Kvantumalgoritmusok
Boole áramkörök szimulációja, faktorizáció, Shor algoritmus

Számítások modellezése

A Boole áramkörök a számítások modellezésére alkalmasak, hasonlóan mint a kvantumáramkörök, Turing gépek, Python programok, stb.

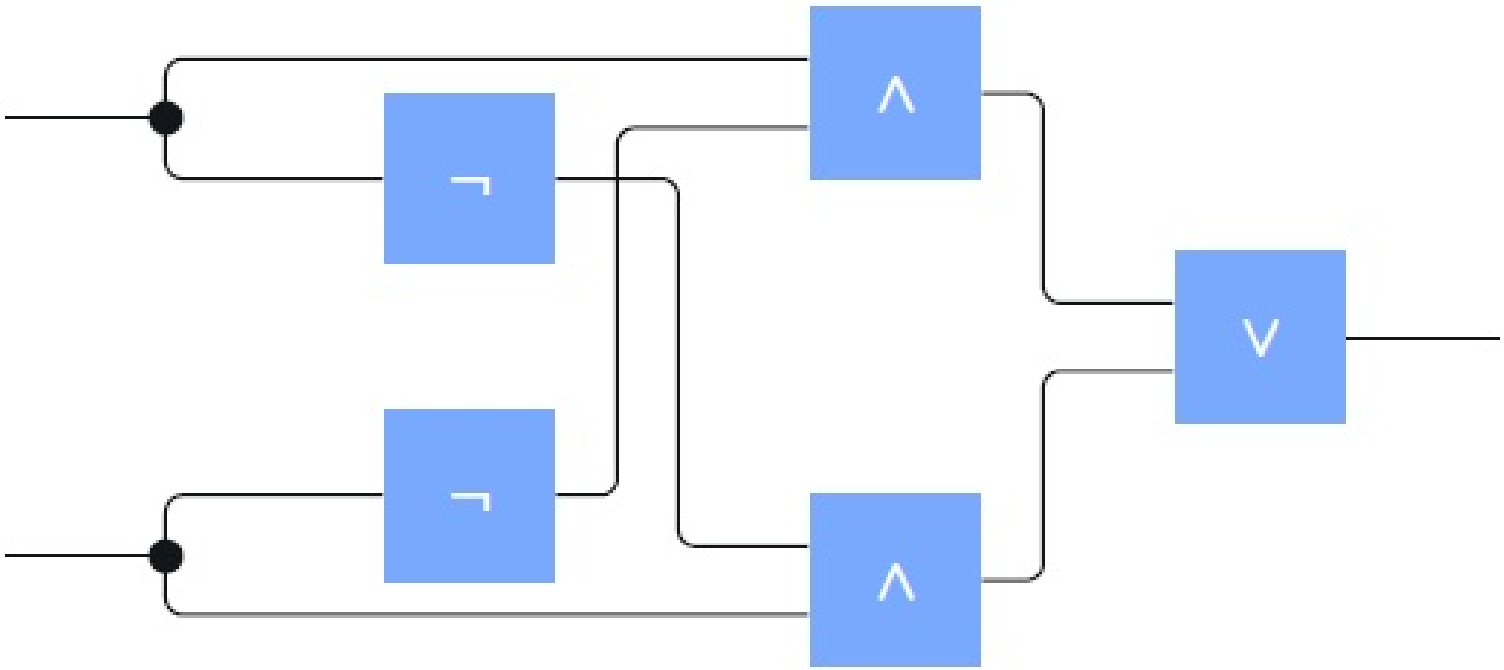
Boole-féle kapuk

1. AND
2. OR
3. NOT
4. FANOUT

Boole áramkör mérete

Egy Boole áramkör mérete az összes kapu száma.
Jelölés: size(C)

Például a következő Boole áramkör mérete 7 :



... és mélysége

A mélysége pedig azon kapuk száma, amelyen legfeljebb átmegy egy input.

Tehát az előző Boole áramkör mélysége 3.

A Boole áramkör mélysége ezért megfelel a párhuzamos futási időigénynek.

Faktorizáció

Input: egész szám n, $n \geq 2$

Output: n prímtényezős felbontása

Példa

$$15 = 3 \cdot 5$$

$$18 = 2 \cdot 3^2$$

Példa

$$340282366920938463463374074317682310984309834 3$$

prímtényezős felbontása

$$3^2 \cdot 7451945066101122 1 \cdot 5073729280707932631243580787$$

Nem ismert felbontás

Az RSA1024 nevezetű szám prímtényezős felbontása nem ismert.
135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

`https://en.wikipedia.org/wiki/RSA_numbers#RSA-1024`

Az RSA gyújteményből eddig legnagyobb felbontott szám az RSA250.

Legnagyobb közös osztó

Input: Két nemnegatív egész szám m és n

Output: m és n legnagyobb közös osztója $lnko(m, n) = gcd(m, n)$

Több hatékoy algoritmus létezik a legnagyobb közös osztó kiszámítására, mint pl. euklideszi algoritmus.

Legnagyobb közös osztó

Input: Két nemnegatív egész szám m és n

Output: m és n legnagyobb közös osztója $lnko(m, n) = gcd(m, n)$

Több hatékoy algoritmus létezik a legnagyobb közös osztó kiszámítására, mint pl. euklideszi algoritmus.

Kérdés: Létezik-e hatékony klasszikus algoritmus egész szám faktorizálására?

Legnagyobb közös osztó

Input: Két nemnegatív egész szám m és n

Output: m és n legnagyobb közös osztója $lnko(m, n) = gcd(m, n)$

Több hatékoy algoritmus létezik a legnagyobb közös osztó kiszámítására, mint pl. euklideszi algoritmus.

Kérdés: Létezik-e hatékony klasszikus algoritmus egész szám faktorizálására?

Válasz: Elképzelhető, de eddig nem ismert.

Klasszikus számítások

Nemnegatív egész számokat bináris jelöléssel ábrázolhatjuk:

szám	kódja	hossza
0	0	1
1	1	1
2	10	2
3	11	2
4	100	3
5	101	3
6	110	3
7	111	3
8	1000	4
9	1001	4
10	1010	4
⋮	⋮	⋮

Kvantum számítások

A számítás kvantumáramkör alapú. Minden *kaput* elemi műveletnek tekintjük.

Kvantum kapuk sztenderd halmaza:

1. Egy-qubites kapuk: $X, Y, Z, H, S, S^\dagger, T, T^\dagger$
2. controlled-NOT kapuk
3. Egy-qubites sztenderd bázis mérések

Kvantum számítások

A számítás kvantumáramkör alapú. Minden *kaput* elemi műveletnek tekintjük.

Kvantum kapuk sztenderd halmaza:

1. Egy-qubites kapuk: $X, Y, Z, H, S, S^\dagger, T, T^\dagger$
2. controlled-NOT kapuk
3. Egy-qubites sztenderd bázis mérések

Az unitér kapuk ebben a halmazban *univerzálisak*, azaz szinte minden unitér leképezés elérhető ezeknek az alkalmazásával.

Költsége

Minden kvantumáramkörnek van egy fix költsége. Szükségünk van egy kvantumáramkörökből álló *családra* $\{C_1, C_2, \ldots\}$ ahhoz, hogy egy algoritmust leírjunk.

Költsége

Minden kvantumáramkörnek van egy fix költsége. Szükségünk van egy kvantumáramkörökből álló *családra* $\{C_1, C_2, \ldots\}$ ahhoz, hogy egy algoritmust leírjunk.

Példa: A faktorizáció egy klasszikus algoritmusát le lehetne írni egy Boole-körökből álló családjával, ahol C_n faktorizál n-bites számokat.

Költsége

Minden kvantumáramkörnek van egy fix költsége. Szükségünk van egy kvantumáramkörökből álló *családra* $\{C_1, C_2, \ldots\}$ ahhoz, hogy egy algoritmust leírjunk.

Példa: A faktorizáció egy klasszikus algoritmusát le lehetne írni egy Boole-körökből álló családjával, ahol C_n faktorizál n-bites számokat.

Egy ilyen algoritmus költsége lenne

$$t(n) = size(C_n)$$

Aszimptotikus jelölés

Legyen $g(n)$ és $h(n)$ két függvény. Azt mondjuk, hogy

$$g(n) = \mathcal{O}(h(n)),$$

ha létezik olyan pozitív konstans c és pozitív egész szám n_0, hogy

$$g(n) \leq c \cdot h(n)$$

minden $n \geq n_0$ esetén.

Példa: $5n^3 - 50n^2 + 3n = \mathcal{O}(n^3)$

Példa Boole körök

Van olyan Boole-féle áramkörökből álló család $\{C_1, C_2, \ldots\}$, ahol C_n két n-bites számot összead úgy, hogy

$$size(C_n) = \mathcal{O}(n)$$

Példa Boole körök

Van olyan Boole-féle áramkörökből álló család $\{C_1, C_2, \ldots\}$, ahol C_n két n-bites számot összead úgy, hogy

$$size(C_n) = \mathcal{O}(n)$$

Ez azt jelenti, hogy n-bites számok összeadását lehet $\mathcal{O}(n)$ költséggel kiszámolni.

További példák

- n bites számok szorzása elvégezhető $\mathcal{O}(n^2)$ költséggel.

További példák

- n bites számok szorzása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok osztása elvégezhető $\mathcal{O}(n^2)$ költséggel.

További példák

- n bites számok szorzása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok osztása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok legnagyobb közös osztójának kiszámítása elvégezhető $\mathcal{O}(n^2)$ költséggel.

További példák

- n bites számok szorzása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok osztása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok legnagyobb közös osztójának kiszámítása elvégezhető $\mathcal{O}(n^2)$ költséggel.
- n bites számok moduláris hatványozása elvégezhető $\mathcal{O}(n^3)$ költséggel.

Shor algoritmus - fő ötletek

Shor faktorizálási algoritmusa egy kvantumalgoritmus, amely nagy számok prímtényezős felbontását végzi hatékonyan.

- Jelentősen gyorsabb, mint a klasszikus algoritmusok
- Kvantumszámítógépen polinomiális időben fut

1. A probléma átalakítása

A faktorizálás visszavezethető egy perióduskeresési problémára.

Adott egy N szám, keresünk egy a-t, amelyre:

$$a^r \equiv 1 \pmod{N}$$

- r a periódus (rend)
- Ez a kulcs a faktorizáláshoz

2. Perióduskeresés mint kulcsfeladat

A lényeg:

- Ha megtaláljuk r-t,
- akkor abból nagy valószínűséggel kiszámíthatók N nemtriviális osztói

Tehát a probléma:

$$\text{Periódus meghatározása } f(x) = a^x \mod N$$

3. Kvantumos párhuzamosság

A kvantumszámítógép szuperpozíciót használ:

- Egyszerre sok x értéket kezel
- Kiszámolja:

$$a^x \mod N$$

- Mindezt egyetlen kvantumállapotban

4. Kvantumos Fourier-transzformáció (QFT)

A kulcslépés:

- A QFT kiemeli a periódust
- Interferenciát használ a helyes r megtalálásához

Ez teszi lehetővé:

r hatékony meghatározását

5. Klasszikus utófeldolgozás

A kvantumos mérés után:

- Klasszikus módszerekkel meghatározzuk r-t (pl. lánctörtek)
- Majd kiszámoljuk az osztókat:

$$\gcd(a^{r/2} \pm 1, N)$$

Miért fontos?

- Klasszikusan: Nincs ismert hatékony faktorizáló algoritmus

Miért fontos?

- Klasszikusan: Nincs ismert hatékony faktorizáló algoritmus
- Shor algoritmusa: Polinomiális időben működik

Miért fontos?

- Klasszikusan: Nincs ismert hatékony faktorizáló algoritmus
- Shor algoritmusa: Polinomiális időben működik
- Következmény: Veszélyezteti az RSA titkosítást

Példa

A cél: a 15 szám prímtényezőkre bontása.

1. Válasszunk egy *a* számot

Legyen $a = 2$, amelyre $\gcd(2, 15) = 1$.

2. Periódus keresése Vizsgáljuk a következő függvényt:

$$f(x) = 2^x \mod 15$$

Számoljuk ki néhány értékét:

$$2^1 \equiv 2 \mod 15$$

$$2^2 \equiv 4 \mod 15$$

$$2^3 \equiv 8 \mod 15$$

$$2^4 \equiv 16 \equiv 1 \mod 15$$

Látható, hogy a periódus:

$$r = 4$$

3. Ellenőrzés r páros, így folytathatjuk.

$$2^{r/2} = 2^2 = 4$$

$$2^{r/2} \not\equiv -1 \mod 15$$

4. Osztók kiszámítása

$$\gcd(4 - 1, 15) = \gcd(3, 15) = 3$$

$$\gcd(4 + 1, 15) = \gcd(5, 15) = 5$$

5. Eredmény

$$15 = 3 \times 5$$

Megjegyzés (kvantumos rész) A periódus (r) meghatározása kvantumszámítógépen történik:

- szuperpozíció létrehozása
- $f(x) = a^x \mod N$ kiszámítása párhuzamosan
- kvantumos Fourier-transzformáció alkalmazása
- mérés $\rightarrow$ periódus információ kinyerése