort tud tárolni, és így az input és az output is csak egységvektor lehet. Tehát csak olyan transzformációkat tudunk elvégezni, ami egységvektorból egységvektorba képez, vagyis unitér transzformációkat. Ekkor $U(\varphi_0) = \varphi_1$, ahol U unitér transzformáció, φ_0 pedig az input, és így transzformációt követően a kapott φ_1 is egységvektor lesz, tehát teljesíti a kvantum regiszterre vonatkozó tulajdonságokat.

Másrészről feltételezzük, hogy a kvantumszámítógépek csak konstans nagyságú mátrixszal leírható transzformációt tudnak kezelni, illetve így azok tenzorszorzatait. Ezzel a feltételezéssel a számítógép gyakorlatban való elkészítésének kritériumait, és így magát az elkészítést is egyszerűsíthetjük.

3.3 Tenzorszorzat

Legyen $v = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$, $w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix}$, ekkor a tenzorszorzatuk $v \otimes w = \begin{pmatrix} v_1 w_1 \\ v_1 w_2 \\ \vdots \\ v_2 w_1 \\ \vdots \end{pmatrix}$.

Például legyen $v = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$, $w = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, így a tenzorszorzatuk $v \otimes w = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ lesz.

Legyen V egy n, W pedig egy m dimenziós vektortér. Ekkor $V \otimes W$ tenzorszorzat egy nm dimenziós vektortér lesz, melynek elemei a $v \otimes w$ vektorok lineáris kombinációi, ahol $v \in V$, $w \in W$. Emellett $v_1, v_2 \in V$, $w_1, w_2 \in W$, és $z \in \mathbb{C}$ teljesíti a következőket:

1. $z(|v_1\rangle \otimes |w_1\rangle) = (z|v_1\rangle) \otimes |w_1\rangle = |v_1\rangle \otimes (z|w_1\rangle)$
2. $(v_1+v_2)\otimes(w_1+w_2) = v_1\otimes(w_1+w_2)+v_2\otimes(w_1+w_2) = \sum_{i=1}^{2}\sum_{j=1}^{2} v_i \otimes w_j$

Megjegyezzük, hogy a tenzorszorzás nem kommutatív. Lineáris operátorokra a következőképpen definiáljuk:

Legyen A, B lineáris operátor rendre V-n, illetve W-n értelmezve. Ekkor az $A \otimes B$ lineáris operátor $V \otimes W$-n. Az $A \otimes B$-t reprezentáló mátrixot a következő képen számíthatjuk ki:

$$A \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & \cdots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & \cdots & a_{2,n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1}B & a_{m,2}B & \cdots & a_{m,n}B \end{pmatrix}.$$

Itt A egy $n \times n$-es, B pedig egy $m \times m$-es mátrix, amik reprezentálják rendre A, illetve B azonos betűkkel jelölt lineáris operátorokat. Így a keletkező $A \otimes B$ mátrix $nm \times nm$-es lesz.

Például legyen $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$,

ekkor az $A \otimes B$ a következő lesz:

$$A \otimes B = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Unitér transzformációk tenzorszorzata is unitér transzformáció. Így unitér transzformációk tenzorszorzatát is alkalmazhatjuk a kvantum regiszterre. Ezzel elérhető, hogy csak a j-edik bitjét módosítsuk a regiszternek $I_{j-1\times j-1}\otimes U_j \otimes I_{n-j\times n-j}$, ahol I az identitás.

3.4 A kvantum kapuk

Elsőként az egy qubites kapukat nézzük. Ebből az identitás után a legalapvetőbb a $\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, ez a NOT kapu. Természetes módon a NOT kaput úgy definiálnánk, hogy $NOT|0\rangle = |1\rangle$, és $NOT|1\rangle = |0\rangle$. Ez teljesül is. Tekintsük a bázisvektoros modellt, ekkor $|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}$, $|1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$. Így

$$NOT\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix}$$

$$NOT\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\end{bmatrix}$$

A Hadamar kapunak hasonlóan fontos a szerepe. Ennek egy q-bitre a mátrixa $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$. Itt

$$H\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$$

$$H\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$$

Vagyis $H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, $H|1\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

Ennek az önmagával való tenzorszorzatait véve hasonló alakra jutunk a $|0\ldots0\rangle$ inputra:

$$H^{\otimes 2}|0\rangle|0\rangle = (H\otimes H)(|0\rangle\otimes|0\rangle) = H|0\rangle\otimes H|0\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}\otimes\frac{|0\rangle+|1\rangle}{\sqrt{2}}$$

$$= \frac{1}{2}(|0\rangle|0\rangle + |0\rangle|1\rangle + |1\rangle|0\rangle + |1\rangle|1\rangle) = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle)$$

$$H^{\otimes n}|0\dots0\rangle = (H|0\rangle)^{\otimes n} = (\frac{|0\rangle + |1\rangle}{\sqrt{2}})^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} |i\rangle$$

Másik fontos 2-qubites kapu a CNOT kapu. Itt az egyik a kontroll qubittől függően változik meg a másik a cél qubit. Hasonlóan működik mint a NOT kapu, csak az inputja a kontroll qubit lesz, míg a cél qubitet módosítja. Tehát ha a kontroll qubit 0, nem történik semmi, viszont ha a kontroll qubit 1, a cél qubit értékét megváltoztatja. Példában bemutatjuk 2-qubites rendszerben a kapu működését. Több qubites rendszerben az identitásokkal tenzorszorozva a 2-qubites $CNOT$ kaput megkaphatunk bármely $CNOT$ kaput, melynek kontroll és cél qubitje rendszer tetszőleges két bitjén hat.

Ekkor tehát a $CNOT|0\rangle|1\rangle = |0\rangle|1\rangle$, és a $CNOT|0\rangle|0\rangle = |0\rangle|0\rangle$, viszont $CNOT|1\rangle|0\rangle = |1\rangle|1\rangle$, és $CNOT|1\rangle|1\rangle = |1\rangle|0\rangle$. Vagyis a $CNOT$ kaput modellező mátrixot a következőképpen írhatjuk fel:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \text{ mivel } |00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

így tehát a $CNOT$ kapu hatása a bázisvektorokra ilyen módon számolható:

$$CNOT \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

$$CNOT \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix},$$

$$CNOT \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

$$CNOT \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

3.5 Hagyományos számítógépek szimulálása kvantumszámítógépen

Ebben az alfejezetben azt tárgyaljuk, hogyan lehet kvantumszámítógéppel ugyanazokat a számításokat elvégezni, mint hagyományos számítógépen. Ezzel arra is rámutatunk, hogy a kvantumszámítógépekkel sokkal bővebb eszköztár van a kezünkben, mint a hagyományos esetben.

A hagyományos számítógép processzora logikai kapukat használ a számítások elvégzéséhez. Alapvetően három logikai kapuról szoktunk beszélni az AND, az OR, és a NOT. Azért elég ezeket a kapukat használni, mert ezek logikailag teljes rendszert alkotnak, vagyis az AND, OR, NOT segítségével bármilyen $f : \{i,h\}^n \mapsto \{i,h\}$ logikai függvény leírható. Itt az OR és az AND közül elég csak az egyik kaput megcsinálni, mert azzal szimulálható a másik is.

Tehát, ha a kvantumszámítógéppel szimulálni szeretnénk egy hagyományos számítógépet, akkor elég a NOT és az AND vagy az OR kapukat megcsinálni. A probléma alapvetően az, hogy a kvantumszámítógépben lévő kapuk unitér transzformációval leírhatóak, vagyis invertálhatóak is, de a logikai függvényeknek legtöbb esetben nincs egyértelmű inverze, így azokat nem tudjuk unitér transzformációkkal leírni. Ezt a problémát úgy oldjuk meg, hogy az outputunkban az input is benne lesz, a logikai művelet eredménye mellett. Így egyértelmű az inverze az outputnak, és ezt már meg tudjuk csinálni unitér transzformációval. Másrészről az outputnak csak azt a részét vesszük figyelembe, ami a logikai művelet végeredménye, és így szimulálni tudjuk a hagyományos számítógépeket.

A NOT kapu szerencsésen hagyományos értelemben is azt a műveletet végzi el, amit a kvantum NOT kapu csinál, ahogy ezt a kapu bevezetésekor is megjegyeztük. Így elég csak az AND vagy az OR kapu valamelyikét szimulálnunk.

Ezekhez a logikai műveletekhez bevezetjük a Fredkin és Toffoli kapukat. Mindkét kapunak 3-qubites az inputja és az outputja is. A Toffoli kapu outputja csak akkor különbözik az inputtól, ha az első két inputbit 1-es, és ekkor az output 3. bitje megváltozik. A Fredkin kapu az input utolsó két bitjét felcseréli, ha az input első bitje 0, és nem változtat semmit, ha az input első bitje 1-es.

Toffoli kapu						Fredkin kapu					
input			output			input			output		
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1	0	1	0
0	1	0	0	1	0	0	1	0	0	0	1
0	1	1	0	1	1	0	1	1	0	1	1
1	0	0	1	0	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0	1	1	0	1
1	1	0	1	1	1	1	1	0	1	1	0
1	1	1	1	1	0	1	1	1	1	1	1

Ekkor a Toffoli kapunál a harmadik inputbitet 1-re állítva, és az outputban kapott utolsó bitet negálva, vagyis alkalmazva rá a NOT kaput, ugyanazt kapjuk, mintha az első két inputbitre alkalmaznánk az AND műveletet. Hasonlóan, ha a Fredkin kapunál az utolsó inputbitet 0-ra állítjuk, akkor az output második bitjének értéke ugyan az, mint amit az első két inputbitre alkalmazott AND művelet ad.

Tehát így tudjuk kvantumszámítógéppel szimulálni az AND kaput is, és a NOT kaput, amit az elöző fejezetben részleteztünk. Ezek pedig logikailag teljes rendszert alkotnak, tehát a számítógépre írt minden algoritmust kvantumszámítógépen tudunk szimulálni.

3.6 Fourier-transzformáció kvantumszámítógépekkel

A Fourier-transzformációnak elég nagy szerepe van a kvantumalgoritmusokban. A Shor-algoritmusban is a leglényegesebb lépésben is a Fourier-transzformációt alkalmazzuk, aminek köszönhetően adott szám rendjét kvantumpolinomiális időben meg tudjuk határozni.

A Diszkrét Fourier-transzformáció hagyományos definíciója alapján egy $|j\rangle$ bázisvektor transzformáltját a következőképpen definiáljuk:

$$DFT(|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

A következőkben a transzformált alakot szeretnénk kiszámolni kvantumszámítógéppel. Ehhez a szummát alakítjuk át egy könnyebben kezelhető alakra, ahol a szummában szereplő bázisvektorok kettes számrendszerbeli alakja $|k\rangle = |k_1 k_2 \dots k_n\rangle$:

$$DFT(|j\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |k\rangle$$

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

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

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

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

$$DFT(|j\rangle) = \left(\frac{|0\rangle + e^{\frac{2\pi i j}{2}} |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle + e^{\frac{2\pi i j}{2^2}} |1\rangle}{\sqrt{2}} \right) \otimes \cdots \otimes \left(\frac{|0\rangle + e^{\frac{2\pi i j}{2^n}} |1\rangle}{\sqrt{2}} \right)$$

Tehát például két qubit esetén a bázisvektorok diszkrét fourier transzformáltja a következő képpen néz ki:

$$DFT(|00\rangle) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right)$$

$$DFT(|01\rangle) = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle + i|1\rangle}{\sqrt{2}} \right)$$

$$DFT(|10\rangle) = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right)$$

$$DFT(|11\rangle) = \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right) \otimes \left(\frac{|0\rangle - i|1\rangle}{\sqrt{2}} \right)$$

Így egy bázisvektorra ki tudjuk számolni n kaput alkalmazva, de mivel 2^n darab bázisvektorunk van, és mindre ki kell számolni, így ez exponenciális időben menne csak. Ezért csinálunk egy olyan algoritmust, ami általános input bázisvektorra annak a fourier transzformáltját adja ki outputnak.

Elsőként a fent levezetett n tagú tenzorszorzat l-edik tagját számoljuk ki,

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

Ehhez bevezetünk két új kaput az R_k-t, és a CR_k-t, amit a $|j_{k+l-1}\rangle$ qubit kontrollál. Ha $j_{k+l-1} = 0$ a CR_k identitás lesz, és nem változtat semmit, ha $j_{k+l-1} = 1$ a $CR_k = R_k$.

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i \frac{1}{2^k}} \end{bmatrix}, CR_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i \frac{j_{k+l-1}}{2^k}} \end{bmatrix}$$

Ekkor nevezzük PR_m-nek azt a modult, amelyben sorban alkalmazzuk a CR_m-től a CR_1-ig a kapukat.

$$PR_{n+1-l} = CR_{n+1-l}CR_{n-l}\dots CR_1$$

$$PR_{n+1-l} = \prod_{k=n+1-l}^{1} \begin{bmatrix} 1 & 0 \\ 0 & exp\Big(2\pi i \frac{j_{k+l-1}}{2^k}\Big) \end{bmatrix}$$

$$PR_{n+1-l} = \begin{bmatrix} 1 & 0 \\ 0 & exp\Big(2\pi i\Big(\frac{j_n}{2^n-l+1} + \frac{j_{n-1}}{2^n-l} + \dots + \frac{j_l}{2}\Big)\Big) \end{bmatrix}$$

$$PR_{n+1-l} = \begin{bmatrix} 1 & 0 \\ 0 & exp\Big(2\pi i\Big(\frac{j}{2^{n-l+1}}\Big)\Big) \end{bmatrix}$$

Legyen $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, ekkor az Hadamard kapu definíciója miatt

$$H|j_l\rangle = \frac{|0\rangle + e^{2\pi i \frac{j_l}{2}}|1\rangle}{\sqrt{2}} = CR_1|+\rangle.$$

Ekkor pedig $PR_{n+1-l}|+\rangle = \frac{|0\rangle + e^{2\pi i \frac{j}{2^{n-l+1}}}|1\rangle}{\sqrt{2}}$. Ezzel előállítottuk a tenzrszorzat $n+1-l$-edik tagját, így az l megfelelő választásával a fenti tenzorszorzat minden tagját elő tudjuk állítani.

Figyelnünk kell arra, hogy ne módosítsuk azokat a biteket, amelyek későbbi lépésben még kontrolláló bitként szerepelnek. Ezért az utolsó bitet módosító modult kell elsőnek alkalmazni a PR_n-et. Itt még az n-edik bit is kontrolláló bitként szerepel, de későbbi PR_k-kra már nincs hatással. Majd sorban alkalmazzuk a modulokat n-től egészen 1-ig, ahol PR_1-nél már csak az első bitnek van kontrolláló szerepe.

Viszont azzal, hogy visszafele alkalmaztuk a kapukat, sajnos a bitsorrend is megfordul. Ahhoz, hogy a Fourier transzformáltat kapjuk, vagy fordított sorrendben olvassuk az outputot, vagy kell egy algoritmus, amely megfordítja a bitek sorrendjét. Ehhez elsőként mutatunk egy modult, amely képes két bitet felcserélni, majd ezután ezt alkalmazzuk az első és az n-edik bitre, majd a második és $(n-1)$-edik bitre, és így tovább, amíg végül az $\lfloor \frac{n}{2} \rfloor$-edik és az $(n+1-\lfloor \frac{n}{2} \rfloor) - edik$ bitre is alkalmaztuk. Így tehát már a biteket fordított sorrendben kapjuk.

A kérdés most tehát, hogy hogyan lehet az l-edik és a k-adik biteket felcserélni. Ehhez használjuk fel a $CNOT$ kaput. A modul három lépésből áll. Elsőként az l-edik bit legyen a kontroll bit, és a k-adik a cél bit, második lépésként a k-adik legyen a kontroll, és az l-edik a cél bit, végül pedig megint legyen az l-edik a kontroll és a k-adik a cél bit.

Ezt alkalmazva a $|0\rangle$ és $|0\rangle$ bitekre a kontroll bit 0 volta miatt a második bit értéke nem változik, és mindhárom lépésben ez történik, így végül a $|0\rangle$ és $|0\rangle$ biteket kapjuk outputként. Ha $|1\rangle$ és $|0\rangle$ az input, az első lépésben a második bit megváltozik és $|1\rangle$ és $|1\rangle$ biteket kapjuk, a második lépésben az első bit változik meg, így $|0\rangle$ és $|1\rangle$-et kapunk, és végül a harmadik lépésben nem módosul az érték így ez lesz az output is. Ha $|0\rangle$ és $|1\rangle$ az input, az első lépésben nem változik az értékük, a második lépésben az első bit megváltozik és így $|1\rangle$ és $|1\rangle$ lesz, végül a harmadik lépésben módosul a második bit, és így az output $|1\rangle$ és $|0\rangle$ lesz. Az utolsó eset pedig, ha $|1\rangle$ és $|1\rangle$ a két bit. Ekkor az első lépésben a második bit módosul, majd a következőben az első bit nem módosul, és végül a harmadik bit megint megváltozik, és így megint $|1\rangle$ és $|1\rangle$ lesznek az output bitek. Tehát minden esetben az input fordítottja az output, így ez a modul tényleg felcseréli a két bitet.

Tehát a Fourier transzformáltat megkaphatjuk a fenti módon megfordítva a PR_k modulokat alkalmazó fenti algoritmus eredeti outputját.

A fenti algoritmusban szereplő PR_k-k maximum n darab kaput használnak, és így az adott kapuk segítségével $O(n^2)$ időben ki tudjuk számolni tetszőleges bázisvektor transzformáltját. A kvantumpárhuzamosság miatt, ezt tetszőleges inputra alkalmazva, kvantumpolinomiálisan megkaphatjuk annak Fourier transzformáltját.

4 Kvantum algoritmus a rend meghatározására

4.1 Az input kezelése, és regiszterek beállítása

Adott N szám faktorizálásának problémáját redukáltuk egy véletlen m szám rendjének meghatározására. Ezen probléma megoldásához használjuk a kvantumszámítógépet. Legyen t az a szám, melyre $N^2 \leq 2^t < 2N^2$, és n pedig olyan, hogy $n = \lceil \log_2 N \rceil$. Továbbá legyen a keresett $O_N m = P$.

Első lépésként két kezdeti állapotban lévő kvantum regisztert inicializálunk. Az elsőt t a másodikat n qubitesnek definiáljuk. Ezeken futtatjuk a kvantum algoritmust, és az algoritmus outputja ezeken a regisztereken mért érték lesz. A kvantumszámítógép állapotait φ-vel jelöljük, ahol φ_k a k-adik lépés utáni állapot.

$$\varphi_0 = |00\ldots0\rangle|00\ldots0\rangle$$

Ezt követően alkalmazzuk a Hadamard kaput az első regiszter minden qubitjére. Ekkor azt kapjuk, hogy

$$\varphi_1 = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|0\rangle.$$

Itt az első regiszter szuperpozícióba kerül azonos amplitúdókkal, vagyis minden bázisvektort azonos valószínűséggel kapnánk meg mérési eredményként, ha most mérnénk meg az első regisztert. A második regiszter ekkor még változatlan.

Legyen V_x az az unitér lineáris operátort, amelyre

$$V_x(|j\rangle|k\rangle) = |j\rangle|k + x^j\rangle.$$

Ezt alkalmazva a két regiszterre a következőt kapjuk:

$$V_x(\varphi_1) = \varphi_2 = \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} |j\rangle|x^j\rangle.$$

Ezzel inicializáltuk és beállítottuk a regisztereket, és innentől futtatjuk az algoritmus lényegi részét.

4.2 A rend adott függvényének meghatározása speciális esetben

A kezdeti beállítás után az első regiszterben lévő j-t bontsuk szét egy összegre, amelynek egyik tagja P többszöröse, a másik tagja pedig a maradék. Vagyis legyen $a, b \in \mathbb{N}$, $0 \leq b < P$, és ekkor $j = aP + b$ teljesüljön. A következőkben $P \mid 2^t$. Ebben a speciális esetben könnyen látható, hogy hogyan egyszerűsödik majd le a kifejezés, és a későbbiekben részletezzük majd az általános esetet.

$$\varphi_2 = \frac{1}{\sqrt{2^t}} \sum_{b=0}^{P-1} \left(\sum_{a=0}^{\frac{2^t}{P}-1} |aP + b\rangle \right) |x^b\rangle.$$

Ezután az első regiszterre alkalmazzuk a Fourier transzformációt, majd a belső kifejezést hatványazonosság szerint szorzattá alakítjuk, hogy a szummából majd később kiemelhessük a $e^{2\pi ijb}$-t:

$$\varphi_3 = \frac{1}{\sqrt{2^t}} \sum_{b=0}^{P-1} \sum_{a=0}^{\frac{2^t}{P}-1} \left(\frac{1}{\sqrt{2^t}} \sum_{k=0}^{2^t-1} e^{\frac{2\pi ik(aP+b)}{2^t}} |k\rangle \right) |x^b\rangle,$$

$$\varphi_3 = \frac{1}{2^t} \sum_{b=0}^{P-1} \sum_{a=0}^{\frac{2^t}{P}-1} \left(\sum_{k=0}^{2^t-1} e^{\frac{2\pi ikaP}{2^t}} e^{\frac{2\pi ikb}{2^t}} |k\rangle \right) |x^b\rangle.$$

A szummákat felcserélve, egy olyan alakot kapunk, ahol az adott x^b modulo N maradékok és az első regiszterben szereplő j értékekhez tartozó összes együtthatót összegezzük.

$$\varphi_3 = \frac{1}{2^t} \sum_{b=0}^{P-1} \sum_{k=0}^{2^t-1} \left(\sum_{a=0}^{\frac{2^t}{P}-1} e^{\frac{2\pi ikaP}{2^t}} e^{\frac{2\pi ikb}{2^t}} |k\rangle \right) |x^b\rangle.$$

$$\varphi_3 = \frac{1}{2^t} \sum_{b=0}^{P-1} \sum_{k=0}^{2^t-1} e^{\frac{2\pi ikb}{2^t}} \left(\sum_{a=0}^{\frac{2^t}{P}-1} e^{\frac{2\pi ikaP}{2^t}} |k\rangle \right) |x^b\rangle.$$

$$\varphi_3 = \frac{1}{2^t} \sum_{b=0}^{P-1} \sum_{k=0}^{2^t-1} e^{\frac{2\pi ikb}{2^t}} \left(\sum_{a=0}^{\frac{2^t}{P}-1} exp\Big(\frac{2\pi ikP}{2^t}\Big)^a |k\rangle \right) |x^b\rangle.$$

A mértani sorozat összegképletébe $\sum_{i=1}^{n} q^i = \frac{q^{n+1}-1}{q-1}$, ahol $q \neq 1$ behelyettesítve a $q = exp(\frac{2\pi ikP}{2^t})$-t, ha a nevező $exp(\frac{2\pi ikP}{2^t}) - 1 \neq 0$,

akkor értelmes lesz a képlet. Ekkor a számláló értéke $exp(\frac{2\pi ikP}{2^t})^{\frac{2^t}{P}} - 1 = e^{\frac{2\pi ikP}{2^t}\frac{2^t}{P}} - 1 = exp(2\pi ik) - 1 = 0$, vagyis az egész összeg 0 lesz.

Most nézzük meg mikor alkalmazhatjuk a fenti összegképletet. Az $exp(\frac{2\pi ikP}{2^t}) - 1 \neq 0$ feltétel akkor teljesül, ha $k \neq l\frac{2^t}{P}$, ahol $l \in \mathbb{Z}$. Egyéb esetben, amikor nem alkalmazhatjuk a fenti módszert $k = l\frac{2^t}{P}$. Ekkor az összegben szereplő tagok $exp(2\pi iak\frac{P}{2^t}) = exp(2\pi ial\frac{2^t}{P}\frac{P}{2^t}) = 1$, vagyis a szumma értéke ekkor $\frac{2^t}{P} \times 1 = \frac{2^t}{P}$. Tehát ezen egyszerűsítéseket, és összevonásokat elvégezve a mérhető értékek könnyebben átlátható alakját kapjuk.

$$\varphi_3 = \frac{1}{\sqrt{P}} \sum_{b=0}^{P-1} \left(e^{\frac{2\pi ijb}{2^t}} |l\frac{2^t}{P}\rangle \right) |x^b\rangle.$$

4.3 A rend adott függvényének meghatározása általános esetben

Általános esetben is, hasonlóan a fenti esethez, az első regiszter eloszlása speciális. Vagyis $|l\frac{2^t}{P}\rangle$ értékeket lényegesen nagyobb valoszínűséggel mérünk, mint más értékeket. Megjegyezzük, hogy általános esetben nullánál nagyobb valószínűséggel mérhetünk más értékeket is. A következőkben végigszámoljuk általános esetet.

A jelölés egyszerűsítése érdekében legyen $2^t = Q$, r legyen olyan, hogy $r \equiv Q \bmod P$, $Q < P$, és legyen $Q_0 = \frac{Q-r}{P}$. Vagyis $Q : P = \frac{Q_0}{P} + r$ egy maradékos osztás. Valamint $w = e^{2\pi i/2^t}$ legyen.

Ekkor a következő képpen írhatjuk fel a φ_3-at:

$$\varphi_3 = \frac{1}{Q}\left[\sum_{a=0}^{\frac{Q_0}{P}-1} \sum_{b=0}^{P-1} \left(\sum_{k=0}^{Q-1} w^{k(aP+b)} |k\rangle \right) |x^b\rangle + \sum_{k=0}^{Q-1} \sum_{b=0}^{r-1} w^{k(P\frac{Q_0}{P}+b)} |k\rangle \ |x^b\rangle \right],$$

$$\varphi_3 = \frac{1}{Q} \sum_{k=0}^{Q-1} \left[\sum_{a=0}^{\frac{Q_0}{P}-1} \sum_{b=0}^{P-1} w^{k(aP+b)} |k\rangle |x^b\rangle + \sum_{b=0}^{r-1} w^{k(P\frac{Q_0}{P}+b)} |k\rangle \ |x^b\rangle \right],$$

$$\varphi_3 = \frac{1}{Q}\sum_{k=0}^{Q-1}\left[\sum_{b=0}^{P-1}\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}w^{kb}|k\rangle|x^b\rangle + \sum_{b=0}^{r-1} w^{kP\frac{Q_0}{P}}w^{kb}|k\rangle\ |x^b\rangle\right],$$

$$\varphi_3 = \frac{1}{Q}\sum_{k=0}^{Q-1}\left[\sum_{b=0}^{P-1} w^{kb}\left(\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}\right)|k\rangle|x^b\rangle + \sum_{b=0}^{r-1} w^{kb}\left(w^{k\frac{Q_0}{P}P}\right)|k\rangle\ |x^b\rangle\right],$$

Ekkor a két szummát összevonhatjuk oly módon, hogy a kerek zárójelekben lévőket egy szummában írjuk fel. Ezt azért tehetjük meg mert a 0-tól $\frac{Q_0}{P}-1$-ig megy, a másik felében az összegnek pedig $\frac{Q_0}{P}$ van. Ezt viszont csak azon b-kre tehetjük meg, amelyek szerepelnek mindkét szummában, vagyis csak r darabban.

$$\varphi_3 = \frac{1}{Q}\sum_{k=0}^{Q-1}\left[\sum_{b=0}^{r-1} w^{kb}\left(\sum_{a=0}^{\frac{Q_0}{P}} w^{kaP}\right)|k\rangle|x^b\rangle + \sum_{b=r}^{P-1} w^{kb}\left(\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}\right)|k\rangle\ |x^b\rangle\right].$$

A kerek zárójelben lévő részeket összegezve, és leosztva Q-val megkapjuk egy adott $|k\rangle|x^b\rangle$ regiszter együtthatóját. A fenti összegzés abszolutértékét a szummák elejére kiemelt Q-val leosztva az együttható abszolutérték kapjuk, és ennek négyzete az adott bitsorozat mérésének valószínűsége. Az összegzést a mértani sorozat összegképletének segítségével számoljuk ki.

Ezt a különböző b-kre összegezve megkapjuk $Prob(k)$-t, amivel azt jelöljük, hogy mekkora valószínűséggel mérünk az első regiszterben k értéket.

$$Prob(k) = \frac{1}{Q^2}\sum_{b=0}^{r-1}\left|\sum_{a=0}^{\frac{Q_0}{P}} w^{kaP}\right|^2 + \frac{1}{Q^2}\sum_{b=r}^{P-1}\left|\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}\right|^2$$

$$Prob(k) = \frac{r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}} w^{kaP}\right|^2 + \frac{P-r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}\right|^2$$

$$Prob(k) = \frac{r}{Q^2}\left|\frac{w^{kP(\frac{Q_0}{P}+1)}-1}{w^{kP}-1}\right|^2 + \frac{P-r}{Q^2}\left|\frac{w^{kP\frac{Q_0}{P}}-1}{w^{kP}-1}\right|^2$$

$$Prob(k) = \frac{r}{Q^2}\left|\frac{e^{\frac{2\pi i}{Q}kP(\frac{Q_0}{P}+1)}-1}{e^{\frac{2\pi i}{Q}kP}-1}\right|^2 + \frac{P-r}{Q^2}\left|\frac{e^{\frac{2\pi i}{Q}kP\frac{Q_0}{P}}-1}{e^{\frac{2\pi i}{Q}kP}}-1\right|^2$$

Legyen $\{k\}_Q \equiv k \bmod Q$, és $-\frac{Q}{2} < \{k\}_Q \leq \frac{Q}{2}$. Vagyis $\{k\}_Q$ a legkisebb abszolutértékű maradékát jelöli k-nak modulo Q-ra.

A következőkben megmutatjuk, hogy azon speciális esetek valószínűsége, amikor $\left|\frac{\pi\{kP\}_Q}{Q}\frac{Q_0}{P}\right| < \frac{\pi}{2}$ az $\frac{1}{P}$ körüli érték. Az egyenlőtlenséget átrendezve a következot kapjuk

$$\left|\frac{\pi\{kP\}_Q}{Q}\frac{Q_0}{P}\right| = \left|\{kP\}_Q\right|\frac{1}{P}\frac{Q_0}{Q}\pi < \frac{\pi}{2}$$

$$\left|\{kP\}_Q\right| < \frac{P}{2}\frac{Q}{Q_0} < \frac{P}{2}$$

Ezen speciális eseteket két részre kell bontanunk a mértani-sorozat összegképletére vonatkozó megkötés miatt, így $0 < |\{Pk\}_Q| \leq \frac{P}{2}(1-\frac{1}{N})$, és $\{Pk\}_Q = 0$ eseteket fogjuk elemezni, és ezek valószínűségeit fogjuk most kiszámolni. Elsőként egy-két egyenlőséget, és egyenlőtlenséget említünk meg, amelyek segítségével fogjuk becsülni ezeket az értékeket.

$$\frac{Q_0}{Q} = \frac{Q-r}{Q} = 1 - \frac{r}{Q} \geq 1 - \frac{N}{N^2} = 1 - \frac{1}{N}$$

Mivel $Q = 2^t$-t úgy választottuk, hogy $N^2 \leq 2^t \leq 2N^2$. Másrészt tudjuk, hogy

$$\begin{aligned}
\left|e^{i\alpha} - 1\right|^2 &= \left|\cos\alpha + i\sin\alpha - 1\right|^2 \\
&= (\cos\alpha - 1)^2 + \sin^2\alpha \\
&= \cos^2\alpha + \sin^2\alpha + 1 - 2\cos\alpha \\
&= 2\Big(1 - \cos\frac{2\alpha}{2}\Big) \\
&= 2\Big(1 - \Big(1 - 2\sin^2\frac{\alpha}{2}\Big)\Big) \\
&= 4\sin^2\left(\frac{\alpha}{2}\right)
\end{aligned}$$

Ezek alapján a fenti becslésben az abszolútérték négyzeteket a szinuszos kifejezésekre cserélve a következőt kapjuk:

$$Prob(k) = \frac{r}{Q^2}\frac{\sin^2\left(\frac{2\pi\{kP\}_Q}{2Q}(\frac{Q_0}{P}+1)\right)}{\sin^2\left(\frac{2\pi\{kP\}_Q}{2Q}\right)} + \frac{P-r}{Q^2}\frac{\sin^2\left(\frac{2\pi\{kP\}_Q}{2Q}\frac{Q_0}{P}\right)}{\sin^2\left(\frac{2\pi}{2Q}\right)}$$

Továbbá $|\alpha| < \frac{\pi}{2}$, esetén $\frac{4}{\pi^2}\alpha^2 \leq \sin^2\alpha \leq \alpha^2$, mivel $\frac{2}{\pi}\alpha \leq \sin\alpha \leq \alpha$. Most azon k-k valószínűségét nézzük, amikre $\left|\frac{\pi\{kP\}_Q}{Q}\frac{Q_0}{P}\right| < \frac{\pi}{2}$, itt felhasználhatjuk az egyenlőtlenséget is.

$$Prob(k) \geq \frac{r}{Q^2}\frac{\frac{4}{\pi^2}\left(\frac{2\pi\{kP\}_Q}{2Q}(\frac{Q_0}{P}+1)\right)^2}{\left(\frac{2\pi\{kP\}_Q}{2Q}\right)^2} + \frac{P-r}{Q^2}\frac{\frac{4}{\pi^2}\left(\frac{2\pi\{kP\}_Q}{2Q}\frac{Q_0}{P}\right)^2}{\left(\frac{2\pi\{kP\}_Q}{2Q}\right)^2}$$

$$Prob(k) \geq \frac{r\frac{4}{\pi^2}\left(\frac{\pi\{kP\}_Q}{Q}(\frac{Q_0}{P}+1)\right)^2 + (P-r)\frac{4}{\pi^2}\left(\frac{\pi\{kP\}_Q}{Q}\frac{Q_0}{P}\right)^2}{Q^2\left(\frac{\pi\{kP\}}{Q}\right)^2}$$

$$Prob(k) \geq \frac{r\left(2\frac{\{kP\}_Q}{Q}(\frac{Q_0}{P}+1)\right)^2 + (P-r)\left(2\frac{\{kP\}_Q}{Q}\frac{Q_0}{P}\right)^2}{\left(\pi\{kP\}_Q\right)^2}$$

$$Prob(k) \geq r\Big(\frac{2}{\pi Q}\Big(\frac{Q_0}{P}+1\Big)\Big)^2 + (P-r)\Big(\frac{2}{\pi Q}\frac{Q_0}{P}\Big)^2$$

$$Prob(k) \geq P\Big(\frac{2}{\pi Q}\frac{Q_0}{P}\Big)^2 \geq \frac{4}{\pi^2 P}\Big(\frac{Q_0}{Q}\Big)^2 \geq \frac{4}{\pi^2 P}\Big(1-\frac{1}{N}\Big)^2$$

Most tekintsük azokat a k értékeket, amelyekre nem alkalmazhatjuk a szummát egyszerűsítő képletet, mivel a képletben szereplo nevező nulla lenne, vagyis $w^{kP} = 1$.

$$Prob(k) = \frac{r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}} w^{kaP}\right|^2 + \frac{P-r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}-1} w^{kaP}\right|^2$$

$$Prob(k) = \frac{r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}} 1^a\right|^2 + \frac{P-r}{Q^2}\left|\sum_{a=0}^{\frac{Q_0}{P}-1} 1^a\right|^2 \geq \frac{Q_0^2}{PQ^2}$$

Fent belátott $\frac{Q_0}{Q} =\geq 1-\frac{1}{N}$ miatt teljesül, hogy

$$Prob(k) \geq \frac{Q_0^2}{PQ^2} \geq \frac{1}{P}\left(1-\frac{1}{N}\right)^2 \geq \frac{1}{P}\left(1-\frac{1}{N^2}\right) \geq \frac{1}{P}\left(1-\frac{1}{N}\right)$$

4.4 A rend meghatározása az outputból leolvasható függvényéből

Legyen most d olyan szám, ami $\frac{kP}{Q}$ kerekített értéke, vagyis $d \approx \frac{kP}{Q}$. Ekkor

$$|kP - Qd| = \{Pk\}_Q \leq \frac{P}{2}$$

Amit tovább írhatunk úgy, hogy

$$\left|\frac{k}{Q} - \frac{d}{P}\right| \leq \frac{1}{2Q} \leq \frac{1}{2N^2} \leq \frac{1}{2P^2}$$

A következőkben egy algoritmust mutatunk, amivel egy valós számot tudunk tetszőlegesen megközelíteni egy tőrtekből álló sorozattal, amely tagoknak nevezője és számlálója relatív prímek. Ez az algoritmus a valós számot átalakítja lánctörté, és úgy adja meg a sorozat tagjait, hogy a lánctört első bizonyos számú elemét veszi csak figyelembe, és az azok által meghatározott törtet adja.

Az algoritmus lépései során minden olyan tőrt értéket felvesz, amelyre igaz, hogy $\left|x - \frac{a}{b}\right| \leq \frac{1}{2b^2}$. Vagyis a fent igazoltak alapján az algoritmus segítségével megkaphatjuk a P értékét, mint nevezőt, ha relatív prím d-hez.

Az alapgondolat az, hogy legyen egy $a_0, a_1, a_2 \ldots$ sorozat, ahol $a_0 \leq 0$, és

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}} \tag{1}$$

Itt az x tetszőleges valós szám. Ekkor $a_0 = \lfloor x \rfloor$, $x_0 = x - a_0$, és ha $x_n \neq 0$, akkor $a_{n+1} = \left\lfloor \frac{1}{x_n} \right\rfloor$, és $x_{n+1} = \frac{1}{x_n} - a_{n+1}$. Az a_n sorozat ezen definiciójával a fenti módon tetszőlegesen megközelíthető az x értéke. Ha a sorozat első néhány tagjára alkalmazzuk a fenti módszert, akkor egy racionális számot kapunk, amelyet felírhatunk $\frac{p_n}{q_n}$ alakban, ahol p_n és q_n relatív prímek.

Úgy adjuk meg ekkor ezt a két számot, hogy $p_0 = a_0$, $p_1 = a_0 a_1 + 1$, $q_0 = 1$, $q_1 = a_1$, és $p_n = a_n p_{n-1} + p_{n-2}$, $q_n = a_n q_{n-1} + q_{n-2}$.

Tehát most ezen algoritmus segítségével, ha P és d relatív prímek, végül kiszámolható a rend, vagyis P.

Mivel tudjuk, hogy egy racionális számot alakítunk lánctörté, így tudjuk, hogy az algoritmus által meghatározott sorozat véges, és a sorozat elemszáma

az input méretétől logaritmikusan függ. Ekkor tehát a sorozatban szereplő tagok nevezőinek valamelyike veszi fel P értékét, így azokra ellenőrizzük a rendre vonatkozó tulajdonságot, vagyis, hogy $m^P \equiv 1 \pmod{N}$. A hatványozást polinom időben ki tudjuk számolni, ezt pedig maximum annyiszor kell kiszámolni, ahány potenciális P van, és ezzel együtt is polinom ídőben kiszámítható.

Most már csak azt kell ellenőriznünk, hogy P és d relatív prímségének a valószínűsége egy konstanssal alulról becsülhető. Ezt fogjuk becsülni a következőben úgy, hogy k legyen az első regiszterben mért érték, d pedig a fentebb leírt módon a k, Q és P-vel definiált érték. Azon k értékekből, amelyek segítségével tudunk számolni, csak azok lesznek jók amik P-hez relatív prím d értéket adnak. Ez tovább szűkíti a jó mérések valószínűségét. A d érték által felvehető P darab érték közül nekünk csak $\phi(P)$ darab lesz jó, és ezzel a jó mérések valószínásége egy $\frac{\phi(P)}{P}$-s szorzót kap.

$$Prob(k : (P, d) = 1) = \frac{4}{\pi^2}\Big(1 - \frac{1}{N}\Big)^2 \frac{\phi(P)}{P}$$

Bizonyos δ konstansokra megmutatták, hogy $Prob\Big(\frac{\phi(P)}{P}\Big) > \frac{\delta}{\log\log P}$. Ezt felhasználva tehát $O(\log\log P)$ időben találunk alkalmas d számot.

Ezen d szám mellett megtaláljuk polinom időben a m rendjét P-t, ami nagy valószínűséggel olyan lesz, hogy segítségével már N-t is szorzattá tudjuk alakítani. Egyéb esetben újabb véletlen m számot választunk, és arra futtatjuk az algoritmust. Így tehát most már polinom időben tudjuk faktorizálni N-t kvantumszámítógép segítségével.

4.5 A rend meghatározása példaszámítással

Előbbiekben egy példán bemutattuk a Shor-algoritmus azon lépéseinek futását, amelyekhez nem szükséges kvantumszámítógép. Most ezekkel a számokkal folytatjuk a példát, és bemutatjuk, hogy mit csinál a kvantumszámítógép ezen inputokon.

A fentebb bemutatott példában $N = 21$-et választottuk, és hozzá véletlenszerűen a 2-t mint a 21-hez relatív prímszám. Mivel $N^2 = 441$, ezért $N^2 = 441 \leq 2^t = 512 = 2^9 < 2N^2 = 882$, vagyis $t = 9$.

Kezdeti állapotban mindkét regiszter 0-n áll.

$$\varphi_0 = |00\ldots0\rangle|00\ldots0\rangle$$

Majd az első regisztert a hadamard kapu segítségével szétbontjuk egyenlő együtthatójú részekre

$$\varphi_1 = \frac{1}{\sqrt{512}} \sum_{j=0} 511|j\rangle|0\rangle$$

Ezután a V_x operátort alkalmazzuk, és így a második regiszterben a 2 azon hatványait kapjuk modulo 21, amelyek hatványa az első regiszterben lévő érték.

$$\begin{aligned}\varphi_2 = \frac{1}{\sqrt{512}}\Big(\quad & |0\rangle|1\rangle + |1\rangle|2\rangle + |2\rangle|4\rangle + |3\rangle|8\rangle + |4\rangle|16\rangle + |5\rangle|11\rangle + \\ & |6\rangle|1\rangle + |7\rangle|2\rangle + |8\rangle|4\rangle + |9\rangle|8\rangle + |10\rangle|16\rangle + |11\rangle|11\rangle + \\ & |12\rangle|1\rangle + \cdots + |511\rangle|2\rangle\Big)\end{aligned}$$

Ezután átcsoportosítottuk a szummát a második regiszter szerint.

$$\begin{aligned}\varphi_2 = \frac{1}{\sqrt{512}}\Big[\quad & (|0\rangle + |6\rangle + |12\rangle + \cdots + |504\rangle + |510\rangle)|1\rangle + \\ & (|1\rangle + |7\rangle + |13\rangle + \cdots + |505\rangle + |511\rangle)|2\rangle + \\ & (|2\rangle + |8\rangle + |14\rangle + \cdots + |506\rangle)|4\rangle + \\ & (|3\rangle + |9\rangle + |15\rangle + \cdots + |507\rangle)|8\rangle + \\ & (|4\rangle + |10\rangle + |16\rangle + \cdots + |508\rangle)|16\rangle + \\ & (|5\rangle + |11\rangle + |17\rangle + \cdots + |509\rangle)|11\rangle\Big]\end{aligned}$$

A mért véletlen érték a második regiszterben legyen a 2, ezen mutattuk be a számolásokat az elöző példánál is. Ekkor nézzük csak azokat a lehetséges állapotokat, ahol a második regiszter 2, és ezekre megszorítva vizsgáljuk a további lépéseket. Itt a megszorítás lényegében annyit jelent, hogy a szumma többi tagját nem írjuk le, így egyszerűsítve a lépések leírását, és mivel feltételeztük, hogy 2 lesz az érték, amit mérünk, így a szumma többi tagja nem is lényeges az algoritmus szempontjából.

$$\varphi_2 = \frac{1}{\sqrt{86}}(|1\rangle + |7\rangle + |13\rangle + \cdots + |505\rangle + |511\rangle)|2\rangle$$

Erre alkalmazzuk most a Diszkrét Fourier-transzformációt:

$$DFT(\varphi_2) = \varphi_3 = DFT\left(\frac{1}{\sqrt{86}} \sum_{a=0}^{85} |6a+1\rangle\right)|2\rangle$$

$$\varphi_3 = \frac{1}{\sqrt{512}} \sum_{j=0}^{511} \left(\left[\frac{1}{\sqrt{86}} \sum_{a=0}^{85} e^{2\pi i \frac{6ja}{512}} \right] e^{2\pi i \frac{j}{512}} |j\rangle \right) |2\rangle$$

Most tekintsük az első regiszter mérésének lehetőségét. Ez már a rend meghatározása szempontjából sokkal nagyobb szerepet játszik, a második regiszterben mért érték lényegében nem fontos. Az első regiszterben mérhető értékek valószínűsége:

$$Prob(j) = \frac{1}{512 \times 86} \left| \sum_{a=0}^{85} e^{2\pi i \frac{6ja}{512}} \right|^2$$

Itt a legtöbb érték közel 0 valószínűséggel mérhető, de a 0, 85, 171, 256, 341, 427 értékeket nagy valószínűséggel mérhetjük. Ezekből, ha 0-t mérünk, akkor újra kell futtatni az algoritmust és új mérést kell végeznünk. Most tegyük fel, hogy a 85 értéket mérjük.

$$Prob(85) = \frac{1}{512 \times 86} \left| \sum_{a=0}^{85} e^{2\pi i \frac{6 \times 85 \times a}{512}} \right|^2 \approx 0,1142$$

Ezután a $\frac{85}{512}$-hez tartozó lánctört segítségével próbáljuk megkeresni a rendet.

$$\frac{85}{512} = \cfrac{1}{6 + \cfrac{1}{42 + \cfrac{1}{2}}} \tag{2}$$

Tehát, ezzel a módszerrel kapott konvergens sorozat tagjai, amelynek tagjai olyan racionális számok, amelyeknek a nevezője és számlálója relatív prímek, a következőek: $\frac{1}{6}, \frac{42}{253}, \frac{85}{512}$. Ezek alapján az első tagra ellenőrizzük le elsőként, hogy teljesíti-e a feltételt. Ha erre nem lenne igaz, akkor a következő tagra is ellenőrizzük, és így tovább. Az első tag alapján $P = 6$, vagyis $2^6 \equiv 1 \pmod{21}$ kongruenciát kell ellenőriznünk, ez pedig teljesül, vagyis a rend 6.

Irodalomjegyzék

[1] Peter W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", Proceedings of the 35th Annual Symposiump on Foundations of Computer Science, Santa Fe (1994)

[2] SAMUEL J. LOMONACO, JR., "A LECTURE ON SHOR'S QUANTUM FACTORING ALGORITHM VERSION 1.1" (2000)

[3] C. Lavor, L.R.U. Manssur, and R. Portugal, "Shor's Algorithm for Factoring Large Integers" (2008)