LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Kvantu algoritmi bumbu meklēšanas modelī

Bakalaura darbs

Autors: **Andrejs Kuzņecovs**
Stud. apl. nr.: ak11525
Darba vadītājs: Andris Ambainis, prof., Dr.dat.

Rīga 2016

Anotācija

Viens no uzdevumiem, kurā kvantu datoriem ir priekšrocības, salīdzinot ar klasiskiem datoriem, ir vaicāšanas uzdevums. Šajā uzdevumā ir dota zināma funkcija f, nezināma bitu virkne x, un melnā kaste, ar kuras palīdzību var piekļūt x bitiem. Mērķis ir uzbūvēt $f(x)$ rēķināšanas algoritmu, izmantojot mazu skaitu melnās kastes vaicājumu .

Viens no modeļiem tādu algoritmu konstruēšanai ir bumbas meklēšanas modelis. Ar šo modeli ir iespējams iegūt kvantu algoritmus ar zemu vaicājumu skaitu dažiem vaicāšanas uzdevumiem.

Šajā darbā šīs modelis ir pielietots dažādām funkcijām f ar mērķi izveidot algoritmus ar mazu vaicājumu skaitu. Iegūtie risinājumi tiek salīdzināti ar risinājumiem, kas izmanto citas kvantu metodes.

Atslēgvārdi: kvantu skaitļošana, vaicājumu algoritmi, vaicājumu sarežģītība, bumbas meklēšanas modelis

Abstract

Quantum algorithms using bomb testing

One of the problems in which quantum computers are better then classical computers is black-box query problem. In this task we are given a known function f, unknown bit string x and a black-box oracle to query bits of x. Our goal is to compute $f(x)$ using low number of black-box queries.

Quantum bomb testing is one of the models to construct these algorithms. This model can be used to construct quantum algorithms that use small number of black-box queries.

In this paper we apply this model to obtain low-query algorithms for several f functions. We compare results with other quantum methods.

Keywords: quantum computing, query algorithms, query complexity, bomb-testing model

Saturs

1 Definīcijas un apzīmējumi

Definīcija 1.1. *Par N-bitu virkni x sauksim kortežu $x = (x_1, x_2, ..., x_N)$, kur $x_i \in \{0, 1\}$. Vienkaršības dēļ rakstīsim $x = x_1x_2...x_N$*

Definīcija 1.2. *Par $N \times M$-bitu matricu A sauksim matricu ar N rindām un M kollonām, kurai visi elementi ir 0 vai 1:*

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,M} \\ A_{2,1} & A_{2,2} & \dots & A_{2,M} \\ \vdots & \vdots & \ddots & \vdots \\ A_{N,1} & A_{N,2} & \dots & A_{N,M} \end{bmatrix}$$

$\forall i, j : A_{i,j} \in \{0, 1\}$

Par A i-to rindu A_i sauksim M-bitu virkni $A_{i,1}A_{i,2}...A_{i,M}$

Tāpat, kā darbā [2], definēsim **1-bloku**:

Definīcija 1.3. *Par N-bitu virknes x 1-bloku sauksim x apakšvirkni $x_l, x_{l+1}, ...x_r$, kurai izpildās:*

- $\forall i : l \leq i \leq r \implies x_i = 1$
- *$l = 1$, vai $x_{l-1} = 0$*
- *$r = N$, vai $x_{r+1} = 0$*

  l un r sauksim par 1-bloka sākumu un beigām, $r - l + 1$ - par bloka izmēru.

Definīcija 1.4 (Leksikografiskais virkņu salīdzinājums)**.** *Pieņemsim, ka x, y ir divas N-bitu virknes. Sakam, ka x ir leksikografiski mazāka par y (rakstām $x < y$), ja $\exists i : 1 \leq i \leq N$, kuram izpildās $\forall j : 1 \leq j < i \implies x_j = y_j$ un $x_i < y_i$.*

$\neg(y < x)$ vietā rakstam $x \leq y$.

2 Ievads

Viens no interesantiem virzieniem mūsdienu datorzinātnē ir kvantu datoru priekšrocību noteikšana un potenciālā izmantošana dažādos skaitļošanas uzdevumos. Ar lielu pārliecību var teikt, ka interesi šai jomai piesaistīja šādi algoritmi: Grovera meklēšanas algoritms [5] un Šora algoritmi [8] skaitļa sadalīšanai pirmreizinatājos un diskrētās logaritmēšanas problēmai.

Grovera meklēšanas algoritms dod iespēju meklēt nesakārtotā sarakstā (datubāzē) ar izmēru N, izmantojot $\approx \sqrt{N}$ vaicājumus, kas ir labāk par N vaicājumiem, kas ir vajadzīgi klasiskiem algoritmiem. Savukārt, Šora piedāvātu algoritmu praktiskā realizācija nozīmē to, ka kriptogrāfiskas sistēmas (RSA, DSA, uz eliptiskām liknēm balstītas kriptosistēmas), kas izmanto hipotēzi par skaitļa faktorizācijas un diskrētās logaritmēšanas problēmas grūtību, vairs nav drošas.

Šajā darbā mēs pielietosim bumbu meklēšanas modeli, kas ir līdzīgs Grovera meklēšanas algoritmam. Šīs modelis potenciāli dod kvadrātisku orākula vaicājumu skaitu samazināšanu dažos uzdevumos, salīdzinot ar klasisku gadījumu.

2.1 Kvantu algoritmi

Iemesls, kāpēc kvantu datori varētu būt labāki par klasiskiem datoriem ir kvantu paralēlisms (detalizētāku skaidrojumu kvantu skaitļošanai un kvantu paralēlismam var atrast vairākos darbos, piemēram darbā [9]).

Kvantu paralēlisma būtība ir šāda: kvantu sistēmai ir N bāzes stāvokļi:

$$|1\rangle, |2\rangle, |3\rangle, ..., |N\rangle$$

Kvantu sistēma var atrasties superpozīcijā no šiem stāvokļiem $|\Psi\rangle = \sum_{i=1}^{N} \alpha_i |i\rangle$, kur $\alpha_i \in \mathbb{C}$ ir i-tā bāzes stāvokļa amplitūda stāvoklim $|\Psi\rangle$.

Papildus tiek prasīts, ka $\sum |\alpha_i|^2 = 1$, un reāls lielums $|\alpha_i|^2 \geq 0$ tiek interpretēts kā varbūtība dabūt bāzes stāvokli i, veicot kvantu stāvokļa mērījumu pret standartbāzi $|1\rangle, |2\rangle, |3\rangle, ..., |N\rangle$. Šajā gadījuma saka, ka sistēma vienlaicīgi atrodas stāvokļos $|1\rangle, |2\rangle, |3\rangle, ..., |N\rangle$ ar varbūtībam $|\alpha_1|^2, |\alpha_2|^2, ..., |\alpha_N|^2$, attiecīgi.

Tādēļ ka kvantu sistēma ar n kubitiem var vienlaicīgi atrasties kādā no $N = 2^n$ bāzes stāvokļiem, tad kvantu dators potenciāli var vienlaicīgi "operēt" ar 2^n stāvokļiem. Līdz ar to, nav izslēgta situācija, ka dažos uzdevumos kvantu datori ir eksponenciāli ātrāki par klasiskiem datoriem.

Grovera meklēšanas algoritms

Kvantu paralēlisma priekšrocība tika izmantota nestrukturētās meklēšanas uzdevumā, kura risinājums ir zināms ka Grovera meklēšanas algoritms [5]. Šajā uzdevumā ir dots nesakārtots saraksts (datubāzē) ar N elementiem, kurā vajag atrast atzīmētus elementus.

Funkcija f aprēķina, vai elements ir atzīmēts:

$$f(i) = \begin{cases} 0, & \text{ja i-tais elements nav atzīmēts} \\ 1, & \text{ja i-tais elements ir atzīmēts} \end{cases}$$

Ir dots orākuls O_x (unitārā transformācija), kas nomaina i-tā bāzes stāvokļa amplitūdu, ja elements ir atzīmēts: $O_x|i\rangle = (-1)^{f(i)}|i\rangle$. Papildus, kvantu stāvoklim ir atļauts pielietot patvaļīgas unitāras transformācijas U_i, kas nav atkarīgas no f: $U_0, U_1, U_2, ...$

Darba [5] pamatrezultāts ir, ka atzīmētu saraksta elementu (ja tāds ir) var atrast ar $O(\sqrt{N})$ vaicājumiem.

2.2 Vaicājumu sarežģītība

Šajā darbā mēs izmantosim sarežģītības definīcijas klasiskiem un kvantu algoritmiem, par pamatu ņemot darbu [7].

Skaitļošanas modeļa formulējums

Ir dota nezināma N-bitu virkne $x_1, x_2, ..., x_N$. Šai virknei ir iespējams piekļūt tikai izmantojot melnas kastes orākulu O_x, kas saņemot ieejā vaicāta bita pozīciju i, izdod x_i.

- Klasiskajā gadījumā var pieņemt, ka ir pieeja "melnai" kastei: funkcijai $O_x(i) = x_i$.
- Kvantu gadījumā var pieņemt, ka ir pieeja orākulam - unitārai transformācijai $O_x|i\rangle = (-1)^{x_i}|i\rangle$, vai līdzīgam orākulam $O_x|i, r\rangle = |i, r \oplus x_i\rangle$.

O_x izmantošanu (f izsaukšanu) sauksim par vaicājumu.

Papildus ir dota **zināma** funkcija $f : \{0,1\}^N \to E$ (funkcija, kas saņemot N-bitu virkni x, izdot rezultātu no kopas E). Ļoti bieži $E = \{0,1\}$, tas ir, funkcija atgriež tikai 1 bitu. Šajā darbā aplūkosim arī funkcijas, kas izdod vairāk par 1 bitu.

Kā piemērs, f var būt OR, AND, XOR, vai jebkāda cita visur definēta funkcija: $f = OR(x_1, x_2, ..., x_N) = x_1 \vee x_2 \vee ... \vee x_N$

Uzdevuma mērķis ir aprēķināt f ar ierobežotu kļūdas varbūtību ϵ, izmantojot orākulu O_x pēc iespējas maz reizes: $\forall u \in E : Pr[\text{algoritms izdod u}|f(x) = u] > 1 - \epsilon$.

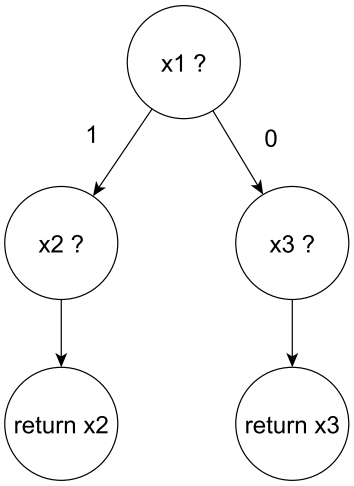
Piezīme: parasti ierobežots ϵ nozīmē, ka $\epsilon < 1/3$. Ja $\epsilon < 1/2$, to var uztaisīt patvaļigi tuvu 0, atkārtojot algoritmu $\mathcal{A}$ C reizes (C ir konstante), un izvēloties no visām atbildēm visbiežāk sastopamu. Vaicājumu skaits palielinās C reizēs, bet konstante C neietekmē asimptotisku novērtējumu vaicājumu skaitam.

Vaicājumu sarežģītība klasiskiem algoritmiem

Jebkādu klasisku algoritmu, kas vaicā x bitus, var aplūkot ka lēmumu koku, kas kārtējā soli izvēlas, kādu mainīgu pavaicāt (lēmumu zara izvēle), izmantojot sistēmas stāvokli - informāciju par jau pavaicātiem x bitiem un papildus informāciju (nejaušības avots).

Atkarībā no tā, vai algoritms izmanto nejaušības avotu, šīs algoritms ir determinēts ($D(f)$ = vaicājumu skaits labākajam iespējamam algoritmam $\mathcal{A}$, kas ir vienāds ar lielāku attālumu līdz koka lapai), vai varbūtisks ($R(f)$ = sagidāmais vaicājumu skaits ceļā līdz koka lapai).

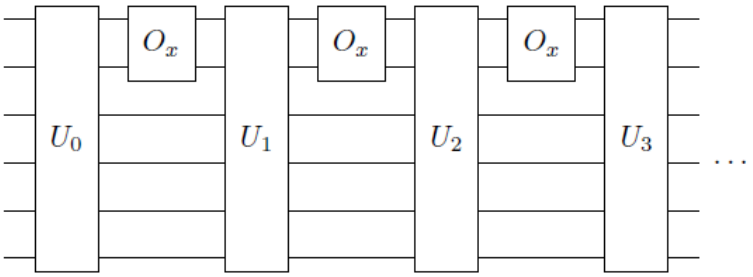
Piemēram, attēlā 1 ir parādīts determinēts vaicājumu algoritms funkcijai $f = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$. Atkarībā no x_1 vērtības, tiek pavaicāts un atgriezts x_2 vai x_3. Šī funkcija ir atkarīga no 3 x bitiem, bet f aprēķināšanai pietiek pavaicāt tikai 2 no tiem.



Att. 1: Determinēts vaicājumu algoritms funkcijai IF x_1 THEN x_2 ELSE x_3

Vaicājumu sarežģītība kvantu algoritmiem

Kvantu algoritmu, kas izmanto orākulu O_x, var aplūkot ka secīgu unitāru transformāciju pielietošanu: $U_0, O_x, U_1, O_x, U_2, O_x, ..., O_x, U_k$, pēc kurām tiek veikts kvantu stāvokļa mērījums (skat. attēlu 2). Šajā gadījumā sarežģītība ir O_x izsaukumu skaits k. Atšķir divu veidu sarežģītības: $Q_E(f)$, kur $\epsilon = 0$ un $Q_2(f)$, kur ϵ ir ierobežots.



Att. 2: Vispārīgā vaicājumu algoritma struktūra

Šajā darbā mēs piedāvājam metodes, kā uzlabot $Q_2(f)$, izmantojot bumbas meklēšanas modeļa [1] priekšrocības.

3 Izmantotās metodes

Šajā nodaļā aprakstīsim metodes, kuras izmantosim, veidojot kvantu algoritmus ar zemu vaicājumu skaitu.

3.1 Bumbas meklēšanas modelis

Viens no perspektīviem modeļiem kvantu algoritmu veidošanai ir bumbas meklēšanas modelis. Tas ir relatīvi nesen izgudrots modelis, kas pirmoreiz tika aprakstīts darbā [1].

Bumbas meklēšanas modeļa pamatā ir idejas par objekta mērījumiem bez mijiedarbības (*angl.: interaction free measurements*), kas tika izmantoti, lai izveidot prāta eksperimentu - Elitzura-Vaidmana bumbas testētāju [3].

Šī modeļa priekšrocības ir salīdzināmas ar Grovera algoritma priekšrocībam (potenciāls kvadrātiskais sarežģītības uzlabojums dažām funkcijam f). Modelis izmanto orākulu CO_x, kas ir līdzīgs orākulam O_x Grovera meklēšanas algoritmā.

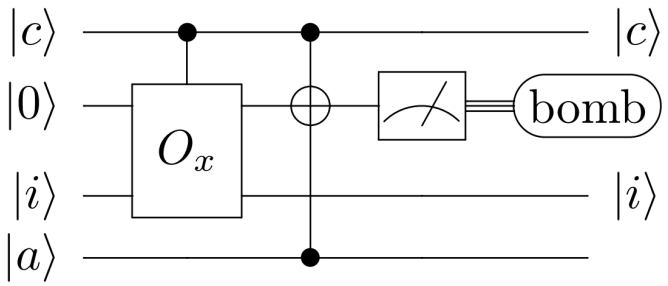
Šīm modelim atbilstošais orākuls CO_x (skat. attēlu 4, izmantojam [1], shēmu 5.1, līdzīgi [1] shēmai 3.2) iedarbojas uz stāvokli, kas satur:

- kontroles bitu c - ar kura palīdzību kontrolējam, vai O_x tiek izpildīts
- ieraksta reģistrs r (tiek prasīts, lai pirms izmantošanas tas saturētu 0)
- pozīcijas reģistru i - tiek vaicāts mainīgais x_i
- a - kas saturēs minējumu par x_i vērtību.

Orākula CO_x iedarbība uz šī stāvokļa ir:

$$CO_x|c, r, i, a\rangle = |c, r \oplus \big[c \cdot (x_i \oplus a)\big], i, a\rangle$$

Uzreiz pēc CO_x pielietošanas, reģistrs r tiek nomērīts, un ja tiek nomērīts stāvoklis 1 (ja $c \cdot (x_i \oplus a)$=1), tad algoritms beidz savu darbību (notiek "bumbas sprādziens").



Att. 3: Kvantu orākula CO_x shēma

Bumbas meklēšanas modelis ļauj izmantot klasiskos vaicājumu algoritmus, lai veidot kvantu algoritmus ar zemu vaicājumu skaitu. Šīs modelis pārveido klasisko vaicājumu algoritmu $\mathcal{AG}$, kas rēķina f, uzminot kārtēja pavaicāta bita vērtību, par kvantu algoritmu. Tagad detalizētāk aprakstīsim, ko tas nozīmē:

Teorēma 3.1 (Bumbu meklēšanas algoritma konstrukcija, [1] teorēma 8)**.** *Pieņemsim, ka f ir funkcija: $f : D \to E$, kur $D \subseteq \{0,1\}^N$.*
Pieņemsim, ka $\mathcal{A}$ ir klasiskais varbūtisks vaicājumu algoritms, kas izrēķina $f(x)$ ar ierobežotu kļūdas varbūtību un izdara ne vairāk par T vaicājumiem.
Pieņemsim, ka $\mathcal{G}$ ir varbūtisks algoritms, kas katrā solī, "zinot" A iekšējo stāvokli (zinot iepriekšējo pavaicātu bitu vērtības, un kārtēja vaicāta bita pozīciju ind), spēj uzminēt bita x_{ind} vērtību, katram x vidēji kļūdoties ne vairāk par G reizēm.
Šādā gadījumā pastāv kvantu algoritms, kas izrēķina f ar ierobežotu kļūdas varbūtību un izdara $O(\sqrt{T \cdot G})$ vaicājumus.

$\mathcal{A}$ un $\mathcal{G}$ tiks aplūkoti kopā kā viens varbūtisks vaicājumu algoritms $\mathcal{AG}$, kas vaicā $\leq T$ no x bitiem, minot kārtēja pavaicāta bita vērtību un izdarot vidēji ne vairāk par G kļūdām visām pieļaujamām x vērtībām.

Par funkcijas f vaicājumu sarežģītību bumbas meklēšanas modelī $B(f)$ sauksim mazāku iespējamu $T \cdot G$ algoritmam $\mathcal{AG}$, kas rēķina f.

Kas šajā modelī būtu "labs" (labāks par klasisku vaicājumu skaita ziņā) algoritms $\mathcal{AG}$? Vajag, lai augšējais novērtējums vidējam kļūdu skaitam G algoritmā $\mathcal{G}$ būtu asimptotiski mazāks par $\mathcal{A}$ vaicājumu skaitu: $G = o(T)$ (citādi var vienkārši pielietot algoritmu $\mathcal{A}$). Protams, $\mathcal{A}$ vaicājumu skaits arī nedrīkst kļūt liels.

Mēs pierādīsim augšējus novērtējumus bumbas meklēšanas sarežģītībai $B(f)$ izvēlētām funkcijam f. Pierādīsim šo novērtējumu, parādot algoritmu $\mathcal{AG}$, kas rēķina f. Pielietojot [1] rezultātus, iegūsim augšējus novērtējumus kvantu vaicājumu sarežģītībai funkcijai f, izmantojot $Q_2(f) = O(B(f))$.

3.2 OR funkcija bumbu meklēšanas modelī

Lai parādīt lasītājam, kāpēc bumbu meklēšanas modeļa pielietošana varētu būt izdevīga, aplūkosim kā piemēru $f = OR(x_1, x_2, ..., x_N)$. Uzbūvēsim algoritmu, kas aprēķina f, izmantojot $\leq N$ vaicājumus, kļūdoties ≤ 1 reizi.

1. Pēc kārtas pavaicājam x_i, $i = 1, 2, ...N$, minot ka $x_i = 0$. Ja uzminējām un $x_i = 0$, turpinām vaicāšanu nākamiem i. $x_i = 1$ gadījumā atgriežam atbildi $true$.

2. Atgriežam kā atbildi $false$.

Katrs x bits tiek vaicāts ne vairāk par vienu reizi, tāpēc $T \leq N$. Pēc pirmās kļūdas algoritms beidz savu darbību, tāpēc $G \leq 1$.

Pielietojot šīm algoritmam teorēmu 3.1, iegūstam kvantu vaicāšanas algoritmu (ar ierobežotu kļūdas varbūtību) ar vaicājumu sarežģītību $O(\sqrt{T \cdot G}) = O(\sqrt{N \cdot 1}) = O(\sqrt{N})$.

Šīs rezultāts nav ārkārtīgi pārsteidzošs, jo darbā [5] tika parādīts algoritms, kas (ar ierobežotu kļūdu) ar $O(\sqrt{N})$ vaicājumiem atrod $i : x_i = 1$ ($OR = 1$), vai paziņo ka tāda i nav ($OR = 0$). Līdzīgi, darbā [6] tika parādīts, ka jebkurš kvantu algoritms, kas izmanto šāda veida melnas kastes orākulus un aprēķina OR ar ierobežotu kļūdu, izmanto vismaz $\Omega(\sqrt{N})$ vaicājumus.

Iespējamais ieguvums no šādas pieejas būtu tāds, ka ir iespējams izdomāt un aprakstīt sarežģītākus kvantu algoritmus, izmantojot intuīciju par klasiskiem algoritmiem $\mathcal{A}$ un "labiem" $\mathcal{G}$ minējumiem.

3.3 Bumbas meklēšanas modeļa paplašinājums divu bitu XOR funkcijai

Lai palielināt savas iespējas ar bumbas meklēšanas modeli atrast efektīvus kvantu vaicāšanas algoritmus, atļausim algoritmam $\mathcal{AG}$ jauna veida vaicājumus ar minēšanu - papildus pie x_i vaicāšanas, atļaujam $x_i \oplus x_j$ vaicājumus. Nosauksim orākulu (skat. attēlu 4), kas atļauj šādus vaicājumus par CO_{xx}.

Līdz ar to, algoritmam $\mathcal{AG}$ kļūst pieejami 4 veida vaicājumi:

- Pavaicāt x_i, minot ka $x_i = 0$, vai $x_i = 1$ (2 veidi)
- Pavaicāt $x_i \oplus x_j$, minot ka $x_i \oplus x_j = 0$, vai $x_i \oplus x_j = 1$ (2 veidi)

Līdzīgi, kā orākulam CO_x, jauniem vaicājumiem atbilstoša orākula darbība uz stāvokļa, kas satur kontroles reģistru c, ieraksta reģistru r, pozīcijas reģistrus i, j un minējuma reģistru a:

$$CO_{xx}|c, r, i, j, a\rangle = |c, r \oplus \big[c \cdot (x_i \oplus x_j \oplus a)\big], i, j, a\rangle$$

Jaunajam orākulam prasām, lai pirms pielietošanas r būtu 0. Uzreiz pēc pielietošanas algoritms beidz darbību ("bumbas sprādziens"), ja $c \cdot (x_i \oplus x_j \oplus a) = 1$.

Līdzīgi, ka [1] 5. lemmā tika definēta bitu virkne $\tilde{x}$, definēsim $2N + 2N^2$-bitu virkni $\hat{x}$, kas saturēs x, x kur katrs bits pamainīts uz pretējo, katru XOR funkciju no diviem x bitiem un katras XOR funkcijas no diviem x bitiem pretējo funkciju.

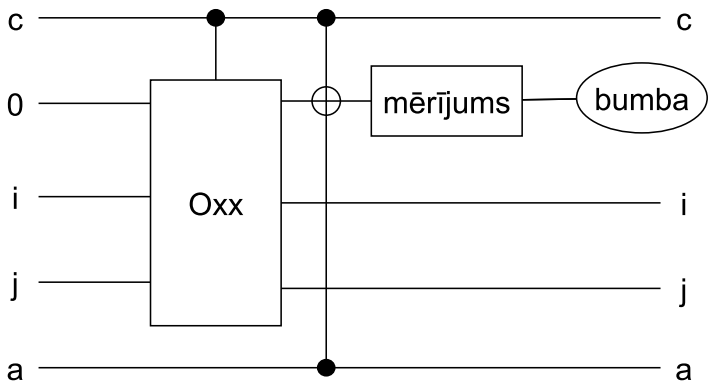
- $\hat{x}_i = x_i$
- $\hat{x}_{i+N} = \neg x_i$
- $\hat{x}_{2N+(i-1)\cdot N+j} = x_i \oplus x_j$
- $\hat{x}_{2N+N^2+(i-1)\cdot N+j} = \neg(x_i \oplus x_j)$

visām $1 \leq i, j \leq N$ vērtībām.

Piezīme: var pamanīt, ka mēs "izšķerdīgi" definējam $\hat{x}$ - jo virkne $\hat{x}$ satur

1. $x_i \oplus x_i = 0$, kas nedod informāciju par x
2. $x_i \oplus x_j$ un $x_j \oplus x_i$ dublē viens otru
3. $x_i \oplus x_j = 1$ minējošie vaicājumi netiek izmantoti šajā darbā
4. Šajā darbā tiek izmantota tikai (asimptotiski) mazā daļa no $x_i \oplus x_j, i < j$ vaicājumiem.

Darba autors tomēr aizstāvēs šādu $\hat{x}$ definīciju, jo punktu 1.,2.,3. optimizācija samazina $\hat{x}$ izmēru tikai ≈ 4 reizēs. Punktu 1. un 2. optimizācija palielina definīcijas sarežģītību, punktu 3. un 4. optimizācija samazina simetriju un iespējamus nākamus modeļa paplašinājuma lietojumus.



Att. 4: Kontrolēta kvantu orākula CO_{xx} shēma

Vēl viens iemesls, kādēļ šo konstrukciju varētu uzskatīt par neefektīvu, ir tas, ka jaunieviestas virknes izmērs palielinājas no $2N = O(N)$ līdz $2N + 2N^2 = O(N^2)$ bitiem. Tomēr, šī konstrukcija tikai ≈ 2 reizēs palielina izmantoto kubitu skaitu kontrolēta orākula shēmā, un [1] algoritmā 8. novērtējumos netiek izmantots paplašīnātas bitu virknes $\tilde{x}$ garums.

Ieviešot CO_{xx} var pieņemt, ka katrs CO_{xx} vaicājums "maksā" tikpat daudz, cik 2 CO_x vaicājumi (līdzīgi ar kļūdu), bet tādēļ ka T un G tiek novērtēti asimptotiski, konstantei 2 nav ietekmes uz algoritma vaicājumu skaitu.

Šajā darbā paplašinātais vaicājumu "komplekts" ar XOR-vaicājumiem tiks pielietots tikai dažos uzdevumos, tādēļ ka darba autoram nav stingru pierādījumu par orākulu O_{xx}, CO_{xx} iespējamu impelementāciju.

3.4 Citas kvantu skaitļošanas metodes

Definīcija 3.1. *Par Būla formulu $F = F(x_1, x_2, ...x_N)$ izmērā S sauksim Būla izteiksmi, kas ir atkarīga no Būla mainīgajiem, un kura sastav tikai no NOT, OR, AND loģiskiem elementiem un sākotnējiem mainīgajiem.*

Formulā prasām, lai katra starpizteksme, kas atkarīga no $x_1, x_2, ...x_N$ tiktu aprēķināta katru reizi, kad tā ir nepieciešama tālākiem aprēķiniem.

Par formulas izmēru sauksim lielumu: $S = \sum_{i=1}^{N} used(i)$, kur $used(i)$ parāda, cik reizes mainīgais x_i ir sastopams šajā formulā.

Piemērs: Būla funkciju (IF x_1 THEN x_2 ELSE x_3) var aprēķināt, izmantojot formulu $(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$. Formulas izmērs: $used(1) + used(2) + used(3) = 2 + 1 + 1 = 4$.

Teorēma 3.2 (Kvantu algoritms formulas rēķināšanai, [4] teorēma 1)**.** *Ja $F(x_1, x_2, ..., x_N)$ ir Būla formula ar izmēru S, tad pastāv kvantu algoritms, kas aprēķina F, izmantojot $O(\sqrt{S})$ vaicājumus.*

Teorēma 3.3 (Grovera meklēšanas algoritms, [5])**.** *Pieņemsim, ka f ir funkcija: $f : \{1, 2, ..., N\} \to \{0, 1\}$, un $f(i) = 1$ K dažādiem i (K var būt nezināms). Tādā gadījumā pastāv kvantu algoritms, kas atrod $i : f(i) = 1$ ar ierobežotu kļūdas varbūtību, izmantojot $O(\sqrt{\frac{N}{K}})$ f vaicājumus pie $K \geq 1$, vai ar ierobežotu kļūdu paziņo, ka tāds i nepastāv pie $K = 0$.*

4 Vaicājumu algoritmi bitu virknēm

Šajā nodaļā aprakstīsim klasiskus vaicāšanas algoritmus dažādām funkcijam f, kurus ar teorēmas 3.1 palīdzību pārveidosim par bumbas meklēšanas algoritmiem.

4.1 Perioda pārbaude

Formulējums

Ir dota N-bitu virkne x. Vajag pārbaudīt, vai tā ir periodiska ar periodu K ($K \leq N$).

Formālāk: vai visiem $i : 1 \leq i \leq N - K$ izpildās $x_i = x_{i+K}$.

Autors grib uzsvert, ka šīs uzdevums nav saistīts ar līdzīgu perioda atrašanas uzdevumu, kas tiek izmantots Šora algortimā [8] tādēļ, ka netiek prasīts, ka $x_i \neq x_{i+p}$, kur $p < K$.

Risinājums

Sākumā parādīsim algoritmu, kā pavaicāt pirmos K bitus. Algoritms 1:

1. Pēc kārtas vaicājam x_i, $i = 1, 2, ..., K$, minot ka $x_i = 1$.
2. Atgriežam visus pavaicātus bitus: $x_1, x_2, ..., x_K$

Izmantojot algoritma 1 pavaicātos mainīgus, vaicāsim atlikušus mainīgus. Algoritms 2:

1. Pēc kārtas vaicājam visus atlikušos x_i ($i = K + 1, K + 2, ..., N$), minot ka $x_i = x_{i-K}$ (x_{i-K} jau tika pavaicāts).
   Ja $x_i = x_{i-K}$, turpinām vaicāšanu nākamiem i. Ja $x_i \neq x_{i-K}$, atgriežam atbildi $false$
2. Ja izgājam visiem i no $K+1$ līdz N un nekur nekļūdījamies, tad atgriežam atbildi $true$.

Tagad apvienosim šos algoritmus: pielietojosim teorēmu 3.1 algoritmiem 1 un 2 dabūsim kvantu algoritmus 1' un 2'.

Nomērīsim algoritma 1' atbildi. Izmantosim šo atbildi algoritmam 2'.

Piezīme: algoritmu 1 varēja aizvietot šādi: pēc kārtas $i = 1, 2, ..., K$ pielietojam orākulu uz stāvokļa $|i\rangle$, nomēram un iegūstam x_i, kurus izmantojam algoritmam 2.

Analīze

Algoritms 1: $T_1, G_1 \leq K$ (tiek pavaicāti pirmie K mainīgie, sliktākajā gadījumā nokļūdāmies K reizes, ja tie visi bija 0).

Algoritms 2:

$T_2 \leq N - K$ (tiek pavaicāti ne vairāk par $N - K$ atlikušiem mainīgie)

$G_2 \leq 1$ (jo pēc pirmās kļūdas algoritms apstājas)

Līdz ar to, ieguvām (kombinēto) bumbu meklēšanas algoritmu ar vaicājumu sarežģītību $O(\sqrt{T_1 \cdot G_1} + \sqrt{T_2 \cdot G_2}) = O(K + \sqrt{N - K}) = O(K + \sqrt{N})$.

Cits risinājums

Lai noteikt, vai virkne ir periodiskā, uzkonstruēsim šādu formulu:

$F(x_1, x_2, ..., x_N) = \bigwedge\limits_{1 \leq i \leq K} \left[\left(\bigwedge\limits_{1 \leq j \leq N \wedge j \equiv i (\mathrm{mod} K)} x_j \right) \vee \left(\bigwedge\limits_{1 \leq j \leq N \wedge j \equiv i (\mathrm{mod} K)} \neg x_j \right) \right]$

Ar izteiksmi $\left(\bigwedge\limits_{1 \leq j \leq N \wedge j \equiv i (\mathrm{mod} K)} x_j \right) \vee \left(\bigwedge\limits_{1 \leq j \leq N \wedge j \equiv i (\mathrm{mod} K)} \neg x_j \right)$ pārbaudām, vai visi elementi, kas ir pozīcijās $i, i + K, i + 2 \cdot K, ...$ ir vienādi. Uztaisot konjunkciju no šādām izteiksmēm priekš $i = 1, 2, ..., K$ pārbaudām, vai virkne ir periodiska.

Tādēļ ka katrs mainīgais x_i tiek izmantots tieši 2 reizes, formulas izmērs ir $O(2 \cdot N) = O(N)$. Ja $K > \frac{N}{2}$, tad no formulas var izmest dažas izteiksmes formā $x_i \vee \neg x_i$, iegūstot formulu izmērā $O(N - K)$

Pielietojot teorēmu 3.2 iegūstam, ka pastāv kvantu algoritms, ka izrēķina šo formulu ar $O(\sqrt{N - K}) = O(\sqrt{N})$ vaicājumiem.

Risinājums ar 2 bitu XOR vaicājumiem

Tagad uzbūvēsim bumbas meklēšanas algoritmu, kas drīkst izmantot arī $x_i \oplus x_j = 0$ vaicājumus:

1. Pēc kārtas vaicājam $x_i \oplus x_{i+K}$, minot ka $x_i \oplus x_{i+K} = 0$, $i = 1, 2, ..., N - K$
   Veiksmīga minējuma gadījumā: $x_i \oplus x_{i+K} = 0 \iff x_i = x_{i+K}$, turpinām vaicāšanu nākamiem i.
   Kļūdas gadījumā: $x_i \oplus x_{i+K} = 1 \iff x_i \neq x_{i+K}$, atgriežam atbildi $false$

2. Ja pārbaudījām visus i un nekļūdījāmies, atgriežam atbildi $true$.

Novērtēsim T un G:

$T \leq N - K$, jo tiek pavaicāti ne vairāk par $N - K$ bitu pāru XOR funkcijas.
$G \leq 1$, jo pēc pirmās kļūdas algoritms beidz darbību.

Iegūstam bumbas meklēšanas algoritmu ar sarežģītību $O(\sqrt{T \cdot G}) = O(\sqrt{N - K}) = O(\sqrt{N})$

Risinājumu salīdzinājums

Izmantojot XOR-paplašinātu bumbas meklēšanas modeli, un teorēmu 3.2 formulu rēķināšanai, dabūjam algoritmus ar sarežģītību $O(\sqrt{N - K})$. Izmantojot nepaplašinātu bumbas meklēšanas modeli, ieguvām algoritmu ar sarežģītību $O(K + \sqrt{N - K})$, kas ir asimptotiski sliktāks, ja $K \neq O(\sqrt{N})$.

4.2 K vieninieki

Formulējums

Ir dota N-bitu virkne x. Vajag pārbaudīt, vai tajā ir tieši $1 \leq K \leq N$ vieninieki: $\sum_{i=1}^{N} x_i = K$.

Risinājums

1. Piešķiram $s \leftarrow 0$
2. Kamēr $s \leq K$, pēc kārtas vaicājam x_i, $i = 1, 2, ..., N$, minot ka $x_i = 0$. Ja neesam uzminējuši un $x_i = 1$, tad palielinām s par 1: $s \leftarrow s+1$. Turpinām vaicāt nākamus bitus.
3. Atgriežam kā atbildi patiesumvērtību $s = K$

Analīze

T: $T \leq N$ - sliktākajā gadījumā tiek pavaicāti visi biti.

G: $G \leq K+1$ - pēc $K+1$ kļūdām esam pārliecinājušies, ka virknē ir vismaz $K+1$ vieninieki.

Pielietojot teorēmu 3.1, iegūstam kvantu algoritmu šīm uzdevumam ar vaicājumu sarežģītību $O(\sqrt{N \cdot (K+1)}) = O(\sqrt{N \cdot K})$

Cits risinājums

Šajā risinājumā vairākkārt pielietosim Grovera algoritmu, katru reizi mēģinot atrast jaunu vienīnieku (līdzīgas idejas autors atrada avotā [10]):

1. Piešķiram $S \leftarrow \varnothing$ (atrasto vienīnieku pozīciju kopa), $Q \leftarrow 0$ (izmantoto vaicājumu skaits visās Grovera meklēšanās)
2. Kamēr $Q \leq C\sqrt{N \cdot K}$ un $|S| \leq K$: (C ir konstante, kuru izvēlēsimies vēlāk):
   Pielietojam Grovera meklēšanas algoritmu funkcijai $f(i) := (x_i = 1) \wedge (i \notin S)$, uzskatam ka šajā meklēšanā tika izmantoti q vaicājumi.
   Atjaunojam Q: $Q \leftarrow Q+q$. Ja tika atrasts i: $(x_i = 1) \wedge (i \notin S)$, atjaunojam S: $S \leftarrow S \cup \{i\}$
3. Atgriežam patiesumvērtību $K = |S|$

Cita algoritma analīze

Ievērojam, ka S satur tikai $i : x_i = 1$, katrs tāds i tiek iekļauts tikai vienu reizi.

Ievērojam, ka piedāvātais algoritms izsauc Grovera meklēšanu $\leq C\sqrt{N \cdot K}$ reizes (jo Q visu laiku palielinās), un izmantotais vaicājumu skaits ir $\leq C\sqrt{N \cdot K} + Q = C\sqrt{N \cdot K} + C\sqrt{N \cdot K} + \sqrt{N} = O(\sqrt{N \cdot K})$ (pirmais saskaitāmais - $C\sqrt{N \cdot K}$ vaicājumi, lai pārbaudīt, ka $x_i = 1$).

Pieņemsim, ka sagaidāmais vaicājumu skaits, lai atrast vienu no U vieniniekiem ar Grovera meklēšanu ir $c\sqrt{\frac{N}{U}}$.

Tad sagaidāmais vaicājumu skaits E, lai atrast $\leq min(U, K+1)$ vieniniekus bitu virknē x ar U atzīmētiem elementiem virknē x ir $E \leq \sum_{i=1}^{K+1} c\sqrt{\frac{N}{i}} \leq \int_0^{K+1} c\sqrt{\frac{N}{x}}dx = c\sqrt{N} \times (2\sqrt{x}\Big|_0^{K+1}) = 2c\sqrt{N}\sqrt{K+1} \leq 2\sqrt{2}c\sqrt{N \cdot K} = O(\sqrt{N \cdot K})$. Līdz ar to, ja tiks izvēlēts $C \geq (2\sqrt{2}c+1)c$, tad ar ar papildus $c\sqrt{N \cdot K} \geq c\sqrt{N}$ vaicājumiem jāpietiek, lai "pārliecināties", ka virknē x tika atrasti visi iespējamie vienīnieki.

Citi lietojumi

Šajā uzdevumā piedāvātie algoritmi atrod $\leq K$ no x vieniniekiem (visus vieniniekus, ja x vienīnieku skaits $\leq K$). Tāpēc ar līdzīgu pieeju varētu risināt šādus uzdevumus:

- Pārbaudīt vai vienīnieku skaits virknē x ir mazāks par K, vai lielāks nekā K.
- Aprēķināt funkciju $f(x)$, kur $f(x)$ ir konstante visiem x, kuros vienīnieku skaits $\geq K$ (šādu lietojumu apskatīsim nākamajā uzdevumā).

4.3 2 vieninieki attālumā $< K$

Formulējums

Ir dota N-bitu virkne x. Vajag noteikt, vai pastāv pozīcijas u, v, kuram izpildās:

$$(x_u = x_v = 1) \wedge (u < v < u + K)$$

Risinājums

Tāpat, kā iepriekšējā uzdevumā, izmantosim stratēģiju, kura min, ka biti ir 0.

1. Piešķiram $u \leftarrow -K$
2. Pēc kārtas vaicājam x_i $(i = 1, 2, ..., N)$ minot ka $x_i = 0$.
   Ja $x_i = 0$, turpinām vaicāšanu nākamiem i.
   Ja $x_i = 1$ un $i < u + K$, atgriežam $true$
   Pēdējā gadījumā ($x_i = 1$, un $u + K \leq i$) piešķiram $u \leftarrow i$ un turpinām vaicāšanu.
3. Atgriežam $false$.

Analīze

Korektība: ievērojam, ka algoritma darbības laikā (izņemot brīdi, uzreiz pēc $u \leftarrow i$) u ir vislielāka pozīcija j tāda, ka $x_j = 1 \wedge j < i$.

Papildus ievērojam, ka algoritms neizdod $-K$ kā pirmo pozīciju, jo $i < -K + K$ nav spēkā pozitīviem i, un (u, i) ir atgriezts tikai pēc visu prasīto nosacījumu pārbaudes.

T: $T \leq N$, jo katrs bits tiek pavaicāts ne vairāk par 1 reizi.

G: $G \leq \lfloor \frac{N}{K} \rfloor + 2 = O(\lfloor \frac{N}{K} \rfloor)$.

Pieņemsim, ka algoritms nokļūdījas pozīcijās $p_1, p_2, ...p_{G-1}, p_G$. Tādā gadījumā $\forall i : 1 \leq i \leq G - 2 \implies p_i + K \leq p_{i+1}$ (citādi pēc $i + 1$ kļūdam algoritms apstātos). Līdz ar to $\forall i : 1 \leq i \leq G - 1 \implies p_i \geq 1 + K \cdot (i - 1)$ un $1 + K \cdot (G - 2) \leq p_{G-1} \leq N$, no kā seko prasītais.

Izmantojot teorēmu 3.1 iegūstam kvantu algoritmu ar vaicājuma sarežģītību $O(\sqrt{T \cdot G}) = O(\frac{N}{\sqrt{K}})$

Cits risinājums - pieeja ar Grovera meklēšanu

Līdzīgi, kā uzdevumā "K vieninieki", atradīsim $\lfloor \frac{N}{K} \rfloor + 2$ vieniniekus, atkārtoti izmantojot Grovera meklēšanu. Atrisināsim šo uzdevumu, izmantojot $O(\sqrt{N \cdot \frac{N}{K}}) = O(\frac{N}{\sqrt{K}})$ vaicājumus. Sakārtosim atrastās pozīcijas, iegūstot sarakstu $p_1, p_2, ..., p_M$. Iziesim cauri sarakstam, atgriežam $true$, ja $\exists i : p_{i+1} < p_i + K$. Iegūtā saraksta kārtošana un meklēšana tajā neprasa orakula vaicājumus, tāpēc vaicājumu sarežģītība ir $O(\frac{N}{\sqrt{K}})$.

Cits risinājums - pieeja ar formulas rēķināšanu

Uzbūvēsim šādu formulu: $F(x_1, x_2, ...x_N) = \bigvee\limits_{1 \le i \le N} \Big[\bigvee\limits_{i<j<min(N,i+K)} (x_i \wedge x_j) \Big]$.

Ar izteiksmi $\bigvee\limits_{i<j<min(N,i+K)} (x_i \wedge x_j)$ pārbaudam vai i var būt vienāds ar meklēto pozīciju u. Uztaisot disjunkciju no šadām izteiksmēm priekš $i = 1, 2, ...N$ pārbaudam, vai tāds i vispār pastāv.

F izmērs ir $O(N \cdot K)$. Pielietojot teorēmu 3.2, iegūstam kvantu algoritmu, kas rēķina F ar $O(\sqrt{N \cdot K})$ vaicājumiem.

Risinājums ar citu bumbas meklēšanas modeļa paplašinājumu

Iespējams, ka šīm uzdevumam būtu izdevīgi ieviest vēl vienu paplašinājumu bumbas meklēšanas modelim - līdzīgi kā XOR paplašinājumam, AND paplašinājumu, kas ļautu algoritmam $\mathcal{AG}$ ar vaicājumu minēt $x_i \wedge x_j$ vērtību.

Šādā gadījumā algoritms varētu aprēķināt iepriekš aprakstītu Būla formulu F, izmantojot $T = O(N \cdot K)$ vaicājumus, un kļūdoties $G \le 1$ reizi, līdz ar to iegūstot kvantu algoritmu ar vaicājumu sarežģītību $O(\sqrt{N \cdot K})$.

Risinājumu salīdzinājums

Ar bumbas meklēšanas metodi uzbūvējam kvantu algoritmu ar $O(\frac{N}{\sqrt{K}})$ vaicājumiem (un, iespējams, ar $O(\sqrt{N \cdot K})$ vaicājumiem), ar citām kvantu metodēm - $O(\frac{N}{\sqrt{K}})$ un $O(\sqrt{N \cdot K})$ vaicājumiem.

- Ja $K \le \sqrt{N}$, ir izdevīgi izmantot $O(\sqrt{N \cdot K})$ metodi.
- Ja $\sqrt{N} \le K$, tad ir izdevīgi izmantot kādu no $O(\frac{N}{\sqrt{K}})$ metodēm.

Izmantojot labāko no metodēm, visiem K ieguvām algoritmu ar $O(N^{3/4})$ vaicājumiem.

4.4 K vieninieki pēc kartas

Formulējums

Ir dota N-bitu virkne x. Vajag pateikt, vai tajā ir 1-bloks ar izmēru $\geq K$.

Risinājums

1. Piešķiram $L \leftarrow 0$
2. Kamēr $L + K \leq N$:
   - Vaicājam x_{L+K}, minot ka tas ir 1.
   - ja neesam uzminējuši, un $x_{L+K} = 0$, tad piešķiram $L \leftarrow L + K$ un turpinām soli 2.
   - ja esam uzminējuši, un $x_{L+K} = 1$, piešķiram $s, t \leftarrow L + K$
   - Atrodam tagadēja 1-bloka kreiso galu: kamēr neesam kļūdījušies, un $s - 1 > L$ vaicājam x_{s-1} minot, ka $x_{s-1} = 1$. Ja uzminējām, tad samazinām s par 1.
   - Atrodam tagadēja 1-bloka labo galu: kamēr neesam kļūdījušies, un $t + 1 \leq N$ vaicājam x_{t+1} minot, ka $x_{t+1} = 1$. Ja uzminējām, tad palielinām t par 1.
   - Ja $t - s + 1 \geq K$, atgriežam *true*.
   - Citādi, esam nokļūdījušies ($x_{t+1} = 0$) piešķiram $L \leftarrow t+1$ un turpinām soli 2.
3. atgriežam atbildi $false$

Analīze

Lemma 4.1. *Piedāvātais algoritms atrod* 1*-bloku izmērā* $\geq K$*, ja tāds ir.*

Pierādījums. Ievērojam šādus faktus:

1. Vienmēr ir spēkā: $L = 0$, vai $x_L = 0$ (jo piešķiram $L \leftarrow ind$ tikai tad, kad ir zināms, ka $x_{ind} = 0$)
2. Pa kreisi no L nekad nav 1-bloka ar izmēru $\geq K$.
   Tas ir patiess, jo vienīgs iemesls, kādēļ bits pozīcijā i varētu tikt izlaists (netiek vaicāts), ir tad, ja kādā brīdī $L < i < L + K$.
   Ja i pieder 1-blokam ar izmēru $\geq K$, tad arī $L + K$ pieder šīm 1-blokam (jo $L = 0$, vai $x_L = 0$), un šīm blokam tiks atrasts sākums un beigas: s un t.

□

Lemma 4.2. *Solis 2. tiek izpildīts ne vairāk par* $\lfloor \frac{N}{K} \rfloor$ *reizēm.*

Pierādījums. Ievērojam to, ka pēc soļa 2. $L \leftarrow L + K$, vai $L \leftarrow t + 1 > L + K$, tātad L katrā soļa 2. iterācijā palielinās vismaz par K. Tādēļ, ka visā izpildes laikā $L \leq N$, seko prasītais. □

Tagad novērtēsim T un G:

T: $T \leq N$ - ievērojam, ka izpildot soli 2., vaicājam tikai elementus ar indeksiem $> L$, un soļa beigās jaunā L vērtība kļūst vienāda ar pēdējā pavaicāta bita indeksu.

G: $G \leq 2 \cdot \frac{N}{K}$. Algoritms kļūdās tikai tad, kad tiek vaicāts x_{L+K} (≤ 1 kļūda uz soļa 2. iterāciju), x_{s-1} (≤ 1 kļūda), x_{t+1} (≤ 1 kļūda).

Papildus ievērojam to, ka nokļūdoties x_{L+K} vaicājumā, s un t netiek meklēti, tātad katrā soļa 2. iterācijā notiek ne vairāk par 2 kļūdām.

Tātad, pielietojot teorēmu 3.1, iegūstam bumbu meklēšanas algoritmu ar vaicājumu sarežģītību $O(\sqrt{T \cdot G}) = O(\sqrt{\frac{2 \cdot N^2}{K}}) = O(\frac{N}{\sqrt{K}})$

Cits risinājums

Lai noteikt, vai virknē ir K vienādi pēc kārtas, uzbūvēsim šādu formulu:

$F(x_1, x_2, ..., x_K) = \bigvee_{1 \leq i \leq N-K+1} (\bigwedge_{i \leq j \leq i+K-1} x_j)$.

Ar izteiksmi $\bigwedge_{i \leq j \leq i+K-1} x_j$ pārbaudām, vai virknē ir K vienādi pēc kārtas, sākot ar pozīciju i. Uztaisot konjunkciju no šādām izteiksmēm priekš $i = 1, 2, ..., N - K + 1$ pārbaudām, vai virknē vispār ir šāda pozīcija.

Iegūtas formulas F izmērs ir $O(K \cdot (N - K + 1)) = O(K \cdot (N - K))$. Pielietojot teorēmu 3.2 iegūstam kvantu algoritmu, kas izrēķina F, izmantojot $O(\sqrt{K \cdot (N - K)})$ vaicājumus.

Risinājumu salīdzinājums

Ieguvām divus dažādus algoritmus:

1. bumbu meklēšanas algoritmu ar vaicājumu skaitu $O(\frac{N}{\sqrt{K}})$
2. "formulas rēķināšanas" algoritmu ar vaicājumu skaitu $O(\sqrt{K \cdot (N - K)})$.

Salīdzināsim (asimptotiski) vaicājumu skaitu šiem algoritmiem pie dažādiem K.

- Ja $K \leq \sqrt{N}$, tad $\sqrt{K \cdot (N - K)} = \Theta(\sqrt{K \cdot N}) \leq \frac{N}{\sqrt{K}}$, un ir izdevīgi izmantot formulas rēķināšanas algoritmu.
- Ja $\sqrt{N} \leq K \leq \frac{N}{2}$, tad joprojām $\sqrt{K \cdot (N - K)} = \Theta(\sqrt{K \cdot N})$, bet $\sqrt{K \cdot N} \geq \frac{N}{\sqrt{K}}$ un ir izdevīgi izmantot pirmo algoritmu.
- Ja $\frac{N}{2} \leq K \leq N$, tad $\frac{N}{\sqrt{K}} \leq \frac{N}{\sqrt{N/2}} = O(\sqrt{N})$, un $\sqrt{K \cdot (N - K + 1)} \geq \sqrt{N/2 \cdot 1} = \Theta(\sqrt{N})$, un joprojām ir izdevīgi izmantot pirmo algoritmu.

Izmantojot labāko no šīm metodēm, visiem K dabūjam algoritmu ar $O(N^{3/4})$ vaicājumiem.

Piezīme: var pamanīt, ka šīm un "2 vieninieki attālumā < K" uzdevumam ir līdzīga struktūra, jo $\left[$ pastāv 2 vieninieki attālumā $< K$ $\right] = \left[$ pastāv 2 vieninieki attālumā 1 $\right] \bigvee \left[$ pastāv 0-bloks (kurus definējām līdzīgi 1-blokiem), kas nav malējais bloks, un tā garums ir $< K$ $\right]$. Iespējams, līdzīgas struktūras dēļ, ieguvam līdzīgus novērtējumus algoritmu sarežģītībai. Autors var minēt, vai šiem diviem uzdevumiem pastāv risinājumi kas būtu labāki par $O(N^{3/4})$ visām K vērtībām.

4.5 Viena 1-bloka meklēšana, kuram zināms minimālais garums

Šajā apakšnodaļa nedaudz uzlabosim vaicājuma sarežģītību darba [2] uzdevumam 4.2.

Formulējums

Ir dota N-bitu virkne x. Zināms, ka tajā ir tieši viens 1-bloks, kura garums ir $\geq L$. Vajag atrast šī bloka sākumu un beigas l un r.

Risinājums

Vispirms parādīsim algoritmu, lai atrast (kaut-kādu) pozīciju, kas pieder vienīgajam 1-blokam. Algoritms 1:

1. Kamēr neesam kļūdījušies, vaicājam $x_{i \cdot L}$, $i = 1, 2, ..., \lfloor \frac{N}{L} \rfloor$, minot ka $x_{i \cdot L} = 0$.

2. Tad kad esam nokļūdījušies, izdodam kā atbildi $i \cdot L$, kur $x_{i \cdot L} = 1$

Izmantojot algoritma 1 atrasto 1-bloka pozīciju (saucam to par p) atrodam bloka sākumu un beigas. Algoritms 2:

1. Izmantojot bināru meklēšanu atrodam vismazāko pozīciju l intervālā $[1, p]$, kurai $x_l = 1$. Katru reizi vaicājot pozīcijā ind, minēsim $x_{ind} = 1$.

2. Līdzīgā veidā ar bināru meklēšanu atrodam vislielāko pozīciju r intervālā $[p, N]$, kurai $x_r = 1$.

Tagad apvienosim šos algoritmus: pielietosim teorēmu 3.1 algoritmiem 1 un 2 dabūsim kvantu algoritmus 1' un 2'.

Nomērīsim algoritma 1' atbildi. Izmantosim šo atbildi algoritmam 2'.

Piebilde: tādēļ ka (vismaz šajā gadījumā) binārai meklēšanai nav kvantu priekšrocību, var veikt klasisku bināru meklēšanu, un pēc katra orākula O_x vaicājuma uz indeksa i, veikt mērījumu un iegūt x_i.

Analīze

Algoritms 1:

$T_1 \leq \lfloor \frac{N}{L} \rfloor$ (sliktākajā gadījumā pavaicāsim visas pozīcijas, kas ir L daudzkārtnis).

$G_1 = 1$ (ir garantēts, ka 1-bloks eksistē).

Algoritms 2:

$T_2, G_2 \leq C \cdot \log N$, kur C ir kaut kāda konstante.

Līdz ar to, ieguvām (kombinēto) bumbu meklēšanas algoritmu ar vaicājumu sarežģītību $O(\sqrt{T_1 \cdot G_1} + \sqrt{T_2 \cdot G_2}) = O(\sqrt{\frac{N}{L}} + \log N)$.

Ja $L \geq \frac{N}{\log^2 N}$, tad vaicājumu sarežģītība ir $O(\log N)$.

Ja $L < \frac{N}{\log^2 N}$, tad vaicājumu sarežģītība ir $O(\sqrt{\frac{N}{L}})$.

Cits risinājums

Līdzīgi, kā bumbas meklēšanas algortimā, atradīsim kaut kādu pozīciju, kas pieder 1-blokam un šī 1-bloka galus.

Izmantojot Grovera meklēšanas algoritmu, atradīsim tādu $i : x_{i \cdot L} = 1$, meklējamā telpa: $i \in \{1, 2, ..., \lfloor \frac{N}{L} \rfloor\}$, izmantojot $O(\sqrt{\frac{N}{L}})$ vaicājumus.

Pēc tam veiksim 2 bināras meklēšanas l un r atrašanai - katra ar vaicājumu sarežģītību $O(\log N)$. Kopējā sarežģītība - $O(\sqrt{\frac{N}{L}} + \log N)$ vaicājumi.

5 Vaicājumu algoritmi bitu matricām

Šajā nodaļā aprakstīsim vaicāšanas algoritmus funkcijām f, kuras interpretēs ieejas $N \cdot M$-bitu virkni $x_1x_2...x_{N \cdot M}$ kā $N \times M$-bitu matricu A:

$$A = \begin{bmatrix} x_1 & x_2 & \dots & x_M \\ x_{M+1} & x_{M+2} & \dots & x_{2 \cdot M} \\ \vdots & \vdots & \ddots & \vdots \\ x_{(N-1) \cdot M+1} & x_{(N-1) \cdot M+2} & \dots & x_{N \cdot M} \end{bmatrix}$$

5.1 Taisnstūra meklēšana

Formulējums

Ir dota $N \times M$ bitu matrica A. Ir zināms, ka visi vieninieki šajā matricā veido taisnstūri.

Formālāk - pastāv skaitļi $x_{low}, x_{high}, y_{low}, y_{high}$, kuriem izpildās:

- $1 \leq x_{low} \leq x_{high} \leq N \wedge 1 \leq y_{low} \leq y_{high} \leq M$
- $\forall (i,j) : A_{i,j} = 1 \iff x_{low} \leq i \leq x_{high} \wedge y_{low} \leq j \leq y_{high}$

Ir nepieciešams atrast šī taisnstūra koordinātes - $x_{low}, x_{high}, y_{low}, y_{high}$.

Šo uzdevumu var apskatīt kā īpašgadījumu uzdevumam 4.4.1 darbā [2] pie $D = 2$. Risinot šo uzdevumu mēs izmantosim to, ka taisnstūris ir "regulārs" saistīts apgabals, un uzlabosim vaicājumu skaitu no $O\sqrt{N \cdot M \cdot (h+w)}$ līdz $O\sqrt{N \cdot M}$ (kur h un w ir taisnstūra, kas sastav no vieniniekiem, garums un platums).

Risinājums

Mēs piedāvājam šādu algoritmu:

1. Ejam caur A bitiem pēc kārtas - ejam caur rindas numuram i pieaugšanas secībā, rindas iekšā ejam caur kolonnas numuram j pieaugšanas secībā:
   - Vaicājam $A_{i,j}$, minam ka tas ir $A_{i,j} = 0$
   - Ja $A_{i,j} = 0$, turpinām soli 1. uz nākamiem bitiem.
   - Ja $A_{i,j} = 1$, tad mēs esam atraduši "kreiso augšejo stūri", piešķiram $x_{low} \leftarrow i, y_{low} \leftarrow j$, aizejam uz soli 2.
2. Kamēr neesam kļūdījušies, vai izejam ārpus matricas robežām, vaicājam pēc kārtas
   $A_{x_{low}+1,y_{low}}, A_{x_{low}+2,y_{low}}, ..., A_{stop,y_{low}}, A_{stop+1,y_{low}}$ minot, ka tie ir vienādi ar 1 ($stop$ ir vislielākā pozīcija u, kurai $A_{u,y_{low}} = 1$).
   Tad, kad esam dabūjuši $stop + 1 > N \vee A_{stop+1,y_{low}} = 0$, piešķiram $x_{high} \leftarrow stop$.

3. Kamēr neesam kļūdījušies, vai izejam ārpus matricas robežām, vaicājam pēc kārtas
$A_{x_{low},y_{low}+1}, A_{x_{low},y_{low}+2}, ..., A_{x_{low},stop}, A_{x_{low},stop+1}$ minot, ka tie ir vienādi ar 1 (*stop* - līdzīgi kā iepriekšējā gadījumā).
Tad, kad esam dabūjuši $stop+1 > M \vee A_{x_{low},stop+1} = 0$, piešķiram $y_{high} \leftarrow stop$.

Algoritma rezultāts:
(x_{low}, y_{low}) - "kreisais augšejais stūris" un (x_{high}, y_{high}) - "labais apakšējais stūris".

Vispārinājumi

Autors var minēt iespējamus virzienus, kādos varētu vispārināt šo uzdevumu:

- Lielāks dimensiju skaits - piemēram, tiek meklēts kubs no vieniniekiem trīsdimensiju masīvā no nullēm un vieniniekiem.
- Vairāki taisnstūri, kas savā starpā nešķeļas un nepieskārās, attiecīgais bumbas meklēšanas algoritms būtu ar vaicājumu sarežģītību $O(\sqrt{N \cdot M \cdot K})$, kur K ir taisnstūru skaits (K var būt nezināms).
- Citi "regulāri" saistīti 1-apgabali, kurus var atrast ar bumbas meklēšanas algoritmu, kas veic kļūdu skaitu asimptotiski mazāku par šī apgabala virsmu.

Analīze

T: Izpildot soli 1, pēc kārtas vaicājam elementus no 1. rindas, 2. rindas, u.t.t. Veicot soļus 2 un 3 vaicājam tikai elementus, kas ir "zemāk vai pa labi" no pirmā atrasta vieninieka. Līdz ar to, katru matricas elementu pavaicājam ne vairāk par 1 reizi, tātad $T \leq N \cdot M$

G: Izpildot soli 1 kļūdāmies tieši 1 reizi (jo ir garantēts, ka vismaz viens matricas elements ir 1). Izpildot soļus 2 un 3 kļūdāmies ne vairāk par 1 reizi katrā solī.

Tātad, $G \leq 1 + 1 + 1 = 3$

Līdz ar to, pielietojot teorēmu 3.1, iegūstam bumbas meklēšanas algoritmu ar vaicājumu sarežģītību $O(\sqrt{T \cdot G}) = O(\sqrt{N \cdot M \cdot 3}) = O(\sqrt{N \cdot M})$

Cits risinājums

Atrisināsim šo uzdevumu, pielietojot Grovera meklēšanas algoritmu 2 reizes.

Ievērosim šādu faktu: noteikt, vai elements pozīcijā (i, j) ir "kreisais augšējais stūris" taisnstūrim no vieniniekiem var, aplūkojot $A_{i,j}$ un divus elementus augša un pa kreisi no $A_{i,j}$ - elementu $A_{i-1,j}$ un $A_{i,j-1}$. Definēsim:

$$f(i,j) = \begin{cases} 1, \text{ ja } A_{i,j} = 1 \wedge (i = 1 \vee A_{i-1,j} = 0) \wedge (j = 1 \vee A_{i,j-1} = 0) \\ 0 \text{ citos gadījumos} \end{cases}$$

Ievērosim, ka $f(i,j) = 1$ tikai ja pozīcijā (i,j) ir meklētais stūris.

Līdzīgi atradīsim "labo apakšējo stūri" (tikai papildus skatīsimies pa labi un lejā) ar funkciju g, kas pieņems vērtību 1 tikai "labajam apakšējam stūrim". Definēsim:

$$g(i,j) = \begin{cases} 1, \text{ ja } A_{i,j} = 1 \wedge (i = 1 \vee A_{i+1,j} = 0) \wedge (j = 1 \vee A_{i,j+1} = 0) \\ 0 \text{ citos gadījumos} \end{cases}$$

Gan f, gan g rēķināšanai katrā pozīcijā ir nepieciešamas ne vairāk par 3 orākula izsaukumiem.

Pielietojot teorēmu 3.3 funkcijām f un g, dabūjam 2 kvantu algoritmus, kas atrod vajadzīgus stūrus ar ierobežotu kļūdu. Abi algoritmi izmanto $O(3 \cdot \sqrt{\frac{N \cdot M}{1}})$ orākula vaicājumus. Kopā sanāk $O(3 \cdot \sqrt{N \cdot M} + 3 \cdot \sqrt{N \cdot M}) = O(\sqrt{N \cdot M})$ vaicājumi.

5.2 Sorted(N, D) uzdevums

Formulējums

Šajā uzdevumā ir dota $N \times D$-bitu matrica A. Vajag noteikt, vai matricas rindas ir sakārtotas leksikografiskajā secībā: $A_1 \leq A_2 \leq ... \leq A_N$.

Risinājums

Sākumā ar algoritmu 1 atrisināsim šo uzdevumu gadījumam $D = 2$ (salīdzināsim divas D-bitu virknes x un y). Algoritms 1:

1. x, y - divas D-bitu virknes, algoritma ieejas dati
2. Pēc kārtas $i = 1, 2, ..., D$ pavaicājam $x_i \oplus y_i$, minot ka $x_i \oplus y_i = 0$.
   Vieksmes gadījumā $x_i = y_i$, turpinām vaicāšanu nākamiem i.
   Kļūdas gadījumā vaicājam x_i, minot ka $x_i = 0$:
   - Veiksmes gadījumā $0 = x_i < x_j = 1$, atgriežam atbildi $true$.
   - Kļūdas gadījumā $1 = x_i > x_j = 0$, atgriežam atbildi $false$
3. Ja nevienu reizi nekļūdījāmies visiem i, atgriežam atbildi $true$.

Izmantojot algoritmu 1, pēc kārtas salīdzināsim A_1 un A_2, A_2 un A_3, ... , A_{N-1} un A_N. Algoritms 2:

1. Pēc kārtas $i = 1, 2, ...N - 1$ salīdzinām A_i un A_{i+1}, izmantojot algoritmu 1 kā apakšprocedūru. Atgriežam $false$, ja iegūtā atbilde ir $false$.
2. Atgriežam atbildi $true$.

Analīze

Algoritms 1:

T: $T \leq D + 1$, jo $x_i \oplus y_i$ tiek pavaicāts ne vairāk par 1 reizi katram i, un pēc x_i vaicāšanas algoritms 1 izdod atbildi.

G: Ja $x \leq y$, tad algoritms var kļūdīties tikai vienu reizi, pavaicājot $x_i \neq y_i$, bet nekļūdās, vaicājot $x_i = 0$, tātad $G \leq 1$.

Ja $x > y$, tad algoritms var arī kļūdīties, vaicājot x_i, un $G \leq 2$.

Algoritms 2:

T: $T \leq (D + 1) \cdot (N - 1) = O(D \cdot N)$, jo algoritms 1 tiek izsaukts $\leq N - 1$ reizi un katrs izsaukums izmanto $\leq D + 1$ vaicājumu.

G: $G \leq (N-1)+1 = N$, jo algoritms 1 kļūdas ne vairāk par 1 reizi $A_i \leq A_{i+1}$ gadījumā, un ne vairāk par 2 reizēm $A_i > A_{i+1}$ gadījumā (šīs variants notiek ne vairāk par 1 reizi algoritma 2 darba laikā).

Līdz ar to, pielietojot teorēmu 3.1 iegūstam, kvantu algoritmu ar sarežģītību $O(\sqrt{T \cdot G}) = O(\sqrt{D \cdot N \cdot N}) = O(N \cdot \sqrt{D})$

Cits risinājums

Šeit x, y joprojām ir D-bitu virknes.

Rekursīvi definējam Būla izteiksmes cmp_i (sākot ar lielākiem $i = D-1, D-2, ...$ un beidzot ar mazākiem $i = ..., 3, 2, 1$), kas salīdzina, vai x sufikss, kas sākas pozīcijā i, ir $\leq$ par attiecīgo y sufiksu (cmp_1 pārbauda, vai $x \leq y$).

$cmp_D(x, y) := (x_D \leq y_D)$

$1 \leq i \leq D-1 \implies cmp_i(x, y) := (x_i < y_i) \vee \big[(x_i = y_i) \wedge cmp_{i+1}(x, y)\big]$.

$cmp_1(x, y)$ izteiksmē aizvietojam šādas apakšizteiksmes, iegūstot Būla formulu $g(x, y)$:

- $a \leq b$ ar $\neg a \vee b$
- $a < b$ ar $\neg a \wedge b$
- $a = b$ ar $(a \wedge b) \vee (\neg a \wedge \neg b)$.

Izmantojot g, uzbūvēsim Būla formulu $F(A)$, kas pārbauda vai matricas A rindas ir sakārtotas: $F(A) = \bigwedge_{i=1}^{N-1} g(A_i, A_{i+1})$.

Tādēļ ka cmp_i izmērs ir $O(D + 1 - i)$ (pēc indukcijas), tad cmp_1 izmērs ir $O(D)$, un g izmērs ir $O(D)$ (katrs aizvietošanas likums palielina izmēru ≤ 2 reizēs). F izmērs ir $O((N-1) \cdot D) = O(N \cdot D)$, un pēc teorēmas 3.1 pastāv kvantu algoritms, kas izrēķina $F(A)$, izmantojot $O(\sqrt{N \cdot D})$ vaicājumus.

Risinājumu salīdzinājums

Ieguvam divus algoritmus ar sarežģītību $O(N \cdot \sqrt{D})$ (bumbas meklēšanas modelis), un $O(\sqrt{N \cdot D})$ (algoritms Būla formulas rēķināšanai). Ja $N \neq O(1)$, tad otrs algoritms ir vaicājumu ziņā efektīvāks par pirmo algoritmu.

Citi pielietojumi

Funkcija Sorted(N, M) bija pirmā funkcija ar sarežģītību, kas būtu labāka par klasisko gadījumu (šīm uzdevumam: $O(N \cdot D)$ vaicājumi).

Sākumā šīm uzdevumam autors nevarēja atrast ekvivalentus vai labākus risinājumus ar Grovera algoritmu (tika atrasti $O(\sqrt{N} \times D)$ un $O(N \log N \cdot \sqrt{D})$ risinājumi).

Ir iespējams, ka citos bumbas meklēšanas algoritmos, kad ir nepieciešams vairākas reizes pēc kārtas salīdzināt bitu virknes, šī algoritma idejas būs noderīgas.

Viens no šādiem uzdevumiem, ko var iedomāties, būtu (šajā darbā neaprakstīts) $Min(N, D)$ uzdevums, kurā vajag atrast pozīciju minimālai rindai $N \times D$-bitu matricā A. Pētot šo uzdevumu, autors atrada līdzīgu bumbas meklēšanas algoritmu ar sarežģītību $O(N \cdot \sqrt{D})$, un divus risinājumus, kas izmanto Grovera meklēšanu ar sarežģītībam $O(\sqrt{N} \times D)$ un $O(N \log N \cdot \sqrt{D})$.

6 Rezultāti un secinājumi

Šī darba izstrādes rezultātā tika iegūti šādi augšējie novērtējumu kvantu vaicājumu sarežģītībai $Q_2(f)$.

Uzdevuma nosaukums	$Q_2(f)$ ar BMM	$Q_2(f)$ ar citiem kvantu modeļiem	$Q_2(f)$ ar paplašinātu BMM
Perioda pārbaude	$O(K+\sqrt{N-K})$	$O(\sqrt{N-K})$	$O(\sqrt{N-K})$
$f = K$ vieninieki	$O(\sqrt{N \cdot K})$	$O(\sqrt{N \cdot K})$	
$f = 2$ vieninieki attālumā $< K$	$O(\frac{N}{\sqrt{K}})$	$O(\frac{N}{\sqrt{K}})$, $O(\sqrt{N \cdot K})$	$O(\sqrt{N \cdot K})$
$f = K$ vieninieki pēc kartas	$O(\frac{N}{\sqrt{K}})$	$O(\sqrt{N \cdot K})$	
$f =$ Viena 1-bloka meklēšana, kuram zināms minimālais garums	$O(\sqrt{\frac{N}{L}} + \log N)$	$O(\sqrt{\frac{N}{L}} + \log N)$	
$f =$ Taisnstūra meklēšana	$O(\sqrt{N \cdot M})$	$O(\sqrt{N \cdot M})$	
$f =$ Sorted(N, D) uzdevums		$O(\sqrt{N \cdot D})$	$O(N \cdot \sqrt{D})$

Att. 5: Iegūto risinājumu sarežģītība. BMM - bumbu meklēšanas modelis. Tukšais tabulas elements - uzdevums netika aplūkots šajā modelī

Tika uzlabots augšējais $Q_2(f)$ novērtējums [2] uzdevumam 4.2. Risinot uzdevumu 4.4, autors atrada algoritmu, kas pie $K > \sqrt{N}$ parametra vērtībam pārspēj labākus atrastos risinājumus citos kvantu modeļos.

Iespējams, ka uzdevumiem 4.3 un 4.4 ir iespējams atrast kvantu algortmus ar sarežģītību $O(N^{3/4})$, vai precīzāk novērtēt apakšējus novērtējumus (autors spēj iedomāties $O(\sqrt{N})$ apakšējos novērtējumus, kas reducē OR vai AND uzdevumus uz šī uzdevuma speciālgadījumiem).

Šajā darbā autors centās aplūkot vispārīgus uzdevumus (f funkcijas), kas palīdzēs talākajos šī modeļa petījumos.

Pēc autora domām, bumbu meklēšanas modelis ir spēcīgs rīks kvantu algoritmu veidošan, kas palīdz izmantot intuīciju par klaskiem vaicājumu algoritmiem, būvējot kvantu algoritmus.

7 Pateicības

Vēlos pateikties darba vadītājam Andrim Ambainim par interesantām iespējam tēmas izvēlē, izciliem semināriem par saistītām tēmām un pacietību, sagaidot pirmos rezultātus un to noformējumu. Vēlos pateikties viņam par piezīmi, kas palīdzēja uzlabot algoritmu nodaļā 4.1 no $O(\sqrt{N \cdot K})$ līdz $O(K + \sqrt{N - K})$.

Vēlos pateikties 2016. gada pavasara semestra kvantu skaitļošanas semināra dalībniekiem, kas sniedza idejas par f funkcijām pētīšanai darbā.

Liels paldies draugam Jevgēnijam Vihrovam par neatkarīgu viedokli un komentāriem.

Literatūras saraksts

[1] Cedric Yen-Yu Lin, Han-Hsuan Lin. Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester. Pieejams http://arxiv.org/abs/1410.0932

[2] Ēriks Gopaks. Algoritmu sarežģītības novērtējumi bumbas meklēšanas modelī. Bakalaura darbs. Pieejams https://dspace.lu.lv/dspace/handle/7/28626

[3] Avshalom C. Elitzur, Lev Vaidman. Quantum Mechanical Interaction-Free Measurements. Pieejams https://arxiv.org/abs/hep-th/9305002

[4] Andrew M. Childs, Shelby Kimmel, Robin Kothari. The quantum query complexity of read-many formulas. Pieejams https://arxiv.org/abs/1112.0548

[5] Lov K. Grover. A fast quantum mechanical algorithm for database search. Pieejams https://arxiv.org/abs/quant-ph/9605043

[6] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, Umesh Vazirani. Strengths and Weaknesses of Quantum Computing. Pieejams http://arxiv.org/abs/quant-ph/9701001

[7] Harry Buhrman, Ronald de Wolf. Complexity measures and decision tree complexity: a survey. Pieejams http://www.sciencedirect.com/science/article/pii/S030439750100144X

[8] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. Pieejams https://arxiv.org/abs/quant-ph/9508027v2

[9] Simon Bone, Matias Castro. A Brief History of Quantum Computing. Pieejams http://www.doc.ic.ac.uk/~nd/surprise_97/journal/vol4/spb3/

[10] Finding all marked elements using Grover's algorithm - diskusija vietnē stackexchange.com. Pieejams http://cs.stackexchange.com/questions/13513/finding-all-marked-elements-using-grovers-algorithm.

DOKUMENTĀRĀ LAPA

Bakalaura darbs „Kvantu algoritmi bumbu meklēšanas modelī” izstrādāts LU Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors:

Andrejs Kuzņecovs _____ ___.___.2016.

Rekomendēju/nerekomendēju darbu aizstāvēšanai

Vadītājs:

profesors Dr. sc. comp. *Andris Ambainis* _____ ___.___.2016.

Recenzents:

Darbs iesniegts Datorikas fakultātē ___.___.2016.
Dekāna pilnvarotā persona:

vecākā metodiķe *Ārija Sproģe* _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē
___.___.2016. prot. Nr. _____-.
Komisijas sekretārs(-e):

_____