Математичка гимназија

Матурски рад
из физике

Модификација Гроверовог алгоритма

Ученик
Михаило Јанчевић, IVд

Ментор
др Игор Салом

Београд, *27.5.2024.*

Садржај

1 Увод **1**

2 Стандардан Гроверов алгоритам **3**

3 Модификација Гроверовог алгоритма **9**
3.1 Квантни део алгоритма 9
3.2 Класични део алгоритма 12

4 Примена **15**
4.1 Задатак из информатике 15

5 Закључак **19**

6 Литература **21**

1

Увод

Квантни рачунари су рачунари који, уместо класичних битова који могу да имају вредности 0 или 1, користе квантне битове (*кубиш̄е*), чије вредности могу бити било која суперпозиција ортогоналних стања $|0\rangle$ и $|1\rangle$. Поред тога што један кубит може да има те вредности, кубити у једном регистру могу бити *п̄овезани* (енг. *entangled*), па је укупно стање регистра од n кубита било који нормиран вектор из 2^n-димензионог векторског простора. Квантни рачунар може да примени било коју унитарну трансформацију на стање неког регистра, и може да изврши било које мерење.

За извршавање било ког мерења је довољно да може да се изврши мерење у бази $|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle$, а сва остала мерења могу да се реализују применом унитарног оператора који жељену базу слика у ову, потом мерењем, а онда применом инверзног оператора.

Најлакше је имплементирати унитарне трансформације на једном кубиту. Најпознатија од њих је NOT капија, једина смислена унарна логичка капија која може да се примени и на класичном рачунару:

$$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Још једна корисна унитарна трансформација за један кубит, која је, као и NOT, сама себи инверзна, је H капија, која од стања $|0\rangle$ прави стање $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, а од стања $|1\rangle$ прави $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Унитарне трансформације са више кубита је много теже имплементирати без декохеренције. Једна од таквих трансформација је $CNOT$ капија, која инвертује кубит само ако је вредност контролниог кубита 1, или прецизније, претвара стања $|10_2\rangle$ и $|11_2\rangle$ једно у друго, где је леви кубит контролни:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Постоје и вишеструке $CNOT$ капије, које инвертују кубит само уз компоненту стања у којој су вредности свих контролних кубита 1:

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{pmatrix}.$$

Што више кубита има у капији, то је теже да се она директно имплементира. Ипак, није неопходно да се ове капије директно имплементирају, јер је сваку унитарну трансформацију могуће представити као композицију унарних капија и $CNOT$ капија.

Теорема 1.1. Све што је могуће имплементирати на класичном рачунару, могуће је и на квантном рачунару.

Доказ. Операцију постављања вредности 0 или 1 на неки кубит је могуће извести тако што се његова вредност измери, а онда се примени NOT капија ако резултат није био оно што вредност тог кубита треба да постане. Ово је иреверзибилна операција коју класичан рачунар може да изведе, а све остало је реверзибилно и може да се представи преко основних логичких капија.

Ако се двострука $CNOT$ капија примени на кубит у стању $|1\rangle$ контролисана кубитима $|a\rangle$ и $|b\rangle$, стање тог кубита постаје $|a\ NAND\ b\rangle$. На основу $NAND$ капије је могуће саставити све класичне логичке капије. □

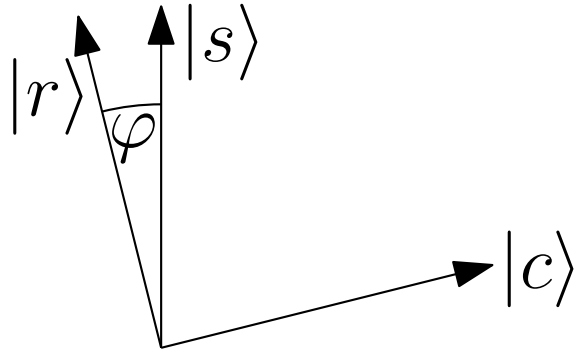
Квантни рачунари могу да ураде све што класични рачунари могу, али могу и да изведу неке ствари које (у тој временској сложености) делују незамисливо. Један добар пример тога је Гроверов алгоритам.

2

Стандардан Гроверов алгоритам

Гроверов алгоритам је један од најпознатијих квантних алгоритама. За дату функцију $f : \{0, 1, \ldots, N-1\} \to \{0, 1\}$ (тзв. *oracle* функција) коју је могуће имплементирати и која k различитих бројева слика у 1, а остале у 0, Гроверов алгоритам проналази неки од бројева који се сликају у 1 у сложености $\mathcal{O}(\sqrt{\frac{N}{k}})$. Ово може да има разне примене као што су тражење одређене вредности у меморији рачунара или тражење неког целобројног решења неке једначине.

Нека је $N = 2^n$ (ако N није степен двојке, може да се повећа до најближег степена двојке, а да се *oracle* функција допуни нулама). Нека је $|s\rangle = \frac{1}{\sqrt{N}}(|0\rangle + |1\rangle + \ldots + |N-1\rangle)$. Ово стање се добија применом H капије на сваки од n кубита. Нека је $|c\rangle = \frac{1}{\sqrt{k}}(|c_1\rangle + |c_2\rangle + \ldots + |c_k\rangle)$, где су c_1, c_2, $\ldots$, c_k бројеви које oracle функција слика у 1 (нпр. решења једначине). На слици 2.1 је приказана раван α у којој се налазе вектори облика $a\,|s\rangle + b\,|c\rangle\,, a, b \in \mathbb{R}$. Вектор $|r\rangle = \frac{1}{\sqrt{N-k}}(\sqrt{N}\,|s\rangle - \sqrt{k}\,|c\rangle) = \frac{1}{\sqrt{N-k}}(|r_1\rangle + |r_2\rangle + \ldots + |r_{N-k}\rangle)$ из те равни је суперпозиција свих бројева од 0 до $N-1$ који се сликају у 0, и ортогоналан је на $|c\rangle$.



Слика 2.1. Раван α

Теорема 2.1. Нека је $|\psi\rangle = a\,|c\rangle + b\,|r\rangle$ неко стање из равни α. Вероватноћа да се приликом мерења $|\psi\rangle$ у бази $|0\rangle$, $|1\rangle$, ..., $|N-1\rangle$ добије неки од бројева c_1, c_2, ..., c_k је иста као вероватноћа да се приликом мерења $|\psi\rangle$ у бази која садржи $|c\rangle$ добије $|c\rangle$.

Први доказ. Нека је прва поменута вероватноћа p_1, а друга p_2.

$$p_1 = \sum_{i=1}^{k} |\,\langle c_i|\psi\rangle\,|^2 = \sum_{i=1}^{k} |\,a\,\langle c_i|c\rangle + b\,\langle c_i|r\rangle\,|^2 = \sum_{i=1}^{k} |a\cdot\frac{1}{\sqrt{k}} + b\cdot 0|^2 = \sum_{i=0}^{k}\frac{a^2}{k} = a^2,$$

$$p_2 = |\,\langle c|\psi\rangle\,|^2 = |a\,\langle c|c\rangle + b\,\langle c|r\rangle\,|^2 = a^2,$$

$$p_1 = p_2.$$

□

Овакав доказ објашњава да су те две вероватноће једнаке, али не даје никакву интуицију о томе зашто је то тако. Следећи доказ показује да то није случајно, и омогућава боље разумевање ове и сличних ситуација.

Други доказ. Нека постоје два квантна регистра величине n у стању $a\,|c\rangle\,|c\rangle + b\,|r\rangle\,|r\rangle$. Леви регистар је код Стевана, а десни код Дејана. Стеван и Дејан мере вредности својих регистара у исто време, у неком референтном систему. Стеван мери у бази која садржи $|c\rangle$ и $|r\rangle$, а Дејан у бази $|0\rangle, |1\rangle, \ldots |N-1\rangle$. Та два мерења су се десила на различитим местима, а у исто време у том референтном систему, па је временско-просторни интервал између та два мерења просторног типа, и њихов поредак је немогуће дефинисати. То значи да резултат Дејановог мерења не сме да зависи од тога да ли се Стевановo мерење већ десило или није.

Кад би се Дејаново мерење десило пре Стевановог, он би мерио стање

$$a\,|c\rangle\,|c\rangle + b\,|r\rangle\,|r\rangle =$$

$$= \frac{a}{\sqrt{k}}\,|c\rangle\,|c_1\rangle + \ldots + \frac{a}{\sqrt{k}}\,|c\rangle\,|c_k\rangle + \frac{b}{\sqrt{N-k}}\,|r\rangle\,|r_1\rangle + \ldots + \frac{b}{\sqrt{N-k}}\,|r\rangle\,|r_{N-k}\rangle\,.$$

Када се то стање овако запише, сваком вектору из Дејанове базе одговара само једно стање Стевановог регистра, и ако је тај вектор резултат мерења, Стеванов регистар ће прећи у стање које је везано за њега. Због овога је вероватноћа за сваки од исхода Дејановог мерења иста као кад би мерио стање

$$\frac{a}{\sqrt{k}}\,|c_1\rangle + \ldots + \frac{a}{\sqrt{k}}\,|c_k\rangle + \frac{b}{\sqrt{N-k}}\,|r_1\rangle + \ldots + \frac{b}{\sqrt{N-k}}\,|r_{N-k}\rangle = a\,|c\rangle + b\,|r\rangle = |\psi\rangle$$

на свом регистру. Вероватноћа да при том мерењу добије неки од вектора $|c_1\rangle, |c_2\rangle, \ldots, |c_k\rangle$ је p_1.

Кад би се Стеваново мерење десило пре Дејановог, он би мерио стање

$$a\,|c\rangle\,|c\rangle + b\,|r\rangle\,|r\rangle\,.$$

Када се то стање овако запише, сваком вектору из Стеванове базе одговара највише једно стање Дејановог регистра, и ако је тај вектор резултат мерења, Дејанов регистар ће прећи у стање које је везано за њега. Због овога је вероватноћа за сваки од исхода Стевановог мерења иста као кад би мерио стање

$$a\,|c\rangle + b\,|r\rangle = |\psi\rangle$$

на свом регистру. Једина стања која он може да добије су $|c\rangle$ и $|r\rangle$, јер су сви остали вектори из његове базе ортогонални на $|c\rangle$ и $|r\rangle$, па самим тим и на целу раван α, и на вектор $|\psi\rangle$ из те равни. Вероватноћа да при том мерењу добије вектор $|c\rangle$ је p_2, и Дејанов регистар онда прелази у стање $|c\rangle$, а вероватноћа да добије вектор $|r\rangle$ је $1 - p_2$, и Дејанов регистар онда прелази у стање $|r\rangle$. Сада се, после Стевановог мерења, дешава и Дејаново, и он мери стање $|c\rangle$ или $|r\rangle$, у зависности од резултата Стевановог мерења. Ако Дејан мери $|c\rangle$, онда сигурно добија неки од бројева $c_1, c_2, \ldots, c_k$, а ако мери $|r\rangle$, сигурно не добија ниједан од бројева $c_1, c_2, \ldots, c_k$. Вероватноћа да Дејан при свом мерењу добије неки од бројева $c_1, c_2, \ldots, c_k$ је, по теореми тоталне вероватноће, једнака $p_2 \cdot 1 + (1 - p_2) \cdot 0 = p_2$.

Пошто је немогуће одредити да ли се Стеваново мерење десило пре Дејановог или не, вероватноћа да Дејан добије неки од бројева $c_1, c_2, \ldots, c_k$ не сме да зависи од тога. Дакле, $p_1 = p_2$. □

Циљ Гроверовог алгоритма је да добије неки од бројева $c_1, c_2, \ldots c_k$ мерењем стања система које је у равни α, па то стање треба да буде што ближе вектору $|c\rangle$. То се постиже ротацијом. Пошто није познато шта је вектор $|c\rangle$ на почетку, једини вектор за који је познато да се налази у равни α је $|s\rangle$, тако да се за почетно стање бира $|\psi_0\rangle = |s\rangle$. Ротација се постиже као композиција две рефлексије, око $|r\rangle$ и око $|s\rangle$.

Оператор који представља рефлексију око $|r\rangle$ се реализује на следећи начин: на регистар се додаје један кубит у стању $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Сада се на регистар и тај додатан кубит примењује оператор U_f који ради следеће:

$$U_f\,|\psi\rangle\,|x\rangle = |\psi\rangle\,|x \oplus f(\psi)\rangle$$

Ако је овај оператор технички могуће имплементирати, онда Гроверов алгоритам може да функционише. Он подсећа на $CNOT$ капију примењену на

додатни кубит, контролисану кубитом $|f(\psi)\rangle$, али се разликује јер не захтева још један кубит који би остао у стању $|f(\psi)\rangle$. Важиће:

$$U_f |r_i\rangle |-\rangle = |r_i\rangle |-\rangle ,$$

$$U_f |r\rangle |-\rangle = \sum_{i=1}^{N-k} \frac{1}{\sqrt{N-k}} U_f |r_i\rangle |-\rangle = \sum_{i=1}^{N-k} \frac{1}{\sqrt{N-k}} |r_i\rangle |-\rangle = |r\rangle |-\rangle ,$$

$$U_f |c_i\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |c_i\rangle \frac{1}{\sqrt{2}}(|1\rangle - |0\rangle) = |c_i\rangle \frac{-1}{\sqrt{2}}(|0\rangle - |1\rangle) = - |c_i\rangle |-\rangle ,$$

$$U_f |c\rangle |-\rangle = \sum_{i=1}^{k} \frac{1}{\sqrt{k}} U_f |c_i\rangle |-\rangle = \sum_{i=1}^{k} \frac{-1}{\sqrt{k}} |c_i\rangle |-\rangle = - |c\rangle |-\rangle .$$

Након примене оператора U_f на вектор $a|r\rangle + b|c\rangle$ из равни α и додатни кубит у стању $|-\rangle$, додатни кубит ће остати у стању $|-\rangle$, а регистар ће прећи у стање $a|r\rangle - b|c\rangle$, односно рефлектоваће се у односу на вектор $|r\rangle$.

Сада треба конструисати оператор који рефлектује вектор у односу на $|s\rangle$. Један од начина да се то уради је композицијом следећих оператора:

1. H капија и NOT капија примењене на свих n кубита, што стање $|s\rangle$ слика у $|N-1\rangle$.

2. Вишеструка $CNOT$ капија на додатан кубит у стању $|-\rangle$ контролисана са свих n кубита. Ово слика $|N-1\rangle|-\rangle$ у $-|N-1\rangle|-\rangle$, а $|b\rangle|-\rangle$ у $|b\rangle|-\rangle$ за свако b из $\{0, 1, \ldots N-2\}$. Дакле, то не мења стање додатног кубита, а множи са -1 компоненту стања регистра уз $|N-1\rangle$.

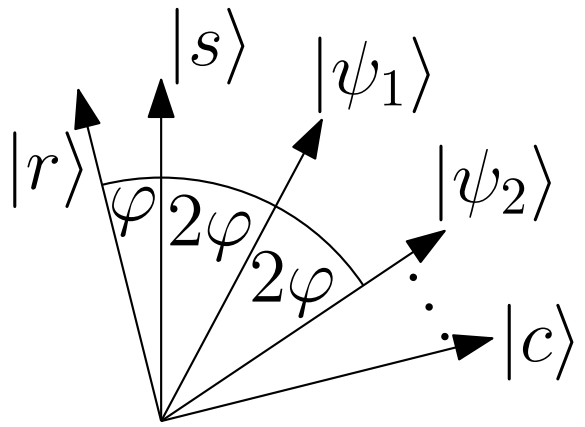
3. NOT капија и H капија примењене на свих n кубита, што стање $|N-1\rangle$ слика назад у $|s\rangle$.

Овај оператор, ако се на њега дода множење са -1, рефлектује стање у односу на $|s\rangle$. То множење са -1 није неопходно јер мења само фазни фактор који је немогуће детектовати.

Нека је φ угао између $|r\rangle$ и $|s\rangle$. Онда је:

$$\sin\varphi = \cos\left(\frac{\pi}{2} - \varphi\right) = \langle s|c\rangle = \sum_{i=1}^{k} \frac{1}{\sqrt{N}} \frac{1}{\sqrt{k}} \langle c_i|c_i\rangle = \sqrt{\frac{k}{N}}.$$

Композиција рефлексије око $|r\rangle$ и рефлексије око $|s\rangle$ је ротација за угао 2φ према вектору $|c\rangle$. Оператор који представља ту ротацију се зове *Гроверов оператор*.



Слика 2.2. Геометријски приказ Гроверовог алгоритма

Након m понављања Гроверовог оператора, за тренутно стање $|\psi_m\rangle$ важи:

$$\angle(|s\rangle, |\psi_m\rangle) = 2m\varphi \implies \angle(|c\rangle, |\psi_m\rangle) = \left|\frac{\pi}{2} - (2m+1)\varphi\right|.$$

Због теореме 2.1, потребно је да угао између $|c\rangle$ и $|\psi_m\rangle$ у тренутку мерења буде што мањи, па m треба да буде што ближе $\frac{\pi}{4\varphi} - \frac{1}{2}$. Дакле, најбоље је применити Гроверов оператор

$$m = \left\lfloor \frac{\pi}{4\varphi} \right\rfloor = \left\lfloor \frac{\pi}{4 \arcsin\sqrt{\frac{k}{N}}} \right\rfloor \leqslant \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{k}} \right\rfloor$$

пута.

Угао између $|c\rangle$ и $|\psi_m\rangle$ је у тренутку мерења највише φ, па је вероватноћа за грешку:

$$1 - |\langle c|\psi_m\rangle|^2 = 1 - \cos^2\angle(|c\rangle, |\psi_m\rangle) = \sin^2\angle(|c\rangle, |\psi_m\rangle) \leqslant \sin^2\varphi = \frac{k}{N}.$$

Иако вероватноћа за грешку постоји, могуће је проверити резултат алгоритма (покретањем *oracle* функције за добијени резултат), и ако је вероватноћа за грешку мања или једнака $\frac{1}{2}$, очекивани број понављања алгоритма док не дође до тачног резултата неће бити већи од 2, па се сложеност не мења због понављања. Претходно ограничење гарантује да ће вероватноћа грешке заиста бити мања или једнака $\frac{1}{2}$ ако је $k \leqslant \frac{N}{2}$. Случај кад је $k > \frac{N}{2}$ треба посебно разматрати. У том случају важи:

$$k > \frac{N}{2} \implies \sin\varphi = \sqrt{\frac{k}{N}} > \frac{1}{\sqrt{2}} \implies \varphi > \frac{\pi}{4} \implies m = \left\lfloor \frac{\pi}{4\varphi} \right\rfloor = 0$$

Стање које се мери је $|\psi_m\rangle = |\psi_0\rangle = |s\rangle$ и резултат тог мерења је насумичан број од 0 до $N-1$. Дакле, у овом случају се Гроверов алгоритам своди на погађање насумичног броја, а вероватноћа да погоди неки од бројева које *oracle* функција слика у 1 ће заиста бити већа од $\frac{1}{2}$ јер је $k > \frac{N}{2}$.

3

Модификација Гроверовог алгоритма

Једно од битних ограничења Гроверовог алгоритма је то што захтева да k буде познато. Такође, он проналази било које решење, што није увек корисно. Ове препреке могу да се превазиђу тако што се алгоритам модификује. У овом поглављу ће бити описана модификација Гроверовог алгоритма која за непознато k проналази **најмањи** број који *oracle* функција слика у 1.

Сложеност овог алгоритма је $\mathcal{O}\left(\sqrt{N}\log\frac{1}{p}\right)$, где је p вероватноћа за грешку. Разлог што овде вероватноћа грешке улази у сложеност а код стандардног Гроверовог алгоритма не улази је то што овде, за разлику од стандардног Гроверовог алгоритма, није могуће лако проверити резултат. Пошто алгоритам налази прво појављивање броја 1 у *oracle* функцији, провера би захтевала одређивање са сигурношћу да ли постоји вредност 1 пре те позиције, а сложеност те провере је већа од сложености овог алгоритма. То, дакле, није мана алгоритма, него нешто што би се појавило у сваком пробабилистичком решењу овог проблема.

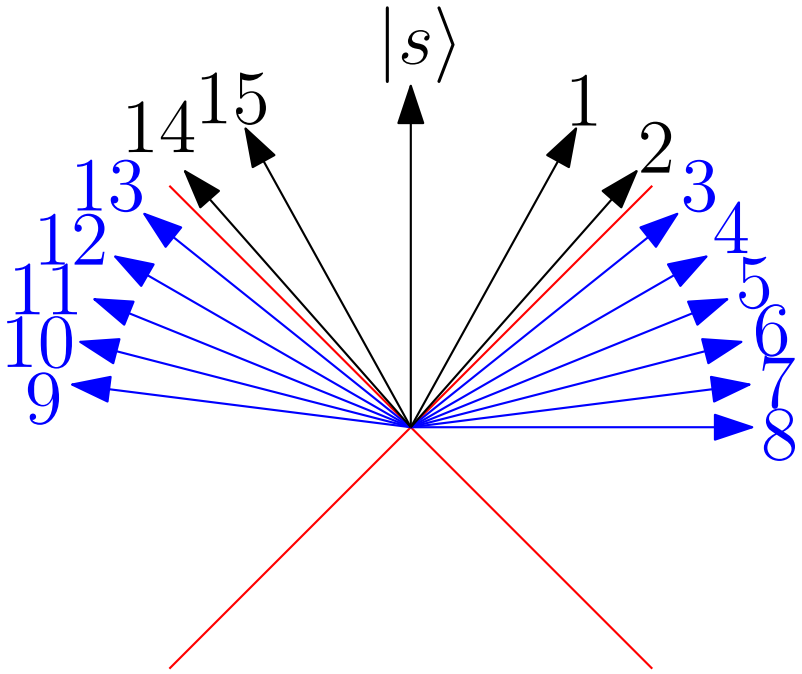
Овај алгоритам се састоји из два дела: квантног и класичног.

3.1 Квантни део алгоритма

Овај део алгоритма треба да, у сложености $\mathcal{O}\left(\sqrt{N}\right)$, врати вредност 0 или 1, и то:

- Ако не постоји вредност коју *oracle* функција слика у 1, резултат је сигурно 0;
- Ако постоји бар једна вредност коју *oracle* функција слика у 1, постоји вероватноћа која је најмање $\frac{1}{2}$ да резултат буде 1.

Први корак овог дела алгоритма је проверавање једне (нпр. прве) вредности *oracle* функције. Ако је та вредност 1, одмах се враћа резултат 1. У наставку овог дела алгоритма се сматра да је добијена вредност била 0, што значи да нису све вредности *oracle* функције 1, па је $k \in \{0, 1, \ldots N-1\}$. Сада се регистар поставља у стање $|s\rangle$, и на њега се примењује Гроверов оператор једном. Добијено стање је ротирано у односу на $|s\rangle$ за неки од N могућих углова из интервала $[0, \pi)$ (тај угао зависи од угла између $|r\rangle$ и $|s\rangle$, што зависи само од k). Тај угао ће бити 0 само ако је $k = 0$ (на почетку је елиминисан случај $k = N$, јер би тада угао исто био 0). На слици 3.1 су приказана могућа добијена стања у равни α, при чему су стања која су ротирана за више од $\frac{\pi}{2}$ помножена са -1, што мења само глобалну фазу, тако да ће све бити исто као да су тамо где би требало да буду.



Слика 3.1. Могућа стања, у зависности од k, након једне примене Гроверовог оператора (за случај $N = 16$).

На овој слици се види да ови вектори нису равномерно распоређени. Њихова расподела је гушћа тамо где је угао између њих и $|s\rangle$ близу $\frac{\pi}{2}$. И стандардан Гроверов алгоритам и ова модификација се базирају управо на тој чињеници. Када би ова стања била равномерно распоређена, број ротација потребан да стање достигне одређени угао у односу на $|s\rangle$ би био $\mathcal{O}\left(\frac{N}{k}\right)$, што није брже од насумичног тражења броја.

Ова модификација се наставља тако што се ово стање добијено једном ротацијом мери у бази која садржи $|s\rangle$. То се постиже тако што се прво на

свих n кубита примени H капија, што $|s\rangle$ претвара у $|0\rangle$, а потом се извршава мерење. Стања чији угао у односу на $|s\rangle$ је између $\frac{\pi}{4}$ и $\frac{3\pi}{4}$ (тај интервал углова је оивичен црвеним линијама на слици 3.1) имају вероватноћу мању од $\frac{1}{2}$ да при овом мерењу дају резултат $|s\rangle$ (односно $|0\rangle$).

Ако при мерењу није добијен $|s\rangle$, враћа се резултат 1, јер то не би било могуће за $k = 0$.

Ако је добијен $|s\rangle$, то значи да је све могућности које су имале шансу мању од $\frac{1}{2}$ да дају $|s\rangle$ (обојене плавом бојом бојом на слици 3.1) могуће одбацити са вероватноћом барем $\frac{1}{2}$.

Сада је стање регистра поново $|0\rangle$ (јер би иначе овај део алгоритма већ био завршен са резултатом 1), па се применом H капије на свих n кубита поново постиже стање $|s\rangle$. У следећем кораку се Гроверов оператор примењује 3 пута више него претходни пут. Стања која нису одбачена су малопре била под углом мањим од $\frac{\pi}{4}$ у односу на $|s\rangle$, па су сада сва разматрана стања под углом мањим од $\frac{3\pi}{4}$ у односу на $|s\rangle$, јер су ротирана за 3 пута већи угао. Мерењем се поново долази или до тога да треба вратити резултат 1, или до тога да треба одбацити стања из интервала углова $\left[\frac{\pi}{4}, \frac{3\pi}{4}\right]$.

Овај процес се наставља, сваки пут примењујући Гроверов оператор 3 пута више него претходни пут, све док не буду одбачене све могућности осим $k = 0$, а онда се враћа резултат 0 (осим ако је процес у неком тренутку прекинут враћајући резултат 1). За сваки број из скупа $\{1, 2, \ldots N-1\}$ је постојао корак када је вероватноћа за добијање $|s\rangle$ била мања од $\frac{1}{2}$ ако k има ту вредност, али је $|s\rangle$ ипак добијен, па је вероватноћа да је алгоритам стигао до овог дела за ту вредност k мања од $\frac{1}{2}$. По теореми тоталне вероватноће, вероватноћа да је алгоритам стигао до овог дела је мања од $\frac{1}{2}$ ако k није 0, односно ако постоји број који *oracle* функција слика у 1.

Дакле, ако нема броја који се слика у 1, алгоритам заиста увек враћа 0, а ако га има, алгоритам заиста има вероватноћу већу од $\frac{1}{2}$ да врати 1.

У последњем кораку овог дела алгоритма, Гроверов оператор се примењује онолико пута колико је потребно да и вектори за $k = 1$ и $k = N - 1$ буду под углом већим од $\frac{\pi}{4}$ у односу на $|s\rangle$. Познато је из претходног поглавља да је за те две вредности:

$$k = 1 \implies \varphi = \arcsin \frac{1}{\sqrt{N}},$$

$$k = N - 1 \implies \varphi = \arcsin \sqrt{\frac{N-1}{N}} = \arccos \sqrt{1 - \left(\sqrt{\frac{N-1}{N}}\right)^2} = \arccos \frac{1}{\sqrt{N}}.$$

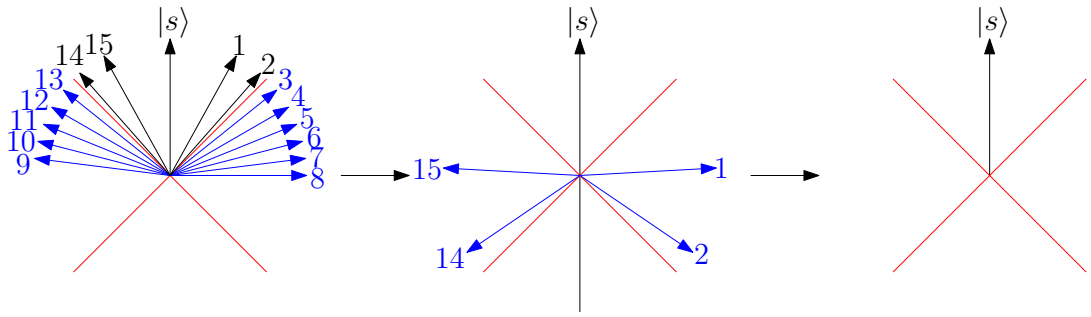
За $k = 1$, угао између стања и $|s\rangle$ након једне ротације је $2\varphi = 2 \arcsin \frac{1}{\sqrt{N}}$, а за $k = N - 1$, тај угао је $\pi - 2\varphi = 2(\frac{\pi}{2} - \arccos \frac{1}{\sqrt{N}}) = 2 \arcsin \frac{1}{\sqrt{N}}$. Број итерација

Гроверовог оператора у последњем кораку је:

$$m = \left\lceil \frac{\frac{\pi}{4}}{2 \arcsin \frac{1}{\sqrt{N}}} \right\rceil = \left\lceil \frac{\pi}{8 \arcsin \frac{1}{\sqrt{N}}} \right\rceil \leqslant \left\lceil \frac{\pi}{8} \sqrt{N} \right\rceil .$$

Укупан број коришћења Гроверовог оператора у свим корацима је:

$$m + \frac{m}{3} + \frac{m}{9} + \ldots \leqslant \left\lceil \frac{\pi}{8} \sqrt{N} \right\rceil \sum_{i=0}^{\infty} \frac{1}{3^i} = \frac{3}{2} \left\lceil \frac{\pi}{8} \sqrt{N} \right\rceil .$$



Слика 3.2. Квантни део алгоритма за $N = 16$.

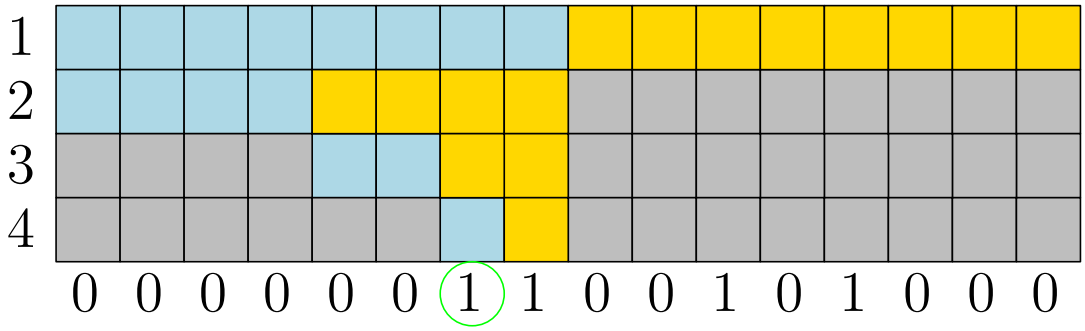
Ово не изгледа толико корисно када се примењује на цео интервал $[0, N-1]$, али не мора да се примењује само на њега. $|s\rangle$ за интервал који чине бројеви којима је првих неколико битова фиксирано на одређене вредности је могуће добити тако што се ти кубити фиксирају, а на остале се примени H капија. Онда и Гроверов оператор може да се направи слично као за цео интервал, тако што првих неколико кубита увек остане фиксирано, а одговарајући оператори се примењују на остале.

3.2 Класични део алгоритма

Класични део овог алгоритма користи квантни део више пута за различите интервале да би бинарном претрагом пронашао прво појављивање броја 1. Нека је тражена вероватноћа грешке $p = \frac{1}{2^x}$. Први корак овог алгоритма је примењивање квантног дела $x + 1$ пута на скуп бројева којима је први бит 0, односно на прву половину целог интервала. Ако је резултат некад био 1, постоји решење у првој половини интервала, и први кубит до краја остаје 0, а ако је резултатат увек био 0, сматра се да не постоји решење у првој половини интервала, па се први кубит до краја фиксира на 1 јер се надаље разматра само друга половина. Даље се, слично овоме, у m-том кораку први

нефиксиран кубит фиксира на 0, квантни део се покреће $x + m$ пута, и у зависности од тога да ли је некада резултат био 1 или није, тај кубит до краја остаје 0 или до краја постаје 1. Кад сви кубити буду фиксирани, добијени број ће бити најмањи број који *oracle* функција слика у 1 (ако је резултат алгоритма $N - 1$, треба проверити и вредност *oracle* функције за тај број, и ако је резултат 0 онда заправо нема бројева који се сликају у 1 на целом интервалу).

На слици 3.3 је приказан пример класичног дела алгоритма. Плава поља су интервал који се проверава у тренутном кораку, а сива она која се више не разматрају.



Слика 3.3. Класични део алгоритма за $N = 16$

У m-том кораку се квантни део примењује $x + m$ пута, па је вероватноћа за грешку мања од $\frac{1}{2^{x+m}}$. Вероватноћа да било где дође до грешке је мања од збира тих вероватноћа, а тај збир је

$$\sum_{m=1}^{n} \frac{1}{2^{x+m}} < \frac{1}{2^x} = p.$$

У m-том кораку се проверава интервал дужине $\frac{N}{2^m}$, $x + m$ пута, а за то је потребно применити Гроверов оператор највише $\frac{3}{2}(x+m)\left\lceil \frac{\pi}{8}\sqrt{\frac{N}{2^m}} \right\rceil \leqslant \frac{3}{2}(x+m) + \frac{3\pi}{16}(x+m)\sqrt{\frac{N}{2^m}}$ пута. Укупан број примена Гроверовог оператора је мањи од:

$$\sum_{m=1}^{n} \left(\frac{3}{2}(x+m) + \frac{3\pi}{16}(x+m)\sqrt{\frac{N}{2^m}} \right) < \frac{3}{2}nx + \frac{3}{4}n(n+1) + \frac{3\pi}{16}\sqrt{N}\sum_{m=1}^{\infty}\frac{x+m}{\sqrt{2^m}} =$$

$$= \frac{3}{4}(1 + 2\log_2\frac{1}{p})\log_2 N + \frac{3}{4}\log_2^2 N + \frac{(3+3\sqrt{2})\pi}{16}\sqrt{N}\left(2 + \sqrt{2} + \log_2\frac{1}{p}\right) =$$

$$= \mathcal{O}\left(\sqrt{N}\log\frac{1}{p}\right).$$

4

Примена

Проблеми које Гроверов алгоритам и његове модификације решавају обично делују као да их није могуће решити у сложености мањој од $\mathcal{O}(N)$. Зато ови алгоритми делују веома моћно на први поглед, али њихове примене нису толико честе. На пример, проблем тражења првог појављивања неке вредности у несортираном низу, који решава описана модификација Гроверовог алгоритма, је могуће решити у сложености $\mathcal{O}(\log N)$ ако се претходно направи сортирана копија низа у $\mathcal{O}(N \log N)$. Тешко је наћи класичан задатак из информатике који се брже решава тим алгоритмом, јер обично буде довољно времена за један пролазак кроз низ, а прављење сортиране копије није много споријe од тога. У наставку је благо промењен задатак са једног такмичења из информатике, и његово решење уз помоћ модификованог Гроверовог алгоритма.

4.1 Задатак из информатике

Задатак. Дати су природни бројеви N, K, T, и Q, низ a дужине N и низови l и r дужине Q. Вукашин се жени. Он има N бомбона, од којих свака припада једном од T типова бомбона. Те бомбоне су поређане у низ, и тип бомбоне i је a_i. Вукашин има K пријатеља, и жели да позове што више њих на свадбу. Сваком госту ће дати све бомбоне из неког интервала. Пријатељ ће се љутити ако постоји тип бомбоне који је неко други добио, а он није (пријатељи који нису позвани се не љуте, јер су мислили да се Вукашин шалио за свадбу). У i-том од Q упита, потребно је одредити који је највећи број пријатеља које Вукашин може да позове на свадбу и да им подели бомбоне из интервала $[l_i, r_i]$, тако да све буду подељене, а да се ниједан пријатељ не наљути.[3]

Решење. У решењу ће бити коришћена модификација Гроверовог алгоритма која је описана раније, за налажење првог појављивања одређене вредности у

неком интервалу у низу a. Да би то било могуће, потребна је *oracle* функција која за то служи. Ако алгоритам тражи вредност X у интервалу $[L, R]$, онда је:

$$f(i) = \begin{cases} 1, & a_i = X \wedge i \in [L, R] \\ 0, & \text{иначе} \end{cases}$$

Ову функцију је лако имплементирати на класичном рачунару у сложености $\mathcal{O}(1)$, па је, по теореми 1.1, то могуће и на квантном рачунару. У наставку, $\mathcal{G}(X, L, R, P)$ ће бити ознака за примену модификације Гроверовог алгоритма за тражење првог појављивања вредности X у интервалу $[L, R]$, са вероватноћом за грешку највише P.

Постојање Q упита омогућава примену одређених спорих операција за припрему пре одговарања на упите, што значајно помаже класичном решењу овог проблема, али за квантно решење није неопходна било каква припрема. У наставку је алгоритам који одговара на i-ти упит:

Прво се примењује $\mathcal{G}(t, l_i, r_i, p)$, за сваки тип бомбона t (p је вероватноћа чија вредност ће касније бити уведена). Они типови бомбона за које је резултат да се уопште не појављују у интервалу $[l_i, r_i]$ неће бити разматрани до краја овог упита, јер њих ниједан пријатељ неће добити. Сваки тип бомбона који се појављује у том интервалу ће бар један пријатељ добити, па тај тип морају да добију сви, да се не би љутили. Такве типове бомбона зовемо *значајни*. Услов да сваки пријатељ мора да добије бар једну бомбону од сваког значајног типа је еквивалентан услову да не постоји тип који неко добије, а неко други не.

Нека је пријатељ P_1 онај који добија бомбону са позиције l_i. Нека је x_1 највећи број међу бројевима који су добијени као резултати примена $\mathcal{G}(t, l_i, r_i, p)$ за значајне типове бомбона j. P_1 ће добити бомбоне из интервала $[l_i, x_1]$. То је у реду јер се у том интервалу налази прво појављивање сваког значајног типа бомбона, што значи да се сваки тип бомбона ту појављује. Такође, тај интервал није могао да буде краћи, јер се онда у њему не би налазила ниједна бомбона оног типа који се први пут појављује на позицији x_1. Интервал за P_1 је могао да буде дужи, али то није потребно из следећег разлога:

Нека је P_1 добио све бомбоне из интервала $[l_i, c]$, за неко $c > x_1$, а пријатељ P_2 из интервала $[c + 1, x_2]$ у неком оптималном решењу. Ако бомбоне из интервала $[x_1 + 1, c]$ пребацимо пријатељу P_2, пријатељ P_1 и даље има све значајне типове (јер му је остао интервал $[l_i, x_1]$), и P_2 и даље има све значајне типове (јер их је већ имао, а сада је само добио још бомбона). Дакле, решење у ком P_1 добија све бомбоне из интервала $[l_i, x_1]$, а P_2 из интервала $[x_1 + 1, x_2]$ је такође валидно и храни једнак број пријатеља, па је и оно оптимално.

Дакле, ако још неки пријатељ осим P_1 добије неке бомбоне, сигурно постоји оптимално решење у ком P_1 добија интервал $[l_i, x_1]$, па ћемо сматрати да је добио тај интервал. Сада се примењује $\mathcal{G}(t, x_1 + 1, r_i, p)$ за све значајне

типове t, и x_2 се дефинише као највећи од резултата тих алгоритама (ако ниједан од њих није рекао да неки значајни тип више не постоји). Сличном аргументацијом као малопре долазимо до тога да, ако је P_2 пријатељ који добија бомбону $x_1 + 1$, постоји оптимално решење у ком он добија тачно интервал $[x_1 + 1, x_2]$.

Овај поступак (у m-том кораку, то је позивање $\mathcal{G}(t, x_{m-1}+1, r_i, p)$, дефинисање x_m као највећег од резултата, и давање пријатељу P_m бомбоне из интервала $[x_{m-1} + 1, x_m]$) се наставља све док се не деси један од два критеријума заустављања:

1. Приликом примене $\mathcal{G}(t, x_m + 1, r_i, p)$ се добија као резултат да неки значајан тип бомбона не постоји више у том интервалу. Овде се зауставља јер следећи пријатељ не би добио тај тип бомбона.

2. $m = K$, односно сви пријатељи су добили свој интервал.

Након заустављања, интервал последњег пријатеља се проширује до краја интервала $[l_i, r_i]$ (нико се неће наљутити јер је он добио те додатне бомбоне). Сада знамо који је одговор на овај упит, а знамо и како је могуће распоредити бомбоне (иако се то не тражи у задатку).

Приликом једног корака претходног поступка (тај корак је одређивање интервала за једног пријатеља), $\mathcal{G}$ се користи највише T пута, јер толико има типова. Током одговора на упит, $\mathcal{G}$ се користи највише $T \cdot K$ пута, јер постоји K пријатеља. Током целог алгоритма, $\mathcal{G}$ се користи највише $T \cdot K \cdot Q$ пута, јер има Q упита. Нека је p_u укупна жељена граница вероватноће грешке. Вероватноћа грешке сваке примене $\mathcal{G}$ је p, па је укупна вероватноћа грешке мања од $p \cdot T \cdot K \cdot Q$, и треба изабрати вредност $p = \frac{p_u}{TKQ}$ за сваку примену $\mathcal{G}$. Дужина интервала $[l_i, r_i]$ је највише N, па је сложеност сваке примене $\mathcal{G}$ највише $\mathcal{O}\left(\sqrt{N}\log\frac{1}{p}\right) = \mathcal{O}\left(\sqrt{N}\left(\log\frac{1}{p_u} + \log(TKQ)\right)\right)$. Пошто се $\mathcal{G}$ примењује највише $T \cdot K \cdot Q$ пута, укупна сложеност алгоритма је $\mathcal{O}\left(\sqrt{N} \cdot TKQ\left(\log(TKQ) + \log\frac{1}{p_u}\right)\right)$. □

У овом решењу се јасно види зашто је важно да вероватноћа грешке буде наведена у сложености: Понављање алгоритма TKQ пута траје приближно $TKQ\log(TKQ)$ пута дуже (за довољно велику вредност TKQ), а не TKQ пута, као што би био случај са детерминистичким алгоритмом. Ово се десило јер је максималан број понављања од почетка био познат, па је вероватноћа грешке на почетку могла да буде постављена на одговарајућу вредност, а ситуација би била још гора да тај број није био познат. Ако је потребно применити алгоритам m пута, где m није на почетку познато, један од начина да се спречи да вероватноћа грешке расте је да се вероватноћа при сваком

понављању преполови, али онда би укупна сложеност била

$$\mathcal{O}\left(\sum_{i=0}^{m-1}\sqrt{N}\log\frac{2^i}{p}\right)=\mathcal{O}\left(\sum_{i=0}^{m-1}\sqrt{N}\left(i+\log\frac{1}{p}\right)\right)=\mathcal{O}\left(m\sqrt{N}\left(m+\log\frac{1}{p}\right)\right).$$

Дакле, ако се неки детерминистички алгоритам понови m пута, то ће трајати m пута дуже, а пробабилистички алгоритам ће трајати $m\log m$ или m^2 пута дуже, у зависности од тога да ли је m унапред познато или не.

Сложеност званичног (класичног) решења поменутог задатка из информатике је $\mathcal{O}(N\log N+Q\log N)$. То што један од сабирака не зависи од Q показује да то решење ради одређену припрему пре одговарања на упите. Ограничења за вредности N, K, T и Q су једнака у оригиналном задатку, и тада је то класично решење знатно боље од описаног квантног решења, али то је очекивано јер је задатак састављен да се тако решава. Међутим, за нека другачија ограничења (на пример, ако је N много веће од K, T и Q), квантно решење ће заиста бити брже од било ког решења које класичан рачунар може да изведе.

5

Закључак

На почетку овог рада је представљен један познати квантни алгоритам. Затим је осмишљена његова модификација, која представља комбинацију класичног и квантног алгоритма, уз доказ временске сложености. Та модификација може имати ширу примену у односу на оригинални алгоритам, што је и приказано применом на један задатак из информатике.

Овом приликом се захваљујем свом ментору др Игору Салому на афирмативном уводу у свет квантне механике.

6

Литература

[1] *IBM - Quantum Computing*
`https://www.ibm.com/topics/quantum-computing`

[2] Grover, Lov K. (1996-07-01). "A fast quantum mechanical algorithm for database search".*Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96.* Philadelphia, Pennsylvania, USA: Association for Computing Machinery. pp. 212–219

[3] 5. задатак са међународне Жаутиковске олимпијаде из информатике 2024. године
`https://codeforces.com/blog/entry/125363`