Laboratorinis darbas: Groverio kvantinio paieškos algoritmo realizavimas ir rezultatai

R. Čiegis

Vilniaus Gedimino technikos universitetas
e-mail: rc@vgtu.lt

Rugsėjo 30 d., 2025, Vilnius

Prieš pradedant vykdyti dirbtuvių užduotis, priminsime pagrindines kvantinių algoritmų dedamąsias dalis ir apibrėžimus.

Prieš pradedant vykdyti dirbtuvių užduotis, priminsime pagrindines kvantinių algoritmų dedamąsias dalis ir apibrėžimus.

- Kubitas yra tiesinės vektorinės erdvės virš kompleksinių skaičių elementas $|\psi\rangle \in \mathbb{C}^2$.

  Jį užrašome kaip tiesinę kombinaciją (superpoziciją) dviejų bazinių būsenų (Dirako *ket* – vektorių):

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle ,$$

  čia $\alpha_j \in \mathbb{C}$ ir tenkina normavimo sąlygą:

$$|\alpha_0|^2 + |\alpha_1|^2 = 1.$$

- n kubitų sistema apibrėžiama panaudojant tenzorinę bazinių vektorių sandaugą, taigi laisvės laipsnių skaičius yra lygus 2^n.

- n kubitų sistema apibrėžiama panaudojant tenzorinę bazinių vektorių sandaugą, taigi laisvės laipsnių skaičius yra lygus 2^n.

- n kubitų sistema apibrėžiama panaudojant tenzorinę bazinių vektorių sandaugą, taigi laisvės laipsnių skaičius yra lygus 2^n. Bazinių vektorių sistema yra:

$$\{ |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle \}, \quad \psi_k \in \{0, 1\}.$$

- n kubitų sistema apibrėžiama panaudojant tenzorinę bazinių vektorių sandaugą, taigi laisvės laipsnių skaičius yra lygus 2^n. Bazinių vektorių sistema yra:

$$\{\, |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle \,\}, \quad \psi_k \in \{0, 1\}.$$

Sistemos būseną $|\psi\rangle$ išreiškiame bazinių *ket*-vektorių tiesine kombinacija

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j \, |j\rangle \,, \quad \alpha_j \in \mathbb{C},$$

čia $|j\rangle = \left|\psi_0^j \, \psi_1^j \cdots \psi_{n-1}^j\right\rangle$.
Koeficientai $\alpha_j \in \mathbb{C}$ ir tenkina normavimo sąlygą:

$$\sum_{j=0}^{2^n-1} |\alpha_j|^2 = 1.$$

- Jeigu atliksime būsenos $|\psi\rangle$ matavimą, tai su tikimybe $|\alpha_k|^2$ gausime reikšmę, atitinkančią vektorių $|k\rangle$.

Po matavimo pasikeičia elemento būsena

$$|\psi\rangle = |k\rangle\,.$$

Kvantinis lygiagretumas

Imkime kvantinę būseną

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle .$$

Kvantinis lygiagretumas

Imkime kvantinę būseną

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle .$$

Tegul U yra tiesinis operatorius, tada

$$U|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j U|j\rangle .$$

Kvantinis lygiagretumas

Imkime kvantinę būseną

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j |j\rangle .$$

Tegul U yra tiesinis operatorius, tada

$$U|\psi\rangle = \sum_{j=0}^{2^n-1} \alpha_j U|j\rangle .$$

Per vieną taktą apskaičiuojame 2^n operatoriaus U vaizdus su visais baziniais vektoriais.

Skaičiuojame bazinių vektorių vaizdų tiesinę kombinaciją, o ne tiesinės kombinacijos vaizdą.

Kvantinių algoritmų ribojimai. Operatoriai

Kvantiniuose algoritmuose naudojami ermitiniai (*Hermitian*) operatoriai

$$U^\dagger = U, \quad U^\dagger U = I$$

.

Kvantinių algoritmų ribojimai. Operatoriai

Kvantiniuose algoritmuose naudojami ermitiniai (*Hermitian*) operatoriai

$$U^\dagger = U, \quad U^\dagger U = I$$

. Ermitiškai jungtinis operatorius $U^\dagger$ apibrėžiamas taip:

$$\langle\psi| U |\phi\rangle = \langle\phi| U^\dagger |\psi\rangle^* .$$

Kvantinių algoritmų ribojimai. Operatoriai

Kvantiniuose algoritmuose naudojami ermitiniai (*Hermitian*) operatoriai

$$U^\dagger = U, \quad U^\dagger U = I$$

. Ermitiškai jungtinis operatorius $U^\dagger$ apibrėžiamas taip:

$$\langle\psi| U |\phi\rangle = \langle\phi| U^\dagger |\psi\rangle^* .$$

Tiesinius operatorius galime susieti su matricomis $M_U = (m_{kj}), 0 \leq k, j \leq 2^n - 1$

$$m_{kj} = \langle k| U |j\rangle .$$

Kvantinių algoritmų ribojimai. Operatoriai

Kvantiniuose algoritmuose naudojami ermitiniai (*Hermitian*) operatoriai

$$U^\dagger = U, \quad U^\dagger U = I$$

. Ermitiškai jungtinis operatorius $U^\dagger$ apibrėžiamas taip:

$$\langle\psi| U |\phi\rangle = \langle\phi| U^\dagger |\psi\rangle^* .$$

Tiesinius operatorius galime susieti su matricomis $M_U = (m_{kj}), 0 \leq k, j \leq 2^n - 1$

$$m_{kj} = \langle k| U |j\rangle .$$

Tada ermitiškai jungtinio operatoriaus matrica yra skaičiuojama taip

$$M_{U^\dagger} = (M_U^*)^T.$$

NOT operatorius X:

$$X: \qquad |0\rangle \to |1\rangle\,, \qquad |1\rangle \to |0\rangle\,,$$

matricinė forma $m_{kj} = \langle k|\, X\, |j\rangle, \quad k, j = 0, 1$:

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

H operatorius:

$$H: \quad |0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$
$$|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

matricinė forma:

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

Dviejų kubitų sistema

Qiskit įrankyje kubitai išdėstomi iš apačios į viršų tvarka – the ordering in little endian

$$|q_1\, q_0\rangle\,.$$

Dviejų kubitų sistema

Qiskit įrankyje kubitai išdėstomi iš apačios į viršų tvarka – the ordering in little endian

$$|q_1\, q_0\rangle\,.$$

Turime keturias skirtingas būsenas

$$|00\rangle\,,\quad |01\rangle\,,\quad |10\rangle\,,\quad |11\rangle\,.$$

$CX(0,1)$ operatorius: jeigu kontrolinio kubito q_0 reikšmė yra lygi vienetui, tai taikinio q_1 reikšmė yra keičiama į priešingą, priešingu atveju jokie ketimai neatliekami:

$$CX(0,1): \quad |00\rangle \to |00\rangle, \quad |01\rangle \to |11\rangle,$$
$$|10\rangle \to |10\rangle, \quad |11\rangle \to |01\rangle.$$

Matricinė operatoriaus forma:

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

$CZ(0,1)$ operatorius:

jeigu kontrolinio kubito q_0 ir taikinio kubito q_1 reikšmės yra lygios vienetui, tai taikinio q_1 reikšmė yra keičiama į (-1), kitais atvejais jokie ketimai neatliekami:

$$CZ(0,1): \quad |00\rangle \to |00\rangle, \quad |01\rangle \to |01\rangle,$$
$$|10\rangle \to |10\rangle, \quad |11\rangle \to -|11\rangle.$$

Matricinė operatoriaus forma:

$$M_{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Funkcijos f(x) reikšmių skaičiavimas

Generuojame visų n kubitų superpoziciją Hilberto vektorinėje erdvėje

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \, .$$

Apskaičiuojame ermitinio operatoriaus U_f vaizdą

$$U_f : \quad |\psi, 0\rangle \quad \rightarrow \quad \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x, f(x)\rangle \, .$$

Groverio iteracijų algoritmas

Kiekvienos iteracijos operatorių G užrašome taip

(Qiskit įrankyje veiksmai atliekami iš kairės į dešinę, priešingai, nei atliekant tradicinius matricinius skaičiavimus pvz. Matlab įrankiu):

$$G = Z_f \, Z_A,$$

Groverio iteracijų algoritmas

Kiekvienos iteracijos operatorių G užrašome taip

(Qiskit įrankyje veiksmai atliekami iš kairės į dešinę, priešingai, nei atliekant tradicinius matricinius skaičiavimus pvz. Matlab įrankiu):

$$G = Z_f \, Z_A,$$

čia Z_f pakeičia amplitudžių ženklus, jei orakulo funkcijos reikšmė yra $f(x) = 1$:

$$Z_f : \quad |x\rangle \quad \to \quad (-1)^{f(x)} \, |x\rangle \, ,$$

Z_A atlieka inversijos transformaciją vidurkio atžvilgiu

$$|\psi\rangle = \sum_{x=0}^{N-1} \alpha_x \, |x\rangle \, , \quad A = \frac{1}{N} \sum_{x=0}^{N-1} \alpha_x ,$$

$$Z_A \, |\psi\rangle = \sum_{x=0}^{N-1} (2A - \alpha_x) \, |x\rangle \, .$$

Pradedame laboratorinio darbo atlikimą.

Pradedame laboratorinio darbo atlikimą.

Pirma, svarbu visus operatorius užrašyti panaudojant standartinius kvantinius vartus.

Pradedame laboratorinio darbo atlikimą.

Pirma, svarbu visus operatorius užrašyti panaudojant standartinius kvantinius vartus.

Ankstesniame seminare skaičiavimus atlikome naudodami klasikinius loginius vartus, bet kvantiniuose kompiuteriuose galime naudoti tik leistinus kvantinius loginius vartus.

Pradedame laboratorinio darbo atlikimą.

Pirma, svarbu visus operatorius užrašyti panaudojant standartinius kvantinius vartus.

Ankstesniame seminare skaičiavimus atlikome naudodami klasikinius loginius vartus, bet kvantiniuose kompiuteriuose galime naudoti tik leistinus kvantinius loginius vartus.

Tai sudėtinė bet kokio kvantinio algoritmo realizavimo dalis.

Priminsiu, kad teoriniame seminare parodėme, jog inversijos transformaciją vidurkio atžvilgiu galime užrašyti ir taip (visi operatoriai yra unitarieji):

$$Z_A = H^{\otimes n}\, Z_{OR}\, H^{\otimes n}, \quad n = \log N.$$

Priminsiu, kad teoriniame seminare parodėme, jog inversijos transformaciją vidurkio atžvilgiu galime užrašyti ir taip (visi operatoriai yra unitarieji):

$$Z_A = H^{\otimes n}\, Z_{OR}\, H^{\otimes n}, \quad n = \log N.$$

Čia apibrėžiame dar vieną operatorių, kuris irgi keičia kvantinių bazinių būsenų amplitudžių ženklus

$$Z_{OR}: \quad |x\rangle \;\to\; \begin{cases} |x\rangle, & x = 0^n, \\ -|x\rangle, & x \neq 0^n. \end{cases}$$

Priminsiu, kad teoriniame seminare parodėme, jog inversijos transformaciją vidurkio atžvilgiu galime užrašyti ir taip (visi operatoriai yra unitarieji):

$$Z_A = H^{\otimes n}\, Z_{OR}\, H^{\otimes n}, \quad n = \log N.$$

Čia apibrėžiame dar vieną operatorių, kuris irgi keičia kvantinių bazinių būsenų amplitudžių ženklus

$$Z_{OR}: \quad |x\rangle \;\to\; \begin{cases} |x\rangle, & x = 0^n, \\ -|x\rangle, & x \neq 0^n. \end{cases}$$

Nesunku patikrinti, kad Z_{OR} yra diagonalinis ir irgi unitarusis.

Nagrinėkime pirmą praktinį uždavinį.

Nagrinėkime pirmą praktinį uždavinį.

Turime du kubitus q_0 ir q_1, tokia sistema gali būti keturiose būsenose

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad |11\rangle.$$

Nagrinėkime pirmą praktinį uždavinį.

Turime du kubitus q_0 ir q_1, tokia sistema gali būti keturiose būsenose

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad |11\rangle.$$

Naudodami Groverio paieškos algoritmą suraskime "11" elementą.

Nagrinėkime pirmą praktinį uždavinį.

Turime du kubitus q_0 ir q_1, tokia sistema gali būti keturiose būsenose

$$|00\rangle, \quad |01\rangle, \quad |10\rangle, \quad |11\rangle.$$

Naudodami Groverio paieškos algoritmą suraskime "11" elementą.

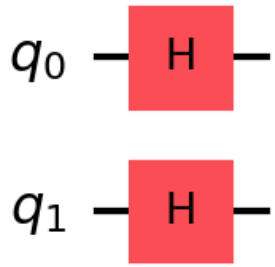
#Required imports

from qiskit import QuantumCircuit

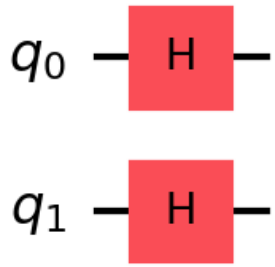
from qiskit import transpile #quantum gates: cx() cz()

circuit = QuantumCircuit(2)
circuit.h(0)
circuit.h(1)

Grandinės sudarymo vizualizacija (lego kaladėlės – blokinė schema)



Grandinės sudarymo vizualizacija (lego kaladėlės – blokinė schema)

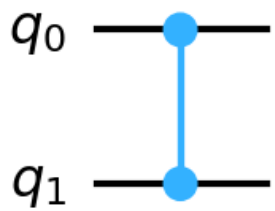


Gauname tokią pradinę dviejų kubitų grandinės būseną: keturių bazinių būsenų superpoziciją su vienodais koeficientais

$$|\psi\rangle = \frac{1}{2}\sum_{x=0}^{3}|x\rangle\,.$$

#Generuojame orakulo operatorių Z_f, atpažįstantį būseną "11".
#Jį aprašome kaip atskirą kvantinę grandinę, naudojančią tuos
#pačius du kubitus q_0, q_1:

```
oracle = QuantumCircuit(2)
oracle.cz(0, 1)
```



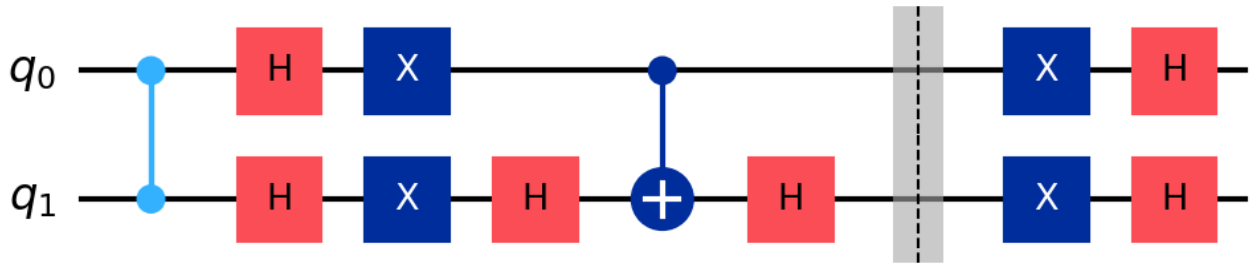
#Generuojame Grover operatorių: pridedame Z_A transformaciją:

```
grover = oracle
grover.h(0)
grover.h(1)
grover.x(0)
grover.x(1)
grover.h(1)
grover.cx(0, 1)
grover.h(1)
grover.x(0)
grover.x(1)
grover.h(0)
grover.h(1)
```

#Generuojame Grover operatorių: pridedame Z_A transformaciją:

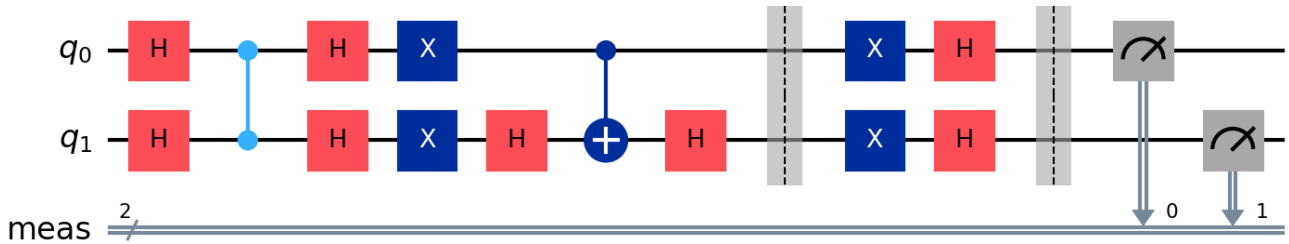
```
grover = oracle
grover.h(0)
grover.h(1)
grover.x(0)
grover.x(1)
grover.h(1)
grover.cx(0, 1)
grover.h(1)
grover.x(0)
grover.x(1)
grover.h(0)
grover.h(1)
```

Realizacijos vizualizacija



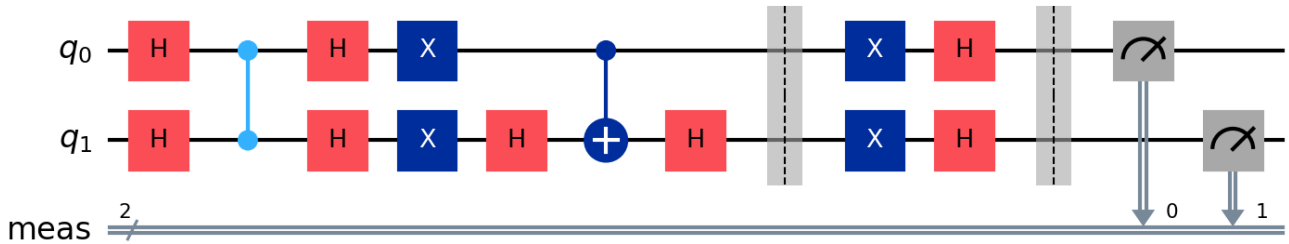
#Atliekame vieną Grover iteraciją ir išmatuojame rezultatą:

circuit.measure_all()



#Atliekame vieną Grover iteraciją ir išmatuojame rezultatą:

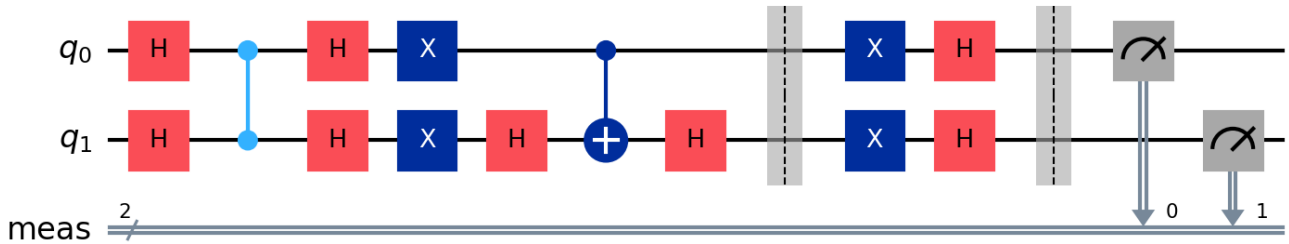
circuit.measure_all()



simulator = AerSimulator()
circuit2 = transpile(circuit, simulator)
job = simulator.run(circuit2, shots=200)
result = job.result()
counts = result.get_counts(circuit2)
print(counts)

#Atliekame vieną Grover iteraciją ir išmatuojame rezultatą:

circuit.measure_all()



simulator = AerSimulator()
circuit2 = transpile(circuit, simulator)
job = simulator.run(circuit2, shots=200)
result = job.result()
counts = result.get_counts(circuit2)
print(counts)

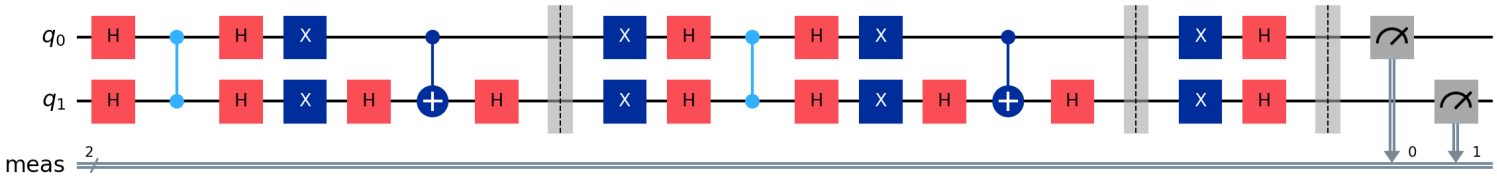
{'11': 200}

Atlikime dvi Groverio iteracijas

$circuit = circuit.compose(grover)$

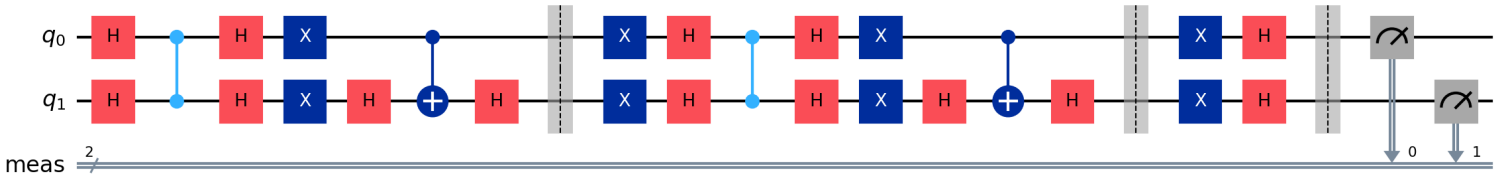
Atlikime dvi Groverio iteracijas

circuit = circuit.compose(grover)



Atlikime dvi Groverio iteracijas

circuit = circuit.compose(grover)



{'11': 44, '10': 52, '00': 44, '01': 60}

Pakeiskime mūsų tikslą, ieškokime elemento "01"

Pakeiskime mūsų tikslą, ieškokime elemento "01"

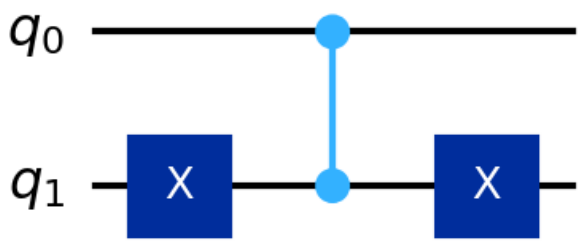
Kadangi algoritmas užrašomas funkciniu/objektiniu būdu, tai užtenka pakeisti tik orakulo dalį

oracle.x(1)
oracle.cz(0, 1)
oracle.x(1)

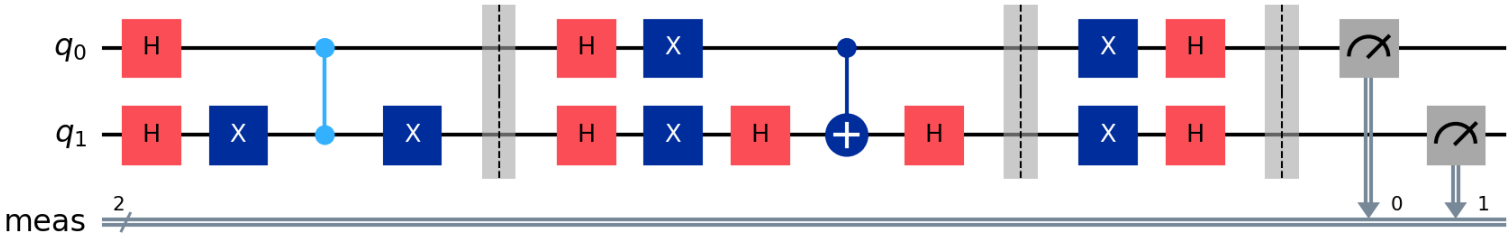
Pakeiskime mūsų tikslą, ieškokime elemento "01"

Kadangi algoritmas užrašomas funkciniu/objektiniu būdu, tai užtenka pakeisti tik orakulo dalį

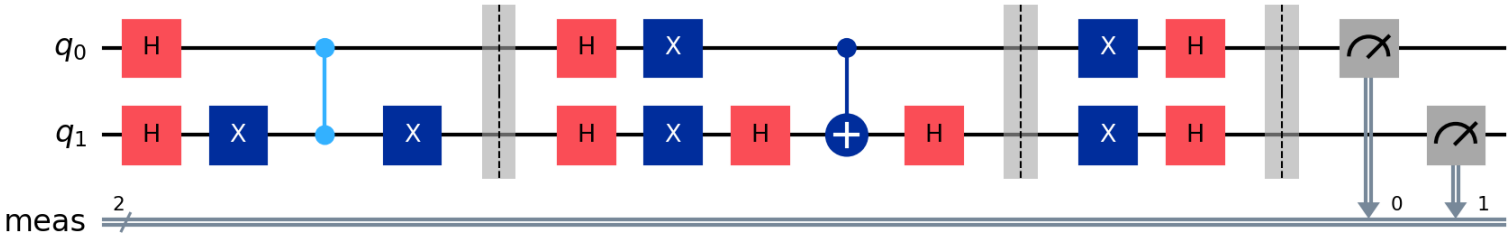
oracle.x(1)
oracle.cz(0, 1)
oracle.x(1)



Atliekame vieną Groverio algoritmo iteraciją



Atliekame vieną Groverio algoritmo iteraciją



Gauname tokius eksperimentų serijos rezultatus:

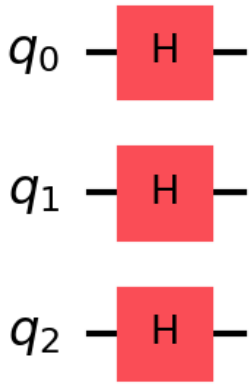
{'01': 200}

Imsime trijų kubitų sistemą ir generuosime aštuonių kvantinių būsenų superpoziją:

$$|\psi\rangle = \frac{1}{2\sqrt{2}}(\,|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle).$$

Imsime trijų kubitų sistemą ir generuosime aštuonių kvantinių būsenų superpoziją:

$$|\psi\rangle = \frac{1}{2\sqrt{2}}(\,|000\rangle + |001\rangle + |010\rangle + |011\rangle + |100\rangle + |101\rangle + |110\rangle + |111\rangle).$$

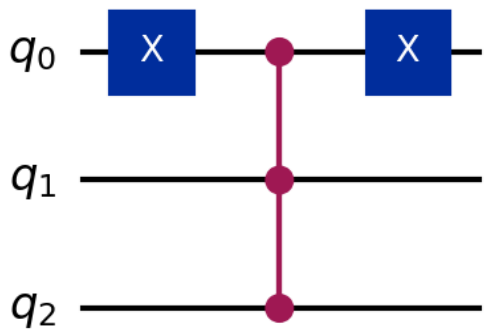


#Generuojame orakulą, kuris atpažįsta elementą "110":

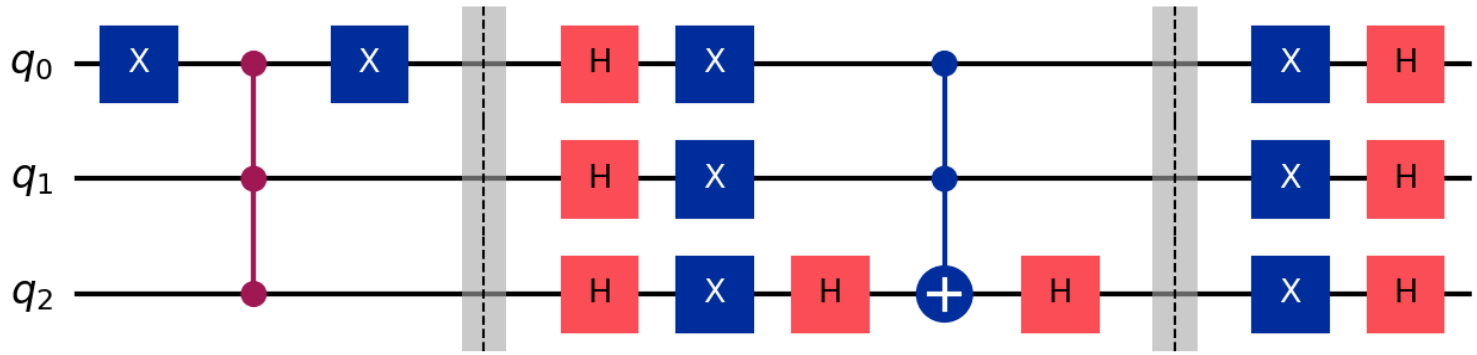
oracle.x(0)
oracle.ccz(0, 1, 2)
oracle.x(0)

#Generuojame orakulą, kuris atpažįsta elementą "110":

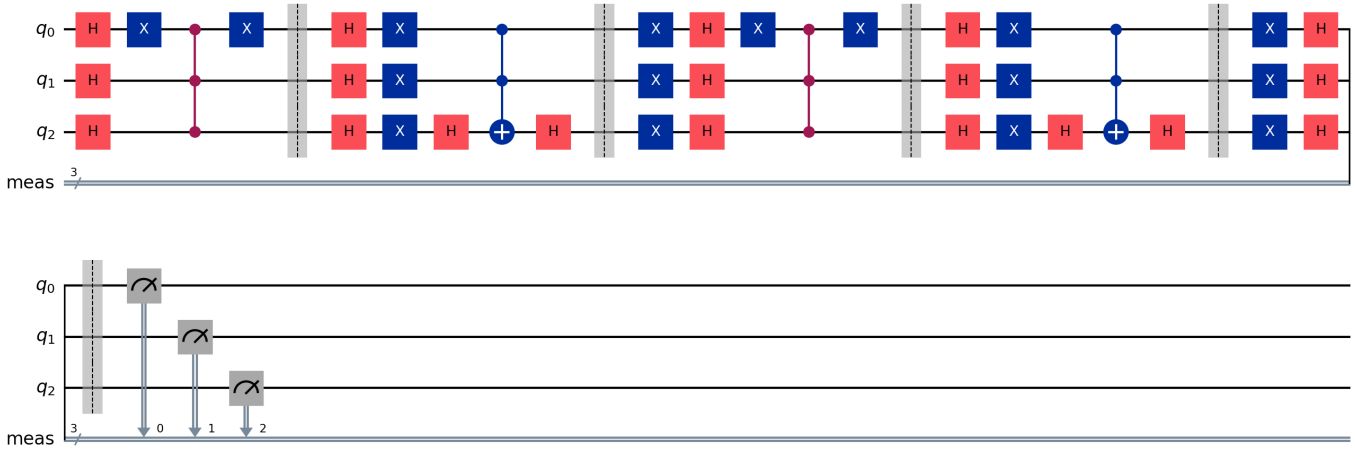
```
oracle.x(0)
oracle.ccz(0, 1, 2)
oracle.x(0)
```



#Generuojame Groverio operatorių :



#Generuojame grandinę su dviem Groverio iteracijomis ir atliekame matavimus:



Dviejų eksperimentinių serijų po 100 bandymų rezultatai:

{'111': 1, '101': 1, '000': 1, '100': 1, '001': 1, '110': 95 }

{'100': 1, '001': 1, '010': 1, '101': 1, '011': 1, '111': 2, '110': 93 }