Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kvantové aplikácie založené na platforme IBM QX

Diplomová práca

2019 **Bc. Vojtech Florko**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Kvantové aplikácie založené na platforme IBM QX

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	9.2.1. Informatika
Školiace pracovisko:	Katedra počítačov a informatiky (KPI)
Školiteľ:	prof. Ing. Ján Kollár, CSc.
Konzultant:	Ing. Milan Spišiak, PhD.

Košice 2019 **Bc. Vojtech Florko**

Názov práce:	Kvantové aplikácie založené na platforme IBM QX
Pracovisko:	Katedra počítačov a informatiky, Technická univerzita v Košiciach
Autor:	Bc. Vojtech Florko
Školiteľ:	prof. Ing. Ján Kollár, CSc.
Konzultant:	Ing. Milan Spišiak, PhD.
Dátum:	26. 4. 2019
Kľúčové slová:	kvantové výpočty, kvantový algoritmus, IBM QX, kvantový bit, kvantové zobrazenie

Abstrakt: Programy pre kvantové počítače, ktoré boli doteraz vyvinuté, ukázali ako odlišný je proces ich tvorby a overovania pri ich aplikovaní, v porovnaní s algoritmami na klasickom počítači. Hľadanie oblastí, kde sa dá kvantový počítač aplikovať, je hlavne za posledné roky perspektívna činnosť, ktorá rozhodne, či praktický kvantový počítač bude súčasťou výpočtovej techniky budúcnosti. Pred niekoľkými rokmi bola spustená platforma IBM QX, ktorá umožňuje experimentovať s vykonávaním kvantových programov a táto služba umožňuje využiť výsledky, ktoré z behu kvantových programov vracia, v rámci aplikácií na klasických počítačov. V rámci hľadania spôsobov využitia kvantových programov ukazujem v tejto práci na niekoľkých aplikáciách s kvantovou zložkou pomocou platformy IBM QX, aký prístup je vhodné zvoliť pri tomto vývoji. V práci ukazujem, ako som postupoval pri transformácií výpočtových problémov z klasických počítačov na kvantový a vyhodnocujem, pri ktorých typoch aplikácií môže mať zmysel pokračovať v experimentovaní.

Thesis title:	Quantum applications based on the IBM QX platform
Department:	Department of Computers and Informatics, Technical University of Košice
Author:	Bc. Vojtech Florko
Supervisor:	prof. Ing. Ján Kollár, CSc.
Tutor:	Ing. Milan Spišiak, PhD.
Date:	26. 4. 2019
Keywords:	quantum computing, quantum algorithm, IBM QX, quantum bit, quantum mapping

Abstract: Programs for quantum computers which were developed so far have shown how different the required process of development and evaluation really is in comparison with algorithms for classical computers. The search for areas where quantum computer could be applied is a prospective field mainly in the last few years, which will decide whether a quantum computer will become a part of information technologies in the future. A few years ago the IBM QX platform was launched, which allows experimenting through executing quantum programs and this service allows to use the results from running the quantum programs in applications hosted on classical computers. In terms of searching for ways quantum programs could be used, I'm showing this in this thesis on multiple applications utilizing a quantum component using the IBM QX platform as well as deciding which approach is appropriate when developing these algorithms. In this thesis I'm showing how I have proceeded with transformation of computational problems from classical to quantum computers and I'm also evaluating, which types of applications are appropriate for further experiments.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra počítačov a informatiky

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Informatika**
Študijný program: **Informatika**

Názov práce:

Kvantové aplikácie založené na platforme IBM QX

Quantum applications based on the IBM QX platform

Študent:	**Bc. Vojtech Florko**
Školiteľ:	**prof. Ing. Ján Kollár, CSc.**
Školiace pracovisko:	**Katedra počítačov a informatiky**
Konzultant práce:	**Ing. Milan Spišiak, PhD.**
Pracovisko konzultanta:	**Katedra počítačov a informatiky**

Pokyny na vypracovanie diplomovej práce:
1. Vysvetliť matematické základy a princípy kvantových výpočtov.
2. Analyzovať kvantovú platformu IBM QX a spôsob jej použitia.
3. Pre riešenie vybraných problémov navrhnúť kvantové algoritmy a implementovať ich využitím IBM QX.
4 Vyhodnotiť dosiahnuté riešenie porovnaním klasického a kvantového prístupu.
5. Záverečnú prácu vypracovať podľa požiadaviek vedúceho.

Jazyk, v ktorom sa práca vypracuje: slovenský
Termín pre odovzdanie práce: 26.04.2019
Dátum zadania diplomovej práce: 31.10.2018

doc. Ing. Jaroslav Porubän, PhD.
vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -10-

prof. Ing. Liberios Vokorokos, PhD.
dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 26.4.2019 .....

Vlastnoručný podpis

Poďakovanie

Na tomto mieste by som rád poďakoval svojmu vedúcemu práce za jeho rady a pripomienky ktoré mi pomohli počas riešenia tejto záverečnej práce.
Špeciálna vďaka patrí môjmu konzultantovi, ktorého znalosti, nadšenie a rady mi otvorili dvere do tejto problematiky a umožnili venovať sa jej intenzívne vďaka pomoci pri riešení problémov, ktoré sa počas riešenia tejto práce vyskytli

Obsah

Zoznam obrázkov

Úvod

Kvantové počítače a kvantové výpočty sa v súčasnosti dostávajú do pozornosti komunity informatiky. Idea využitia kvantových javov pre výpočty na úrovni počítača je ale prekvapujúco stará. Pochádza z roku 1959 keď kvantový fyzik Richard Feynman poukázal na využitie počítačov v súvislosti s potrebami pre fyzikálne výpočty. Myšlienky využitia kvantových javov pre algoritmizáciu pretrvávali v odbornom povedomí ďalšie desaťročia, aj keď len na teoretickej úrovni. V roku 1981 Richard Feynman poukázal na to, že simulácie prírodných javov na počítačoch by boli najlepšie vykonateľné hlavne na systémoch, ktorých výpočty sú vykonávané nad princípmi kvantovej mechaniky, nakoľko tá je súčasťou prírody a nie je teda výtvorom človeka.

Dlhé roky teoretické základy kvantovej mechaniky a princípy kvantových výpočtov boli skúmané teoreticky. Hoci nebol k dispozícii kvantový počítač, prínosom tohto vedeckého bádania bolo poznanie princípov, ktoré by mal využívať, ako aj vývoj dôležitých kvantových algoritmov Algoritmy, ako napríklad Shorov algoritmus z roku 1994 poskytujúci masívne zrýchlenie generovania celých čísel pre kryptografické účely alebo Groverov algoritmus z roku 1996 pre vyhľadávanie v neusporiadanej databáze poukázali na oblasti výpočtových problémov, pre ktoré by bolo využitie kvantových počítačov efektívne v budúcnosti. Prelomovým bodom bol rok 2016, keď spoločnosť IBM predstavila platformu IBM QX. Táto vznikla hlavne vďaka pokrokom od roku 2012 v rámci výskumu IBM v oblasti využitia supravodičov v rámci architektúry kvantových čipov a meraní kvantových chýb. Momentálne sú na tejto platforme dostupné kvantové čipy o veľkosti 5 kvantových bitov a 16 kvantových bitov. Ako vysvetľujú vo svojej práci Ferrari a Amoretti [1], obidve tieto zariadenia sú založené na supravodičmi realizovaných kvantových bitoch, ktoré nie sú citlivé na elektrický šum, pričom sú pravidelne kalibrované. Platforma IBM QX umožňuje pomocou webového rozhrania

vykonávať reálne kvantové programy na jednom z poskytovaných plne funkčných kvantových čipov. Tieto programy možno vyvíjať pomocou kvantového asemblera QASM a následne ich vykonávať na fyzických kvantových počítačoch, umiestnených v laboratóriách IBM v USA. Platforma poskytuje dokumentáciu kvantových programov a kontakty na komunitu, v rámci ktorej je možné si vymieňať akékoľvek informácie a poznatky z už vykonaných experimentov.

Kľúčovým problémom pred experimentovaním s akýmkoľvek druhom počítača, teda aj s kvantovým počítačom je to, na aké problémy je výhodné ho použiť. V súčasnosti pretrváva názor, že kvantové počítače nie sú príliš vhodné pre všeobecné používanie. Sú však efektívne napríklad pre riešenie vybraných NP problémov, pretože vedia na rozdiel od klasických počítačov riešenie týchto problémov na kvantovom počítači prebieha v polynomiálnom čase a to aj s využitím masívneho paralelizmu. Výsledky experimentov na platforme IBM QX ukazujú, ako sa kvantové javy podieľajú na tvorbe výsledkov, poskytovaných reálnymi kvantovými obvodmi. Táto nebývalá možnosť práce s kvantovým počítačom poskytuje priestor pre objavovanie a experimentovanie so zákonitosťami kvantových javov aj pre bežných programátorov.

V tejto práci uvádzam matematické základy a princípy kvantových výpočtov v minimálnej miere potrebnej na pochopenie realizácie kvantových algoritmov na platforme IBM QX. Zameriavam sa na realizáciu vybraných problémov kvantovým spôsobom a ich riešenie v spolupráci s klasickým počítačom. Cieľom experimentov je ukázať, do akej miery sú vybrané problémy vhodné pre využitie kvantového prístupu, čo takýto prístup znamená z hľadiska implementácie a potenciálnych paradigiem vývoja softvéru v budúcnosti.

1 Ciele práce

V tejto práci som si dal za úlohu na základe experimentovania s kvantovým počítačom prostredníctvom platformy IBM QX poukázať na možnosti tvorby programov s využitím kvantového výpočtu.

V rámci analýzy je v prvej časti práce nevyhnutné uviesť základnú teóriu danej problematiky a určiť výhody a nevýhody využívaných programových prístupov, ako aj hardvérových kvantových topológií, ktoré mám pri riešení algoritmických problémov k dispozícii.

V ďalšej časti práce uvádzam výber kvantových programov s príslušnou argumentáciou, prečo majú byť vybrané pre prácu s kvantovým počítačom, ktorý je momentálne využiteľný. Na základe vyvinutých kvantových algoritmov a zhromaždených výsledkov vykonaných programov uvádzam zhodnotenie, aká forma využiteľnosti je dôležitá do budúcnosti pre vývoj kvantových programov.

2 Základy kvantových výpočtov

V súčasnosti sa pri práci s kvantovými počítačmi a hlavne s kvantovou platformou IBM QX nedá považovať úroveň abstrakcie programovacieho prístupu za príliš vysokú. Jazyk alebo konštrukcia kvantových algoritmov sú, ako už z názvu rozhrania IBM QX vyplýva, na úrovni inštrukcií kvantového asemblera, alebo QASM. Tento jazyk umožňuje priamo pracovať s kvantovými bitmi a to využívaním kvantových hradiel (často označovaných aj bránami). Pre pochopenie spôsobov návrhu a vytvorenia aplikácií pre kvantový počítač je potrebné vedieť pracovať s kvantovými bitmi. Pre tento účel je potrebné pochopiť, v čom sa kvantový bit líši od bitu, s ktorým pracuje klasický počítač. Nadväzne na to je potrebné pochopiť význam operácií s kvantovými bitmi a aké základné brány máme k dispozícii pre prácu nad nimi. Zhŕňam taktiež nežiadúce javy ovplyvňujúce správnosť kvantových výpočtov v zmysle presnosti ich výsledkov, ktoré vychádzajú z najnovších architektúr kvantových počítačov.

2.1 Matematické základy kvantových výpočtov

Pochopenie matematických základov kvantových výpočtov je nevyhnutné v minimálnej miere, a to kvôli zameraniu na praktickú implementáciu v tejto práci. Základné poznatky o vlastnostiach a reprezentáciách kvantových bitov sú potrebné pre vytváranie použiteľných kvantových algoritmov.

2.1.1 Komplexné čísla

Kvantová mechanika, ktorej teória je základ pre kvantové výpočty, využíva vo výpočtoch komplexné čísla, pričom ich obsahuje aj definícia kvantového bitu. Komplexné číslo $Z \in \mathbb{C}$ je číslo v tvare $Z = a + bi$, kde $a, b \in \mathbb{R}$ a i je imaginárna jednotka, pre ktorú platí $i^2 = -1$.

Yanofsky vo svojej práci [2] vyjadruje význam komplexných čísel prostredníctvom pravdepodobností stavov a prechodov v kvantovej mechanike. Tieto nie sú definované reálnymi číslami medzi 0 a 1, ale sú im pridelené komplexné čísla také, že $|c|^2$ je reálne číslo medzi 0 a 1. Použitie komplexných čísel vyplýva z kvantovej teórie, nakoľko pridávaním reálnych čísel získame väčšie reálne čísla, avšak viac komplexných čísel môže znižovať výslednú pravdepodobnosť.

2.1.2 Vektorový priestor

Pretože kvantové stavy reprezentované kvantovými bitmi sú definované ako vektory vo vektorovom priestore, je potrebné ho definovať. Kupča [3] aplikoval pri kvantových stavoch definíciu vektorového priestoru za pomoci množiny vektorov V, množiny komplexných čísel $\mathbb{C}$ a operácií súčtu $+$ a násobenia "."pričom štvorica $(V, \mathbb{C}, +, \cdot)$ predstavuje vektorový priestor. Všetky vektory $x \in V$, majú priradené číslo $||x|| \in \mathbb{R}$, ktoré vyjadruje jeho dĺžku, nazývanú $norma$. Vektorovému priestoru, ktorý má normu, sa hovorí normovaný vektorový priestor.

Na vektorovom priestore je ľubovoľným dvom vektorom $x, y \in V$ priradené komplexným skalárnym súčinom komplexné číslo zapísané ako (x, y). Dva vektory $x, y \in V$ sú vzájomne ortogonálne, ak $(x, y) = 0$. Potom je ortogonálna báza množina vektorov, kde ľubovoľné dve z nich majú skalárny súčin rovný 0.

2.1.3 Hilbertov priestor

Kvantové stavy definujeme, ako už bolo uvedené, vo vektorovom priestore. Tento priestor sa nazýva Hilbertov komplexný priestor. Kupča pri odvodzovaní Hilbertovho priestoru [3] definuje najprv úplný vektorový priestor. Cauchyho postupnosť je taká, kde rozdiely medzi číslami sa s ďalšími pribúdajúcimi číslami zmenšujú (napríklad postupnosť $\{3, 3.1, 3.14, 3.141, 3.1415, \ldots, \pi\}$). Vektorový priestor V je úplný, ak každá Cauchyho postupnosť vektorov $x \in V$ konverguje k vektoru, ktorý je prvkom V. Ak priestory, ktoré majú definovaný skalárny súčin nazývame unitárne, potom Hilbertov priestor $\mathcal{H}$ je taký, ktorý je úplný a zároveň unitárny.

2.1.4 Diracov zápis vektorov

Kvantové stavy sú teda definované ako vektory v Hilbertovom priestore. Označenie využívané pri definovaní kvantových stavov $|\varphi\rangle$ pochádza z Diracovho zápisu vektorov $bra - ket$. Vektor $\vec{\varphi}$ sa v zápise kvantovej mechaniky zapisuje ekvivalentne $|\varphi\rangle$ a nazýva sa ket. Kvantový stav môžeme vyjadriť aj stĺpcovým vektorom:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Vektory Hilbertovho priestoru sa však dajú opísať aj riadkovo. V práci Kupču [3] sa pojednáva, že ak sú stĺpcové vektory prvky priestoru V, potom riadkové vektory predstavujú prvky takzvaného duálneho priestoru V^*. Ak je pre stavy $|x\rangle, |y\rangle$ definovaná funkcia $f_{|y\rangle} : V \to C$ pričom $f_{|y\rangle}(|x\rangle) = (|x\rangle, |y\rangle)$, potom $f_{|y\rangle}$ je ekvivalentná riadkovému vektoru, ktorý zapisujeme $\langle y|$ a nazývame bra.

$$\text{Ket-u } |x\rangle = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \text{ zodpovedá vektor bra } \langle x| = |x\rangle^\dagger = (x_1^*, \ \ldots \ , x_n^*),$$

kde x_i^* je združené komplexné číslo k x_i

2.1.5 Tenzorový súčin

Jeden kvantový bit existuje vo vlastnom Hilbertovom priestore. Pri dvoch kvantových bitoch však máme ďalší Hilbertov priestor, v ktorom sú tieto bity spojené. Výsledok tenzorového súčinu dvoch vektorových priestorov reprezentuje nový Hilbertov vektorový priestor, ktorý obsahuje takzvané elementárne tenzory.

$$\text{A} \otimes B = \begin{pmatrix} a_{1,1}B & a_{1,2}B & a_{1,3}B & \ldots & a_{1,n}B \\ a_{2,1}B & a_{2,2}B & a_{2,3}B & \ldots & a_{2,n}B \\ a_{3,1}B & a_{3,2}B & a_{3,3}B & \ldots & a_{2,n}B \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n,1}B & a_{n,2}B & a_{n,3}B & \ldots & a_{n,n}B \end{pmatrix}$$

Tenzorovým súčinom vykonávame transformáciu vektorov opisujúcich kvantové stavy, pričom z kvantových stavov produkuje ďalšiu maticu reprezentujúcu nový stav. V zápise kvantového stavu východiskového z $|\varphi\rangle \otimes |\psi\rangle$ sa bude udávať ako $|\varphi\rangle|\psi\rangle$ alebo väčšinou ako $|\varphi\psi\rangle$.

2.2 Princípy kvantových výpočtov

V predošlej časti už boli naznačené pojmy ako napríklad kvantový stav alebo kvantový bit. Ide o základné prvky kvantových počítačov v ktorých sa odlišujú od klasických počítačov. V tejto kapitole sa venujem základnou prácou s nimi a spôsobom vytvárania kvantového výpočtu.

2.2.1 Kvantový stav

Keďže princípy kvantových výpočtov, teda práce s kvantovými bitmi, vychádzajú z kvantovej mechaniky, je dobré si vedieť predstaviť fyzikálnu podstatu kvantového stavu, ktorý je z informačného hľadiska reprezentovaný hodnotou kvantového bitu. Najjednoduchšiu definíciu zhrnul Laforest [4] vo svojej práci, kde uvádza kvantový stav ako spojenie všetkých relevantných fyzikálnych vlastností kvantového systému. Medzi tieto patria predovšetkým pozícia, polarizácia alebo napríklad spin a mnoho ďalších. Pre definíciu kvantového bitu je ešte potrebné vyjadriť výnimočné stavy. Stavy sú výnimočné práve vtedy, ak prítomnosť v jednom stave jednoznačne indikuje, že je nulová pravdepodobnosť, aby bol v inom stave.

2.2.2 Kvantový bit

Pri kvantových výpočtoch kvantový bit predstavuje najmenšiu jednotku kvantovej informácie. Na rozdiel od klasického bitu, ktorý má hodnotu 0 alebo 1, kvantový bit je reprezentovaný inak. Kvantový bit je definovaný dvojrozmerným Hilbertovým vektorovým priestorom nad komplexnými číslami $\mathbb{C}^2$, ako ukazuje vo svojej práci Valiron [5]. Pre vyjadrenie kvantového bitu to znamená, že je plne reprezentovateľný až dvoma komplexnými číslami. Kvantový bit môže prejsť do dvoch výnimočných stavov a to $|0\rangle$ a $|1\rangle$. Tieto stavy majú vo vektorovej reprezentácii dve hodnoty, pričom tvoria štandardnú výpočtovú bázu $\{|0\rangle, |1\rangle\}$, kde:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Coles a kolektív [6] v svojej práci definujú kvantový bit zaujímavejším spôsobom, a síce že ide o dvojrozmerný systém kvantovej mechaniky, ktorý je v nasle-

dujúcom stave

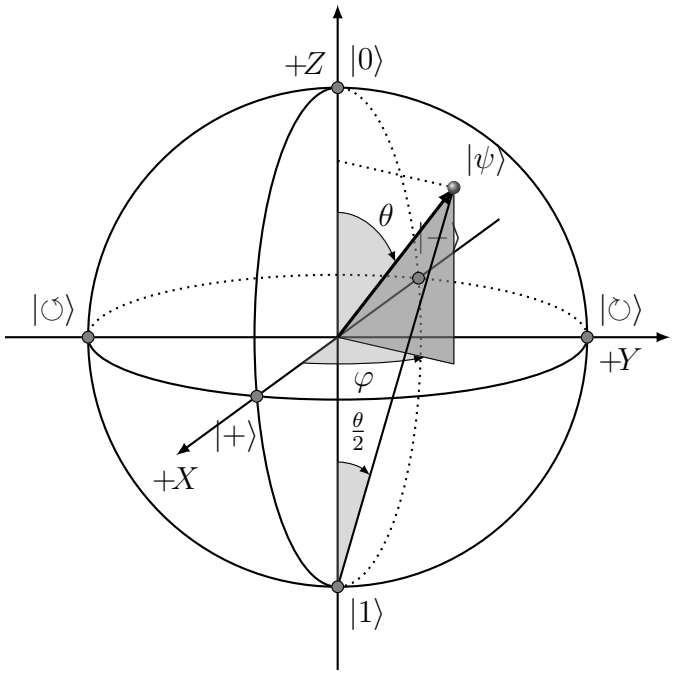
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}^2, |\alpha|^2 + |\beta|^2 = 1$$

Obe definície na prvý pohľad opisujú kvantový bit inak, ale vo svojej podstate sa nevylučujú. Očividný rozdiel v nich je ten, že kým prvá uvedená definícia uvádza, že kvantový bit môže prejsť do jedného z dvoch možných stavov, druhá definícia využíva horeuvedenú formuláciu ďalšieho celistvého stavu skladajúceho sa z týchto stavov. Táto rovnosť súvisí s kvantovým javom superpozície, ktorý kvantové výpočty majú za cieľ s výhodou využiť v prospech výsledkov a v tejto práci sa mu venujem v nasledujúcich kapitolách.

2.2.3 Blochova guľa

Jeden kvantový bit možno zobraziť bodom na povrchu Blochovej gule (obr. 2.1). Stav kvantového bitu $|\psi\rangle$ je vyjadrený:

$$|\psi\rangle = cos(\theta/2)|\theta\rangle + e^{i\phi}sin(\theta/2)|1\rangle$$



Obr. 2.1: Blochova guľa

2.2.4 Viacero kvantových bitov

Blochová guľa slúži vyobrazenie stavu práve jedného kvantového bitu, avšak viacero kvantových bitov sa na nej zobraziť nedá. V praktických kvantových výpočtoch využívam viacero kvantových bitov. V platforme IBM QX sú k dispozícii zložitejšie topológie a teda má význam zaoberať sa z teoretického hľadiska kvantovými systémami s viacerými kvantovými bitmi.

Pri reprezentácii stavov jedného kvantového bitu je stav $\mathbb{C}^2$ v báze $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ vyjadrený ako $\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\}$. Vo svojej práci o teórii kvantových brán vyjadruje Ming-Tseng [7] stav systému viacerých kvantových bitov tenzorovým súčinom (alebo Knoeckerovým súčinom) individuálnych kvantových bitov, pretože tenzorový súčin je definovaný vo vektorovom priestore. V prípade systému dvoch kvantových bitov je kvantový stav vektorov bázy vyjadrený tenzorovým súčinom $\mathbb{C}^2 \times \mathbb{C}^2 = \mathbb{C}^4$, teda:

$$
\begin{aligned}
|00\rangle &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \\
|01\rangle &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \\
|10\rangle &= \begin{pmatrix} 1 \\ 0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \\
|11\rangle &= \begin{pmatrix} 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}
\end{aligned}
$$

Tieto páry kvantových bitov predstavujú v tomto priestore takzvané kvantové superpozície základných stavov a tvoria bázu $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$.

2.2.5 Superpozícia

V predošlom zápise dvoch kvantových bitov $|xy\rangle$ bol umiestnený na druhej pozícii ďalší stav. Jeden kvantový bit však obsahuje okrem $|0\rangle$ a $|1\rangle$ aj všeobecný kvantový stav, ktorého princíp je, že je lineárnou kombináciou základných kvantových stavov. Tento stav bol už vyjadrený v kapitole 2.1.1. Kvantový stav ψ, ktorý je superpozíciou základných stavov $|0\rangle$ a $|1\rangle$ možno vyjadriť v tvare

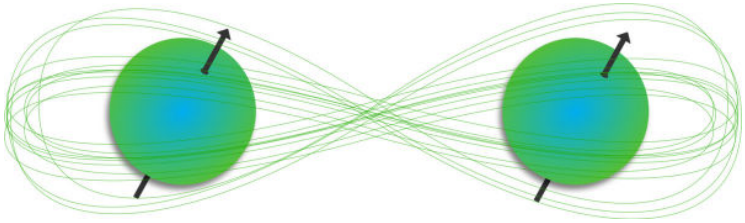
$$\psi = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in \mathbb{C}^2, |\alpha|^2 + |\beta|^2 = 1$$

Reálny stav kvantového bitu je pri meraní v kvantovej superpozícii na výstupe vždy 0 alebo 1. α a β vyjadrujú pravdepodobnosť, že sa jeden z týchto stavov vyskytnú vo výsledku merania stavu. Laforest [4] tento stav opisuje ako ľubovoľný stav, konkrétne že princípy kvantovej mechaniky dovoľujú kvantovému systému nadobúdať akúkoľvek hodnotu stavu $\alpha|0\rangle + \beta|1\rangle$.

Ako opisuje v svojej práci Sarma [8], systémy založené na kvantových bitoch vo svojej podstate predstavujú masívny výpočtový systém porovnateľný s klasickými superpočítačmi postaveným na princípe klasických počítačov. Je to práve vďaka možnosti disponovať viacerými stavmi naraz, pretože práve superpozícia kvantových bitov umožňuje kvantovému počítaču dosiahnuť neporovnateľne vysokú mieru paralelizmu. Tento princíp predstavuje fundamentálny poznatok využívaný pri získavaní výsledkov z kvantových výpočtov, teda meraní kvantových stavov zo vstupných kvantových bitov. Dôležité je poznamenať, že nevieme predpovedať presný stav na kvantovom bite v superpozícii pred jeho meraním, vieme určiť jedine pravdepodobnosť, s akou sa jeden zo stavov prejaví vo výsledku.

2.2.6 Previazanie

Kvantové previazanie je ďalší jav, ktorý klasické počítače a bity nepoznajú a kvantové výpočty majú často krát za cieľ využiť ho ako výhodu vo výpočtoch. Princíp spočíva v zoskupení kvantových bitov (najjednoduchšia simulácia tohto javu využíva jeden pár), ktoré sú generované takým spôsobom, že sú fyzikálne prepojené a kvantový stav jedného kvantového bitu nie je nezávisle vyjadriteľný od stavu ostatných kvantových bitov. Kvantové bity sa v tejto situácii správajú ako jeden celistvý systém a toto platí aj napriek potenciálnej fyzickej vzdialenosti medzi nimi, čím dochádza ku korelácii ich fyzikálnych vlastností, ktoré možno pozorovať, ako je ilustrované na obrázku 2.2.



Obr. 2.2: Abstraktná vizualizácia previazania kvantových bitov [9]

Ak teda máme napríklad stav dvoch kvantových bitov: $(|00\rangle + |11\rangle)$ pričom ak sú tieto kvantové bity previazané, tento stav nemôže byť vyjadrený ako súčin $|\phi\rangle \times |\psi\rangle$. Presná formulácia tohto javu je uvedená v práci o kvantovom previazaní Horodecki, et.al [10], ktorá previazanosť charakterizuje ako stav nerozložiteľnosti kvantového systému do subsystémov, špecifickejšie kedy nie je možné zmeniť separátne vektor jedného kvantového bitu, ktorý je previazaný s aspoň jedným iným kvantovým bitom.

2.2.7 Operácie a brány nad kvantovým bitom

Kvantové výpočty prebiehajú na kvantových bitoch v niekoľkých fázach. Prvá z nich zahŕňa vytváranie a nastavovanie stavov kvantových bitov. Kvantové bity menia svoj stav prostredníctvom operácií, ktoré ich konfigurujú otáčaním okolo osí, ako bolo ilustrované Blochovou guľou. Už ľubovoľné vytvorenie kvantových bitov na stavy $(|0\rangle$ a $|1\rangle)$ predstavuje formu operácie, nakoľko ich kvantový stav sa zmenil. Kvantové stavy sú pre potreby výpočtov vyjadrené maticovo, rovnako ako operácie s nimi. Vykonanie operácie je získané výpočtom tenzorového súčinu matíc. Operácie, ktoré majú pre kvantové výpočty význam sú reprezentované a realizované kvantovými bránami.

2.2.8 Pauliho brány

Brány sa delia na viac kategórií, pričom prvá z nich sú reverzibilné alebo unitárne brány. Medzi najvýznamnejšie patria Pauliho brány X, Y, Z:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Táto brána slúži na negáciu kvantových stavov, pretože nastavuje stav $|0\rangle$ na $|1\rangle$ a naopak stav $|1\rangle$ na $|0\rangle$. Niekedy je známa aj pod označením N, alebo preklopenie bitu. Na Blochovej guli je ekvivalentná otočeniu okolo osi X o uhol π.

$$Y = \begin{pmatrix} 1 & -i \\ i & 0 \end{pmatrix}$$

Brána Y je ekvivalentná otočeniu okolo osi Y o uhol π na Blochovej guli.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Brána Z spôsobuje otočenie okolo osi Z o uhol π na Blochovej guli. Nemení základný stav $|0\rangle$ a mení stav $|1\rangle$ na stav $-|1\rangle$. Kvôli tomuto sa označuje ako takzvané preklopenie fázy.

2.2.9 Hadamardova brána

Na základe použitia Pauliho brán možno povedať, že sa kvantové bity chovali ako klasické bity, pretože operáciami neboli využité kvantové javy, ako je superpozícia alebo previazanie. Jedna z brán, ktorá umožnuje využitie výhod kvantového výpočtu je napríklad brána H:

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

Táto brána je známa pod názvom Hadamardova brána. Jej využitím nastavujeme stav $|0\rangle$ na stav $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ a stav $|1\rangle$ na stav $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Táto brána umožňuje vytvoriť stav superpozície, je teda pre kvantové výpočty nesmierne dôležitá. Jej význam popisuje Williams [11] vzťahom viacerých kvantových bitov; teda ak pripravíme n kvantových bitov, každý v základnom stave $|0\rangle$ a na každom kvantovom bite $paralelne$ vykonáme operáciu prostredníctvom Hadamardovej brány, výsledný stav je ekvivalentná superpozícia všetkých celých čísel v rozsahu od 0 do $2n - 1$.

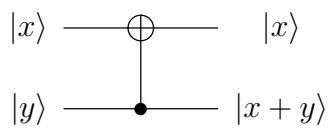
$$H|0\rangle \times H|0\rangle \times \ldots \times H|0\rangle = \frac{1}{\sqrt{2n}} \sum_{j=n}^{2^n-1} |j\rangle$$

Na Blochovej guli je táto operácia ekvivalentná dvom otočeniam, a to otočeniu π okolo osi X a následne otočeniu $\frac{\pi}{2}$ okolo osi Y. Hadamardova brána mení os X na os Z a os Z na os X.

2.2.10 CNOT brána

Stav vykonateľný Hadamardovou bránou umožňuje využiť kvantový jav superpozície pre kvantové výpočty. Pre kvantový jav previazania slúži ďalšia brána, a

to brána CNOT, alebo riadená NOT. Yanofsky v práci [2] vyjadruje princíp brány CNOT na základe jej účinku v kvantovom obvode (obr 2.3):



Obr. 2.3: Definícia brány CNOT [2]

Všimnime si na vyobrazenom obvode, že má dva vstupy a dva výstupy.

$Vstupy$		$Vystupy$	
x	y	x	y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

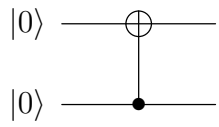
Vrchný vstup je v tejto situácii takzvaný riadiaci bit, teda riadi výstup. V prípade $|x\rangle = |0\rangle$ bude výstup rovnaký ako vstup, pretože stav tohto kvantového bitu neovplyvňuje výstup bitu $|y\rangle$. V prípade $|x\rangle = |1\rangle$ bude však výstup $|y\rangle$ opačný. Zodpovedajúca matica k tejto bráne je:

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

Po zostavení kvantového výpočtu a jeho vykonaní možno merať kvantové hodnoty, teda na výstupe dostávame hodnoty indikujúce, na ktorom kvantovom bite je aká pravdepodobnosť kvantových stavov $|0\rangle$ a $|1\rangle$. Aplikovanie brány možno ilustrovať nasledovne (obr. 2.4):

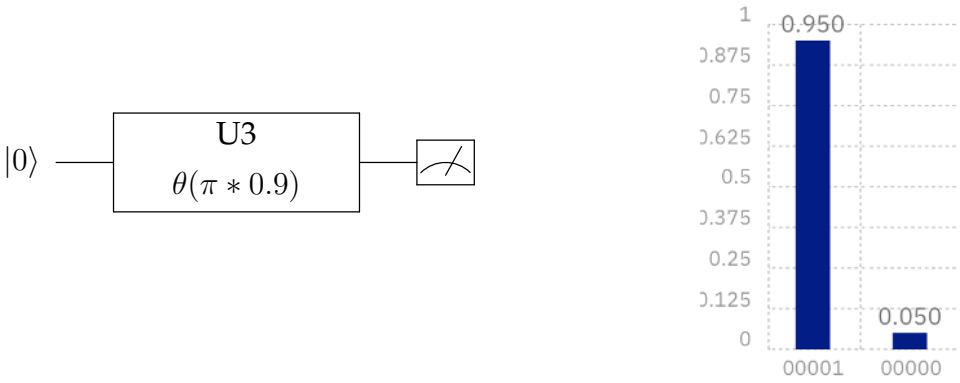
2.2.11 Brány U

Pri zostavovaní kvantového obvodu v rozhraní IBM QX máme k dispozícii ešte niekoľko brán, ktoré sú označované aj ako fyzické, a to U1, U2 a U3. Číslo v ich



Obr. 2.4: Dva kvantové bity prepojené bránou CNOT

označení určuje počet ich parametrov pri ich aplikovaní. Ostatné brány sú špeciálne prípady týchto brán. V implementačnej časti tejto práce využívam bránu U3, ktorá umožňuje nastaviť uhly θ, φ a λ. Ak teda použijeme bránu U3 s nastavením θ napríklad hodnotu 0.9, táto hodnota ovplyvní výsledok pri meraní (obr. 2.5):



Obr. 2.5: Vplyv použitia brány U3 na výsledok po meraní

2.2.12 Veta o nemožnosti klonovania

V spojitosti s bránou CNOT je dobré poukázať na vetu o nemožnosti klonovania, teda nemožnosti vytvárania kópií kvantových bitov. Podľa O'Donnell [12] je otázka, či je pri kvantovom bite $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ možné kopírovať hodnotu $|\psi\rangle$ tak, aby obidva tieto nepreviazané kvantové bity mali hodnotu vstupného bitu $|\psi\rangle$. Bolo by potrebné, aby tieto kvantové bity boli v celkovom stave $|\psi\rangle \otimes |\psi\rangle$. Ukazuje sa, že pre ľubovoľné $n \in N$ neexistuje kvantový obvod, kde pri vstupe $|\psi\rangle \otimes |0^{n-1}\rangle$ dostaneme výstup $|\psi\rangle \otimes |\psi\rangle \otimes f(|\psi\rangle)$. Jednoduchšie povedané, neexistuje taký algoritmus, ktorý na základe vstupu $|\psi\rangle$ bude mať výstup $|\psi\rangle \otimes |\psi\rangle$.

2.2.13 Operácie a brány nad viacerými kvantovými bitmi

Ako už bolo uvedené, systém kvantových bitov je vyjadrený Hilbertovým vektorovým priestorom C. Pri systéme nad viacerými kvantovými bitmi pojednávame o vektorovom priestore C^{2^n}. Základ pre systém dvoch kvantových bitov je:

$$\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$$

Pre systém troch kvantových bitov je štandardnou bázou množiny:

$$|000\rangle, |001\rangle, |010\rangle, |100\rangle, |011\rangle, |101\rangle, |110\rangle, |111\rangle$$

Analogicky rastie rozsah Hilbertovho priestoru kvantového systému. Je evidentné, že tento rast je exponenciálny, teda pri systéme šiestich kvantových bitov je už rozsah pomerne vysoký. Aj preto je simulácia kvantového počítača pri takomto a vyššom rozsahu na klasickom počítači bez využitia reálnych kvantových javov na hardvérovej úrovni náročná, až nemožná.

2.3 Princíp kvantového merania

Meranie predstavuje z praktického hľadiska úkon, pomocou ktorého získavame výsledky z kvantového systému. Presnejšie povedané, určujeme pravdepodobnosti, že kvantový stav je $|0\rangle$ alebo $|1\rangle$. Laforest [4] definuje kvantové meranie pomocou Bornovho pravidla. Toto pravidlo hovorí, že pri kvantovom stave $|\psi\rangle$ a ortonormálnej báze $\{|\phi_1\rangle, ..., |\phi_n\rangle\}$, je pravdepodobnosť pri meraní kvantového stavu $|\phi_i\rangle$ určená takto:

$$P(\phi_i) = |\langle\phi_i|\psi\rangle|^2$$

Po samotnom vykonaní merania pravdepodobnosti dochádza ku vlnovému kolapsu, teda pôvodný kvantový stav $kolabuje$ na odmeraný stav. Aktuálny stav sa v konečnom dôsledku dostal do niektorého stavu z $|\phi_1\rangle, ..., |\phi_n\rangle$. Kvantové meranie je teda z matematického hľadiska operácia, ktorá pozostáva z aplikácie Bornovho pravidla a následného vlnového kolapsu.

Pre ilustráciu aplikácie Bornovho pravidla použijem príklad z práce [4] Laforesta. Nech je daný kvantový systém so stavmi $|+\rangle, |-\rangle$, pričom:

$$\text{Stav } |+\rangle \text{ je definovaný ako } \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$\text{Stav } |-\rangle \text{ je definovaný ako } \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

$$\text{Stav } |\psi\rangle \text{ definovaný ako } \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Bornovo pravidlo nám hovorí v tomto prípade, že pravdepodobnosť pri tomto kvantovom systéme je definovaná ako:

$$P(\pm) = |\langle \pm || \psi \rangle|^2$$

Pretože prvý vektor je vektorom bra, stĺpcový vektor sa mení na riadkový. Dosadením vektorov dostávame teda vzťah:

$$|\langle \pm || \psi \rangle|^2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & \pm 1 \end{bmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{\alpha \pm \beta}{\sqrt{2}}$$

Z tohto vzťahu vyplýva, že dostaneme kvantový stav $|\pm\rangle$ s pravdepodobnosťou

$$\frac{|\alpha \pm \beta|^2}{2}.$$

Zákonitosti kvantových javov a z nich vyplývajúce pravidlá kvantových výpočtov sú známe už desaťročia. Spôsob konštrukcie kvantového počítača, teda systému, kde vieme simulovať kvantové javy a získavať výsledky, je hardvérová záležitosť, kde ešte budú potrebné veľké posuny. Veľkým problémom pri už existujúcom kvantovom počítači je práve meranie. Kvantové javy je totiž možné simulovať realizovaním rôznych architektúr kvantových počítačov, avšak získavanie zmysluplných výsledkov je tá najnáročnejšia časť tejto realizácie. Rieffel uvádza vo svojej práci [13] ohraničenia získavania výsledkov z kvantového počítača. Kvantový počítač je schopný masívneho paralelizmu, avšak akékoľvek meranie spôsobuje zmenu existujúceho celkového kvantového stavu. Meranie je totiž výrazne pravdepodobnostná operácia a teda si nevieme ani vybrať, ktorý výsledok chceme na výstupe. Tento problém bol hlavne v posledných rokoch riešený rôznymi softvérovými aj hardvérovými technikami, ktoré čiastočne upravujú spôsob vykonávania merania a tým obmedzujú dekoherenciu kvantového systému.

2.3.1 Dekoherencia

Dekoherencia je dôležitý a často používaný pojem pri hardvérovej realizácii kvantových počítačov. Dekoherencia má vplyv na výsledky, ktoré získavame z dostupného kvantového počítača a je vhodné vysvetliť ju. Je zrejmé, že dekoherencia kvantového počítača je stratou koherencie, teda že výsledky dosiahnuté kvantovými javmi nie sú v rozsahu hodnôt, v ktorom sme ich očakávali. Prakticky sa tomuto javu hovorí šum a pretože dnešné kvantové počítače sú stále vzdialené dokonalej hardvérovej realizácii, dekoherencia bude v ďalších rokoch predstavovať problém pri ich miniaturizácii a narastaní počtu kvantových bitov. V práci Rieffelovej a Polaka [14] je uvedené, ako je nemožné aj do budúcna fyzický kvantový počítač absolútne izolovať od svojho prostredia, pretože na všetkých fyzických kvantových bitoch nastáva interakcia s okolím. Okolie alebo prostredie je v kvantovom počítači považované za subsystém, ktorý sa nedá riadiť programom, teda nemôžeme predpovedať jeho účinok. Prekonanie problémov dekoherencie je možné v prípadoch, kde je možné chyby korigovať napríklad reverzibilnými operáciami. Najjednoduchšiu teoretickú metodiku opisuje Avaliani [15], kedy pri nízkej chybovosti vieme kvantový počítač považovať za dostatočný. Z kódov odstraňujúcich chyby často býva využívaný napríklad takzvaný repetičný kód, ktorého výhody a prístupy ukazuje Wooton [16]. Hodnota 0 je zakódovaná ako 000 a 1 ako 111. Ak sa vyskytne chyba na jednom bite, napríklad 011, je možné ho opraviť na jeho pôvodnú hodnotu 111. Tieto technológie a postupy predstavujú práve kľúčovú zložku, vďaka ktorej je hardvérová realizácia kvantových počítačov dnes možná, nakoľko chybovosť kvantových výpočtov sa znížila na akceptovateľnú úroveň pri istej miere chybovosti na jednotlivých kvantových bitoch.

2.3.2 Chybovosť

Chybovosť kvantového bitu je určená ako pravdepodobnosť nežiadúcej zmeny kvantového stavu, pričom chyby môžu byť dvoch druhov, a to retenčné alebo operačné. Práca Tannu a Qureshi o problematike, či sú kvantové bity identické [17] ich definuje nasledovne: Pri retenčnej chybovosti je dôležitý fakt, že kvantový bit vie udržať svoju hodnotu iba na obmedzený čas, ktorému sa hovorí čas koherencie a je v priamom súvise s citlivosťou kvantového bitu na hardvérovej úrovni. Napríklad kvantový bit vo vysoko-energetickom stave $|1\rangle$ prirodzene klesá na

nízko-energetický stav $|0\rangle$ a čas tohto klesania je opísaný koherenčným časom T1, ktorý určuje mieru prirodzeného uvoľnenia tejto energie. Chybovosť spočíva v možnosti, že na kvantovom bite nežiadane nastane interakcia s prostredím a vyskytne sa na ňom dekoherencia skôr, ako klesne do stavu $|0\rangle$. Tento jav je definovaný koherenčným časom T2, ktorý určuje dobu, počas ktorej je kvantový bit ovplyvnený prostredím. Zaujímavosťou je, ako sa koherenčné časy za posledné roky zlepšili. Devoret a Schoelkopf uvádzajú vo svojej práci o aplikovaní supravodičov na kvantové účely [18] zlepšenie z 1 nanosekundy na 100 mikrosekúnd pri supravodičových kvantových počítačoch. Medzi tieto počítače patria aj počítače IBM QX.

Operačná chybovosť je určená ako pravdepodobnosť výskytu chýb, ktoré môžu nastať pri operáciách, teda napríklad aplikovaním brán v kvantových obvodoch. Operácia využívajúca rotácie vie pri istom uhle vniesť do tejto operácie prebytočnú rotáciu. Kvantové počítače IBM QX majú napríklad dnes operačnú chybovosť jedno-bitových operácií na úrovni $10^{-}3$, pričom dvoj a viac-bitové operácie ako napríklad brána CNOT je vykonávaná na úrovni chybovosti $10^{-}2$. Tieto údaje sú uvedené v dokumentácii pre kvantové počítače rozhrania IBM QX.

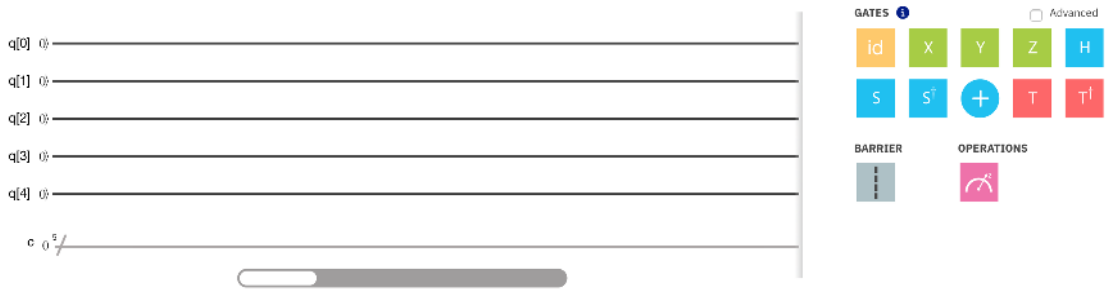
2.4 Kvantové počítače IBM QX a práca s nimi

Kvantové počítače, ako už bolo v úvode naznačené, sa stali reálnym nástrojom na vykonávanie kvantových experimentov v roku 2016. Vtedy spoločnosť IBM umožnila prístup k jej kvantovým architektúram prostredníctvom webového rozhrania a knižnice pre programovací jazyk Python, ktorá pristupuje k API tohto rozhrania. Od tohto momentu neexistuje alternatíva v podobe inej platformy pre prácu s ozajstnými kvantovými počítačmi na tak vysokej úrovni, teda je očividné bez ďalšej analýzy, prečo bola zvolená práve táto platforma.

2.4.1 Kompozícia obvodov a vykonávanie experimentov

Platforma IBM QX pre základnú prácu s kvantovými počítačmi poskytuje rozhranie známe ako kompozitor:

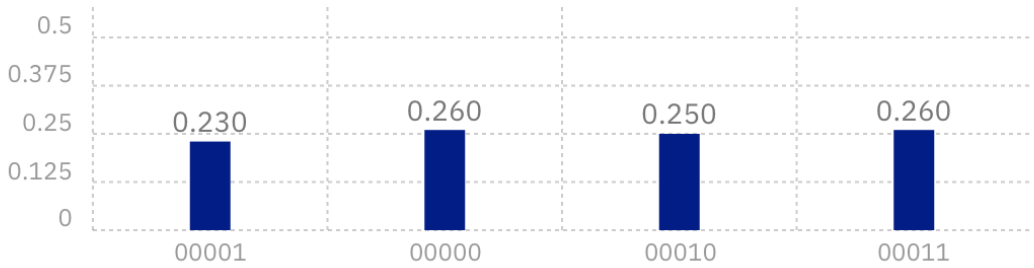
Na kompozitore na obrázku 2.6 je vidno, že pracujeme s kvantovým počítačom s piatimi kvantovými bitmi $q[0]...q[4]$. Na pravej strane kompozitora sú k dispozícii



Obr. 2.6: Rozhranie pre kompozíciu kvantových obvodov

známe kvantové brány ako Pauliho, Hadamardova alebo CNOT (označená symbolom +). Presunom brán na jednotlivé kvantové bity tvoríme kvantový obvod. Daný obvod je možné potom vyhodnotiť troma spôsobmi. Môžeme vykonať simuláciu (teda len softvérovo simulovaný kvantový počítač), čo dáva výsledky okamžite ale presnosťou iba orientačne, keďže sa na ich vyhodnotení nepodieľajú ozajstné kvantové javy. Druhou možnosťou je opakovanie vykonávania kvantového obvodu, ktorý už bol v histórii spustený, pričom výsledky sú presné lebo pochádzajú z reálnej kvantovej simulácie. Treťou možnosťou je vykonať experiment na ozajstnom kvantovom čipe, akurát treba rátať s potenciálnou čakacou dobou v závislosti od vyťaženia daného kvantového počítača a počtu obvodov čakajúcich v postupnosti na vykonanie.

Vyhodnotenie výsledkov dostávame v podobe hodnôt pravdepodobností na daných kvantových bitoch. V rámci platformy IBM QX dostávame graf distribúcie jednotlivých hodnôt:



Obr. 2.7: Výsledky vykonania kvantového obvodu

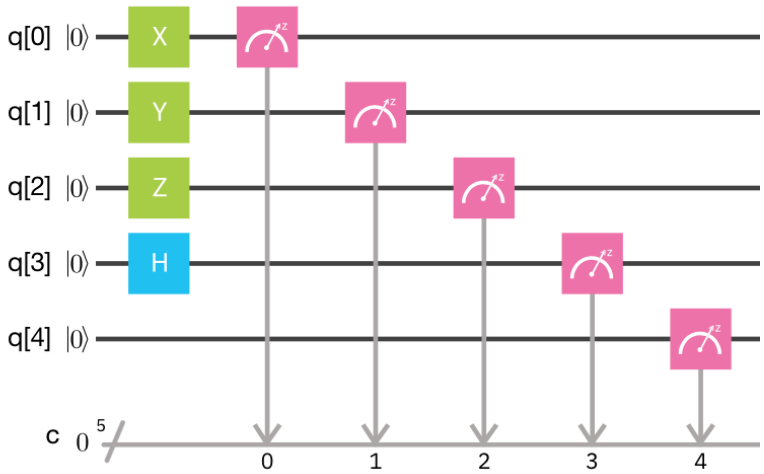
Z obrázku 2.7 vieme vyčítať, aké pravdepodobnosti sa nachádzajú na kvantových bitoch obvodu. Kvantový bit je identifikovaný vo výsledkoch poradím v postupnosti čísel, teda pri piatich kvantových bitoch je kvantový bit $q[0]$ definovaný

ako 00001, kvantový bit $q[1]$ ako 00010 a podobne. Je teda evidentné, že kvantový bit $q[1]$ má pravdepodobnosť $0.250 + 0.260 = 0.510$, pretože sa ho týka tretí a štvrtý stĺpec.

Pre prácu s kvantovými počítačmi je však potrebné poznať možnosti a hranice, ktoré určujú, do akej miery je práca s nimi efektívna. Medzi limitujúce faktory nepatrí totiž len počet kvantových bitov, ale aj architektúra kvantového systému.

2.4.2 Architektúry kvantových počítačov IBM QX

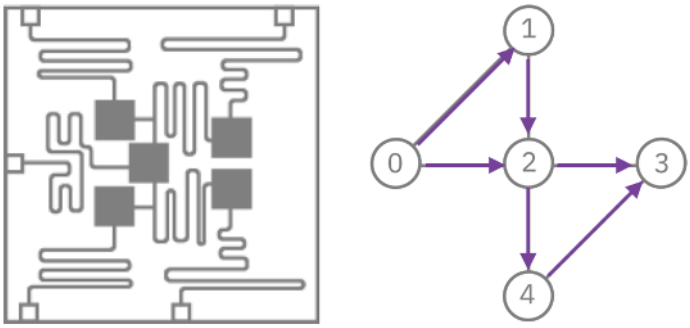
Z predošlých kapitol o kvantových bitoch sa dá usúdiť pravdivosť tvrdenia, že kvantový počítač je takpovediac výkonnejší s narastajúcim počtom kvantových bitov. Je to tak preto, že logicky narastá možná miera využitia paralelizmu v danom kvantovom obvode, pretože pri viacerých kvantových bitoch dokážeme pracovať s viacerými kvantovými stavmi. Pri začiatkoch práce s platformou IBM QX mali používatelia k dispozícii počítač ibmqx2 s 5 kvantovými bitmi. Umiestňovanie brán na dostupné kvantové bity je pred meraním voľné v každom bode obvodu, a teda je možné umiestňovať na kvantové bity napríklad Pauliho kvantové brány alebo Hadamardovu bránu (obr. 2.8):



Obr. 2.8: Ukážkový kvantový obvod na kvantovom počítači o 5 kvantových bitoch so štyrmi bránami a piatimi úkonmi merania

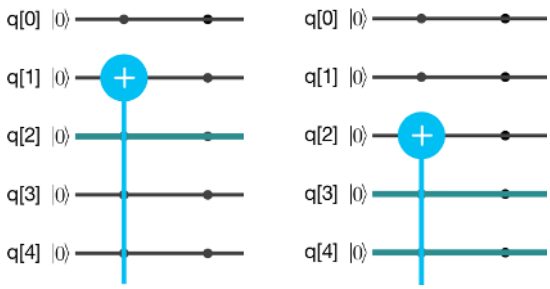
Najnáročnejšia úloha pri práci s kvantovým počítačom je vykonať efektívne a funkčné zobrazenie problému do programu. Vo svojej práci o efektívnom zobra-

zovaní problémov na IBM QX architektúry uvádza Zulehner [19], že ide o naozaj náročnú úlohu práve kvôli architektúram kvantových čipov. Na jednej strane musí byť problém s vysokoúrovňovým pohľadom na riešenie dekomponovaný na podoperácie, ktoré sa vôbec dajú využiť na IBM QX, na strane druhej sa však treba prispôsobiť aj fyzickým obmedzeniam kvantových bitov na kvantových počítačoch IBM QX. Problém môže nastať pri aplikovaní brány CNOT, ktorá slúži na previazanie kvantových bitov. Na každom kvantovom čipe sú totiž realizácie kvantových bitov fyzicky prepojené rôznym spôsobom (obr. 2.9):



Obr. 2.9: Grafické zobrazenie architektúry počítača IBM QX s 5 kvantovými bitmi z platformy IBM QX[20] obohatené o grafické vyobrazenie možných smerov previazaní z kvantových bitov

Je vidieť, že aj keď dokážeme druhý kvantový bit previazať na bity 3 a 4, pri bite číslo 4 je možné previazanie len do bitu číslo 3 a pri bite číslo 3 je nemožné vykonať na tejto architektúre operáciu CNOT (obr. 2.10):



Obr. 2.10: Zobrazenie pokusu o umiestnenie brány CNOT na architektúre z obrázka 2.6. Modrou farbou rozhranie IBM QX vyobrazuje možné prepojenia na kvantové bity.

3 Návrh kvantových aplikácií

Kvantové algoritmy, ktorých výsledky v rámci tejto práce integrujem do klasických aplikácií, som zostavil a vykonal prostredníctvom knižnice QuantumCSharp. Najpoužívanejšie knižnice na získanie výsledkov z kvantových výpočtov platformy IBM Q, a hlavne oficiálne riešenie sú knižnice pre jazyk Python. Pre riešenie nasledujúcich úloh však bola zvolená horeuvedená knižnica pod platformou .NET Core. Voľba použitej knižnice vzišla hlavne z vlastností jazyku Python, ktorý neposkytuje využívanie generického programovania a programovacích šablón/vzorov (templates) a taktiež nie je silno typovaný. Platforma knižnice využíva jazyk C#, ktorý v porovnaní s inými jazykmi ako napríklad C++ alebo Java poskytuje vyšší stupeň abstrakcie a jednoduchší zápis z hľadiska syntaxe. Knižnica je taktiež v porovnaní s ostatnými najjednoduchšia z hľadiska použitia, pretože pri ostatných knižniciach je potrebné isté množstvo nadbytočných konfigurácií, v knižnici QuantumCSharp je po pripojení pomocou užívateľských údajov a výbere kvantového čipu/simulátora možné okamžite vytvoriť pole s požadovanou konfiguráciou kvantových bitov a odoslať ho na vykonanie a vyhodnotenie.

Nasledovná časť programov v úvode praktickej časti práce reprezentuje výsledky z kvantového počítača, ktoré sú vykonateľné aj na bežnom počítači von neumannovského typu. Tieto programy poskytujú náhľad na spôsob zobrazenia problémov, ktoré svojím spracovaním a výsledkami poukazujú, či a na aké výpočtové problémy má význam použiť kvantový počítač. Druhá časť programov realizuje algoritmy, ktorých osobitá vlastnosť je, že spoľahlivosť alebo rýchlosť ich výsledkov ukazujú, že ich realizácia má zmysel pre spúšťanie na kvantovom počítači. Ak sú aj vypočítateľné na klasickom počítači, ich kľúčová zložka by sa mala prevažne získať iba na kvantovom hardvéri v rozumnom čase.

3.1 Operácie s reťazcami

Práca s reťazcami znakov patrí dnes v najrozšírenejších programovacích jazykoch k bežnej praxi. Pretože výsledky kvantových algoritmov sú vyjadrené číselne, je z tohto pohľadu kvantové zobrazenie rádu znakov pomerne zaujímavá úloha. Ako príklady pre implementáciu na kvantovom počítači som si zvolil dve operácie, a to prevrátenie reťazca a získanie znaku s daným indexom. Obe predstavujú rozdielne prístupy k rozloženiu a zobrazeniu výpočtového problému do programu.

3.1.1 Špecifikácie problémov

V probléme s prevrátením reťazca sa na vstupe vyberá reťazec znakov, napríklad "ráno". Nech indexy tvoria postupnosť 0123. Je potrebné určiť, ktorý znak má akú počiatočnú pozíciu/index, čiže v tomto prípade 0123. Úlohou kvantového počítača je teda na základe dĺžky reťazca korektne invertovať postupnosť indexov, teda na výstupe je očakávaná hodnota 3210, na základe čoho klasický počítač vypíše invertovaný reťazec.

Pri získaní znaku pod indexom sa na vstupe nachádza reťazec znakov, napríklad "ráno". Nech zodpovedajúca postupnosť indexov je 0123. Úlohou kvantového počítača je na základe zakódovaných znakov reťazca na vstupe a vyžiadaní konkrétnych indexov poskytnúť vo výsledkoch požadované znaky.

3.1.2 Metodika postupu pri prevrátení reťazca

Pre každý index reťazca vykonáme kvantové invertovanie samostatne pre viditeľnosť korektnosti výsledku. Pretože kvantový počítač vracia výsledky v intervale <0; 1>, je potrebná, ako býva zvykom pri kvantovom zobrazení, nejaká forma kódovania. V tomto prípade zobrazujem index lineárne, teda každému indexu patrí normalizovaný index z intervalu výsledkov. Index získavame pomocou faktora na základe dĺžky reťazca:

$$faktor = \frac{dlzka}{10}$$

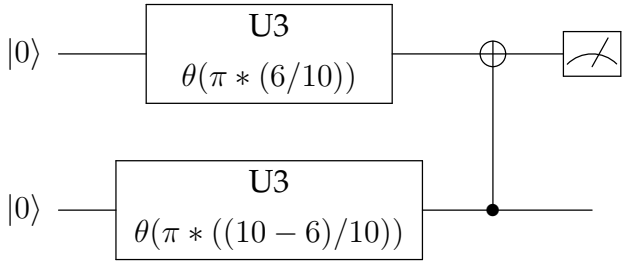
Index potom normalizujeme delením faktorom:

$$normindex = \frac{index}{faktor}$$

Následne sa vytvoria dva kvantové bity aplikáciou brány U3 s hodnotami:

$$q[0] = U3[\theta(\pi * (normindex/10))]$$
$$q[1] = U3[\theta(\pi * ((10 - normindex)/10))]$$

Vykonáme operáciu CNOT prepájajúcu kvantový bit q[0] na q[1] a získame výsledky pre daný index (obr. 3.1):



Obr. 3.1: Operácia CNOT z kvantových bitov q[1] a q[2] pre index znaku

Na základe toho, či je výstup na prvom kvantovom bite vyšší ako 0,5 alebo nie sa výsledok invertovania indexu bude nachádzať na jednom z dvoch výsledkov kvantových bitov. Na konci sa spätne výsledok násobí faktorom a predstavuje invertovaný index. Povedzme teda, že z reťazca o dĺžke 5 invertujeme index 3:

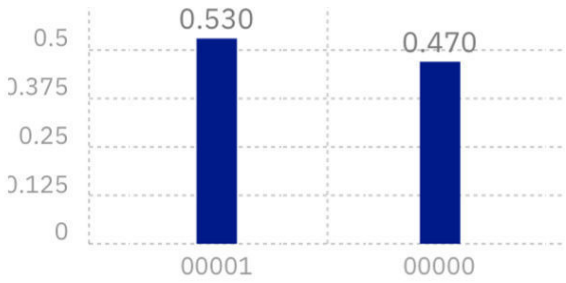
Faktor a normalizácia indexu:

$$faktor = \frac{5}{10}$$

$$normindex = \frac{3}{0,5} = 6$$

Hodnoty kvantových bitov (výsledky na obr. 3.2):

$$q[0] = U3[\theta(\pi * (6/10))]$$
$$q[1] = U3[\theta(pi * ((10 - 6)/10))]$$



Obr. 3.2: Výsledky simulácie operácie prevrátenia reťazca

Je potrebné poznamenať, že kód v projekte bol viacnásobne posielaný na kvantový simulátor aj na ozajstný kvantový počítač. Výpočty sú kvôli presnosti a vyrovnávaniu kvantovej dekoherencie normalizované zaokrúhľovaním. Experimentálne bol taktiež optimalizovaný výpočet v projekte, konkrétne pripočítavaním hodnoty 0.5 k prvej hodnote prvého kvantového bitu a odpočítavaním hodnoty 0.5 od prvej hodnoty druhého kvantového bitu dostávame pri dekoherencii a aj pri rastúcej dĺžke reťazca presnejšie výsledky.

3.1.3 Metodika postupu pri získaní znaku pre daný index

Pre reprezentáciu každého znaku na kvantovom počítači je potrebná istá forma kódovania. Pre potreby tohto programu to budú čísla nad 0.5. Teda napríklad pri reťazci "ráno"bude kódovanie nasledovné:

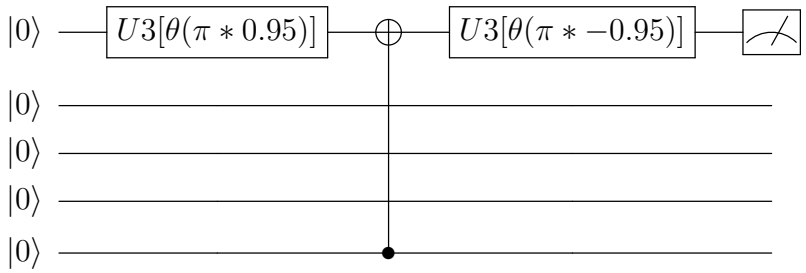
$$r = 0.95$$
$$a = 0.90$$
$$n = 0.85$$
$$o = 0.8$$

Kódovanie som zvolil nad hranicou 0.5, pretože som experimentálne zistil, že čím sú zvolené kódovacie čísla bližšie 1.0, tým dochádza k menšiemu skresleniu výsledkov. Buď vplyvom dekoherencie alebo iných javov vysvitlo, že ak má písmeno kód 0.95 a vyžiadam si ho na výstupe, dostanem číslo v intervale cca (0,99; 0,92). Problém je, ak má písmeno nižšiu hodnotu kódu, teda napríklad pri 0,7 už je možné dostať nespoľahlivý výsledok, z experimentov napríklad 0,83 alebo 0,6. Kvantový výpočet prebieha naraz, teda jedným výpočtom som získal celý výsledok, na rozdiel od výpočtu prevrátenia reťazca. Zobrazenie týmto spôsobom

som zvolil preto, lebo napriek využitiu každého kvantového bitu pre zakódovanie jedného písmena v konečnom dôsledku ovládam výber znakov pod indexmi na vstupe nezávisle na poslednom kvantovom bite, ktorý na výstupe vracia iba žiadané výsledky, teda spracovanie výstupu na klasickom počítači nie je časovo náročné.

Ukážem najprv zobrazenie reťazca o dĺžke 1, teda zobrazujem prvé písmeno na kvantový bit s indexom 0. Otočením osi o hodnotu 0,95 operáciou U3 na riadiaci kvantový bit (ktorý je momentálne pre nás q[4]) a vykonaním operácie CNOT som zakomponoval písmeno do výsledku, inak povedané, v tomto momente som riadiaci kvantový bit prepojil s naším nultým kvantovým bitom, čiže riadiaci kvantový bit sa otočil o rovnaký uhol na osi (obr. 3.3):

$$q[0] = U3[\theta(\pi * 0,95)]$$
$$q[0] = CNOT(q[4])$$
$$q[0] = U3[\theta(\pi * -0,95)]$$



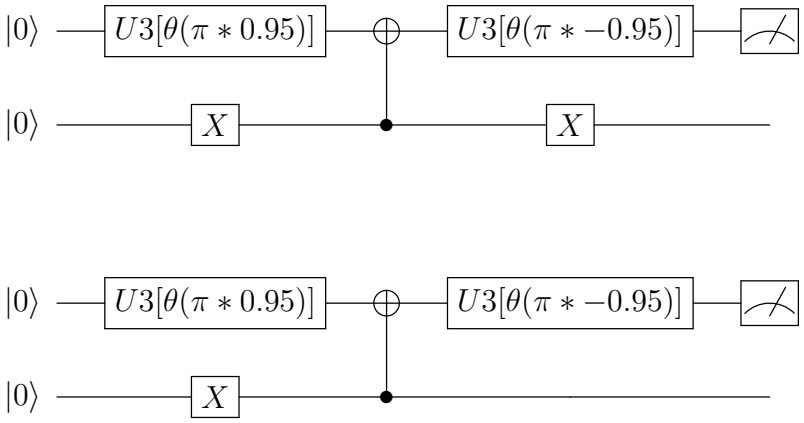
Obr. 3.3: Operácia zobrazenia jedného znaku reťazca

Inverzným otočením pomocou operácie U3 následne nulujem rotáciu nastavenú na prvom kvantovom bite. Toto vynulovanie je potrebné preto, že kvantový bit q[4] kvôli aplikovaniu brány CNOT je prepojený s kvantovým bitom q[0], preto ho môže riadiť. V tomto bode riešenia sa na riadiacom kvantovom bite q[4] nevyžiadava riadenie pred aplikovaním brány CNOT, teda vo výsledku tohto obvodu nebude reprezentovaný stav kvantového bitu q[0].

V situácii že očakávam konkrétne písmeno z reťazca na výstupe, ho očakávam v zakódovanej forme, pričom kód tohto písmena je práve číslo 0.95. Na riadiacom kvantovom bite som umiestnil bránu X pre začiatok a pre koniec takzvaného

riadenia daného písmena, teda jeho výskyt vo výsledku po meraní. Umiestnenie brány X pred operáciu CNOT znamená začiatok riadenia, umiestnenie po operácii CNOT znamená ukončenie riadenia.

Pre jediný kvantový bit (resp. posledný z postupnosti kvantových bitov) sú nasledujúce konfigurácie ekvivalentné, pretože na konci obvodu nemusí byť ukončiť riadenie (obr. 3.4):

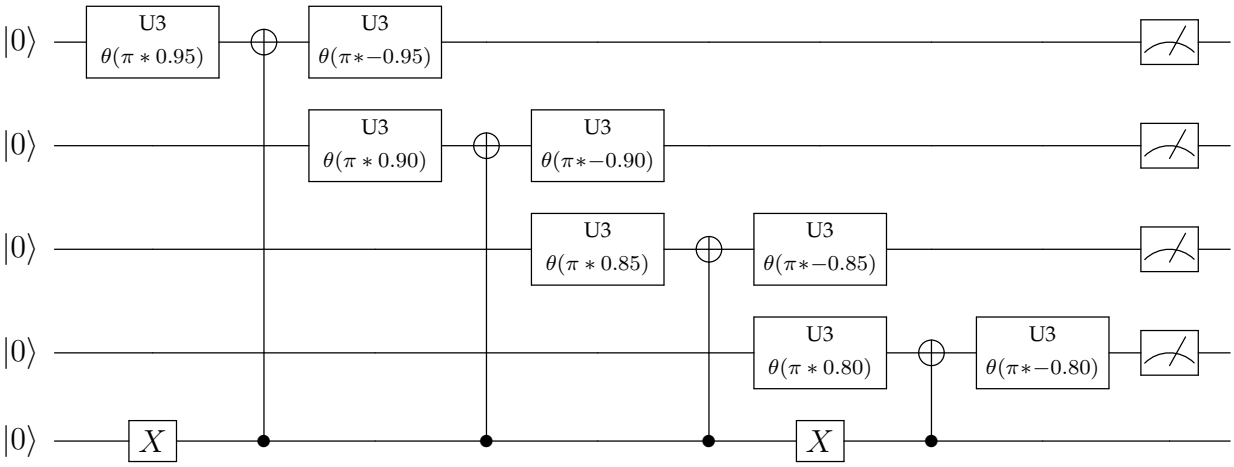


Obr. 3.4: Ekvivalentný zápis pre riadenie prvého a posledného kvantového bitu

Výsledky (obr. 3.5):

Obr. 3.5: Výsledky pri riadení jedného znaku s hodnotou 0.95

V tomto reťazci bol riadený prvý kvantový bit, čiže bol na výstup vyžiadaný prvý znak z reťazca. Vo výsledku bola očakávaná zakódovaná hodnota tohto znaku, pričom ide o hodnotu približne 0,95. Analogicky pre 4 písmená:



Obr. 3.6: Zakódovanie reťazca s dĺžkou štyroch znakov, pričom prvé tri znaky sú očakávané vo výsledku

Preto som do výsledku zakomponoval znaky od prvého po tretie (obr. 3.6). Očakávam teda výsledky na nultom kvantovom bite 0,95, na prvom 0,9 a na druhom 0,85 (obr. 3.7):

Obr. 3.7: Výsledky vyžiadania prvých troch znakov zo štvorznakového reťazca

Teda:

$$q[0] = 0,700 + 0,010 + 0,080 + 0,200 = 0,990$$
$$q[1] = 0,700 + 0,010 + 0,200 = 0,910$$
$$q[2] = 0,700 + 0,010 + 0,080 = 0,790$$

Je evidentné preukázanie javu, že čím kódujem písmená do čísel bližších hodnote 1, tým sú výsledky presnejšie. Pri hodnotách 0,99 (očakávané 0,95) a 0,91 (očakávané 0,90) sú ešte vysledky akceptovateľné, ale pri hodnote 0,79 (očakávané 0,85) už vidno mierne skreslenie. Zvýšením čísel kódovania a spustením na reálnom kvantovom čipe sú výsledky presnejšie. Po získaní výsledkov teda je možné určiť, ktoré zakódované hodnoty sa vrátili, priradiť ich spätne na klasickom počítači ku znakom reťazca a vypísať požadované znaky na výstup.

3.1.4 Súhrn

Z predošlých dvoch experimentov možno vyvodiť závery pre zhodnotenie, či má zmysel algoritmy daného typu vykonávať na kvantovom počítači. V prvom rade tieto kvantové aplikácie nepracovali s kvantovými javmi superpozície a previazania. Ich kódovanie je možné označiť za neekonomické, pretože jeden kvantový bit bol použitý ako základ pre jeden znak reťazca. Teda pri kvantovom počítači s piatimi kvantovými bitmi, ktorý sa dá považovať za relatívne pokročilý, je možné zakódovať nanajvýš reťazce o štyroch znakoch, čo jasne ukazuje, že klasický počítač je pre tieto úlohy oveľa vhodnejší. Pri tomto predpoklade taktiež netreba zabudnúť na fakt, že je potrebné prispôsobiť sa topológii kvantových bitov, teda nie je možné vykonávať operáciu CNOT ľubovoľnými smermi z každého kvantového bitu. Aplikácie využívali len rotáciu osí na jednotlivých kvantových bitoch a pomocou riadiaceho kvantového bitu boli vyžiadané zakódované znaky na výstupe. Výhody efektívneho kódovania informácií, ktoré môžu byť uložené do kvantových bitov, teda neboli využité. Lepší prístup k návrhu kvantových algoritmov by spočíval vo využívaní výhod, ktoré kvantové počítače poskytujú, napríklad zakomponovaním superpozície alebo využitím paralelizmu.

3.2 Generovanie náhodných čísel

Generovanie náhodných čísel je jeden z najpoužívanejších typov algoritmov. Využívajú ho rôzne aplikácie od hier cez štatistiku až po kryptografické algoritmy. Dôležitý aspekt pri využívaní generovania takýchto čísel je miera náhodnosti, ktorá je potrebná na výstupe. Niektoré algoritmy nevyžadujú vysokú mieru náhodnosti a dokonca je akceptovateľný prípad, ak sa čísla v niektorých momentoch zopakujú. Do tejto triedy patrí napríklad algoritmus, ktorý vyberá náhodnú

skladbu z hudobnej kolekcie. Ďalšie dôležité kritérium pre generovanie je typ čísel na výstupe. Niektoré kryptografické algoritmy napríklad požadujú veľké náhodné prvočíslo ako základ pre šifrovací kľúč. Pri náhodných prvočíslach je o to potrebnejšie, aby ich miera náhodnosti bola čo najvyššia.

3.2.1 Špecifikácia problému

Klasické počítače produkujú iba takzvané pseudonáhodné čísla. Ide o čísla, ktoré na základe daného algoritmu disponujú akceptovateľnou mierou náhodnosti, ale nikdy nejde o absolútne náhodné čísla. Úloha bola za pomoci kvantového počítača vygenerovať také náhodné čísla, ktorých náhodnosť je absolútna.

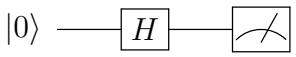
3.2.2 Metodika

Absolútne náhodné čísla je možné získať jedine na základe fyzikálne založenej udalosti, teda napríklad hod kockou alebo mincou, zatočenie kolesom a podobne. Tieto udalosti sú absolútne náhodné, ako opisuje vo svojej práci Pathak [21] pri implementácii kvantových náhodných čísel, práve preto že nedokážeme prepočítať výsledok na základe odporu vzduchu, hmotnosti mince, aplikovanej sily, výšky hodu a podobne. Kvantová náhodnosť je nielen inherentná ale aj absolútna, práve preto, že jej výsledky nedokážeme presne predpovedať pred meraním. Ukazuje sa, že kvantový počítač je vhodný na generovanie absolútne náhodných čísel, pretože výsledky na výstupe sú založené na simulácii a meraní fyzikálne založených kvantových javov a nie softvérového algoritmu. Algoritmická časť kvantového programu realizujúceho generovanie náhodných čísel iba nastaví kvantové stavy na požadované hodnoty, ale úkon merania na základe kvantových princípov napríklad superpozície, poskytuje absolútne náhodné výsledky. Začal som preto generovaním malých čísel, pričom ukazujem, ako sa dostať k väčším.

3.2.3 Generovanie núl a jednotiek

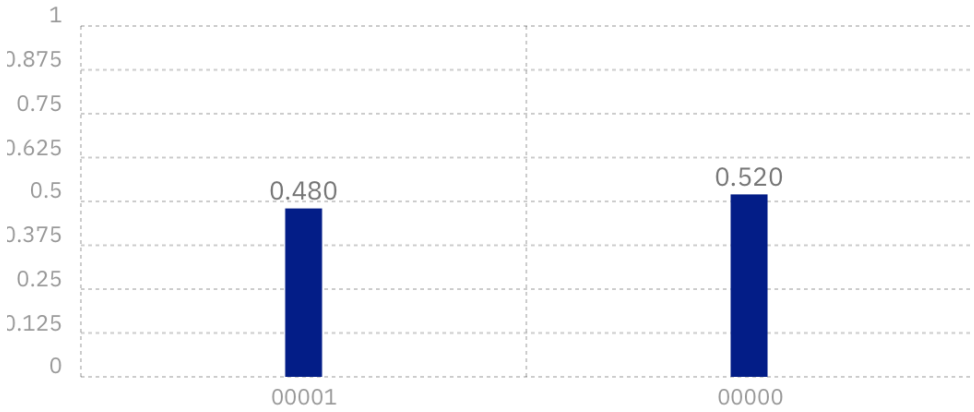
Pre začiatok je dobré uviesť, aký algoritmický prístup je vhodný pre efektívne generovanie malých náhodných čísel. Pri pohľade na brány, ktoré máme na praktickom kvantovom počítači k dispozícii, je možné generovať náhodné čísla rôznymi spôsobmi. V algoritme, o ktorom sa dá povedať, že potrebuje kvantový počítač, je potrebné využiť kvantové javy, ktoré na klasickom počítači nemáme k dispozícii.

Pri generovaní môžeme využiť taktiež dekoherenciu, teda získavanie neočakávaných hodnôt v porovnaní s tými, ktoré boli nastavované na kvantových bitoch. Najjednoduchší prístup je využitie superpozície. Hadamardova brána už zo svojej podstaty dostane kvantový bit do neurčitého kvantového stavu, z ktorého je možné meraním získať kolaps do náhodnej hodnoty (obr. 3.8):



Obr. 3.8: Najjednoduchší algoritmus generovania kvantových čísel

Aplikovanie Hadamardovej brány na kvantový bit a opakované následné meranie ukazujú, že výsledky očakávane oscilujú okolo hodnoty 0.5 (obr. 3.9):

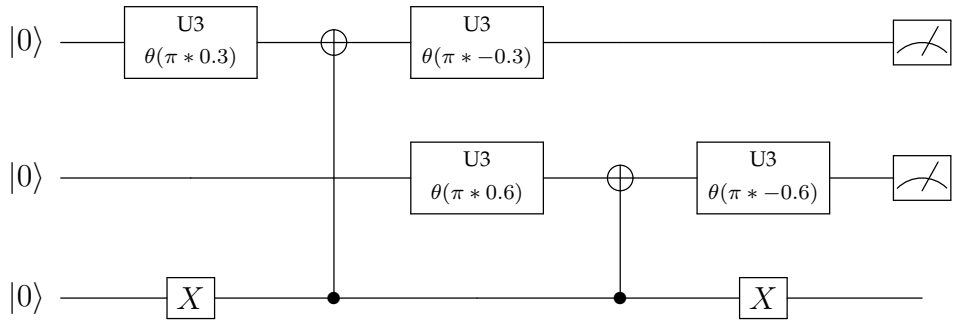


Obr. 3.9: Výsledky po aplikovaní jednej Hadamardovej brány

Ak berieme za smerodajný výsledok na prvom kvantovom bite, dostávame náhodné číslo 0.48. Pretože aplikovaním Hadamardovej brány budú na výstupe zväčša čísla blízke hodnote 0.5, zaokrúhlením výsledku dostávame buď hodnotu 0 alebo 1, čo predstavuje naše absolútne náhodné číslo. Nie je možné z pochopiteľných dôvodov zobrať do úvahy obidve čísla výsledku pri meraní jedného kvantového bitu, teda kvantový stav odmeraný na prvom kvantovom bite a zároveň hodnotu, ktorá nie je na žiadnom kvantovom bite, ako komponenty generovaného náhodného čísla. Je to preto, že nikdy nedostaneme výsledok 00 alebo 11, teda druhá hodnota po výsledku bude vždy jej opačná hodnota, čo je predvídateľné a nenáhodné. Väčšie náhodné číslo môžeme získať napríklad opakovaným meraním. Je očividné, že pri n kvantových bitoch dostaneme n náhodných jednotiek a núl, teda séria týchto čísel predstavuje väčšie náhodné číslo.

3.2.4 Generovanie náhodných čísel pomocou dekoherencie

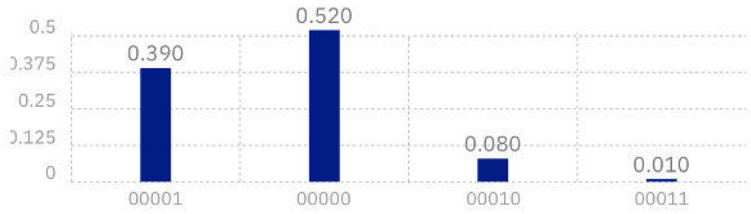
Otázka pri každom algoritme je, či je dostatočne efektívny, pričom pri kvantových algoritmoch pojednávame o efektivite zobrazenia. Ak by sme chceli využiť menej kvantových bitov pre generovanie väčších čísel, je možné ísť cestou využitím dekoherencie, teda zakomponovania nepredpokladaných výsledkov vyplývajúcich nedokonalostí fyzickej realizácie kvantového počítača. Pri využití javu z predošlých príkladov operácií s reťazcami v kapitole 3.1.2 sa v prípade otočenia okolo osi Y, operácií previazania s prvým kvantovým bitom a spätnom otočení o rovnaký uhol vyskytne kolaps. Pri tomto kolapse kódujem počiatočnú hodnotu 0.5, avšak na výstupe môžem dostať vyššie alebo nižšie hodnoty s výraznou odchýlkou, pričom čím nižšie počiatočné hodnoty využívam, tým vyššia je odchýlka a teda aj náhodnosť. Pri práci so znakmi reťazcov tieto odchýlky neboli žiadúce, pretože vstupné hodnoty sme po vyžiadaní očakávali na výstupe v pokiaľ možno nezmenenej forme, avšak pri generovaní náhodných čísel môžeme využiť práve tieto nižšie hodnoty:



Obr. 3.10: Generovanie pomocou nízkych rotácií

Tento prístup využíva dve počiatočné čísla 0.3 a 0.6, pričom pri dvoch číslach je potrebné využiť tri kvantové bity kvôli riadiacemu bitu, ktorý určuje, ktoré hodnoty sa budú brať do výsledku. Hodnota na jednom kvantovom bite v predošlom príklade nie je dostatočné náhodná, ale náhodnosť z hodnôt v tomto príklade je postačujúca, pretože odchýlka od ich počiatočných hodnôt je vyššia (obr.3.11).

Z čísel 0.3 a 0.6 sme získali náhodné čísla 0.39 a 0.52, pričom aj zvyšky 0.08 a 0.01 sa dajú zakomponovať do výsledného náhodného čísla, pretože z prvých dvoch náhodných hodnôt nie je možné vydedukovať ich hodnoty pred meraním. Pretože výsledky z kvantového počítača spracúvavame na klasickom počítači, naše



Obr. 3.11: Zobrazenie dvoch počiatočných hodnôt pre vygenerovanie vyššieho náhodného čísla

výsledné náhodné číslo môže byť 3952, alebo 39520801. Zmysluplnou modifikáciou tohto čísla neznížime ani nezvýšime jeho náhodnosť, ale môžeme ho obfuskovať a pracovať s ním ďalej.

3.2.5 Generovanie pomocou iterácie

Predošlé dva prístupy boli vykonávané ako jedno meranie jedného obvodu. Kedže ale klasický počítač môže čakať na viacero výsledkov, kde sa zoskupia do jedného čísla, je možné spojiť výsledky z n meraní do väčšieho výsledku, pričom miera jeho náhodnosti bude identická s náhodnosťou jeho komponentov. Pri vykonávaní kvantového obvodu máme k dispozícii takzvaných nástrelov, ktoré prakticky znamenajú opakovanie získavania výsledkov do takej miery, až kým sa nedosiahne požadovaná presnosť výsledku. Kvantový výsledok je pri implementácii generátora náhodných čísel korektný aj keď sú nástrely nastavené na hodnotu 1, nástrel mení len rozsah tohto výsledku. Pri nastavení nástrely na hodnotu napríklad 2048 sa kvantový výsledok vo webovom rozhraní IBM QX nemení, pretože implementácia zobrazenia výsledkov ich vždy zobrazuje medzi hodnotami 0 a 1. Pri vytváraní kvantových obvodov napríklad pomocou knižnice $QuantumCSharp$ však pri nastavení vysokej hodnoty nástrelov vracajú z kvantového počítača väčšie, ešte neznormalizované čísla (obr. 3.12):

[0]	{QuantumCSharp.Ibm.QubitValueResult}
Index	0
One	993
Zero	1055

Obr. 3.12: Programaticky získané výsledky pri nástreloch nastavených na 2048

3.3 Hra míny

Populárnou aplikáciou nových technológií sú hry, práve preto, že ľudia na nich môžu pochopiť, aký význam daná technológia má. Samotná spoločnosť IBM má na mobilných zariadeniach a webovej stránke rôzne edukačné hry, ktorými ukazujú princípy kvantových brán a algoritmov. Samotná hra nemôže samozrejme, ako pri predošlých experimentoch, celá byť spustená na kvantovom počítači, ale kvantový počítač môže poslúžiť ako miesto pre medzivýpočty. Nemá zmysel vytvárať hru s novými pravidlami, je lepšie je využiť existujúci koncept hry s overenými princípmi.

3.3.1 Špecifikácia problému

V komunite programátorov na IBM QX už vznikli niektoré herné projekty, napríklad implementácia populárnej hry loďky, kde kvantový počítač vyhodnocuje, či hráč dokázal trafiť loď cudzieho hráča náhodným výstrelom. Úlohou bolo teda vytvoriť jednoduchú hru na klasickom počítači, kde bolo potrebné jednotlivé podvýpočty súčastí hry realizovať na kvantovom počítači.

3.3.2 Metodika vytvorenia herného poľa

Pre účel vytvorenia interaktívnej aplikácie som si zvolil základ implementácie populárnej hry míny, kde sa hráč na predvygenerovanom poli $n \times n$ pokúša odkryť všetky políčka tak, aby neodkryl také, pod ktorým je mína. V klasickej verzii hry sa v prípade odkrytia ukážu v jeho okolí počty mín (obr. 3.13).

Obr. 3.13: Klasická verzia hry míny vo vyriešenom stave

V kvantovej verzii mín sa odkrývajú políčka, pod ktorými sú čísla indikujúce stav hry. Pretože hra je spustená v termináli, ovládanie musí byť jednoduché a teda nemá zmysel pri tak malej veľkosti herného poľa označovať políčka vlajkami. Odkryté čísla ukazujú, aká je pravdepodobnosť výskytu míny vo vzťahu na aktuálne miesto. Pre jednoduchosť hry som teda zaviedol pravidlo, že odkryté číslo indikuje najvyššiu pravdepodobnosť výskytu míny zo všetkých okolitých políčok. Teda ak odkryjem políčko a dostanem nízku pravdepodobnosť, môžem bezpečne odkryť hociktoré z okolitých políčok. Ak je pravdepodobnosť vysoká, bolo by lepšie začať odkrývať mimo tejto zóny. V originálnej verzii hry míny slúži klasický počítač na kľúčové výpočty generovania hracieho poľa, teda generujeme náhodné čísla. Pri vytváraní hry bol k dispozícii kvantový počítač IBM QX s kódovým názvom Rüschlikon so 16 kvantovými bitmi, teda bolo možné vygenerovať hracie pole o veľkosti 4x4. Kvantový program na generovanie poľa teda vytvorí 16 náhodných čísel:

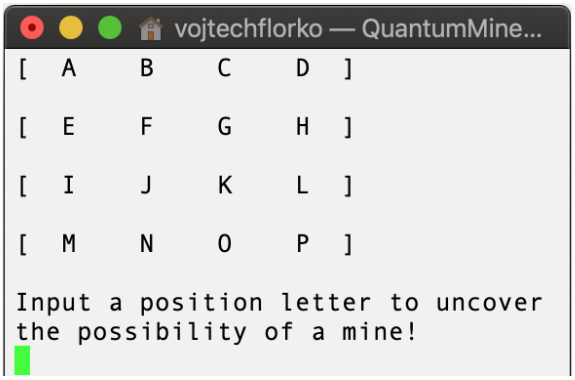
```
Qubit[] qubits = new Qubit[QubitNumber];
for(int i = 0; i < QubitNumber; i++)
{
     qubits[i] = new Qubit(program, i);
     qubits[i].H();
}
```

V tomto kóde sa alokuje pole pre naše kvantové bity a v cykle sa na nich aplikuje Hadamardova brána. Program nastavíme na 1024 nástrelov, teda v aplikácii získavam trojciferné až štvorciferné výsledky. Po vykonaní kvantového programu pozostávajúceho z týchto kvantových bitov je potrebné výsledky zozbierať. Pravdepodobnosť prirodzene vyjadrujem pre herné políčka percentuálne, teda pri výsledkoch náhodne generovaných čísel vyšších ako sú percentuálne hodnoty je potrebná normalizácia. Ako vhodná sa javí operácia modulo s číslom 100, teda náhodné percentuálne vyjadrenie pravdepodobnosti výskytu míny je tvorené zvyškom po delení týmto číslom. Pri veľkosti výsledku napríklad 953 alebo 1115 dostanem teda pravdepodobnosti 53% a 15%.

```
short[] quantum_values = new short[QubitNumber];
for(int i = 0;i < QubitNumber;i++)
{
```

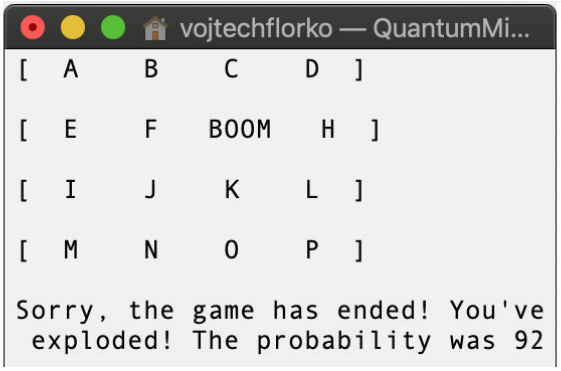
```
        quantum_values[i] = (short)(tmp[i].One % 100);
}
```

V tomto bode vykonávam alokáciu dátovej štruktúry pre finálne náhodné čísla pre naše herné pole. V dátovej štruktúre s názvom tmp mám už kvantové výsledky po meraní z kvantového počítača, pričom .One predstavuje prvú hodnotu z kvantového výsledku. Po vygenerovaní čísel program prechádza ku vykresleniu herného poľa (obr. 3.14):



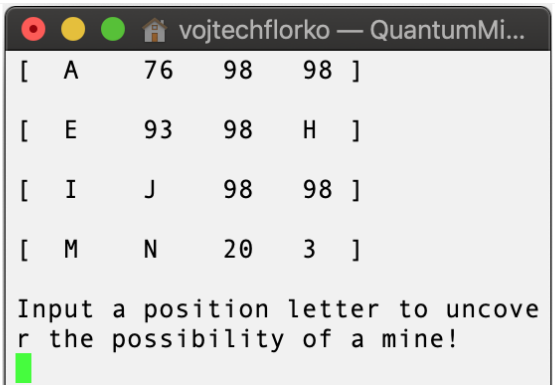
Obr. 3.14: Počiatočný stav kvantovej hry míny

Princíp kvantovej verzie hry míny je odkryť všetky, alebo čo najviac políčok bez vybuchnutia. Problém hry v aktuálnom stave je, že herné pole obsahuje 16 náhodných čísel rôznych veľkostí. Prvé odkrytie políčka je podobne ako v klasických mínach náhodné. To znamená, že po okamžitom odkrytí míny musí hráč hru reštartovať (obr. 3.15):



Obr. 3.15: Odkrytie kvantového výsledku s vysokou pravdepodobnosťou výskytu míny

Aby hra pripomínala klasické míny v zmysle odkrývania políčok bez mín, bolo by optimálne, aby pravdepodobnosti výskytu mín pri približovaní sa k míne stúpali a pri oddiaľovaní sa od mín klesali. Z tohto dôvodu odkrývanie políčok v tejto hre neukazuje hodnotu pravdepodobnosti výskytu míny pod nimi, ale hodnotu políčka s najvyššou pravdepodobnosťou v bezprostrednom okolí. Odkrytím míny sa rozumie nájdenie dostatočne vysokého čísla, ktoré indikuje, že mína sa s najväčšou pravdepodobnosťou bude nachádzať na tomto políčku. V tomto programe ide o číslo vyššie ako 85. Po odkrytí prvého políčka sa všetky okolité hodnoty znormalizujú na klasickom počítači za použitia kvantových výsledkov. Ak teda nájde hráč vyššiu hodnotu, napríklad 59 a políčko vedľa obsahuje číslo 71, zvolí sa zvyšok kvantového výsledku, teda nie hodnota .One, ale .Zero ako počiatočná hodnota, aby sa znížila šanca, že hráč narazí okamžite na mínu. Ak odkryje príliš nízku hodnotu, pravdepodobnosť výskytu míny môže stúpnuť na políčku pri ďalšom odkrytí (obr. 3.16):

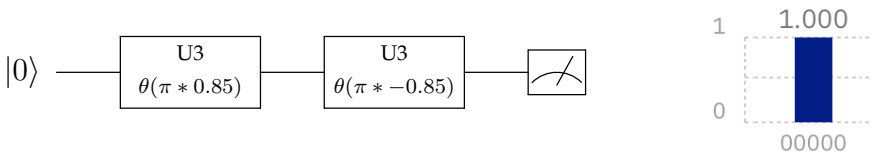


Obr. 3.16: Postup odkrývaním hracieho poľa a klesanie hodnôt pri posúvaní sa od odkrytej hodnoty

Na obrázku 3.16 začínam odkrytím políčka s označením F a posunul som sa na vedľajšie políčko s označením G. Je očividné, že pravdepodobnosť 93 sa nevyskytuje na spoločných najbližších políčkach znakov F a G, pretože najvyššia pravdepodobnosť nie je rovnaká. Takto postupne strategicky odkrývam zakryté políčka a postupujem k úspešnému ukončeniu hry. Je teda vidieť, že pod políčkom s označením H sa jednoznačne nachádza mína s pravdepodobnosťou 98%, ako to ukazujú okolité odkryté políčka. Takouto izoláciou všetkých mín hra môže úspešne skončiť. Kvantový počítač som však v tejto hre, podobne ako v už spomínaných lodičkách, využil aj na vyhodnotenie hry.

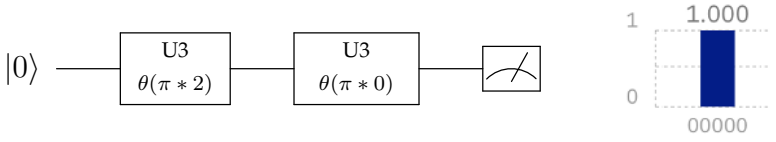
3.3.3 Metodika vyhodnotenia

Pravidlo pre výskyt míny je odkrytie políčka s príliš vysokou pravdepodobnosťou, teda musí existovať nejaká vopred vybraná hranica, po ktorej považuje hra odkrytie za výbuch. Ako hranicu som zvolil hodnotu 85%. Kvantový algoritmus musí byť v prípade vyhodnotenia postavený na základe dvoch vstupov, ktoré predstavujú náhodu a číslo, ktoré vyhodnocujem pre výskyt míny. Keď som chcel vykonať porovnanie na základe predurčenej hodnoty 85, musel som určiť, ktorá hodnota z dvojice (naše políčko, predurčená hodnota), je vyššia. Pretože kvantový počítač nedisponuje porovnaním dvoch čísel, je potrebné využiť iné vlastnosti nastavovania a vyhodnocovania kvantových bitov. Keďže kvantový počítač vracia výsledky v rozmedzí 0 a 1, jasný výsledok je najjednoduchšie reprezentovaný kvantovým stavom a zvyškom, pričom hodnota vyčísleného kvantového stavu nad 0.5 znamená pozitívny výsledok a v opačnom prípade negatívny výsledok. V kapitole 3.1.2 bolo ukázané, ako pomocou rotácie bránou U3 je možné nastaviť uhol θ po osi Y a vynulovať jeho hodnotu, čo som využil aj v tomto prípade (obr. 3.17):



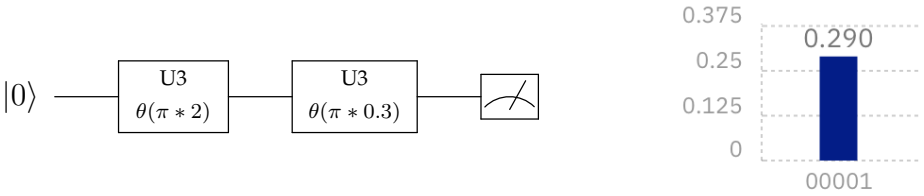
Obr. 3.17: Rotácia θ o 0.85π nasledovaná rotáciou o -0.85π s výsledkom 0

Veľkosť jedného z čísel som teda mohol určiť práve z hodnoty, ktorú dostanem vo výsledku. Ak je celkový výsledok nulový, obidve čísla sa rovnajú. V kapitole 3.1.2 som však aj uviedol experimentálne zistenie, že pri nižších hodnotách ako 0.8 sa výsledky výrazne skresľujú, teda je potrebné túto metodiku vyhodnocovania modifikovať kvôli potrebnej presnosti. Ak zvolím vyššie číslo, výsledok by mal byť presnejší. Tento problém bolo potrebné riešiť opačnou rotáciou (obr. 3.18):



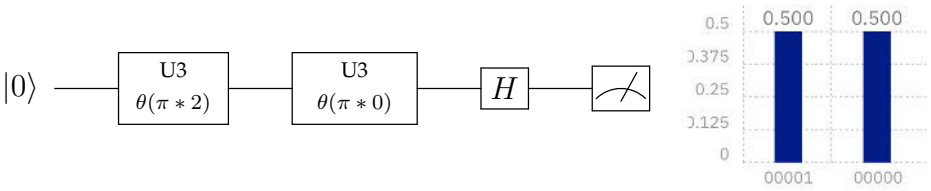
Obr. 3.18: Porovnanie dvoch núl

Do prvej rotácie (v tomto prípade 2π) zatiaľ nebola zakomponovaná prvá hodnota, ktorú je potrebné porovnávať. Pričítaním tejto hodnoty k rotácii v prvej bráne U3 dochádza k dodatočnej rotácii s otáznou presnosťou výsledku (obr. 3.19):



Obr. 3.19: Rotácia o 2π nasledovaná rotáciou o 0.3π s výsledkom ± 0.3

Ak je potrebné porovnať dve hodnoty a je vidieť, že druhá dodatočná rotácia zvyšuje hodnotu výsledku, je potrebné odčítať jednu hodnotu od druhej a zanalyzovať výsledok. Pre umiestnenie prvej hodnoty do prvej rotácie je potrebné odčítať túto hodnotu od 2π. Pri porovnávaní hodnôt 0.6 a 0.2 získavam $(2 - 0.6)\pi$ a 0.2π, čiže 1.8π. Prvé číslo musí byť menšie, pretože druhé nedokázalo výslednú rotáciu posunúť do ďalšieho kvadrantu. Pri opačnom prípade, teda $(2 - 0.2)\pi$ a 0.6π získavam 2.4π, čiže 0.4π, to znamená, že rotácia sa nachádza v predošlom kvadrante. Je očividné, že 2π je hodnota, ktorá reprezentuje rovnosť medzi porovnávanými hodnotami. Nastavenie rotácií na nulový stav pre obe hodnoty (teda 2π a π) však ukazuje ďalší problém. Výsledok je nesprávny kvôli tomu, že výsledky sú merané z pohľadu smerom na os X. Výsledky sa nedajú odmerať, pretože je na to potrebný pri meraní pohľad na osi X a Y. Tento pohľad sa dá nastaviť rotáciou smerom na os Z. Táto operácia je dosiahnuteľná použitím Hadamardovej brány (obr. 3.20):



Obr. 3.20: Využitie Hadamardovej brány na zmenu bázy a pozorovania z osi Z

Týmto spôsobom je odkrytie políčka analyzované pre kontrolu výskytu míny. Po každom odkrytí sa na kvantovom počítači porovnáva odkrytá hodnota s vybranou hodnotou potrebnou pre koniec hry. Ak je odkrytá hodnota vyššia, hra končí. Ak je hodnota nižšia, hra pokračuje ďalej až kým sa neminú všetky políčka, na ktorých sa nenachádza mína, čo značí úspešné dohranie hry.

3.4 Optimálna cesta na mape

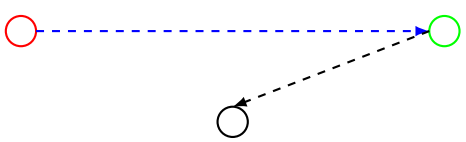
K problémom, ktoré predstavujú obtiažne výpočtové úlohy pre klasický počítač a zároveň ich teoreticky možno realizovať kvantovým obvodom, by mohli v budúcnosti patriť problémy optimalizácie ciest. S týmto úzko súvisí známy problém obchodného cestujúceho, kde je cieľom nájsť z ľubovoľnej siete najkratšiu cestu medzi dvoma bodmi a naspäť, pričom jednotlivé cesty majú rôzne dĺžky.

3.4.1 Špecifikácia problému

Úlohou bolo vyvinúť rôzne prístupy k výberu vhodnej reprezentácie siete pomocou kvantového obvodu, pričom bolo potrebné vyhodnotiť ich a vybrať najvhodnejší z nich na základe pokiaľ možno najnižšej hardvérovej aj časovej náročnosti.

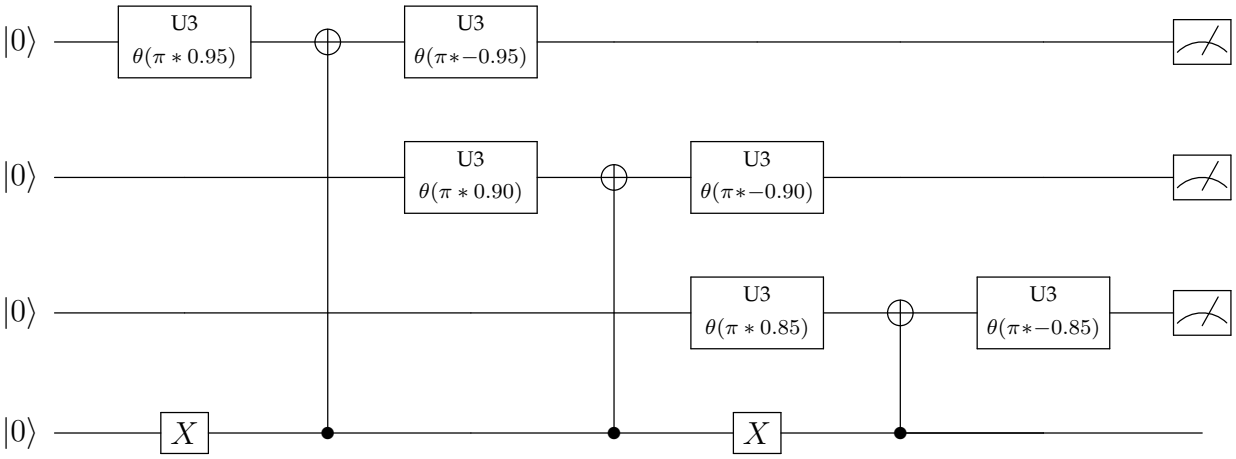
3.4.2 Naivná reprezentácia celej siete

Kým klasický programovací prístup by prehľadával graf do hĺbky, kvantový počítač potrebuje iba efektívny spôsob reprezentácie problému (alebo jeho častí). Tento proces je možné začať napríklad naivným spôsobom, podobne ako pri reprezentovaní reťazcov znakov, využívaním kvantových bitov ako uzlov a riadiacej vetvy ako zvolenou cestou. Uvažujme nasledujúci graf (obr. 3.21):



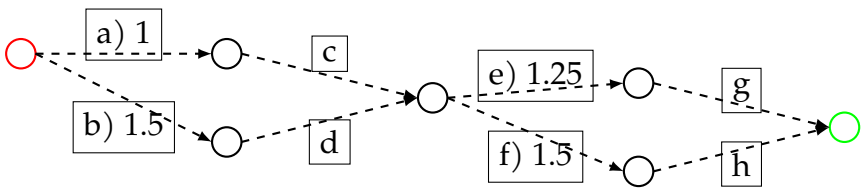
Obr. 3.21: Jednoduchý graf pre priamu reprezentáciu kvantovými bitmi

Daná sieť/mapa disponuje troma uzlami/mestami a dvoma cestami medzi nimi. Jej jednoduchosť samozrejme neumožňuje výber medzi dvoma rôznymi cestami, teda neobsahuje križovatku, avšak slúži na ukázanie možného prístupu ekvivalencie jednotlivých prvkov grafu, ktoré sa mapujú na kvantové bity, kde cesta sa vyberá pomocou riadenia na riadiacom kvantovom bite (obr. 3.22):



Obr. 3.22: Pokus o zobrazenie uzlov z grafu na obrázku 3.21

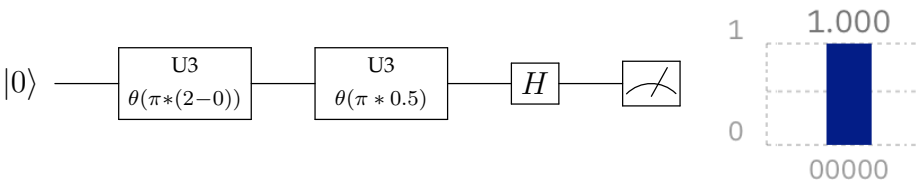
Prvý uzol je reprezentovaný kvantovým bitom s rotáciou θ o 0.95π a analogicky to platí pre ostatné uzly grafu na ostatných kvantových bitoch. Riadenie cesty, ktorá je indikovaná modrou šípkou na obrázku 3.21 sa nachádza medzi dvoma bodmi s rotáciami 0.95π a 0.90π. Pri zhodnotení prístupu jednoznačne klesá efektivita využitia kvantových bitov. Pri veľkom množstve kvantových bitov by bolo možné expanzívnejšie zobrazenie, avšak nie je k dispozícií dostatočný počet kvantových bitov. Z algoritmu v kapitole 3.3.2 je ukázané, ako je možné rozhodovať pri výbere medzi dvoma vetvami. Uvažujme zložitejší graf (obr. 3.23):



Obr. 3.23: Rozhodovanie medzi dvoma nasledovnými križovatkami

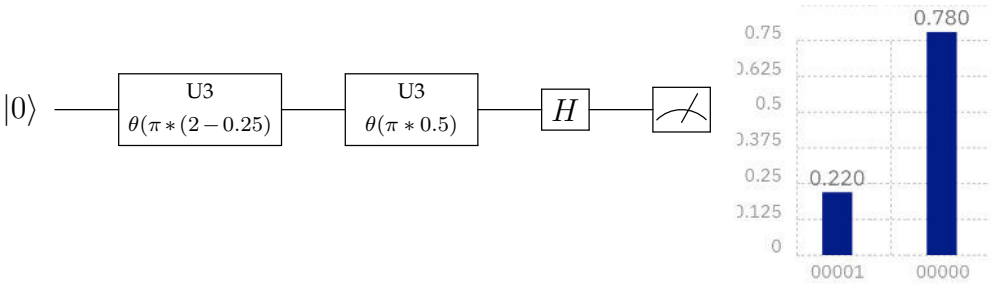
3.4.3 Granulárne zobrazenie križovatiek

V tomto prípade som sa opäť snažil dostať z uzla označeného červenou farbou do uzla označeného zelenou farbou. Pri prechode stoja v ceste dve križovatky, pričom graf bol navrhnutý tak, že z každej dvojice medziuzlov križovatky je dĺžka cesty smerujúca do výsledného uzla identická, teda cesty c) a d) a cesty g) a h) sú dĺžkou navzájom rovnaké. Celkový počet možných ciest medzi začiatočným a cieľovým uzlom je 4, ak sa samozrejme neberie do úvahy návrat opačným smerom, čo by bolo pri hľadaní najkratšej cesty kontraproduktívne. Pri riešení prvej križovatky sa naskytlo riešenie v podobe porovnania, ktoré bolo vysvetlené v kapitole 3.3.3 (obr. 3.24):



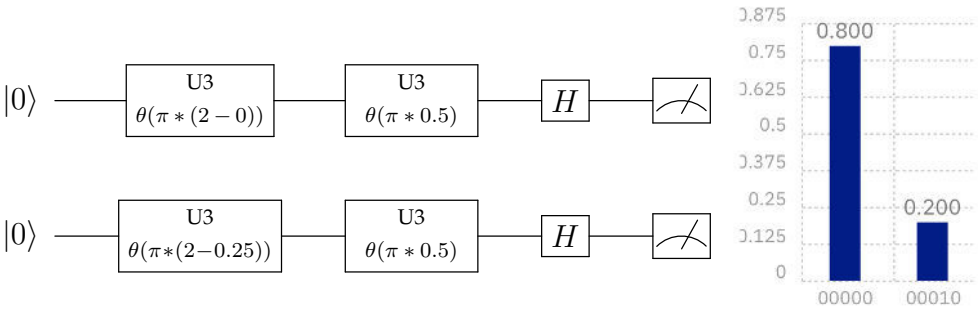
Obr. 3.24: Riešenie prvej križovatky

V tomto obvode je viditeľné, že na bránach U3 sú nastavené hodnoty 0 a 0.5. Je to z dôvodu, že rotácia o 1 a 1.5 je zbytočná a je dobré pre stabilitu algoritmu nerotovať prehnane, teda porovnávať len hodnoty bez spoločnej celej časti, nakoľko výsledok by bol aj tak korektný. Riešenie druhej križovatky je teda (obr. 3.25):



Obr. 3.25: Riešenie druhej križovatky

Výsledky riešení ukázali, že najkratšie cesty sú a) a e). Zápisy riešení sú uvedené ako samostatné operácie, čo je výpočtovo menej náročné pre klasický počítač, avšak trvá dvojnásobne dlhšie na kvantovom počítači. Pretože kvantový stav na prvom kvantovom bite z dvojice výsledkov predstavuje hľadaný výsledok, operácie je možné vykonávať paralelne a čas trvania výpočtu oproti jednému behu o jednom kvantovom bite sa nemení (obr. 3.26):



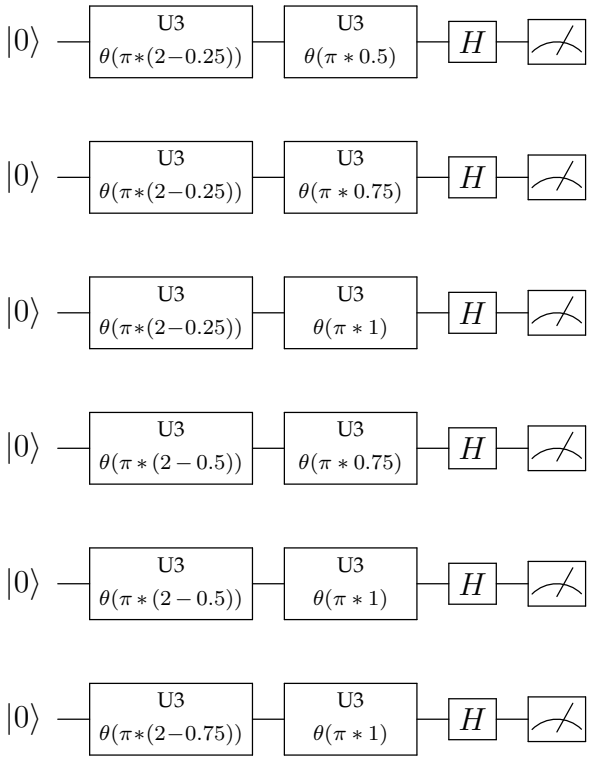
Obr. 3.26: Riešenie oboch križovatiek naraz v jednom výpočte

Pretože ide o jednoduchú sadu križovatiek, forma výsledkov nie je príliš komplikovaná avšak pri zvyšujúcom sa paralelizme by stúpala komplexita rozloženia kvantových výsledkov. Pre prvý kvantový bit vyšla hodnota 0 a pre druhý hodnota 0.2, z čoho je možné vyčítať, že v oboch kvantových bitoch pre nás vyšla pozitívne cesta zakódovaná prvou bránou U3.

3.4.4 Zobrazenie všetkých ciest

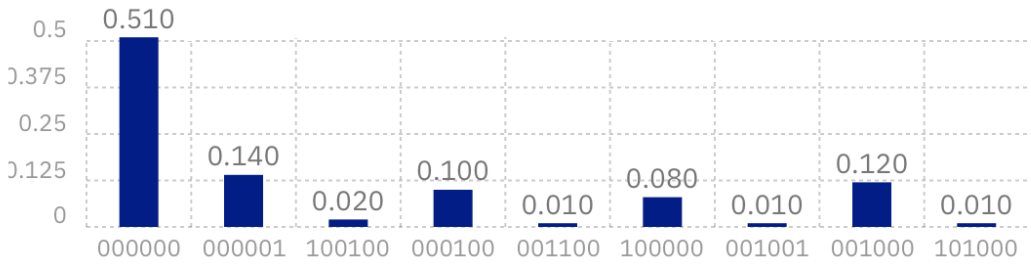
Keďže máme štyri možné cesty a zvyšné nevyužité kvantové bity, bolo možné vyvinúť aj ďalší, rigoróznejší paralelný výpočet, ktorý namiesto jednotlivých križovatiek spracoval všetky cesty grafu horizontálnym prístupom. Cesty sa pohybujú v rozmedzí od hodnôt 2 do 3, preto som odčítal z každej výslednej cesty hodnotu 2. Pri takomto jednoduchom príklade sa môže zdať, že vypočítavam už výsledné cesty na klasickom počítači a tie posielam na kvantový počítač, čo sa javí zbytočné, pretože môžeme výsledné cesty zoradiť na klasickom počítači a netrápiť sa s kvantovou zložkou. Tento argument je sčasti pravdivý, avšak riešenie je možné škálovať na väčšie grafy a cesty zadávať do kvantového počítača anonymným spôsobom, teda napríklad hodnota cesty s dĺžkou 2.25 nebude predprocesovaná ako

$2.25 - 2$ a umiestnená do brány U3$(pi * 0.25; 0; 0)$, ale výpočet bude vykonaný v samotnej bráne: U3$[\theta(pi * ((1 + 1.25) - 2))]$. V tomto bode predpokladám, že existuje program, ktorý na klasickom počítači spracuje graf (alebo podgraf grafu) a vytvorí zoznam všetkých jeho možných ciest na vstupe, pričom tento zoznam bude vstupom pre kvantový algoritmus a hodnoty na bránach budú obsahovať výpočty jednotlivých porovnaní. Obvod, ktorý sa pošle v prípade grafu z obrázka 3.23, bude vyzerať nasledovne (obr. 3.27):



Obr. 3.27: Obvod riešenia so všetkými cestami naraz

Porovnával som každú cestu s každou cestou, čo činilo dokopy šesť výpočtov za jednu časovú jednotku. Ako bolo uvedené, je očividné, že cesta s dĺžkou 0.25 je najkratšia, avšak cieľom tejto časti práce bolo navrhnúť algoritmus, ktorý na brány mapoval súčty parciálnych ciest, ktoré nie sú dopredu známe a vypočítané. Tento výpočet teda využil 6 kvantových bitov, čo nie je možno najefektívnejší spôsob zobrazenia, avšak demonštruje odlišný prístup. Výsledky tohto behu sú nasledovné (obr. 3.28):



Obr. 3.28: Výsledky zobrazenia križovatky z obr. 3.23 pomocou šiestich kvantových bitov

Pri čítaní výsledkov je rozhodujúce, pri ktorom kvantovom bite je ktorá hodnota menšia, pričom víťazí cesta, ktorej reprezentujúca hodnota sa opakuje najčastejšie. Po spracovaní sčítaním hodnôt na jednotlivých kvantových bitoch boli výsledky nasledovné:

cesta a-e vs. cesta a-f 000001 = 0.15
cesta a-e vs. cesta b-e 000010 = 0
cesta a-e vs. cesta b-f 000100 = 0.31
cesta a-f vs. cesta b-e 001000 = 0.15
cesta a-f vs. cesta b-f 010000 = 0
cesta b-e vs. cesta b-f 100000 = 0.11

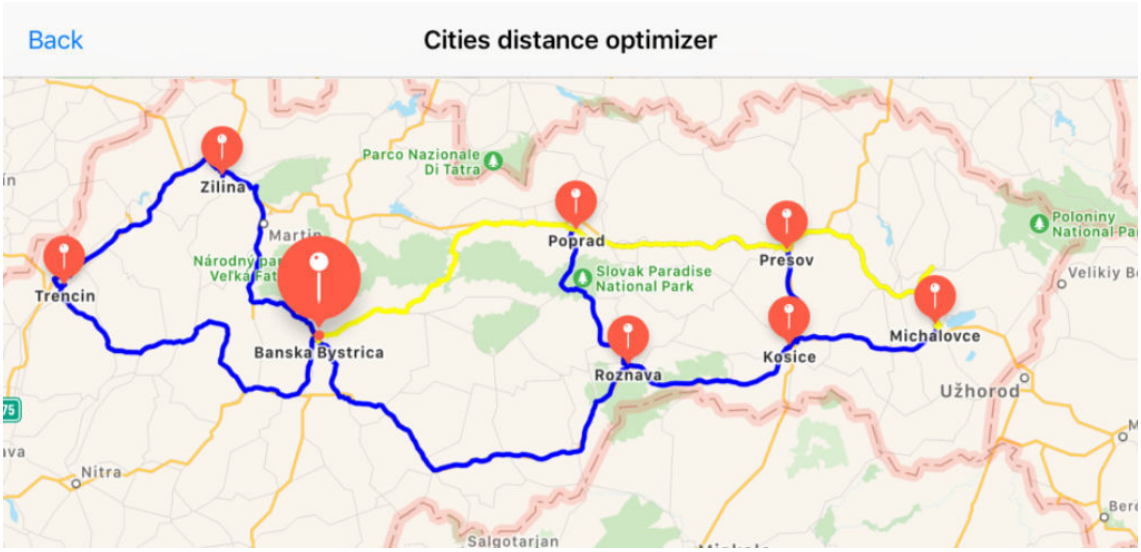
Z výsledkov sa dá vyčítať, že z každej dvojice ciest je tá prvá kratšia, pretože hodnota na kvantovom bite po meraní neprevyšuje hodnotu 0.5 a ani k nej nie je blízka. Ak by výsledok na hociktorom kvantovom bite bol napríklad 0.85, s určitosťou by bola druhá cesta kratšia. Z výsledkov sa dá usúdiť, že napriek možnej dekoherencii sú moje vyhľadávané cesty zakódované prvou bránou U3. Pri pohľade na výsledky je očividné, že cesta, ktorá sa najčastejšie opakuje ako najkratšia, je cesta a)e), čo je pravdivý výsledok.

3.4.5 Praktické riešenie

Daná metodika samozrejme nie je zatiaľ riešením pre komplexné problémy optimalizácie ciest, ale ukazuje prístup, akým je vhodné sa uberať pri tvorbe algoritmov, ktoré sa snažia tieto problémy riešiť. V rámci tejto iniciatívy som teda vytvoril aj testovaciu aplikáciu, ktorej cieľom je ukázať, ako by mohli byť v budúcnosti

zavedené dané metodiky do praktickej formy. Aplikácia, ktorú som vytvoril, je postavená na tablete s mobilným operačným systémom iOS, pretože ide o praktickú a fyzicky prenosnú formu a taktiež umožňuje dotykové ovládanie aplikácie. Knižnicu QuantumCSharp, na ktorej boli postavené predošlé riešenia tejto práce, som prerobil do jazyka Swift a teda som týmto umožnil integráciu kvantových výpočtov na platforme IBM QX do mobilných aplikácií, čo môže predstavovať základ pre ďalší posun v komunitnej a edukačnej tvorbe kvantových aplikácií. Knižnica QuantumSwift je, podobne ako QuantumCSharp, voľne dostupná a otvorená na internete na portáli GitHub.

Základ aplikácie je komponent mapy Slovenskej republiky, kde je vyznačených niekoľko vybraných väčších miest, pričom tieto mestá sú prepojené výberom hlavných cestných komunikácií. Aplikácia teda pripomína množstvo GPS klientskych riešení pre automobily. Princíp aplikácie je zvoliť dotykom dve ľubovoľné mestá v poradí, aby aplikačná logika mala informáciu o začiatku a konci cesty. Následne je vykonané vytvorenie kvantového obvodu na základe križovatiek, ktoré sa medzi mestami nachádzajú. Výsledok je demonštrovaný na obrazovke, kde žltou farbou mapa ukazuje odporúčanú cestu (obr. 3.29):



Obr. 3.29: Obrazovka aplikácie pri zvolení cesty z mesta Michalovce do mesta Banská Bystrica

4 Vyhodnotenie

V rámci realizácie vybraných riešení v podobe aplikácií s kvantovou zložkou v tejto práci som načrtol rôzne prístupy, ktoré sa dajú použiť pre zobrazenie konkrétnych podproblémov do podoby kvantových obvodov. Zobrazovanie a riešenie pomocou kvantových obvodov bolo ukázané na vybraných problémoch, ktoré sú prakticky realizovateľné aj na klasickom počítači. Dôvodom pre takýto výber je, aby bolo možné porovnať obe prístupy a určiť, ktorý je lepší pre ozajstnú praktickú implementáciu v rámci väčšieho programu. Toto určenie je vykonané analýzou dvoch faktorov. Prvý spočíva v zohľadnení efektívnosti využitia kvantových bitov. Ide o dôležitú informáciu pri potenciálnom škálovaní z hľadiska veľkosti problému v podobe množstva dát na vstupe, pretože ak aj algoritmus funguje, nemusí byť univerzálne aplikovateľný pre rastúce nároky na spracovanie. Druhý faktor je úvaha, či vlastnosti a výstup, ktorý dostávame z algoritmu, majú dostatočný význam pre ďalšie skúmanie a vykonávanie na kvantovom počítači, pretože aj pri zostavení univerzálneho kvantového algoritmu s dobrou škálovateľnosťou môže byť stále oveľa výhodnejšie ostať pri klasickom počítači.

4.1 Vyhodnotenie operácií s reťazcami znakov

Ako prvé príklady kvantovej realizácie operácií sú uvedené operácie s reťazcami znakov, konkrétne prevrátenie reťazca a získanie znaku pre daný index v reťazci. Tieto boli vybrané, ako už bolo spomenuté v práci, pretože ide o jednoduché operácie, ktoré sa používajú často v programovaní na klasických počítačoch. Cieľom bolo ukázať na danom spôsobe zobrazenia do podoby kvantového obvodu, či má tento druh problému zmysel riešiť kvantovým prístupom. Z hľadiska využívania zdrojov je tento algoritmus šetrný v rámci jedného výpočtu, pretože používa dva kvantové bity. Keďže sa ale pre invertovanie reťazca s n znakmi vykoná n behov,

tento prístup je veľmi nevýhodný z hľadiska škálovania pri rastúcom počte znakov. Problém získania znaku pre daný index v reťazci využíva naivné zobrazenie, teda v tomto prípade každý znak reťazca využíva jeden kvantový bit. Riešenie je zaujímavé v tom, že poukazuje na spôsob, akým je možné súčasti problému zakódovať a tieto kódy zachytiť v obvode nastavením kvantových bitov. V tomto algoritme je pre reťazec s n znakmi potrebných $n + 1$ kvantových bitov. Výhoda oproti pôvodnému algoritmu je, že sa obvod vykonáva len v jednom behu. Efektívnosť ale nie je momentálne veľká, pretože pri dostupných piatich kvantových bitoch je možné zakódovať v ideálnom prípade len reťazec so štyrmi znakmi. Ide o ideálnu situáciu, pretože je dôležitá aj architektúra kvantového čipu kvôli použitiu brán CNOT, ako som spomenul v kapitole 2.4.2. Z hľadiska významu pre vykonávanie na kvantovom počítači je možné prehlásiť, že vykonávanie na kvantovom počítači neprináša pri bežnej práci s reťazcami výhody ani z hľadiska času a momentálne ani z hľadiska možnosti spracovať rozsiahlejšie reťazce. Pri optimálnejšom algoritme a kvantových počítačoch s veľkým počtom kvantových bitov by bolo v budúcnosti zaujímavé skúmať a hľadať výhody v práci s rozsiahlymi reťazcami.

4.2 Vyhodnotenie generovania náhodných čísel

Generovanie náhodných čísel bolo zvolené, pretože náhodnosť je dôležitá v mnohých oblastiach informatiky. Použitím vlastnosti Hadamardovej brány bolo ukázané, ako je možné dosiahnuť náhodnosť a pretransformovať ju do náhodného čísla. V tejto časti práce bolo opísané, ako je možné rozšírením tohto algoritmu dostávať rozsiahlejšie náhodné čísla, buď pomocou iterácie alebo dekoherencie. Tento algoritmus je teda možné veľmi flexibilne expandovať. Klasické počítače dokážu generovať čísla, ktoré sú náhodné len do dostatočnej miery. Pre kryptografické účely pri generovaní veľkých náhodných kľúčov ide primárne o mieru náhodnosti, než o rýchlosť. Dnes je za relatívne rozumný čas možné pomocou klasického počítača prelomiť metódou overovania každého kľúča 32 bitové kľúče náhodne vygenerované klasickými počítačmi. Čo sa druhého vyhodnocovacieho kritéria týka, je pri aplikáciách, ktoré vysokú náhodnosť potrebujú, možné preferovať použitie kvantovej zložky, ktorá slúži ako primárny generátor náhodnosti.

4.3 Vyhodnotenie hry míny

Hry sú jedna z oblastí, kde je preferované používať klasický počítač, ak takéto aplikácie sú vytvárané len za oddychovým a nie napríklad za edukačným účelom. Ak ale hra obsahuje kľúčovú zložku, ktorá je realizovateľná spoľahlivo na kvantovom počítači, môže predstavovať základ pre ďalšie skúmanie tvorby kvantových algoritmov. Na tejto aplikácii bola ukázané nastavenie a aplikácia generovania náhodných čísel a taktiež spôsob kvantového rozhodovania o veľkosti medzi dvoma hodnotami, teda spôsob ako jednoduché porovnanie pretransformovať do kvantového algoritmu. V rámci tvorby hier je do budúcnosti možné pokúšať sa o presunutie čo najviac podvýpočtov hry na kvantový počítač a taktiež sa zamerať na tvorbu hier, ktoré majú edukačný charakter, hlavne čo sa týka kvantových počítačov.

4.4 Vyhodnotenie techník zobrazenia optimálnej cesty na mape

V časti práce o zobrazení mapy na kvantový obvod som prezentoval tri rôzne spôsoby, ako je možné riešiť problémy ciest medzi bodmi s kvantovou zložkou. Pri porovnaní a výbere z týchto troch techník sú podstatné kritériá efektivity, rýchlosti a škálovateľnosti algoritmu. Taktiež je dôležitý faktor náročnosti vytvorenia obvodu a spracovania výsledkov na klasickom počítači. Prvý algoritmus sa dá vylúčiť prakticky okamžite, pretože nedisponuje riadiacou zložkou pre vetvenie. Tento zobrazovací algoritmus by bolo možné rozšíriť o ďalší riadiaci kvantový bit, avšak pri každom ďalšom uzle grafu a aj pri každej ďalšej vetve rastie požiadavka o nový kvantový bit. Druhý navrhovaný algoritmus, ktorý granulárne mapuje jednotlivé riešenia križovatky grafu na kvantové bity je optimálny z hľadiska množstva využitých kvantových bitov. Klasický počítač potrebuje iba disponovať informáciami, ktorý kvantový bit rieši ktorú z množstva križovatiek. Čo sa týka škálovania, je možné s n kvantovými bitmi vyriešiť n križovatiek, ale úloha vyskladať z týchto výsledkov najkratšie cesty a vybrať tú najpriaznivejšiu je pridelená klasickému počítaču. Toto predstavuje dodatočné časové a výkonnostné požiadavky na celkový výpočet. Tretí algoritmus mapujúci celé cesty je z hľadiska využitia kvantových bitov relatívne náročný, pretože ak porovnávame dvojice, po-

čet kvantových bitov je rovný kombináciám C$(n, 2)$, pričom n je počet ciest, ktoré máme porovnať. Pri 5 cestách teda potrebujeme až 10 kvantových bitov, čo nie je úplne optimálne. Výhoda algoritmu spočíva v tom, že mapujeme na brány súčty všetkých možných podciest, teda klasický počítač len musí vytiahnuť najčastejšie opakujúcu sa kratšiu cestu vo výsledkoch. Pretože na reálnych kvantových čipoch sú dostupné len obmedzené topológie, druhá metóda z kapitoly 3.4.2 predstavuje vhodnú techniku na prehľadávanie ciest, ktorou by sa mohli zaoberať budúce experimenty.

5 Záver

Cieľ tejto práce bol na podklade princípov kvantových výpočtov vyvinúť aplikácie s kvantovou zložkou spustiteľnou na platforme IBM Q. V kapitole 2 boli ukázané základné princípy tejto problematiky spolu s matematickým aparátom. Všetky aplikácie z kapitoly 3 tejto práce boli vyvinuté tak, aby využívali dostupné kvantové počítače v rámci výpočtov v danom programe. V rámci týchto riešení som poukázal na spôsob, akým je možné vyvíjať tieto algoritmy od prvotného štádia návrhu pri zobrazovaní problému pomocou kvantového obvodu. Niektoré z týchto algoritmov bolo možné využiť vo väčšom riešení, ako napríklad programe na klasickom počítači.

Z výsledkov práce sa dá usúdiť, že vývoj kvantových aplikácií má zmysel, ak sa nimi rieši špecifický prípad použiteľnosti, na ktorom je overená výhoda použitia kvantového počítača. Táto práca taktiež ukazuje, do akej vysokej miery je aplikácia kvantových počítačov neprebádaný odbor. Na riešeniach v praktickej časti práce je vidieť, ako potrebné je zvoliť si úplne iný spôsob rozmýšľania nad rozkladom a riešení problémov. Do budúcna je potrebné s týmto základom neustále hľadať nové prípady použitia a s pribúdajúcim počtom kvantových bitov na dostupných kvantových počítačoch je potrebné vyvíjať čoraz efektívnejšie a univerzálnejšie kvantové algoritmy a pokúšať sa zavádzať ich do bežnej praxe.

Literatúra

[1] Davide Ferrari a Michele Amoretti. "Demonstration of Envariance and Parity Learning on the IBM 16 Qubit Processor". In: *CoRR* abs/1801.02363 (2018). arXiv: 1801.02363. URL: http://arxiv.org/abs/1801.02363.

[2] Noson S Yanofsky. "An Introduction to Quantum Computing". In: *Proof, Computation and Agency - Logic at the Crossroads*. 2011, s. 145–180. DOI: 10.1007/978-94-007-0080-2_10. URL: https://doi.org/10.1007/978-94-007-0080-2%5C_10.

[3] Vojtěch Kupča. "Teorie a perspektiva kvantových počítačů". In: (jan. 2001).

[4] Martin Laforest. *The mathematics of quantum mechanics*. 2015.

[5] Benoıt Valiron. "Quantum Computation: a Tutorial". In: *New Generation Comput.* 30.4 (2012), s. 271–296. DOI: 10.1007/s00354-012-0401-7. URL: https://doi.org/10.1007/s00354-012-0401-7.

[6] Patrick J. Coles et al. "Quantum Algorithm Implementations for Beginners". In: (apr. 2018).

[7] Chao-Ming Tseng, Chih-Sheng Chen a Chua-Huang Huang. "Quantum Gates Revisited: A Tensor Product Based Interpretation Model". In: *ESA/VLSI*. CSREA Press, 2004.

[8] K.J. Sarma. "Understanding Quantum Computing". In: *International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-6*. Sept. 2015.

[9] Matt DeCross, July Thomas a Tiffany Wang. "Quantum Entanglement". In: (2019). [online]. Dostupné z: https://brilliant.org/wiki/quantum-entanglement/.

[10] Ryszard Horodecki et al. "Quantum entanglement". In: (2007).

[11] Colin P. Williams. *Explorations in Quantum Computing*. 2nd. Springer Publishing Company, Incorporated, 2008. ISBN: 184628886X, 9781846288869.

[12] Ryan O'Donnell. "Quantum Computation CMU 18-859BB Lecture 3: The Power of Entanglement". In: (sept. 2015). URL: https://www.cs.cmu.edu/~odonnell/quantum15/lecture03.pdf.

[13] Eleanor Rieffel a Wolfgang Polak. "An Introduction to Quantum Computing for Non-physicists". In: *ACM Comput. Surv.* 32.3 (sept. 2000), s. 300–335. ISSN: 0360-0300. DOI: 10.1145/367701.367709. URL: http://doi.acm.org/10.1145/367701.367709.

[14] Eleanor Rieffel a Wolfgang Polak. *Quantum Computing: A Gentle Introduction*. 1st. The MIT Press, 2011. ISBN: 9780262015066.

[15] Archil Avaliani. "Quantum Computers". In: *CoRR* cs.AI/0405004 (2004). URL: http://arxiv.org/abs/cs.AI/0405004.

[16] James R. Wootton a Daniel Loss. "Repetition code of 15 qubits". In: *Phys. Rev. A* 97 (5 máj 2018), s. 052313. DOI: 10.1103/PhysRevA.97.052313. URL: https://link.aps.org/doi/10.1103/PhysRevA.97.052313.

[17] Swamit S Tannu a Moinuddin K Qureshi. "Not All Qubits Are Created Equal". In: (2018).

[18] Michel H. Devoret a Robert J. Schoelkopf. "Superconducting circuits for quantum information: an outlook." In: *Science* 339 6124 (2013), s. 1169–74.

[19] Alwin Zulehner, Alexandru Paler a Robert Wille. "Efficient Mapping of Quantum Circuits to the IBM QX Architectures". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (dec. 2017). DOI: 10.1109/TCAD.2018.2846658.

[20] IBM. *IBM QX Full user guide*. 2016. [online]. Dostupné z: https://quantumexperience.ng.bluemix.net/qx/tutorial?sectionId=full-user-guide&page=002-The_Weird_and_Wonderful_World_of_the_Qubit~2F005-The_Bloch_Sphere.

[21] Anirban Pathak. "Experimental quantum mechanics in the class room: Testing basic ideas of quantum mechanics and quantum computing using IBM quantum computer". In: (máj 2018). URL: https://arxiv.org/pdf/1805.06275.pdf.

Zoznam príloh

Príloha A: Používateľská príručka k projektom/aplikáciám na CD

Príloha B: CD obsahujúce nasledujúce prílohy:

- Projekt aplikácie s riešením operácií so znakmi z kapitoly 3.1
- Projekt aplikácie s hrou míny z kapitoly 3.3
- Projekt aplikácie s cestami medzi Slovenskými mestami z kapitoly 3.4.5

A Prilohy

Používateľská príručka ku kvantovým aplikáciám

Prvé dve aplikácie v diplomovej práci, na ktorých bol vysvetlený postup pri vytváraní kvantových programov, boli vytvorené pod platformou .NET Core. Z tohto dôvodu je potrebné nainštalovať si najnovšiu verziu vývojového prostredia Visual Studio z nasledovnej stránky: https://visualstudio.microsoft.com/downloads/. Je nutné poznamenať, že aplikácie boli vytvorené a spúšťané primárne v prostredí Visual Studio for Mac na operačnom systéme macOS 10.14. Projekty boli v úvodnej fáze ich vytvorenia spúšťané aj na Visual Studio bežiacim pod OS Windows a kompatibilita s týmto operačným systémom bola pri odovzdávaní práce testovaná. Projekty sa otvárajú pomocou súboru .sln.

Posledný projekt bol vytvorený pod platformou iOS, je teda potrebné na počítači s operačným systémom macOS stiahnuť vývojové prostredie XCode: https://itunes.apple.com/us/app/xcode/id497799835?mt=12. Aplikácia musí bežať taktiež na pripojenom zariadení s operačným systémom iOS, pretože knižnica QuantumSwift je pribalená v riešení a je skompilovaná na konkrétnu hardvérovú architektúru fyzických zariadení, teda nemôže bežať v simulátore (je to tak aj z výkonnostných hľadísk, mapa by nebežala stabilne v simulátore zariadenia). Projekt sa otvára pomocou súboru IBMQuantumSampleApp.xcodeproj.

Pre spúšťanie aplikácií v súčinnosti s kvantovým počítačom je taktiež potrebné vytvoriť si účet na platforme IBM QX: https://quantumexperience.ng.bluemix.net/qx/login. V každom projekte je potom potrebné Vaše používateľské meno a heslo použiť pri inicializácií kvantového programu (ukázané v dokumentácii).

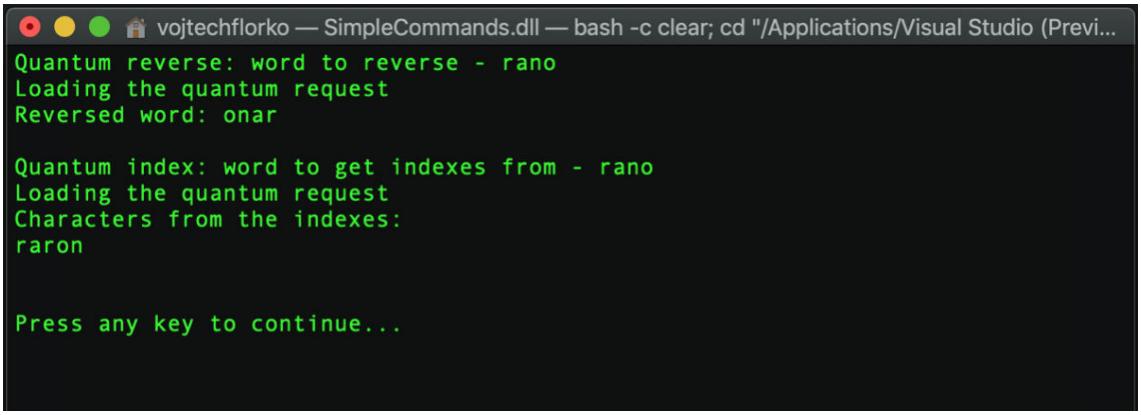
A.0.1 Aplikácia/projekt s kvantovým spracovaním reťazcov

Keďže kvantové programy ukazujúce dva spôsoby spracovania reťazcov sú rovnakého druhu, nie sú obšírne a slúžia predovšetkým na vysvetlenie práce s kvantovým počítačom, nachádzajú sa oba v jednom projekte, pričom projekt sa nazýva SimpleCommands. V projekte sa nachádza súbor Program.cs, ktorý obsahuje reťazec určený na spracovanie. V súbore Commands.cs sa na riadkoch 76 a 128 nachádzajú nasledovné volania kvantových programu, kde je potrebné zadať do parametrov Email a Password vytvorené používateľské meno a heslo z platformy IBM QX:

```
QuantumProgram program = new QuantumProgram(new QuantumProgramOption()
{
    Email = "",
    Password = "",
    Device = IbmQXDevices.Simulator,
    MaxCredits = 15,
});
```

Obr. A.1: Inicializácia kvantového programu

Spustený program vyzerá nasledovne:

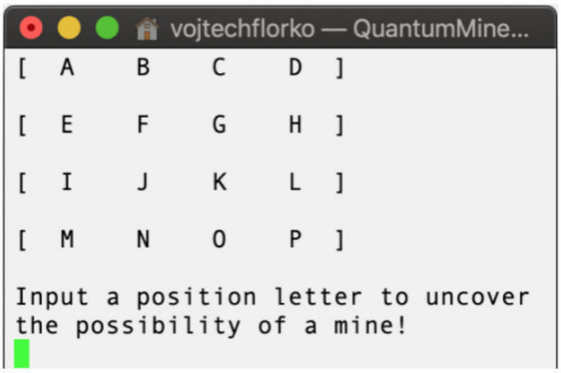


Obr. A.2: Program pracujúci s reťazcami

A.0.2 Aplikácia/projekt kvantovej hry míny

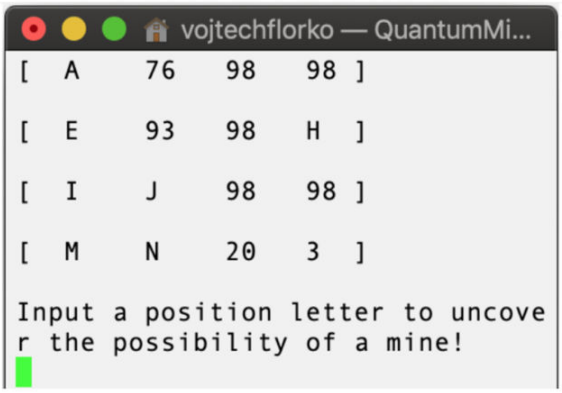
Podobne ako pri predošlom kvantovom programe, hra míny je spúšťaná ako projekt Visual Studio. Ak sa projekt nedá spustiť, je potrebné odstrániť z oboch projektov súbor AssemblyInfo.cs. V súbore Credentials.cs je potrebné ako v predošlom

projekte pridať používateľské údaje do parametrov inicializácie kvantového programu: QuantumMinesweeper/Credentials.cs. Po spustení si program vygeneruje herné pole a hra vyzerá nasledovne:



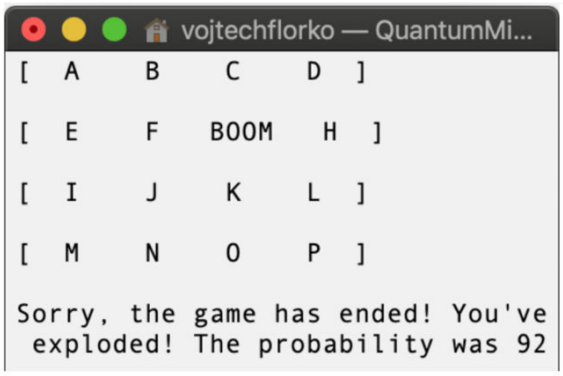
Obr. A.3: Herné pole mín

Hra sa ovláda zadávaním znaku pre odkrytie políčka, pričom políčko s hodnotou vyššou ako 85 obsahuje mínu. Číslo pod políčkom indikuje najvyššiu hodnotu vo svojom okolí, teda ak je číslo 76, jedno z okolitých políčok má najvyššiu hodnotu práve 76, pričom ostatné ju majú nižšiu. Analogicky ak dostaneme hodnotu 98, jedno z okolitých políčok obsahuje mínu a preto treba postupovať opatrne a pokúsiť sa odkryť okolie tohto políčka.



Obr. A.4: Rozohrané herné pole mín

Hra sa končí odkrytím všetkých políčiek, ktoré nie sú míny, alebo odkytím míny a následným vybuchnutím:



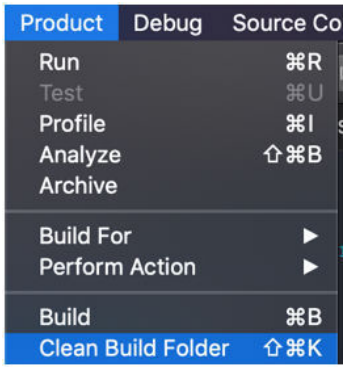
Obr. A.5: Skončená hra

A.0.3 Aplikácia mapa slovenska

Tato aplikacia musí byť otvorená vo vývojovom prostredí XCode. Je potrebné aby toto vývojové prostredie malo pripojené zariadenie s operačným systémom iOS, čo sa dá verifikovať jeho zobrazením v lište zariadení pre spustenie aplikácie:

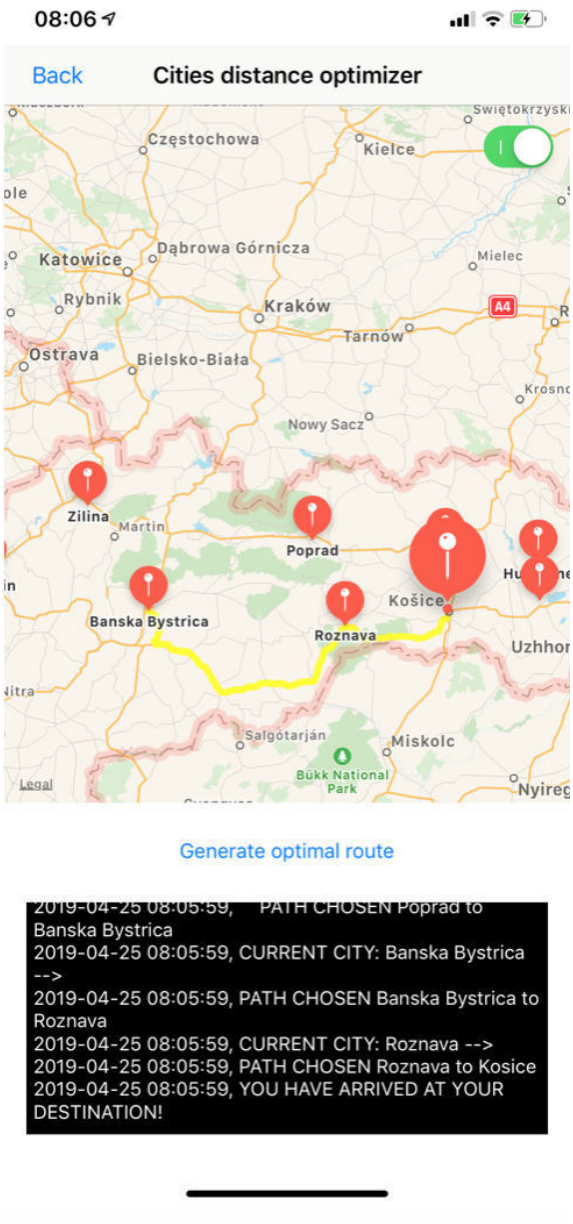
Obr. A.6: Lišta pre spúšťanie aplikácie

Je dobré pred kompiláciou a spúšťaním vyčistiť projekt a Váš build folder v naslednovnej ponuke:



Obr. A.7: Vyčistenie projektu

Po spustení projektu je potrebné vojsť do sekcie Cities Distance Optimization a mapa by sa mala vykresliť aj s mestami. Stlačením jedného mesta a za ním druhého mesta potvrdí výber a je možné stlačiť tlačítko Generate Optimal Route. Ak sa vykreslenie neuskutoční kvôli niekedy nespoľahlivo fungujúcemu vykresľovaniu ciest, výsledok je možné vidieť v malej konzole pod mapou:



Obr. A.8: Mapa Slovenska s vybranou cestou