LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

ALGORITMU SAREŽĢĪTĪBAS NOVĒRTĒJUMI BUMBAS MEKLĒŠANAS MODELĪ

BAKALAURA DARBS

Autors: **Ēriks Gopaks**
Studenta apliecības Nr.: eg11041
Darba vadītājs: profesors Dr. sc. comp. Andris Ambainis

RĪGA 2015

ANOTĀCIJA

Viena no aktuālākajām problēmām kvantu skaitļošanas nozarē ir kvantu datoru priekšrocību noteikšana salīdzinājumā ar klasiskajiem datoriem. Priekšrocības bieži tiek demonstrētas, pierādot kvantu algoritmu sarežģītības novērtējumu no augšas, kurš ir labāks par novērtējumu jebkuram klasiskajam algoritmam tādas pašas problēmas risināšanai.

Darba mērķis ir izpētīt vairākas skaitļošanas problēmas nesen izgudrota „bumbas meklēšanas" skaitļošanas modeļa ietvaros, lai iegūtu tiem atbilstošu algoritmu kvantu vaicājumu sarežģītības novērtējumus, kuri ir potenciāli labāki par zināmajiem.

Pētīto algoritmu starpā ir vairāki algoritmi uz simbolu virknēm, daži algoritmi uz grafiem, kā arī vairāku bitu binārās operācijas AND, OR, XOR.

Rezultātā visiem darbā aprakstītajiem algoritmiem ir pierādīti kvantu vaicājumu sarežģītības novērtējumi no augšas.

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

ABSTRACT

Estimating algorithm complexity in bomb-testing model

One of the most intriguing problems in the field of quantum computing is estimating advantages of quantum computers over classical ones. The advantages are frequently shown by proving complexity upper bounds for some quantum algorithms, which is better than the best known upper bound for any classical algorithm that solves the same problem.

The goal of this work is to research several computational problems within a recently developed “bomb-testing” computing model in order to prove their quantum query complexities, which may be better than known results.

Among the studied algorithms there some algorithms on strings, some graph algorithms as well as bitwise operations AND, OR and XOR for variable number of bits.

As a result, author has proved upper bounds of quantum query complexity for all the algorithms described in the current work.

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

SATURS

APZĪMĒJUMU SARAKSTS

Darbā tiek izmantoti sekojoši apzīmējumi un saīsinājumi:

- BFS – (angl. *breadth-first search*) meklēšana plašumā;
- DFS – (angl. *depth-first search*) meklēšana dziļumā;
- C_n^k – kombināciju skaits no n pa k, jeb $C_n^k = \binom{n}{k}$;
- $O(n)$ – funkcijas novērtējums no augšas;
- $\Theta(n)$ – funkcijas novērtējums no abām pusēm;
- $\tilde{O}(n)$ – funkcijas novērtējums no augšas, ignorējot logaritmiskus reizinātājus;
- $\bar{X}$ – gadījumlieluma X vidēja vērtība.

IEVADS

Mūsdienās kvantu skaitļošanai ir pievērsta ļoti liela uzmanība. Īpašā interese ir kvantu skaitļošanas zinātniekiem un pasaules pašiem lielākajiem uzņēmumiem, jo pirmo kvantu datoru izveidošanas datums strauji tuvojas. Konceptuāli jaunie datori sola būtiskas pārmaiņas jebkurā cilvēces darbības jomā, kas lielā mērā paļaujas uz informācijas tehnoloģijām.

Viens no populārākiem iemesliem ticēt tuvākās nākotnes pārmaiņās ir Šora kvantu algoritms skaitļu sadalīšanai pirmreizinātājos, kuram neeksistē analogu klasisko algoritmu pasaulē. Mūsdienu kriptogrāfija lielā mērā balstās uz pieņēmuma, ka skaitļa pirmskaitļu reizinātāju atrašana ir pietiekoši sarežģīts uzdevums. Savukārt, Šora algoritma efektivitātes dēļ šis pieņēmums vairs nav spēkā, līdz ar to ir svarīgi paredzēt potenciālas nepatikšanās un plānot to pārvaldības procesus jau tagad.

Darba autoram likās interesanti un svarīgi saprast kvantu skaitļošanas principus, un pašam iesaistīties kvantu skaitļošanas problēmu pētījumos. Izvēlētā tēma ir cieši saistīta ar vienu no pašiem jaunākiem darbiem kvantu skaitļošanas jomā, divu Masačūsetsas Tehnoloģiju Institūta pētnieku publikācijas [3], kurā tiek piedāvāts jauns un unikāls modelis kvantu algoritmu analīzei – „bumbas meklēšanas" modelis. Autoram likās intriģējoši un izaicinoši piedalīties tik jaunā un līdz galam neizpētītā zinātnes virzienā.

Pētījuma mērķis ir bijis iepazīties ar jaunākajiem pētījumiem par kvantu algoritmiem, un apskatīt dažas algoritmu klases, kuras var pārtulkot uz vaicājumu sarežģītības modeli. Vislielākā uzmanība tika pievērsta algoritmiem uz grafiem un uz simbolu virknēm, fokusējoties uz kopā ar darba vadītāju nodefinētajām skaitļošanas problēmām.

Autors ir pielietojis savas zināšanas, prasmes, intuīciju un pieredzi kombinatorikā, varbūtības teorijā, algoritmu teorijā un kvantu skaitļošanā, bez kā izvesto novērtējumu formulu pierādīšana nevarētu būt iespējama. Vairāku formulu pārbaudei autors ir papildus izmantojis vairākas informācijas tehnoloģijas, ieskaitot pašu uzrakstītās datorprogrammas izdomāto algoritmu simulācijai un formulu vērtību rēķināšanai lielām argumentu vērtībām. Bet, ņemot vērā to, ka katrs aprakstītais novērtējums tika matemātiski pierādīts, autors ir nolēmis izvairīties no nematemātisko pētniecības metožu iekļaušanas darbā.

Darba specifika paredz nopietnu uzsvaru uz matemātiskām darbībām. Visi faktoloģiskā materiāla avoti ir aprakstīti darba pašās beigās (skat. „*Izmantotā literatūra un avoti*"), un ir pārsvarā nepieciešami pirmajās divās nodaļās. Tālākās nodaļās tiek izmantota tikai viena teorēma no tiem avotiem, ar kuras palīdzību arī notiek visu algoritmu kvantu vaicājumus sarežģītības rēķināšana.

Pirmajā nodaļā tiek izskaidrots bumbas meklēšanas skaitļošanas modelis, ar kura palīdzību ir iegūti tālāko nodaļu novērtējumi. Tiek parādīta saikne starp modeļa matemātiskajām īpašībām un tā fizisko realizāciju interferometra veidā. Saprast materiālu lasītājam var palīdzēt iepriekšējās zināšanas par tādu fizikālo parādību kā staru interference, kā arī par vienkāršākiem kvantu pasaules likumiem.

Visa otrā nodaļa ir veltīta kvantu vaicājumu sarežģītības definēšanai. Starp algoritma laika un vaicājumu sarežģītību ir konceptuālās atšķirības, līdz ar to šī nodaļas saprašana ir nepieciešama, lai lasītājam rastos pareiza iegūto rezultātu interpretācija.

Trešā, ceturtā un piektā nodaļas ir veltītas attiecīgi bināro operatoru, algoritmu uz simbolu virknēm un algoritmu uz grafiem sarežģītības novērtēšanai. Katrā no tām tiek definēta risināmā problēma, piedāvāts risināšanas algoritms un pierādīts labākais no autora iegūtajiem kvantu vaicājumu sarežģītības novērtējumiem. Visi pierādījumi ir izklāstīti secīgi un pietiekami detalizēti, lai lasītājam būtu ērtāk sekot tiem līdzi.

Svarīgākie darba rezultāti, kā arī autora secinājumi par darba rezultātiem un pētījuma turpmākās attīstīšanas iespējām ir aprakstīti darba nobeigumā.

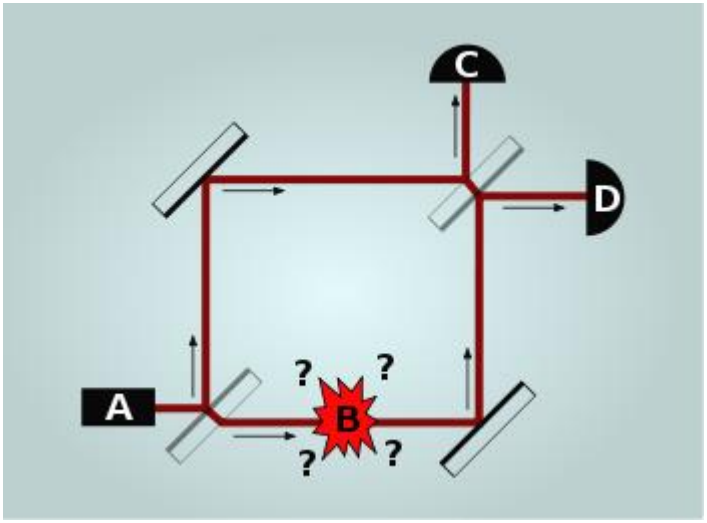
1. BUMBAS MEKLĒŠANAS MODELIS

Šajā nodaļā tiek aprakstīts tā saucamais „bumbas meklēšanas" modelis, kura ietvaros notiks šī darba turpmākā algoritmu analīze.

Elitzura-Vaidmana bumbas meklēšanas problēma [1] ir plaši pazīstams domāšanas eksperiments, kurš demonstrē, cik lielā mērā kvantu mehānika atšķiras no klasiskās mehānikas. Šī problēma labi parāda iespēju veikt mērījumus, kuros nenotiek mijiedarbības ar objektu, kura īpašības tiek mērītas. Tam klāt sistēmas stāvoklis mērīšanas rezultātā netiek sabojāts.

1.1. Elitzura-Vaidmana bumbas testētājs

Elitzurs un Vaidmans savā rakstā [1] ir aprakstījuši jaunu iespēju noteikt objekta eksistenci, neveicot ar doto objektu mijiedarbības. Piedāvātā paņēmiena pamatā ir kvantu mehānikas parādība, kuru sauc par Aharonova-Boma [2] efektu.



1.1. att. **Interferometra konfigurācija**

Aprakstītā metode balstās uz daļiņu interferometra, kurš ir līdzīgs Maha-Zendera klasiskās optikas interferometram. Tiek apgalvots, ka tas spēj strādāt ar jebkura veida daļiņām. Daļiņa sasniedz pirmo staru sadalītāju, kura caurlaides koeficients ir $\frac{1}{2}$. Daļiņas vilnis sadalās atstarotajā un neatstarotajā daļās, kuras pēc atstarošanās no spoguļiem atkal apvienojas noslēdzošajā staru sadalītājā (*1.1. att.*). Divi detektori salasa daļiņas pēc tam, kad tie ir izgājuši caur otro staru sadalītāju. Spoguļi un staru sadalītāji tiek speciāli izvietoti tā, lai destruktīvas interferences dēļ, neviena daļiņa netiktu novērota ar kādu vienu no detektoriem, piemēram, D, bet visi tiktu pamanīti ar otro – C.

Ja, nemainot spoguļu un staru sadalītāju pozīcijas, tiek nobloķēts viens no diviem daļiņu ceļiem interferometrā, tad tās daļiņas, kuras veiksmīgi iziet cauri interferometram, tiek novēroti ar katru no detektoriem C un D ar vienādu varbūtību. Tātad, detektors D spēj pamanīt daļiņas tikai tajā gadījumā, ja viens no interferometra ceļiem tiek bloķēts.

Procedūra, ar kuras palīdzību var noskaidrot objekta atrašanos dotajā vietā, neizlaižot caur to nevienu fotonu, ir sekojoša:

Mēs izmantojam iepriekš aprakstīto interferometra konfigurāciju, par daļiņām izvēlējoties fotonus, tātad detektors D nenovēro fotonus, kad abi interferometra ceļi ir atvērti, un nodrošinām tādu interferometra izvietojumu, lai viens no šiem ceļiem ietu caur to telpu, kurā mēs vēlamies pārbaudīt objekta eksistenci. Šādā konfigurācijā nosūtot vienu fotonu, ir iespējami tikai sekojoši trīs iznākumi:

1. neviens no detektoriem nenovēro fotonu,
2. fotonu novēro detektors C,
3. fotonu novēro detektors D.

Pirmajā gadījumā fotons tika absorbēts (vai izkliedēts) ar objektu un nav sasniedzis nevienu no detektoriem. Šī iznākuma varbūtība ir $\frac{1}{2}$.

Otrais gadījums ir iespējams gan tad, ja objekts atrodas apskatāmajā telpā, gan arī tad, ja tas tur neatrodas. Mijiedarbība ar objektu nenotiek abās situācijās, tātad mēs varam mēģināt eksperimentu no jauna. Šī iznākuma varbūtība ir $\frac{1}{4}$.

Beidzot, trešajā gadījumā, ja detektors D novēro fotonu, mēs esam konstatējuši objekta eksistenci apskatāmajā telpā, šim objektam „nepieskaroties". Kā jau tika secināts augstāk, detektors D spēj novērot fotonu tikai pie nosacījuma, ja viens no interferometra ceļiem ir bloķēts, tātad objektam tur ir jābūt. Šis mērījums nemijiedarbojas ar objektu, jo, tikmēr fotons mijiedarbotos ar objektu, tas nesasniegtu detektoru D (pirmais gadījums). Šī iznākuma varbūtība ir $\frac{1}{4}$.

Savā rakstā Elitzurs un Vaidmans šī interferometra konfigurāciju pielieto aktīvu bumbu meklēšanai starp visām bumbām, daļa no kurām var būt neaktīvas. Fotonam mijiedarbojoties ar aktīvu bumbu, tā uzsprāgst [1]. Šādā veidā autori dramatizēja savus argumentus, apgalvojot, ka objekta eksistences noteikšana tiešām notiek bez mijiedarbības ar objektu.

1.2. Bumbas testētājs kvantu stāvokļu valodā

Visus tos pašus daļiņas (fotona) pārveidojumus augstāk aprakstītajā interferometrā var ērti un vienkārši aprakstīt kvantu stāvokļu valodā [1].

Apzīmēsim ar $|1\rangle$ tādu fotona stāvokli, kurā tas kustās virzienā pa labi, un ar $|2\rangle$ – fotona stāvokli, kurā tas kustās virzienā uz augšu (skat. attēls 1).

Aprakstītajā interferometra konfigurācijā, katru reizi fotonam atstarojoties no spoguļa virsmas, tā viļņa funkcijas fāze mainās par $\frac{\pi}{2}$. Tāpēc mēs varam aprakstīt daļēji apsudraboto spoguļu darbību kā

$$|1\rangle \to \frac{1}{\sqrt{2}}[|1\rangle + i|2\rangle]$$

$$|2\rangle \to \frac{1}{\sqrt{2}}[|2\rangle + i|1\rangle]$$

Pilnībā apsudraboto spoguļu darbība ir vienkārši

$$|1\rangle \to i|2\rangle$$

$$|2\rangle \to i|1\rangle$$

Ja objekts neatrodas pētāmajā telpā, tad sākotnējais fotona stāvoklis, izejot caur interferometru, pārvēršas par

$$|1\rangle \to \frac{1}{\sqrt{2}}[|1\rangle + i|2\rangle] \to \frac{1}{\sqrt{2}}[i|2\rangle - |1\rangle] \to \frac{1}{\sqrt{2}}\left[i \cdot \frac{1}{\sqrt{2}}[|2\rangle + i|1\rangle] - \frac{1}{\sqrt{2}}[|1\rangle + i|2\rangle]\right]$$

$$= \frac{i}{2}|2\rangle - \frac{1}{2}|1\rangle - \frac{1}{2}|1\rangle - \frac{i}{2}|2\rangle = -|1\rangle$$

Rezultātā fotons iziet no pēdējā staru sadalītāja virzienā pa labi, tātad tiek novērots ar detektoru D.

Pretējā gadījumā, ja objekts atrodas pētāmajā telpā, tad transformācijas ir sekojošas:

$$|1\rangle \to \frac{1}{\sqrt{2}}[|1\rangle + i|2\rangle] \to \frac{1}{\sqrt{2}}[i|2\rangle + i|izkliedēts\rangle] \to \frac{1}{2}[i|2\rangle - |1\rangle] + \frac{i}{\sqrt{2}}|izkliedēts\rangle$$

kur $|izkliedēts\rangle$ apzīmē fotona stāvokli pēc mijiedarbības ar objektu. Saskaņā ar standarta kvantu mērījumu pieeju [6], detektori ietekmē fotona kvantu stāvoklis tiek sabojāts:

$$\frac{1}{2}[i|2\rangle - |1\rangle] + \frac{i}{\sqrt{2}}|izkliedēts\rangle \to \begin{cases} |2\rangle, \ \ detektēts\ ar\ D, & varbūtība\ \frac{1}{4} \\ |1\rangle, \ \ detektēts\ ar\ C, & varbūtība\ \frac{1}{4} \\ |izkliedēts\rangle, \ \ netiek\ detektēts, & varbūtība\ \frac{1}{2} \end{cases}$$

Mēs redzam, ka objekta eksistenci var noteikt ar detektoru D tikai tad, ja objekts tur tiešām atrodas. Rezultātā, detektoram D novērojot fotonu, mēs iegūstam mums vajadzīgo informāciju – objekts atrodas kādā no interferometra ceļiem. Nepieciešamības gadījumā šo procedūru ir iespējams pielāgot tam, lai precīzāk noteiktu objekta pozīciju: lokāli jāpārbauda, vai objekts neatrodas visos citos interferometra iekšējos reģionos [1].

1.3. Veiksmīgā mērījuma varbūtība

Pilnu varbūtību noteikt objekta eksistenci ar Elitzura un Vaidmana aprakstīto metodi, neveicot ar objektu nekādas mijiedarbības, var ļoti vienkārši izrēķināt:

1. 1.-ajā solī fotons var nonākt uz detektoru D ar varbūtību $\frac{1}{4}$,
2. 2.-ajā solī fotons var nonākt uz detektoru D, ja iepriekšējā solī bija nonācis uz detektoru C. Šī notikuma varbūtība ir $\frac{1}{4} \cdot \frac{1}{4} = \frac{1}{4^2}$
3. 3.-ajā solī fotons var nonākt uz detektoru D, ja katrā no iepriekšējiem soļiem tas bija nonācis uz detektoru C. Šī notikuma varbūtība ir $\frac{1}{4^2} \cdot \frac{1}{4} = \frac{1}{4^3}$
4. utt. ...

Tātad pilna varbūtība ir visu šo notikumu summa:

$$\frac{1}{4} + \frac{1}{4} \cdot \frac{1}{4} + \frac{1}{4^2} \cdot \frac{1}{4} + \cdots = \sum_{i=1}^{\infty} \frac{1}{4^i} = \frac{\frac{1}{4}}{1 - \frac{1}{4}} = \frac{1}{3}$$

Turpmāk savā rakstā Elitzurs un Vaidmans parāda, ka objekta eksistences pārbaudes varbūtību var padarīt bezgalīgi tuvu $\frac{1}{2}$, interferometrā izmantojot staru sadalītājus ar citādāku sudraba pārklājumu [1].

Vēlāk [7] tika atrasta labākā pieeja, kurā veiksmīgā mērījuma varbūtību $P = \cos^2 \frac{\pi}{2N}$ var padarīt patvaļīgi tuvu vieniniekam, izvēloties pietiekami lielu soļu skaitu N, jeb

$$\lim_{N \to +\infty} P = \lim_{N \to +\infty} \cos^2 \frac{\pi}{2N} = \cos^2 0 = 1$$

Šī labākās pieejas pamatā ir kvantu Zeno efekts [8].

2. KVANTU VAICĀJUMU SAREŽĢĪTĪBA

Kvantu vaicājumu sarežģītība ir ļoti svarīgs modelis, kurš palīdz novērtēt kvantu datoru jaudu. Šajā modelī mums ir dota melna kaste ar bināru simbolu virkni $x = x_1x_2 \dots x_N$, un mēs vēlamies izrēķināt kādas funkcijas $f(x)$ vērtību, piekļustot šai melnajai kastei pēc iespējas mazāku skaitu reizes [3].

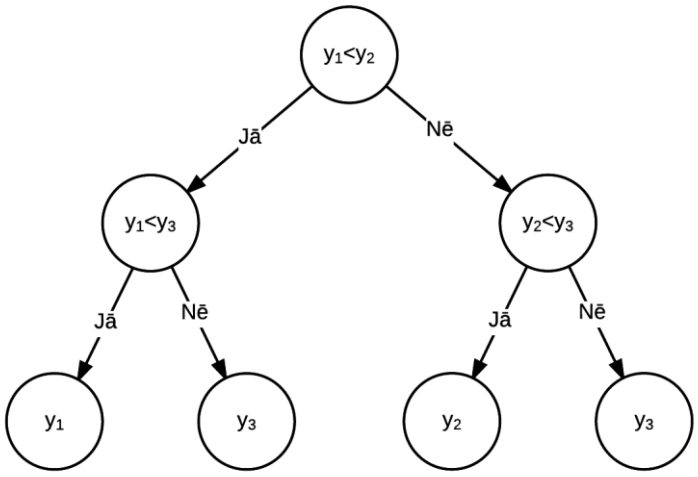
Pirms konkrēti definēt vaicājumu sarežģītību kvantu pasaulē, apskatīsim klasisko vaicājumu sarežģītības jēdzienu.

2.1. Vaicājumu sarežģītība

Ja ir dots uzdevums atrast vienu elementu no virknes $y_1, y_2, \dots, y_M$, tad varam pieņemt, ka divu šīs virknes elementu savstarpēja salīdzināšana ir dārgākā no visām algoritmam pieejamām operācijām. Vērtēsim algoritma rēķināšanas sarežģītību ar kopējo salīdzināšanas skaitu elementiem $y_1, y_2, \dots, y_M$.

Determinētiem algoritmiem, kuri precīzi risina šāda veida uzdevumu, var izveidot atbilstošu lēmumu koku. Viena elementa meklēšanas gadījumā, atbilstošs lēmumu koks sanāk binārs. Tad vaicājumu sarežģītība tiek definēta kā garākais ceļš no koka saknes līdz kādai no lapām. Uzdevuma vaicājumu sarežģītība tad ir vienāda ar garāko ceļu pašam zemākajām iespējamām lēmumu kokam, kurš risina doto uzdevumu.

Attēlā (*2.1. att.*) ir parādīts lēmumu koka piemērs vienkāršam mazākā elementa meklēšanas algoritmam pie $M = 3$.



2.1. att. **Lēmumu koka piemērs**

Katra koka virsotne, izņemot lapas, atbilst vienai salīdzināšanas operācijai. Veicot pārvietošanos kokā atkarībā no salīdzināšanas rezultātiem, algoritms nonāk vienā no lapām, rezultātā atgriežot lapā norādīto elementu.

2.2. Kvantu vaicājumu sarežģītības definīcija

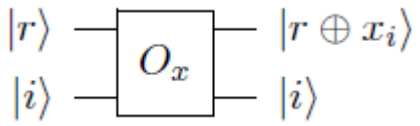
Apskatīsim funkcijas $f: D \to E$, kur $D = \{0,1\}^N$ ir visi iespējamie binārie vārdi garumā N, un E ir patvaļīga netukša kopa. Kvantu orākuls O_x ir unitārs operators, kurš darbojas uz diviem reģistriem:

1. ieraksta reģistrs r (1-dimensijas kvantu reģistrs);
2. pozīcijas reģistrs i (N-dimensiju kvantu reģistrs).

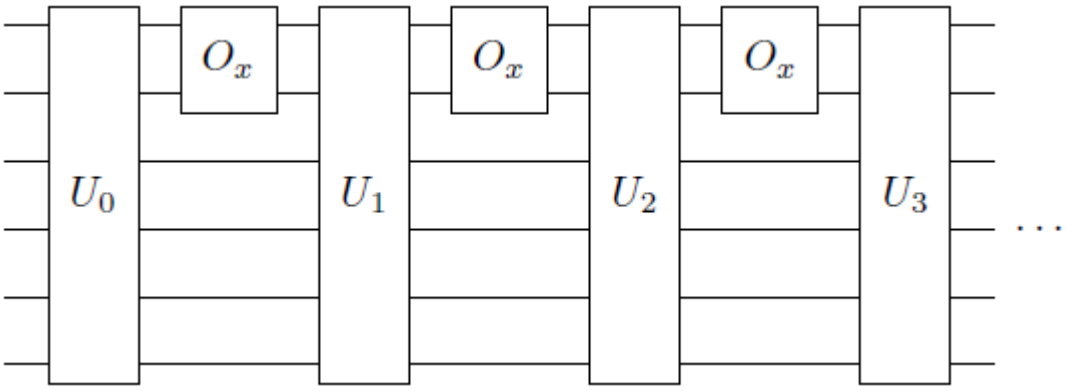
Šī kvantu orākula darbināšanas rezultātā iegūst

$$O_x|r, i\rangle = |r \oplus x_i, i\rangle$$

kur $\oplus$ ir binārais XOR operators. Kvantu orākulam atbilstoša kvantu shēma ir



Tad visiem algoritmiem, kas rēķina funkciju f, ir sekojoša vispārīga struktūra (*2.2. att.*):



2.2. att. **Vispārīga vaicājumu algoritmu kvantu shēma**

Kvantu vaicājumu sarežģītība $Q_\delta(f)$ ir mazākais orākula O_x izmantošanas reižu skaits, kāds nepieciešams funkcijas $f(x)$ vērtības rēķināšanai ar maksimāli pieļaujamo klūdu δ. Parasti tiek pieņemts, ka $\delta = 0.01$, kaut arī δ var būt jebkura konstante $\delta \leq \frac{1}{10}$. Bieži $Q_{0.01}(f)$ vietā tiek vienkārši rakstīts $Q(f)$, kas ļauj vienkāršot apzīmējumus [3].

Diezgan bieži ir vienkāršāk noteikt algoritma kvantu vaicājumu sarežģītību nevis tā laika sarežģītību, un secinājumi no vaicājumu sarežģītības bieži noder kvantu datoru iespēju un ierobežojumu izpratnē. Viens pazīstamais piemērs ir Grovera algoritms nestrukturētai meklēšanai [4] – pārtulkojot šo problēmu uz kvantu vaicājumu modeli, tika pierādīts, ka šim

algoritmam nepieciešami $O(\sqrt{N})$ vaicājumi, šādā veidā pierādot, ka Grovera algoritms ir optimāls [5].

2.3. Kvantu vaicājumu sarežģītības novērtēšana

Elitzura-Vaidmana bumbas testētājs rod jaunas iespējas kvantu algoritmu analīzei. Kā jau tika pieminēts, šī darba ietvaros veiktie algoritmu sarežģītības novērtējumi pārsvarā balstās uz divu Masačūsetsas Tehnoloģiju institūta pētnieku rezultātiem [3]. Lai pie šiem rezultātiem nonāktu, pētnieki ir nodefinējuši specifisku kvantu vaicājumu sarežģītības modeli, kuru viņi nosauca par „bumbas vaicājumu sarežģītību" (angl. *bomb query complexity*). Šī modeļa ietvaros definētie algoritmi tiek saukti par „bumbas meklēšanas algoritmiem" (angl. *bomb query algorithms*).

Šī darba ietvaros tas modelis netiks definēts, bet tiks aktīvi izmantota viena no rakstā pierādītajām teorēmām. Šī teorēma ļauj iegūt kvantu vaicājumu algoritmu sarežģītības novērtējumu no augšas, izmantojot tiem ekvivalentu klasisku vaicājumu algoritmu.

Teorēma. (formālāk skat. [3], „Theorem 8")

Pieņemsim, ka eksistē klasisks algoritms, kas izrēķina $f(x)$, veicot ne vairāk par T vaicājumiem, un algoritms min katra vaicājuma rezultātu (0 vai 1), kļūdoties vidēji ne vairāk ka G reizes pa visiem x. Tādā gadījumā mēs varam uzkonstruēt bumbas meklēšanas algoritmu, kurš veic $O(TG)$ vaicājumus. Izmantojot šo algoritmu, var uzkonstruēt kvantu algoritmu ar vaicājumu sarežģītību $Q(f) = O(\sqrt{TG})$.

Raksta autori turpat arī parāda, ka atbilstošs kvantu vaicājumu algoritms ar iegūto sarežģītību vienmēr eksistē, un apraksta procedūru, kā šādu algoritmu var uzkonstruēt (skat. [3], nodaļa 5.3).

Visi turpmākie šī darba kvantu algoritmu vaicājumu sarežģītības novērtējumu tiks iegūti izmantojot šo teorēmu.

3. VIENKĀRŠĀKIE ALGORITMI

Šajā nodaļā tiek analizēti paši vienkāršākie algoritmi – elementārās bitu operācijas. Šo algoritmu izpēte ir bijusi autoram svarīga pētījuma sastāvdaļa, un, iespējams, labi kalpos lasītājam kā piemērs apskatāmā skaitļošanas modeļa pielietošanai.

3.1. Binārā AND operācija

Apskatīsim tādu funkciju f, kura ir definēta kā $f(x) = x_1$ AND x_2 AND ... AND x_N. Zinām, ka $f(x) = 1$ tad un tikai tad, ja $x_1 = x_2 = \cdots = x_N = 1$, un pretējā gadījumā $f(x) = 0$.

Novērtēsim kvantu vaicājumu sarežģītību, rīkojoties saskaņā ar sekojošu stratēģiju:

1. Pieprasām katru bitu pēc kārtas, minot, ka tas ir vienāds ar 1.
2. Ja kārtējā vaicājumā būsim kļūdījušies, tad tas nozīmē, ka esam atraduši tādu x_i: $x_i = 0$, un varam secināt, ka dotajam x ir spēkā $f(x) = 0$. Šajā gadījumā pieļaujam tikai vienu kļūdu un varam izvairīties no turpmākiem vaicājumiem.
3. Pretējā gadījumā mēs nevienā vaicājumā nebūsim kļūdījušies, līdz ar to $\forall_{1 \le i \le N}(x_i = 1)$, tātad $f(x) = 1$.

Secinām, ka pirmajā gadījumā veikto vaicājumu skaits ir $T_1 \le N$ un vidējais kļūdu skaits ir $G_1 = 1$. Otrajā gadījumā: $T_2 = n$ un $G_2 = 0$. Šie gadījumi ir neatkarīgi, tātad kopējo binārās AND operācijas kvantu vaicājumu sarežģītību varam novērtēt kā $Q(f) = O\left(\max\left(\sqrt{T_1 \cdot G_1}, \sqrt{T_2 \cdot G_2}\right)\right) = O\left(\sqrt{N \cdot 1}\right) = O\left(\sqrt{N}\right)$.

3.2. Binārā OR operācija

Apskatīsim tādu funkciju f, kura ir definēta kā $f(x) = x_1$ OR x_2 OR ... OR x_N. Zinām, ka $f(x) = 0$ tad un tikai tad, ja $x_1 = x_2 = \cdots = x_N = 0$, un pretējā gadījumā $f(x) = 1$.

Spriežot līdzīgi, ka binārās AND operācijas gadījumā, izvēlamies sekojošu vaicāšanas stratēģiju:

1. Pieprasām katru bitu pēc kārtas, minot, ka tas ir vienāds ar 0. Arī šeit iegūstam divus iespējamos gadījumus:
2. Ja kārtējā vaicājumā neesam pareizi uzminējuši (t.i. $x_i = 1$), tad vairs vaicājumus neveicam, un šajā gadījumā $T_1 \le N$ un $G_1 = 1$.
3. Pretējā gadījumā visas N minēšanas bija pareizas, tātad $\forall_{1 \le i \le N}(x_i = 0)$.

Secinām, ka arī binārās OR operācijas kvantu vaicājumu sarežģītība ir $Q(f) = O\left(\max\left(\sqrt{T_1 \cdot G_1}, \sqrt{T_2 \cdot G_2}\right)\right) = O\left(\sqrt{N \cdot 1}\right) = O\left(\sqrt{N}\right)$.

3.3. Binārā XOR operācija

Apskatīsim tādu funkciju f, kura ir definēta kā $f(x) = x_1 \text{ XOR } x_2 \text{ XOR } \ldots \text{ XOR } x_N$.

Šoreiz mums nav iespējas izvairīties no vaicāšanas pēc kāda bita vērtības, jo katrs atsevišķais var ietekmē rezultātu (ja tas ir vieninieks), līdz ar to ir jāmin visu N bitu vērtības: $T = N$. Tagad atliek tikai novērtēt mazāko iespējamo vidējo kļūdu skaitu, ko mēs pieļausim, optimāli minot bitu vērtības.

Skaidrs, ka katram $x \in \{0,1\}^N$ atsevišķu bitu vērtības ir savā starpā neatkarīgas, līdz ar to, neatkarīgi no izvēlētās minēšanas stratēģijas, varbūtība kļūdīties i-tajā vaicājuma ir $\frac{1}{2}$. Tātad veicot visus N minējumus, vidēji kļūdīsimies $G = N \cdot \frac{1}{2} = \frac{N}{2}$ reizes.

Rezultātā iegūstam, ka binārās XOR operācijas kvantu vaicājumu sarežģītība ir $Q(f) = O\left(\sqrt{TG}\right) = O\left(\sqrt{N \cdot \frac{N}{2}}\right) = O\left(\frac{N}{\sqrt{2}}\right) = O(N)$.

4. ALGORITMI UZ SIMBOLU VIRKNĒM

Šajā nodaļā tiek atsevišķi apskatīti daži algoritmi, kuri uztver funkcijas f argumentu x kā bināru ciparu virkni. Kaut arī šie algoritmi veido tikai nelielu apakškopu no visu simbolu virkņu algoritmu kopas, autors ir centies izvēlēties tādus algoritmus, kuri var būtu vispārināmi un tam klāt arī pietiekami lietderīgi.

Definīcija 1. *Par **1-bloku** dotajam vārdam x sauksim tādu secīgu vārda x apakšvirkni s, kura sākas vārda pozīcijā l un beidzas pozīcijā r, un kurai izpildās sekojoši nosacījumi:*

1. *s pierakstā ir tikai vienīnieki, jeb $\forall_{l\le i\le r}(x_i = 1)$,*
2. *vai nu $l = 1$ (t.i. s ir vārda x prefikss), vai nu $x_{l-1} = 0$,*
3. *vai nu $r = N$ (t.i. s ir vārda x sufikss), vai nu $x_{r+1} = 0$.*

Piemēram, vārdā „00001111100" ir viens vienīgais 1-bloks garumā 5. Savukārt, vārdā „100110111000101" ir pieci 1-bloki.

4.1. Viena 1-bloka meklēšana

Apskatīsim sekojošu 1-bloka meklēšanas problēmu. Ir dota virkne $x_1, x_2, \dots, x_N$, kur $\forall_{1\le i\le N} x_i \in \{0,1\}$. Uzdevums ir atrast l un r – šī virknes *vienīgā* 1-bloka sākuma un beigu pozīcijas.

Autors piedāvā sekojošu vaicājumu stratēģiju:

1. Vaicājam katru bitu x_i pēc kārtas, sākot ar $i = 1$. Kamēr nav bijusi neviena kļūda, minam, ka $x_i = 0$. Tad, pirmoreiz kļūdoties, atrodam 1-bloka sākumpozīciju $l: x_l = 1$.
2. Kamēr ir bijusi tieši viena kļūda, minam, ka $x_i = 1$. Tad, kļūdoties otrreiz, atrodam 1-bloka beigu pozīciju $r = i - 1$.
3. Pēc otrās kļūdas varam vairs neveikt vaicājumus, jo kortežs (l, r) jau ir atrasts.

Ja esam pavaicājuši visus bitus, kļūdoties tikai vienreiz, tad tas nozīmē, ka 1-bloka beigas sakrīt ar vārda x beigām (t.i. 1-bloks ir vārda x prefikss). Tādā gadījumā $r = N$.

Vispārīgā gadījumā sagaidāmais kļūdu skaits ir $G = 2$, izņemot speciālgadījumus, kad 1-bloka beigas sakrīt ar virknes beigām (šajā gadījumā kļūdu skaits ir 1), tāpēc precīzs

vidējais kļūdu skaits $G_{precīzs} < G$, bet šī neprecizitāte nav būtiska asimptotiskajos sarežģītības novērtējumos. Tāpēc mūsu nolūkiem pilnīgi pietiek ar $G = 2$.

Sliktākajā gadījumā (ja vārdā x ir tikai viens vienīnieks priekšpēdējā pozīcijā) ir nepieciešams vaicāt katru bitu, tātad $T = N$.

Iegūstam kvantu vaicājumu sarežģītības novērtējumu $Q = O(\sqrt{TG}) = O(\sqrt{2N}) = O(\sqrt{N})$.

4.2. Viena 1-bloka meklēšana, kuram zināms minimālais garums

Tagad apskatīsim līdzīgu 1-bloka meklēšanas problēmu, tikai šoreiz pieņemsim, ka 1-blokam ir iepriekš zināms minimālais garums. Tāpat, kā iepriekšējā uzdevumā, ir dota virkne $x_1, x_2, \dots, x_N$, kur $\forall_{1 \le i \le N} x_i \in \{0,1\}$. Zināms, ka tajā ir tieši viens 1-bloks, pie tam tā garums ir vismaz L, un $L \geq 2$. Uzdevums ir atrast l un r – meklējamā 1-bloka sākuma un beigu pozīcijas.

Autors piedāvā veikt meklēšanu saskaņā ar sekojošu algoritmu:

1. Kamēr neesam kļūdījušies, vaicājam $x_{i \cdot L}$, $i = 1,2,3, \dots, \left\lfloor \frac{N}{L} \right\rfloor$, minot, ka $x_{i \cdot L} = 0$.
2. Tiklīdz esam kļūdījušies, tad esam atraduši tādu $x_p = 1$, kur $p = kL$ un $k \in \{1,2,3, \dots\}$, kurš pieder meklējamam 1-blokam. Atliek tikai atrast šī 1-bloka sākuma un beigu pozīcijas l un r. Pateicoties faktam, ka vārdā x ir tieši viens 1-bloks, varam atrast šīs pozīcijas ar binārās meklēšanas palīdzību.
3. Meklējam pozīciju l, kura var būt intervālā $p - L + 1 \le l \le p - 1$, jeb $(k - 1)L + 1 \le l \le kL - 1$. Šajā intervālā ir $(p - 1) - (p - L + 1) + 1 = L - 1$ dažādas pozīcijas. Ar bināro meklēšanu atradīsim tādu pozīciju l, kur $x_l = 1$ un vai nu $x_{l-1} = 0$, vai arī $l = 1$. Tas prasīs $\Theta(\log(L - 1)) = \Theta(\log L)$ vaicājumus, katrā no kuriem minēsim, ka $x_t = 1$. Tātad kļūdīsimies tad, ja vaicāsim par pozīcijām, kuras nepieder 1-blokam. Kļūdu skaits tad sanāk $\Theta(\log L)$. Sliktākajā gadījumā visos x_t vaicājumos kļūdīsimies, tātad $\forall_{p-L+1 \le j \le p-1}(x_j = 0)$, un tad $l = p$.

   Meklējam pozīciju r līdzīgi, kā meklējām l, tikai otrā pusē no x_p. Pozīcija r var būt intervālā $p + 1 \le r \le N$. Šī intervala garums ir $N - (p + 1) + 1 = N - kL$. Ar bināro meklēšanu atradīsim pozīciju r, kur $x_r = 1$ un vai nu $x_{r+1} = 0$, vai arī $r = N$. Tas prasīs $\Theta(\log(N - kL)) = \Theta(\log N)$ vaicājumus, kuros minēsim, ka

$x_t = 1$. Kļūdu skaits sanāk $\Theta(\log N)$, un sliktākajā gadījumā $\forall_{p+1 \le j \le N}(x_j = 0)$, tātad $r = p$.

Sliktākajā gadījumā ir jāveic $T = \mathrm{O}\left(\left\lfloor \frac{N}{L} \right\rfloor + \log L + \log N\right) = \mathrm{O}\left(\frac{N}{L} + \log N\right)$ vaicājumi. Tas var gadīties, piemēram, ja meklējamais 1-bloks atrodas virknes pašās beigās.

Vidējo kļūdu skaitu varam novērtēt kā $G = \Theta(1 + \log L + \log N) = \Theta(\log N)$.

Varam secināt, ka algoritma kvantu vaicājumu sarežģītība ir $Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{\left(\frac{N}{L} + \log N\right)\log N}\right) = O\left(\sqrt{\frac{N}{L}\log N + \log^2 N}\right)$.

Interesanti, ka pie $L \le \frac{N}{\log N}$ novērtējums sanāk

$$Q = O\left(\sqrt{\frac{N \log N}{L}}\right)$$

savukārt, pie $L \ge \frac{N}{\log N}$ iegūstam

$$Q = O\left(\sqrt{\log^2 N}\right) = O(\log N)$$

4.3. Vairāku 1-bloku meklēšana

Ir dota virkne $x_1, x_2, \dots, x_N$, kur $\forall_{1 \le i \le N} x_i \in \{0,1\}$. Šoreiz uzdevums ir atrast visu virknē x esošo 1-bloku sākuma un beigu pozīcijas, t.i. i-tajam 1-blokam jāatrod l_i un r_i, un jāatgriež kortežu (l_i, r_i) kopa. Ja virknē nav neviena 1-bloka, tad jāatgriež tukša kopa.

Līdzīgi, kā iepriekšējā uzdevumā, autors piedāvā sekojošu stratēģiju visu 1-bloku meklēšanai:

1. Vaicājam katru bitu x_i pēc kārtas, sākot ar $i = 1$. Kamēr nav bijusi neviena kļūda, minam, ka $x_i = 0$. Tad, pirmoreiz kļūdoties, atrodam pirmā 1-bloka sākumpozīciju $l_1 : x_l = 1$.

2. Kamēr ir bijusi tieši viena kļūda, minam, ka $x_i = 1$. Tad, kļūdoties otrreiz, atrodam pirmā 1-bloka beigu pozīciju $r_1 = i - 1$.

...

$2j - 1$. Kamēr ir bijušas tieši $2(j-1)$ kļūdas, minam, ka $x_i = 0$. Tad, kļūdoties $(2j-1)$-to reizi, atrodam j-tā 1-bloka sākumpozīciju $l_j : x_j = 1$.

$2j$. Kamēr ir bijušas tieši $2j - 1$ kļūdas, minam, ka $x_i = 1$. Tad, kļūdoties $2j$-to reizi, atrodam j-tā 1-bloka beigu pozīciju $r_j = i - 1$.

...

Mainīgais j var pieņemt vērtības no 1 līdz K. Aprakstītie soļi ir jāveic kamēr mēs nebūsim pavaicājuši visus N bitus.

Ja esam pavaicājuši visus bitus un neesam kļūdījušies, tad tas nozīmē, ka visi virknes simboli ir nulles, jeb $\forall_{1\le i\le N}(x_i = 0)$. Tādā gadījumā virknē nav neviena 1-bloka, un jāatgriež tukša kopa.

Ja esam pavaicājuši visus bitus, kļūdoties nepāra reižu skaitu, tad tas nozīmē, ka pēdējā (K-tā) 1-bloka beigas sakrīt ar vārda x beigām (t.i. šis 1-bloks ir vārda x prefikss). Tādā gadījumā $r_K = N$.

Kā var viegli redzēt no aprakstītās stratēģijas, katrā solī tiek pieļauts ne vairāk par vienu kļūdu, tātad pavisam ir jāveic tieši $2K$ soļi, līdz ar to algoritms kļūdās vidēji ne vairāk par $2K$ reizēm.

Tālāk apskatīsim šī problēmas divas iespējamas variācijas.

4.3.1. 1-bloku meklēšana, ja to skaits ir zināms iepriekš

Ja pašā sākumā ir zināma K vērtība (t.i. mēs zinām vārda x 1-bloku skaitu jau iepriekš), tad varam uzskatīt K par konstanti. Tad, saskaņā ar teorēmu, K bloku meklēšanas algoritma kvantu vaicājumu sarežģītība ir $Q = O(\sqrt{TG}) = O(\sqrt{2KN}) = O(\sqrt{N})$.

Daudz interesantāk (un arī sarežģītāk) ir gadījumā, kad vārda 1-bloku skaits nav iepriekš zināms.

4.3.2. 1-bloku meklēšana, ja to skaits nav zināms iepriekš

Ja sākumā nav zināma K vērtība, tad tas būtiski maina risināmo uzdevumu – šoreiz ir jāizrēķina vidējais kļūdu skaits pa visiem vārdiem x. Kļūdu skaits tad ir divas reizes lielāks par vidējo 1-bloku skaitu $\overline{K}$.

$$G = 2\overline{K} = 2 \cdot \frac{1 \cdot s_1 + 2 \cdot s_2 + 3 \cdot s_3 + \cdots + h \cdot s_h}{2^N} = \frac{1 \cdot s_1 + 2 \cdot s_2 + 3 \cdot s_3 + \cdots + h \cdot s_h}{2^{N-1}}$$

kur h ir lielākais iespējamais 1-bloku skaits vienam vārdam garumā N, un s_i ir tādu vārdu skaits, kuros ir tieši i 1-bloki.

Viegli redzēt, ka lielākais 1-bloku skaits ir sasniedzams tad, ja katra 1-bloka garums ir 1, un starp katriem blakus esošiem 1-blokiem ir tikai viena nulle. Piemēram, vārda „1010101" garumam $N = 7$ lielākais iespējamais 1-bloku skaits ir 4.

Tāpēc lielākais 1-bloku skaits ir $h = \left\lceil \frac{N}{2} \right\rceil$. Apskatītajā piemērā $N = 7$, tātad $h = \left\lceil \frac{7}{2} \right\rceil = 4$.

Autors piedāvā sekojošu paņēmienu, kas ļauj izrēķināt vērtības $\{s_i\}$ kombinatoriski.

Apzīmēsim katra 1-bloka pirmo vieninieku ar simbolu ⊛. Ar to pašu simbolu apzīmēsim arī katram 1-blokam sekojošo nulli. Jāņēm vērā, ka pēc pēdēja 1-bloka var nebūt simbolu (ja tas ir vārda x prefikss). Tāpēc pierakstīsim vārdam x vēl vienu, $(N+1)$-to, simbolu, kurš vienmēr būs 0. Tad vārdā x vienmēr visiem 1-blokiem sekos vismaz viena 0.

s_1 ir vienāds ar visu to vārdu skaitu, kuros ir tieši viens 1-bloks. Dažādiem vārdiem 1-bloku pozīcijas atšķiras, tāpēc mēs interesējamies par visu iespējamo viena 1-bloka sākuma un beigu pozīciju kombinācijām.

$$s_1 = C_{N+1}^2$$

Piemēram, ja $N = 3$, tad $s_1 = C_4^2 = 6$. Atbilstība (bijekcija) starp kombinācijām un vārdiem ir nodemonstrēta zemāk.

1. ⊛⊛ 00 → 100
2. ⊛ 0 ⊛ 0 → 110
3. ⊛ 00 ⊛ → 111
4. 0 ⊛⊛ 0 → 010
5. 0 ⊛ 0 ⊛ → 011
6. 00 ⊛⊛ → 001

Viegli pārliecināties, ka citu vārdu garumā 3 ar vienu 1-bloku nav. No $2^3 = 8$ vārdiem atliek tikai vārds 000, kurā nav neviens 1-bloks, un vārds 101, kurā ir divi 1-bloki.

Divu 1-bloku gadījumā $N+1$ pozīcijās jāizvieto 4 simboli ⊛ (2 sākumpozīcijas un 2 beigu pozīcijas). Skaidrs, ka šo simbolu izlikšanas secība nav svarīga, jo pirmās divas simbolu ⊛ pozīcijas nosaka pirmā 1-bloka sākumu un beigas, bet pēdējās divas pozīcijas – pēdējā 1-bloka sākumu un beigas. Tātad $s_2 = C_{N+1}^4$.

Tātad, ja 1-bloku skaits ir i, tad $N+1$ pozīcijās ir jāizvieto $2i$ simboli ⊛, tāpēc iegūstam $s_i = C_{N+1}^{2i}$.

Tātad vidējais kļūdu skaits ir

$$G = \frac{1 \cdot C_{N+1}^2 + 2 \cdot C_{N+1}^4 + 3 \cdot C_{N+1}^6 + \cdots + \left\lceil \frac{N}{2} \right\rceil \cdot C_{N+1}^{2\left\lceil \frac{N}{2} \right\rceil}}{2^{N-1}}$$

Lai šo izteiksmi novienkāršotu, ar matemātiskās indukcijas palīdzību pierādīsim, ka

$$1 \cdot C_{N+1}^2 + 2 \cdot C_{N+1}^4 + 3 \cdot C_{N+1}^6 + \cdots + \left\lceil \frac{N}{2} \right\rceil \cdot C_{N+1}^{2\left\lceil \frac{N}{2} \right\rceil} = (N+1)2^{N-2}$$

Bet pirms tam nodefinēsim dažas lemmas, kuras noderēs pierādījumā.

Lemma 1.

$$\sum_{k=0}^{n} C_n^k = 2^n$$

Šī ir klasiskā vienādība, kura tiek mācīta kombinatorikas pamatu mācību kursos. Tā kā visas saskaitāmās vērtības sakrīt ar $(1+x)^n$ koeficientiem pie attiecīgām x pakāpēm, tad vienādība ir spēkā, pielietojot substitūciju $x = 1$.

Lemma 2.

$$\sum_{k=0}^{n} k \cdot C_n^k = n \cdot 2^{n-1}$$

Šī arī ir klasiskā vienādība, kuras pierādījumu var viegli atrast kombinatorikas mācību grāmatās vai citos viegli pieejamos avotos. Darba ietvaros tā netiks pierādīta.

Lemma 3 (Paskāla identitāte).

$$C_{n-1}^{k-1} + C_{n-1}^{k} = C_n^k$$

Vēl viena klasiskā vienādība (tā ir Paskāla trijstūra veidošanas pamatā), kuras pierādījums ir ļoti vienkāršs:

$$C_{n-1}^{k-1} + C_{n-1}^{k} = \frac{(n-1)!}{(k-1)!(n-k)!} + \frac{(n-1)!}{k!(n-k-1)!} = \frac{(n-1)!}{(k-1)!(n-k-1)!}\left(\frac{1}{n-k} + \frac{1}{k}\right) = \frac{n!}{k!(n-k)!}.$$

Atgriežoties pie apskatāmās summas,

$$1 \cdot C_{N+1}^2 + 2 \cdot C_{N+1}^4 + 3 \cdot C_{N+1}^6 + \cdots + \left\lceil \frac{N}{2} \right\rceil \cdot C_{N+1}^{2\left\lceil \frac{N}{2} \right\rceil} = \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} i \cdot C_{N+1}^{2i}$$

Pareizināsim to ar $\frac{2}{2}$, lai koeficienti pirms kombināciju formulām sakristu ar to augšējiem indeksiem:

$$\sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} i \cdot C_{N+1}^{2i} = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} 2i \cdot C_{N+1}^{2i}$$

Tagad katram saskaitāmajam vienreiz pielietosim 3. lemmu:

$$\frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} 2i \cdot C_{N+1}^{2i} = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} 2i \cdot \left(C_N^{2i-1} + C_N^{2i}\right) = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left(2i \cdot C_N^{2i-1} + 2i \cdot C_N^{2i}\right)$$

Vēlreiz pielietosim 3. lemmu, bet tikai „daļēji", lai visiem saskaitāmajiem ar nepāra augšējo kombinatorisko indeksu būtu atbilstošs koeficients priekšā:

$$\frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left(2i \cdot C_N^{2i-1} + 2i \cdot C_N^{2i}\right) = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left(C_{N-1}^{2i-2} + C_{N-1}^{2i-1} + (2i-1) \cdot C_N^{2i-1} + 2i \cdot C_N^{2i}\right) =$$

$$= \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left(C_{N-1}^{2i-2} + C_{N-1}^{2i-1}\right) + \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left((2i-1) \cdot C_N^{2i-1} + 2i \cdot C_N^{2i}\right)$$

Kā redzam, summa ir sadalījusies divās pazīstamās virknēs:

$$A = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left(C_{N-1}^{2i-2} + C_{N-1}^{2i-1} \right) = \frac{1}{2} \cdot \sum_{i=0}^{2\left\lceil \frac{N}{2} \right\rceil - 1} C_{N-1}^{i}$$

un

$$B = \frac{1}{2} \cdot \sum_{i=1}^{\left\lceil \frac{N}{2} \right\rceil} \left((2i-1) \cdot C_N^{2i-1} + 2i \cdot C_N^{2i} \right) = \frac{1}{2} \cdot \sum_{i=1}^{2\left\lceil \frac{N}{2} \right\rceil} i \cdot C_N^i = \frac{1}{2} \cdot \sum_{i=0}^{2\left\lceil \frac{N}{2} \right\rceil} i \cdot C_N^i$$

Atsevišķi apskatīsim divus gadījumus:

1. ja N ir pāra skaitlis, tad, saskaņā ar 1. lemmu,

$$A = \frac{1}{2} \cdot \sum_{i=0}^{N-1} C_{N-1}^{i} = \frac{1}{2} \cdot 2^{N-1} = 2^{N-2}$$

un

$$B = \frac{1}{2} \cdot \sum_{i=0}^{N} i \cdot C_N^i = \frac{1}{2} \cdot N \cdot 2^{N-1} = N \cdot 2^{N-2}$$

Tātad visa summa sanāk

$$A + B = 2^{N-2} + N \cdot 2^{N-2} = (N+1) \cdot 2^{N-2}$$

2. pretējā gadījumā, ja N ir nepāra skaitlis, tad

$$A = \frac{1}{2} \cdot \sum_{i=0}^{N} C_{N-1}^{i} = \frac{1}{2} \cdot 2^{N-1} + 0 = 2^{N-2}$$

un

$$B = \frac{1}{2} \cdot \sum_{i=0}^{N+1} i \cdot C_N^i = \frac{1}{2} \cdot N \cdot 2^{N-1} + 0 = N \cdot 2^{N-2}$$

Arī šajā gadījumā iegūstam

$$A + B = 2^{N-2} + N \cdot 2^{N-2} = (N+1) \cdot 2^{N-2}$$

Esam pierādījuši izvirzīto vienādību. Tagad varam atgriezties pie vidējās kļūdas rēķināšanas un pierakstīt to elegantāk:

$$G = \frac{1 \cdot C_{N+1}^2 + 2 \cdot C_{N+1}^4 + 3 \cdot C_{N+1}^6 + \cdots + \left\lceil \frac{N}{2} \right\rceil \cdot C_{N+1}^{2\left\lceil \frac{N}{2} \right\rceil}}{2^{N-1}} = \frac{(N+1) \cdot 2^{N-2}}{2^{N-1}} = \frac{N+1}{2}$$

Beidzot varam novērtēt algoritma kvantu vaicājumu sarežģītību $Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{N \cdot \frac{N+1}{2}}\right) = O(N)$.

Interesanti, ka šajā (vispārīgākā) uzdevuma variantā sarežģītības novērtējums ir vājāks, nekā iepriekš apskatītajā.

4.4. 1-bloku meklēšanas vispārinājums uz vairākām dimensijām

Iepriekšējos uzdevumos mēs apskatījām 1-bloku meklēšanas algoritmus, kuri darbojas vienā dimensijā. Šeit mēs apskatīsim pētāmo problēmu vispārinājumus patvaļīgam dimensiju skaitam un pielāgosim iepriekš aprakstītos algoritmus vairākām dimensijām.

Ir dotas N^D dažādu punktu koordinātes D-dimensiju telpā. $D \geq 1$. Apzīmēsim doto punktu kopu ar P. Šie punkti veido $N \times N \times ... \times N$ režģi: attālums starp katriem diviem blakus esošiem režģa punktiem ir vienāds ar kaut kādu konstanti $\Delta \in \mathbb{R}$, $\Delta > 0$, un katram režģa iekšējam punktam blakus ir tieši $2D$ punkti. Katram punktam tiek piekārtots cipars 0 vai 1. Pieejamais vaicājumu orākuls O_x uz punkta vaicājumu atbild ar šim punktam piekārtoto ciparu.

Ērtībai pieņemsim, ka $\Delta = 1$, un ka režģa vienā no stūriem esošā punkta koordinātes ir $(1,1,1, \dots ,1)$, nezaudējot uzdevuma vispārīgumu (nepieciešamības gadījumā izvēlēto punktu var pārvietot uz patvaļīgu citu pozīciju D-dimensiju telpā, un arī attālumu starp punktiem Δ var pareizināt ar patvaļīgu pozitīvu konstanti). Varam uzskatīt, ka punkti, kuriem piekārtots 1, ir atrodami apskatāmajā telpā, un citi punkti nav.

Vispārināsim 1-bloku definīciju uz vairākām dimensijām.

Definīcija 2. *Par **D-dimensiju 1-bloku** dotajai punktu kopai sauksim sekojošu netukšu dotās kopas apakškopu S:*

1. *visiem kopas S punktiem ir piekārtots cipars 1,*
2. *katra punkta $s \in S$ visiem blakus punktiem $p \in P$ izpildās viens no nosacījumiem:*
   - *(a) punktam p ir piekārtots cipars 0 un $p \notin S$, vai*
   - *(b) punktam p ir piekārtots cipars 1 un $p \in S$.*

Intuitīvi D-dimensiju 1-bloku var uztvert kā atsevišķu dotajā telpā „iekrāsotu" apgabalu (tas nešķeļas ne ar vienu citu „iekrāsotu" apgabalu).

Vairāku dimensiju gadījumā punktu skaits var būt ļoti liels, un tāpēc nebūtu lietderīgi algoritmam atgriezt visus tos punktus, kuri ierobežo D-dimensiju 1-blokus. Tāpēc nodefinēsim mērķi nedaudz savādāk: uzdevums ir atrast visu punktu skaitu, kuriem ir piekārtots cipars 1, jeb jāatrod visu D-dimensiju 1-bloku izmēru summa.

4.4.1. Vairāku D-dimensiju 1-bloku meklēšana

Uzdevums ir līdzīgs tam, ko jau atrisinājām nodaļā 4.3 vienas dimensijas gadījumā: dotajā punktu kopā P ir vairāki D-dimensiju 1-bloki, kuri ir jāatrod. Atgādināsim, ka 1-

dimensijas gadījumā tika meklētas 1-bloka sākuma un beigu pozīcijas, bet tagad, D-dimensiju gadījumā interesēsimies par 1-blokam piederošu punktu skaitu (1-bloka izmēru).

Izmantosim līdzīgu stratēģiju, ka iepriekšējos apakšuzdevumos, ar nelielu modifikāciju. Apzīmēsim dotajā telpā esošo D-dimensiju 1-bloku skaitu ar K, un visu atrasto 1-bloku punktu kopu ar F.

> Kamēr nebūsim kļūdījušies, vaicājam katru punktu pēc kārtas, sākot ar punktu, kura koordinātes ir $(1,1,1,\dots,1,1)$. Nākamie punkti tad būs $(1,1,1,\dots,1,2)$, $(1,1,1,\dots,1,3),\dots,(1,1,1,\dots,1,N)$, $(1,1,1,\dots,2,1)$, $(1,1,1,\dots,2,2),\dots,$ $(N,N,N,\dots,N,N)$. Minam, ka punktam ir piekārtots cipars 0.
>
> (a) Ja esam pareizi uzminējuši, tad apskatāmais punkts mūs neinteresē, jo tas nepieder nevienam no meklējamajiem 1-blokiem. Turpinām vaicāt, pārejot pie nākamā punkta.
>
> (b) Ja esam kļūdījušies, tad apskatāmajam punktam $p = (p_1, p_2, \dots, p_D)$ ir piekārtots cipars 1, tātad tas pieder kādam 1-blokam, kuru mēs vēl neesam līdz šim atraduši. Pievienojam p atrasto punktu kopai F. Šajā gadījumā izmantosim iespēju un palaidīsim klasisko „flood fill" algoritmu (DFS vai BFS – tas šeit nav būtiski), atrodot visus blakus esošos punktus ar piekārtotiem vieniniekiem. Formālāk, katram atrastajam 1-bloka punktam ar koordinātēm $(p_1, p_2, \dots, p_D)$ vaicājam arī visus tā kaimiņpunktus $(p_1 \pm 1, p_2, \dots, p_D)$, $(p_1, p_2 \pm 1, \dots, p_D)$, $\dots, (p_1, p_2, \dots, p_D \pm 1)$, kuri pieder dotajam režģim P, un par kuriem vēl neesam līdz šim vaicājuši. Katru atrasto 1-blokam piederošo punktu pievienojam kopai F.

Algoritma rezultāts ir atrasto punktu kopas izmērs $|F|$.

Ja mums ir iepriekš zināms visu 1-bloku skaits K, tad varam pabeigt algoritmu tiklīdz būsim atraduši K-tā 1-bloka visus punktus. Savādāk ir jāvaicā visi punkti $p \in P$.

Sliktākajā gadījumā algoritma izpildes laikā nāksies pavaicāt katru no $N \cdot N \cdot \dots \cdot N = N^D$ punktiem, tātad $T = N^D$. Lai novērtētu vidējo kļūdu skaitu, ieviesīsim tādu jēdzienu kā 1-bloka virsmas laukums.

Definīcija 3. *Par D-dimensiju 1-bloka **virsmu** sauksim visu to punktu kopu, kuriem ir piekārtots cipars 0 un kuriem blakus ir vismaz viens dotā 1-bloka punkts. Tad ar D-dimensiju 1-bloka **virsmas laukumu** sapratīsim šī 1-bloka virsmas punktu skaitu.*

Vienas dimensijas gadījumā 1-bloka virsmu veido nulles pirms un pēc dotā 1-bloka (ja tādas ir), tātad 1-dimensijas 1-bloka virsmas laukums nepārsniedz 2. Divu dimensiju gadījumā 1-bloka virsmas laukumu varam intuitīvi salīdzināt ar dotā 1-bloka „perimetru". Trīs

dimensiju gadījumā 1-bloka virsma ir līdzīga ģeometriskajai virsmai. Šīs intuitīvās analoģijas palīdz definīcijas saprašanā.

No algoritma apraksta varam izsecināt, ka katra 1-bloka atrašanai mēs pieļaujam tik daudz kļūdas, cik ir punkti šī 1-bloka virsmā. Tātad i-tais 1-bloks izraisa līdz s_i kļūdām, kur s_i – dotā 1-bloka virsas laukums (ja kārtējā apskatāmā 1-bloka virsma pārklājas ar kādu no iepriekš apskatīto 1-bloku virsmām, tad kļūdu skaits būs mazāks, jo mēs nevaicāsim vienus un tos pašus punktus vairākkārt).

Tad algoritma kļūdu skaits ir vienāds ar

$$S = \sum_{i=1}^{K} s_i$$

jeb visu apskatāmajā telpā esošo D-dimensiju 1-bloku virsmas laukumu kopsumma. Līdz ar to algoritma kvantu vaicājumu sarežģītība sanāk $Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{N^D S}\right) = O\left(N^{\frac{D}{2}}\sqrt{S}\right)$.

4.4.2. Viena D-dimensiju 1-bloka meklēšana, kuram zināmi minimālie izmēri

Apskatīsim nodaļas 4.2. uzdevuma vispārinājumu patvaļīgam dimensiju skaitam: dotajā punktu kopā P ir tieši viens D-dimensiju 1-bloks, kurš arī ir jāatrod. Par 1-bloku ir zināms, ka tas ir tik liels, ka iekļauj sevī 1-apakšbloku izmērā $L \times L \times ... \times L = L^D$. Formālāk, tas nozīmē, ka režģis izmērā $L \times L \times ... \times L = L^D$, kura visiem punktiem ir piekārtots cipars 1, veido punktu kopu R, kura ir meklējamā D-dimensiju 1-bloka punktu kopas apakškopa, jeb $R \subseteq P$. Piemēram, 2-dimensiju gadījumā tas nozīmētu, ka meklējamais 3-dimensiju 1-bloks ietver sevī „kvadrātu" ar malas garumu L. 3-dimensiju gadījumā – „kubu" ar malas garumu L.

Apzīmēsim dotajā telpā esošo D-dimensiju 1-bloku skaitu ar K, un visu atrasto 1-bloku punktu kopu ar F. Izmantosim sekojošu minēšanas stratēģiju:

> Kamēr nebūsim kļūdījušies, vaicājam punktus tā, lai nekādi divi vaicājamie punkti neatrastos viena režģa izmērā L^D ietvaros – sākot ar punktu, kura koordinātes ir $(L, L, L, \dots, L, L)$. Nākamie punkti tad būs $(L, L, L, \dots, L, 2L)$, $(L, L, L, \dots, L, 3L), \dots, \left(L, L, L, \dots, L, \left\lfloor \frac{N}{L} \right\rfloor L\right)$, $(L, L, L, \dots, 2L, L)$, $(L, L, L, \dots, 2L, 2L), \dots$, $\left(\left\lfloor \frac{N}{L} \right\rfloor L, \left\lfloor \frac{N}{L} \right\rfloor L, \left\lfloor \frac{N}{L} \right\rfloor L, \dots, \left\lfloor \frac{N}{L} \right\rfloor L, \left\lfloor \frac{N}{L} \right\rfloor L\right)$. Minam, ka punktam ir piekārtots cipars 0.
>
> (a) Ja minējums ir pareizs, tad apskatāmais punkts mūs neinteresē, jo tas nepieder meklējamajām D-dimensiju 1-blokam. Turpinām vaicāt, pārejot pie

nākamā punkta.

(b) Ja mēs esam kļūdījušies, tad apskatāmajam punktam $p = (p_1, p_2, \dots, p_D)$ ir piekārtots cipars 1, tātad tas pieder kādam 1-blokam, kuru mēs vēl neesam līdz šim atraduši. Pievienojam p atrasto punktu kopai F. Šajā gadījumā izmantosim iespēju un palaidīsim „flood fill" algoritmu, atrodot visus blakus esošos punktus ar piekārtotiem vieniniekiem. Formālāk, katram atrastajam 1-bloka punktam ar koordinātēm $(p_1, p_2, \dots, p_D)$ vaicājam arī visus tā kaimiņpunktus $(p_1 \pm 1, p_2, \dots, p_D)$, $(p_1, p_2 \pm 1, \dots, p_D)$, ..., $(p_1, p_2, \dots, p_D \pm 1)$, kuri pieder dotajam režģim P, un par kuriem vēl neesam līdz šim vaicājuši. Katru atrasto 1-blokam piederošo punktu pievienojam kopai F.

Algoritma rezultāts ir atrasto punktu kopas izmērs $|F|$.

Sliktākajā gadījumā algoritma izpildes laikā jāpavaicā katrs no $\left\lfloor \frac{N}{L} \right\rfloor \cdot \left\lfloor \frac{N}{L} \right\rfloor \cdot \dots \cdot \left\lfloor \frac{N}{L} \right\rfloor = \left\lfloor \frac{N}{L} \right\rfloor^D$ punktiem, kamēr nebūsim atraduši pirmo 1-blokam punktu. Apzīmēsim to ar $T_1 = \left\lfloor \frac{N}{L} \right\rfloor^D$. Tad uzreiz tiek darbināts „flood fill" algoritms, kura izpildes laikā tiek vaicāti visi 1-blokam piederošie punkti, kā arī punkti, kas veido šī 1-bloka virsmu. Apzīmēsim to ar $T_2 = |V| + |S|$, kur V – 1-blokam visu piederošo punktu kopa („tilpums") un S – 1-bloka virsma.

Tad sliktākajā gadījumā pieprasījumu skaits ir $T = O(T_1 + T_2) = O\left(\left\lfloor \frac{N}{L} \right\rfloor^D + |V| + |S|\right)$. Nav grūti pamanīt, ka $O(|V| + |S|) = O(|V|)$. Neformāls pamatojums varētu būt tāds, ka katram D-dimensiju 1-bloka virsmas punktam blakus ir vismaz viens 1-bloka iekšējais punkts (un ne vairāk par konstantu skaitu – $2D$ – punktu). Savukārt, neobligāti katram iekšējam punktam blakus ir kaut viens ārējais punkts. No tā izriet $|S| \leq 2D|V|$. Tāpēc

$$T = O\left(\left\lfloor \frac{N}{L} \right\rfloor^D + |V| + |S|\right) = O\left(\left\lfloor \frac{N}{L} \right\rfloor^D + |V|\right)$$

Kļūdas aprakstītajā algoritmā ir tikai atrodot pirmo 1-bloka iekšējo punktu, kā arī visu 1-bloka virsmas punktos, dēļ „flood fill" algoritma pielietošanas. Kļūdu vidējais skaits tad ir novērtējams kā

$$G = \Theta(1 + |S|) = \Theta(|S|)$$

Līdz ar to algoritma kvantu vaicājumu sarežģītība sanāk

$$Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{\left(\left\lfloor \frac{N}{L} \right\rfloor^D + |V|\right) \cdot |S|}\right)$$

Nav pārsteidzoši, ka iegūtais novērtējums ir daudzos gadījumos labāks par iepriekšējā uzdevuma sarežģītības novērtējumu, jo informācija par minimāliem 1-bloka gabarītiem ļauj izvairīties no lielā punktu skaita vaicāšanas.

5. ALGORITMI UZ GRAFIEM

Šajā nodaļā tiek apskatīti daži algoritmi uz grafiem, un tiek novērtēta to kvantu vaicājumu sarežģītība. Ja nav norādīts savādāk, tad turpmāk uzskatīsim, ka visi apskatāmie grafi ir neorientēti, un ka neorientētā grafā šķautne $(u, v) = (v, u)$.

5.1. Trijstūru meklēšana grafā

Definīcija 4. *Par **trijstūru** sauksim tādu grafa saistītu apakšgrafu, kurš:*

1. *sastāv no 3 virsotnēm un 3 šķautnēm,*
2. *veido ciklu garumā 3.*

Mums ir dots grafs, kuram ir zināma V – virsotņu kopa, un par kuru ir zināms, ka tajā ir tieši viens trijstūris. Grafa incidenču matrica A ir uzdota melnās kastes (orākula) veidā: vaicājot par šķautni (u, v), kur $u, v \in V$, orākuls atgriež 1, ja dotajā grafā eksistē šķautne no u uz v. Pretējā gadījumā orākuls atgriež 0. Visu grafā esošo šķautņu kopu apzīmēsim ar $E \subseteq V \times V$. Sākumā mēs nezinām, kuri kortežu kopas $V \times V$ elementi pieder E. Šķautņu kopas izmēru apzīmēsim ar $N = |E|$.

Autors piedāvā sekojošu vienkāršu vaicājumu stratēģiju:

> Katrā solī patvaļīgi izvēlamies šķautni $(u, v) \in V \times V, u \neq v$, par kuru līdz šim vēl neesam vaicājuši, un vaicājam par to orākulam A, minot, ka $(u, v) \notin E$, t.i. ka orākuls atgriezīs 0.
>
> (c) ja orākuls atgriež 0, tad mēs esam pareizi uzminējuši un tiešām $(u, v) \notin E$. Šajā gadījumā atrasto šķautņu skaits nav mainījies, tāpēc ir jāmēģina vēlreiz.
>
> (d) ja orākuls atgriež 1, tad mēs esam kļūdījušies, un šķautne (u, v) patiesībā pieder grafam, t.i. $(u, v) \in E$. Šajā gadījumā esam atraduši jaunu grafa šķautni. Tad pārbaudām, vai šī šķautne veido trijstūri ar divām citām iepriekš atrastajām šķautnēm:
>
> (i) ja atrodam trijstūri, tad esam izpildījuši uzdevumu, un varam pabeigt algoritma izpildi,
>
> (ii) ja neatrodam trijstūri, tad jāmēģina vēlreiz – jāatrod cita šķautne.

Lai novērtētu šim algoritmam atbilstošo vidējo kļūdu skaitu, autors piedāvā sekojošu paņēmienu.

Apskatīsim virkni no visām dotajam grafam piederošām šķautnēm $e_{p_1}, e_{p_2}, \dots, e_{p_N}$, kur šķautnes ir sakārtotas to vaicāšanas secībā ($p_1, p_2, \dots, p_N$ ir patvaļīga virknes $1, 2, \dots, N$

permutācija). Aprakstītais algoritms veic vaicājumus, kamēr nebūs atradis tās trīs šķautnes e_{p_a}, e_{p_b} un e_{p_c}, kuras veido meklēto trijstūri.

Tagad pārrakstīsim šo virkni, aizvietojot e_{p_a}, e_{p_b} un e_{p_c} ar ciparu 1 un pārējās šķautnes ar ciparu 0. Jaunā virkne var izskatīties, piemēram, sekojoši:

$$\mathbf{001010001}000$$

Šajā piemērā $N = 11$, un algoritms būs kļūdījies 8 reizes (iezīmētas treknrakstā).

Lai izrēķinātu vidējo kļūdu skaitu G, apskatīsim šajā virknē visas iespējamās trešā (pēdējā) vieninieka pozīcijas:

- ja pēdējais vieninieks ir 3.-ajā pozīcijā, tad pārējos 2 vieniniekus var izvietot iepriekšējās 2 pozīcijās C_2^2 dažādos veidos;
- ja pēdējais vieninieks ir 4.-ajā pozīcijā, tad pārējos 2 vieniniekus var izvietot iepriekšējās 3 pozīcijās C_3^2 dažādos veidos;
- …
- ja pēdējais vieninieks ir i-ajā pozīcijā, $3 \leq i \leq N$, tad pārējos 2 vieniniekus var izvietot iepriekšējās $i-1$ pozīcijās C_{i-1}^2 dažādos veidos.

Tad vidējo kļūdu skaitu varam izteikt kā

$$G = \frac{3 \cdot C_2^2 + 4 \cdot C_3^2 + \cdots + N \cdot C_{N-1}^2}{C_N^3}$$

Lai vienkāršotu šo izteiksmi, parādīsim, ka

$$3 \cdot C_2^2 + 4 \cdot C_3^2 + \cdots + N \cdot C_{N-1}^2 = \frac{(N-2)(N-1)N(N+1)}{8}$$

Pierādīsim to ar matemātiskās indukcijas palīdzību.

Indukcijas bāze. Vienādība ir spēkā pie $N = 3$:

$$3 \cdot 1 = \frac{(3-2)(3-1)3(3+1)}{8} \Leftrightarrow 3 = 3$$

Induktīvā pāreja. Pieņemot, ka vienādība ir spēkā pie $N = k$, parādīsim, ka tā paliek spēkā arī pie $N = k+1$:

$$\frac{(k-2)(k-1)k(k+1)}{8} + (k+1) \cdot C_k^2$$

$$= \frac{\big((k+1)-2\big)\big((k+1)-1\big)(k+1)\big((k+1)+1\big)}{8}$$

$$\Leftrightarrow \frac{(k-2)(k-1)k(k+1)}{8} + (k+1) \cdot \frac{k(k-1)}{2} = \frac{(k-1)k(k+1)(k+2)}{8}$$

$$\Leftrightarrow (k-2)(k-1)k(k+1) + 4(k+1)k(k-1) = (k-1)k(k+1)(k+2)$$

$$\Leftrightarrow (k-1)k(k+1)\big((k-2)+4\big) = (k-1)k(k+1)(k+2)$$

$$\Leftrightarrow \boldsymbol{(k-1)k(k+1)(k+2) = (k-1)k(k+1)(k+2)}$$

No tā izriet, ka vienādība ir spēkā patvaļīgām N vērtībām, $N \geq 3$.

Iegūstam, ka

$$G = \frac{3 \cdot C_2^2 + 4 \cdot C_3^2 + \cdots + N \cdot C_{N-1}^2}{C_N^3} = \frac{\frac{(N-2)(N-1)N(N+1)}{8}}{\frac{N(N-1)(N-2)}{6}} = \frac{\boldsymbol{3(N+1)}}{\boldsymbol{4}}$$

Tātad meklētais algoritma kvantu vaicājumu sarežģītības novērtējums ir $Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{|V|^2 \frac{3(N+1)}{4}}\right) = O\left(|V|\sqrt{N}\right) = O\left(|V|\sqrt{|E|}\right)$

Piezīmes: varam pamanīt, ka iegūtais novērtējums ir diezgan vājš. Izrādās, ka visiem grafu algoritmiem, kuriem ir zināma virsotņu kopa V un ir pieejama incidenču matrica A orākula veidā, ir spēkā tāds pats kvantu vaicājumu sarežģītības novērtējums no augšas, jo vienkārši vienmēr atrodot visas šķautnes E ar $O(|V|^2)$ pieprasījumiem orākulam, un tad pielietojot jebkuru klasisku algoritmu (neobligāti efektīvu) atrastajām šķautnēm, vidējais kļūdu skaits $G = |E|$, un sarežģītības novērtējums ir tieši tāds pats: $Q = O\left(\sqrt{TG}\right) = O\left(|V|\sqrt{|E|}\right)$.

Trijstūru meklēšanas problēmai neorientētā grafā ir zināms būtiski labāks kvantu vaicājumu sarežģītības novērtējums - $\widetilde{O}\left(|V|^{5/4}\right)$ [9]. Darbā iegūtais novērtējums $O\left(|V|\sqrt{|E|}\right) = O\left(|V| \cdot \sqrt{|V|^2}\right) = O(|V|^2)$ ir objektīvi sliktāks.

5.2. K-stūru meklēšana grafā

Definīcija 5. *Par **K-stūru** sauksim tādu grafa saistītu apakšgrafu, kurš:*

1. *sastāv no K virsotnēm un K šķautnēm,*
2. *veido ciklu garumā K.*

Mums ir dots grafs, kuram ir zināma V – virsotņu kopa, un par kuru ir zināms, ka tajā ir tieši viens K-stūris. Tāpat, kā iepriekšējā uzdevumā, grafa incidenču matrica A ir uzdota melnās kastes (orākula) veidā: vaicājot par šķautni (u, v), kur $u, v \in V$, orākuls atgriež 1, ja dotajā grafā eksistē šķautne no u uz v. Pretējā gadījumā orākuls atgriež 0. Visu grafā esošo šķautņu kopu apzīmēsim ar $E \subseteq V \times V$. Sākumā mēs nezinām, kuri kortežu kopas $V \times V$ elementi pieder E. Šķautņu kopas izmēru apzīmēsim ar $N = |E|$.

Izmantosim līdzīgu stratēģiju, kā iepriekšējā uzdevumā, tikai šoreiz pārtrauksim vaicāšanu tajā gadījumā, kad būsim atraduši vajadzīgo K-stūri.

Arī sarežģītības novērtējumam izmantosim līdzīgu pieeju, apskatot grafa šķautņu virkni to vaicāšanas secībā, un aizvietojot K-stūrim piederošas šķautnes ar 1 un pārējās šķautnes ar 0. Aprakstītā virkne var izskatīties, piemēram, sekojoši:

010101000110000

Šajā piemērā $K = 5$, $N = 15$, un algoritms būs kļūdījies 11 reizes (iezīmētas treknrakstā), kamēr neatradīs meklēto 5-stūri.

Tāpat, kā iepriekšējā uzdevumā, bet jau vispārīgāk, lai izrēķinātu vidējo kļūdu skaitu G, apskatīsim šajā virknē visas iespējamās pēdējā vieninieka pozīcijas:

- ja pēdējais vieninieks ir K-ajā pozīcijā, tad pārējos $K-1$ vieniniekus var izvietot iepriekšējās $K-1$ pozīcijās C_{K-1}^{K-1} dažādos veidos;
- ja pēdējais vieninieks ir $(K+1)$-ajā pozīcijā, tad pārējos $K-1$ vieniniekus var izvietot iepriekšējās K pozīcijās C_K^{K-1} dažādos veidos;
- ...
- ja pēdējais vieninieks ir i-ajā pozīcijā, $K \le i \le N$, tad pārējos $K-1$ vieniniekus var izvietot iepriekšējās $i-1$ pozīcijās C_{i-1}^{K-1} dažādos veidos.

Tad vidējo kļūdu skaitu varam izteikt kā

$$G = \frac{K \cdot C_{K-1}^{K-1} + (K+1) \cdot C_K^{K-1} + \cdots + N \cdot C_{N-1}^{K-1}}{C_N^K}$$

Lai šo izteiksmi novienkāršotu, parādīsim, ka

$$K \cdot C_{K-1}^{K-1} + (K+1) \cdot C_K^{K-1} + \cdots + N \cdot C_{N-1}^{K-1} = K \cdot C_{N+1}^{K+1}$$

ir spēkā visiem K, $3 \le K \le N$.

Šoreiz nav nepieciešams izmantot matemātiskās indukcijas paņēmienu, bet pietiek pamanīt, ka

$$i \cdot C_{i-1}^{K-1} = K \cdot C_{i+1}^{K+1} - K \cdot C_i^{K+1}$$

Šo vienādību var pierādīt, pārrakstot vienādības labo pusi:

$$K \cdot C_{i+1}^{K+1} - K \cdot C_i^{K+1} = K \frac{(i+1)!}{(K+1)!\,(i-K)!} - K \frac{i!}{(K+1)!\,(i-K-1)!}$$

$$= \frac{K}{(K+1)!}\left(\frac{(i+1)!}{(i-K)!} - \frac{i!}{(i-K-1)!}\right)$$

$$= \frac{K}{(K+1)!}\left(\frac{i!\,(i+1)}{(i-K)(i-K-1)!} - \frac{i!}{(i-K-1)!}\right)$$

$$= \frac{K}{(K+1)!} \cdot \frac{i!}{(i-K-1)!}\left(\frac{i+1}{i-K} - 1\right) = \frac{K}{(K+1)!} \cdot \frac{i!}{(i-K-1)!} \cdot \frac{K+1}{i-K}$$

$$= \frac{i!}{(K-1)!\,(i-K-1)!\,(i-K)} = i \cdot \frac{(i-1)!}{(K-1)!\,(i-K)!} = i \cdot C_{i-1}^{K-1}$$

Tagad varam pārrakstīt pētāmo summu sekojoši:

$$K \cdot C_{K-1}^{K-1} + (K+1) \cdot C_K^{K-1} + \cdots + N \cdot C_{N-1}^{K-1}$$

$$= (K \cdot C_{K+1}^{K+1} - K \cdot C_K^{K+1}) + (K \cdot C_{K+2}^{K+1} - K \cdot C_{K+1}^{K+1}) + \cdots$$

$$+ (K \cdot C_{N+1}^{K+1} - K \cdot C_N^{K+1}) = K \cdot C_{N+1}^{K+1} - K \cdot C_K^{K+1} = K \cdot C_{N+1}^{K+1}$$

Beidzot, izmantojot iegūto vienādību, varam eleganti novienkāršot vidējā kļūdu skaita izteiksmi:

$$G = \frac{K \cdot C_{K-1}^{K-1} + (K+1) \cdot C_K^{K-1} + \cdots + N \cdot C_{N-1}^{K-1}}{C_N^K} = \frac{K \cdot C_{N+1}^{K+1}}{C_N^K} = \frac{K \cdot \frac{(N+1)!}{(K+1)!\,(N-K)!}}{\frac{N!}{K!\,(N-K)!}}$$

$$= K \cdot \frac{N+1}{K+1} = \frac{\boldsymbol{K}}{\boldsymbol{K+1}} \cdot (\boldsymbol{N+1})$$

Tad K-stūru meklēšanai grafā kvantu vaicājumu sarežģītības novērtējums ir $Q = O\left(\sqrt{TG}\right) = O\left(\sqrt{|V|^2 \cdot \frac{K}{K+1}(N+1)}\right) = O\left(|V|\sqrt{N}\right) = O\left(|V|\sqrt{|E|}\right)$.

REZULTĀTI

Apkopojot iegūtos rezultātus, autors ir pierādījis, ka zemāk pārskaitītajiem uzdevumiem eksistē kvantu algoritms (saskaņā ar [3], „Algorithm 11") ar tam sekojošo kvantu vaicājumu sarežģītības novērtējumu:

- AND vai OR operācijas N bitiem $O\left(\sqrt{N}\right)$
- XOR operācija N bitiem $O(N)$
- Viena 1-bloka meklēšana $O\left(\sqrt{N}\right)$
- Viena 1-bloka meklēšana, kuram zināms minimālais garums $O\left(\sqrt{\frac{N}{L}\log N + \log^2 N}\right)$
- Vairāku 1-bloku meklēšana, ja to skaits ir zināms iepriekš $O\left(\sqrt{N}\right)$
- Vairāku 1-bloku meklēšana, ja to skaits nav zināms iepriekš $O(N)$
- Vairāku 1-bloku meklēšana D dimensijās $O\left(N^{\frac{D}{2}}\sqrt{S}\right)$
- Viena 1-bloka meklēšana D dimensijās ar zināmiem minimāliem izmēriem $O\left(\sqrt{\left(\left\lfloor\frac{N}{L}\right\rfloor^{D} + |V|\right)\cdot|S|}\right)$
- Trijstūru meklēšana grafā $O\left(|V|\sqrt{|E|}\right)$
- K-stūru meklēšana grafā $O\left(|V|\sqrt{|E|}\right)$

Par svarīgākiem rezultātiem darba autors uzskata iegūtos novērtējumus algoritmiem uz simbolu virknēm un to vispārinājumiem vairākām dimensijām. It īpaši interesanti izskatās novērtējums 1-bloka meklēšanai ar zināmu minimālo garumu, jo tas ir asimptotiski ievērojami labāks par citiem apskatītajiem algoritmiem, un atbilstošajai problēmai var būt plašs praktisks pielietojums.

Uzdevumiem uz grafiem nav izdevies atrast algoritmu, kura novērtējums būtu labāks par triviāla algoritma novērtējumu $O\left(|V|\sqrt{|E|}\right)$, kur $|V|$ – virsotņu skaits un $|E|$ – šķautņu skaits grafā, bet pētījumu gaitā autors ir parādījis, ka tāds novērtējums ir spēkā jebkuram algoritmam uz grafiem, kuram vienīgais veids gūt informāciju par grafa šķautnēm ir vaicājot incidenču matricas orākulu. Varam uzskatīt, ka ir atrasti jauni algoritmi, jo līdz šim neviens nebija pētījis algoritmus šīm problēmām.

SECINĀJUMI

Pētījuma gaitā tika konstatēts, ka izmantotais bumbas meklēšanas modelis ļauj ātri noteikt kvantu vaicājumu sarežģītības novērtējumus, kaut arī salīdzinoši vājus. Labāku novērtējumu sasniegšanai šī modeļa ietvaros ir nepieciešams izmantot sarežģītākus algoritmus un vēlams apskatīt specifiskākās problēmas.

Autors saskata plašas iespējas turpmākajiem algoritmu pētījumiem bumbas meklēšanas modelī:

- sarežģītāku datu struktūru pielietošana efektīvākai vaicājumu rezultātu minēšanas stratēģijai,
- varbūtisku algoritmu analīze, kuri rēķina $f(x)$ ar ierobežotu kļūdu,
- datorsimulāciju izstrāde minēšanas stratēģiju interaktīvai analīzei, kā arī novērtējumu uzlabošanas iespēju meklēšanai.

Autora pieredzē šis ir pirmais tīri zinātnisks darbs. Viens no sākotnējiem mērķiem ir bijis pārbaudīt savas spējas zinātniskos pētījumos, kā arī izsekot līdzi savu personisku īpašību atbilstību tīri zinātniskā ilgtermiņa darbam. Gūtās atziņas un pārdomas par procesu kopumā ir diezgan pozitīvas.

PATEICĪBAS

Es gribētu izteikt savu pateicību mana darba vadītājam un pasniedzējam – profesoram Andrim Ambainim – par veltītu laiku, sapratni, pacietību un palīdzību darba tapšanas laikā, kā arī par interesantu darba tēmu piedāvājumu.

IZMANTOTĀ LITERATŪRA UN AVOTI

[1] A. C. Elitzur, L. Vaidman, „Quantum mechanical interaction-free measurements," *Foundations of Physics 23 no. 7*, (July, 1993) 987–997, arXiv:hep-th/9305002.

[2] Y. Aharonov and D. Bohm, *Phys. Rev.* 115, 1804 (1959).

[3] Cedric Yen-Yu Lin, Han-Hsuan Lin, „Upper bounds on quantum query complexity inspired by the Elitzur-Vaidman bomb tester," in *Proceedings of the 18th Conference on Quantum Information Processing.* Jan, 2015. arXiv:abs/1410.0932v2.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC)*. May, 1996. arXiv:quant-ph/9605043.

[5] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM Journal on Computing 26 no. 5*, (1997) 1510–1523, arXiv:quant-ph/9701001.

[6] J. von Neumann, „Mathematical Foundations of Quantum Theory and Measurement," Princeton, NJ (1983).

[7] P. Kwiat, H. Weinfurter, T. Herzog, A. Zeilinger, and M. A. Kasevich, "Interaction-free measurement," *Physical Review Letters 74 no. 24*, (1995) 4763.

[8] B. Misra and E. C. G. Sudarshan, "The Zeno's paradox in quantum theory," *Journal of Mathematical Physics 18 no. 4*, (1977) 756.

[9] François Le Gall, „Improved Quantum Algorithm for Triangle Finding via Combinatorial Arguments," in *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science (FOCS)*. 2014. arXiv:abs/1407.0085v2.

DOKUMENTĀRĀ LAPA

Bakalaura darbs „Algoritmu sarežģītības novērtējumi bumbas meklēšanas modelī” izstrādāts LU Datorikas fakultātē.

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

Autors:

Ēriks Gopaks _____ ___.___.2015.

Rekomendēju / nerekomendēju darbu aizstāvēšanai

Vadītājs:

profesors Dr. sc. comp. *Andris Ambainis* _____ ___.___.2015.

Recenzents: profesors Dr. sc. comp. *Juris Vīksna*

Darbs iesniegts Datorikas fakultātē ___.___.2015.

Dekāna pilnvarotā persona:

vecākā metodiķe *Ārija Sproģe* _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē

___.___.2015. prot. Nr. _____-.

Komisijas sekretārs(-e):

_____