VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INTERAKTIVNÍ SIMULÁTOR DNA VÝPOČTU

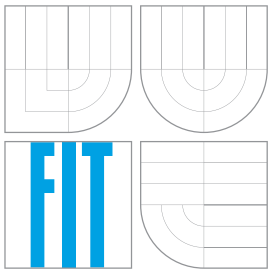
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KOVÁCS

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

INTERAKTIVNÍ SIMULÁTOR DNA VÝPOČTU

INTERACTIVE DNA COMPUTATION SIMULATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN KOVÁCS
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL BIDLO, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce je zaměřena na shrnutí biologických operací nad DNA a jejich nasazení ve výpočtu složitých matematických problémů. Ukázkovým příkladem, jehož řešení je v práci demonstrováno, je problém hamiltonovské cesty grafem, také znám jako problém obchodního cestujícího. Při řešení tohoto problému budou popsány použité operace nad DNA a postup, který poprvé představil Leonard Adleman. Jeho práci je možné považovat za první experiment v oboru, který je teď známý jako DNA počítání. Cílem této bakalářské práce je implementovat interaktivní simulační program (založený na principech a formálním modelu Adlemanovy práce) pro řešení hamiltonovské cesty grafem a zhodnotit jeho možnosti vzhledem na prostorovou složitost při různých instancích problému obchodního cestujícího.

Abstract

The aim of this work is to summarize the basic principles of operations performed over DNA molecules and to demonstrate their usage in solving some hard mathematical problems. In particular, the Hamiltonian Path Problem – HPP (also known as the Traveling Salesman Problem) will be considered as a case study. A fundamental approach introduced by Leonard Adleman will be desctribed to solve the HPP using the DNA operations. His work may be considered as the first experiment in the area that is currently known as DNA computing. The goal of this bachelor thesis is to implement an interactive software simulator (based on the principles and formal models of Adleman's work) for solving the HPP and to evaluate its abilities with respect to area complexity considering various instances of HPP.

Klíčová slova

DNA výpočet, operace nad DNA, graf, hamiltonovská cesta grafem

Keywords

DNA computation, DNA operations, graph, Hamiltonian path

Citace

Martin Kovács: Interaktivní simulátor DNA výpočtu, bakalářská práce, Brno, FIT VUT v Brně, 2014

Interaktivní simulátor DNA výpočtu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D. Uvedl jsem všechny literátní prameny, ze kterých jsem čerpal.

.....
Martin Kovács
17. mája 2014

Poděkování

Děkuji Ing. Michalovi Bidlovi, Ph.D. za jeho odbornou pomoc, férový přístup a cenné rady při psaní této bakalářské práce.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Predpokladajme, že chceme zabezpečiť ubytovanie pre skupinu štyristo študentov. Na internáte je však miesto obmedzené a preto iba sto z týchto študentov dostane lôžko v izbe. Veci sa ďalej komplikujú tým, že niektorí študenti spolu nemôžu bývať na jednej izbe. Je jednoduché zistiť, či konkrétna sada sto študentov vyhovuje daným požiadavkom, no je niečo celkom iné, vytvoriť zoznam všetkých vyhovujúcich sád sto študentov. Pokúsiť sa takýto zoznam vytvoriť metódou „pokus–omyl“, t.j. kontrolovať každú možnú kombináciu sto študentov, by znamelo zkontrolovať viac kombinácií, než je počet atómov v známom vesmíre. Takýto problém patrí do množiny nedeterministicky polynomiálnych (ďalej NP) problémov. [4]

Lepší spôsob riešenia NP problémov ponúka nasadenie operácií nad DNA, ktoré budú interpretované ako výpočet. Pomocou DNA je možné vykonať milióny výpočtových krokov takmer okamžite, vďaka masívnemu paralelismu. Syntéza oligonukleotidov (postupnosť n nukleotidiov) ponúka užívateľovi možnost tvorby ľubovoľne dlhých kombinácií nukleotidov. Žíhanie umožňuje spojiť komplementárne oligonukleotidy do dvojzávitnice, a naopak pomocou denaturácie je možné tento spoj znehodnotiť. Triedenie vlákien (výber oligonukleotidov obsahujúcich určitú kombináciu nukleotidov) poskytuje užívateľovi možnosť ponechať iba tie oligonukleotidy, ktoré predstavujú medzikrok výpočtu, alebo samotný výsledok, a ostatné zahodiť.

Táto práca sa zaoberá návrhom a implementáciou aplikácie s grafickým užívateľským rozhraním na simulovanie výpočtov za pomoci DNA. Kontrétnym problémom riešeným s použitím DNA výpočtov je problém hamiltonovskej cesty grafom. Účelom aplikácie je demonštrovať užívateľovi postup, akým sú DNA výpočty realizované. Ďalšie využitie aplikácie je možné pri výučbe biológiou inšpirovaných systémov.

Štruktúra práce je nasledujúca. V kapitole 2 je podrobne popísaná štruktúra DNA a biologický základ operácií, ktoré umožňujú využitie DNA ako výpočtový prostriedok. Kapitola 3 obsahuje popis problému Hamiltonovskej cesty grafom, detailne uvádza postup Leonarda Adlemana pri riešení tohto problému za pomoci DNA a uvádza formalizáciu operácií nad DNA. Kapitola 4 popisuje program, vytvorený v rámci práce za účelom demonštrácie Adlemanovho experimentu. Sú v nej uvedené vybrané implementačné detaily zaujímavých alebo kľúčových častí programu. Posledná kapitola sa zaoberá experimentami s vytvoreným programom. Uvádza nastavenia parametrov simulácie, simulačné štatistiky a výsledky každého experimentu.

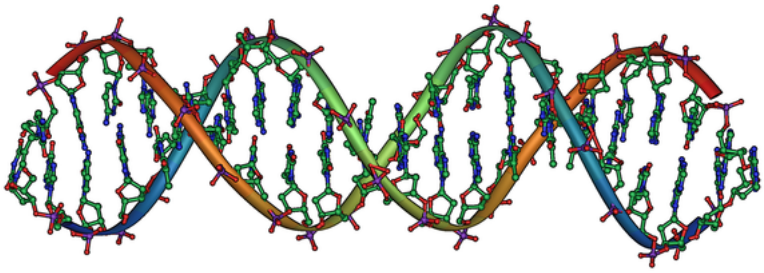
Kapitola 2

DNA výpočty

Táto kapitola podrobne rozoberá štruktúru DNA a operácie, ktoré umožňujú využitie DNA ako výpočtový prostriedok. Popisuje biologickú podstatu týchto operácií a pojednáva o formalizácii DNA výpočtov, teda o spôsobe ako zaviesť operácie nad DNA do prostredia číslicových počítačov pre potreby simulácie DNA výpočtov. Informácie uvedené v tejto kapitole sú prevzaté z [2].

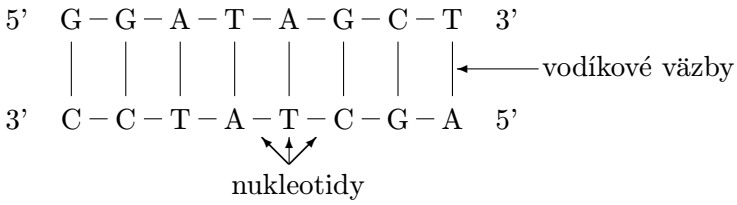
2.1 Štruktúra DNA

Deoxyribonukleová kyselina sa bežne v prírode vyskytuje ako dvojzávitnica (obrázok 2.1) zložená z dvoch polymérových vlákien. Každé vlákno je tvorené reťazcom takzvaných nukleotidov (dusíkatých bází) pripojených ku kostre z fosfátu a deoxyribózy pomocou vodíkových väzieb. Takýto reťazec niekoľkých nukleotidov sa tiež nazýva oligonukleotid.



Obrázok 2.1: Dvojzávitnica DNA [8]

Štyri nukleotidy deoxyribonukleovej kyseliny sú adenín, guanín, cytozín a tymín, bežne uvádzané ako A,G,C, respektíve T. Každé vlákno má dva konce. Chemická konvencia ich uvádza ako 5' koniec a 3' koniec, čo poskytuje DNA prirodzenú orientáciu. Z chemických vlastností nukleotidov vyplýva, že adenín sa dokáže spárovať s tymínom a guanín s cytozínom. Adenín a tymín sú spojené dvomi vodíkovými väzbami (purínová väzba), a guanín s cytozínom sú spojené tromi vodíkovými väzbami (pyrimidínová väzba). Aby sa jednotlivé oligonukleotidy spojili do dvojzávitnice, musia mať v miestach spojenia komplementárne nukleotidy, ako je to znázornené na obrázku 2.2.



Obrázok 2.2: Schéma štruktúry dvojzávitnice DNA

2.2 Syntéza oligonukleotidov

Oligonukleotidy s ľubovoľnými kombináciami nukleotidov je možné synteticky vyrobiť strojom o veľkosti mikrovlnej rúry. Tomuto stroju sú v roztoku dodávané štyri nukleotidy DNA, ktoré sú pospájané do užívateľom definovaného poradia. Stroj vytvorí milióny kópií požadovaného oligonukleotidu a uloží ich do roztoku v malej ampulke.

2.3 Denaturácia a renaturácia

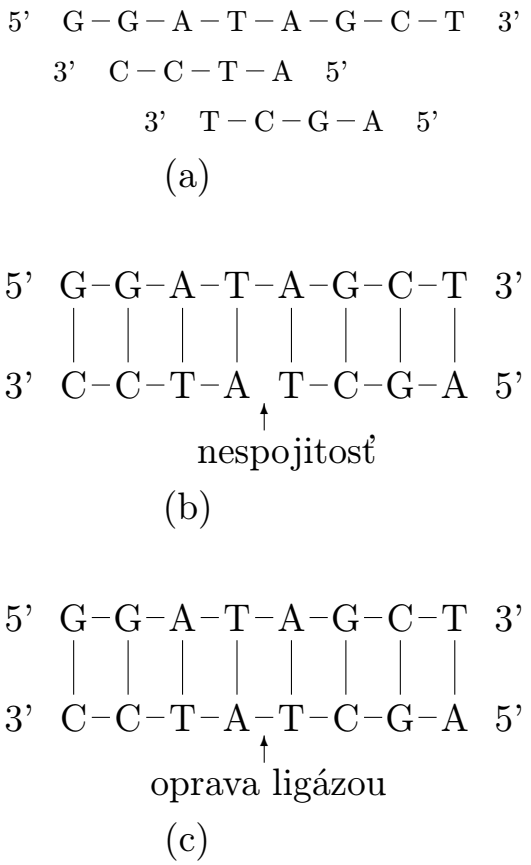
Dvojzávitnica DNA môže byť rozložená na jednotlivé vlákna zohriatim roztoku na teplotu danú zložením vlákien. Zohriatím sa zničia vodíkové väzby medzi komplementárnymi nukleotidmi vlákien. Keďže medzi guanínom a cytozínom existuje silnejšia väzba ako medzi adenínom a tymínom, na jej prelomenie je potrebná väčšia teplota. Tento proces označujeme denaturácia. Opakom denaturácie je renaturácia, teda proces ochladenia jednotlivých vlákien DNA za účelom vytvorenia spoju medzi komplementárnymi prvkami. Výsledkom renaturácie je dvojzávitnica.

2.4 Spájanie a štiepenie vlákien

Enzýmy ligázy sú primárne zodpovedné za spájanie nespojitostí, ktoré vznikajú počas replikácie, opráv a rekombinácie DNA. Obecne ligáza katalyzuje tvorbu spojov medzi susednými nukleotidmi. Prácu ligázy je možné rozdeliť do troch krokov. Najprv je vytvorený prostredník medzi enzýmom a nukleotidom (darcom). Potom je presunom adenylynovej skupiny z enzýmu aktivovaný 5’ koniec darcovského nukleotidu. Na záver je nukleofilnou reakciou 3’ koncu nukleotidu príjemcu vytvorený spoj medzi darcom a príjemcom. Tento proces je detailnejšie popísaný v [7].

Operácia štiepenia vlákien DNA je tiež vykonávaná pomocou enzýmou. Restrikčné endonukleázy rozpoznajú špecifickú postupnosť nukleotidov, veľmi často o dĺžke 4-8, v jednotlivých vláknach DNA. Pri výskyte takejto postupnosti v konkrétnom vlákne dvojzávitnice je daná dvojzávitnica v tomto mieste rozštiepená. Podľa druhu restrikčného enzýmu môžu mať odštiepené časti dvozávitnice, buď hladké konce (obe vlákna majú po odštiepení rovnakú dĺžku), alebo „lepkavé“ konce (jedno vlákne je odštiepené o kúsok ďalej ako druhé).

Vo výpočtoch využívajúcich DNA je enzým ligázy využívaný na riešenie nasledujúceho problému. Pri skladaní vlákien z viacerých oligonukleotidov dochádza k výskytu nespojitostí medzi nukleotidmi dvoch rôznych oligonukleotidov vo vlákne. Takéto nespojitosti môžu byť zacelené pomocou enzýmu ligázy. Priebeh tvorby dvojzávitnice znázorňuje obrázok 2.3.



Obrázok 2.3: Tvorba dvojzávitnice; (a) jednotlivé oligonukleotidy (b) spojenie nukleotidov do dvojzávitnice (c) zacelenie medzery pomocou enzýmu ligázy

2.5 Triedenie

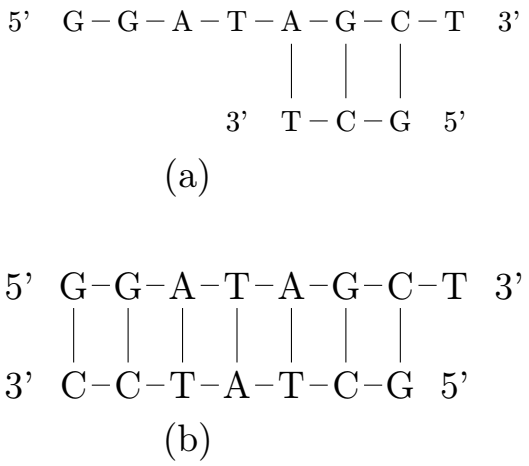
Jednou zo základných operácií je aj triedenie, teda extrakcia jednotlivých vlákien, obsahujúcich požadovanú postupnosť nukleotidov, z roztoku v skúmavke. Spôsob ako takúto extrakciu vykonať je určiť si požadovanú postupnosť x a k nej komplementárnu postupnosť $\overline{x}$. K postupnostiam $\overline{x}$ sú následne pripojené molekuly, ktoré vytvoria pevný spoj s ukotvenou maticou. Rozliatie roztoku v skúmavke na túto maticu potom spôsobí vytvorenie väzieb medzi postupnosťami x v oligonukleotidoch zo skúmavky a postupnosťami $\overline{x}$ pripevnených k matici. Nevyhovujúcich vlákien sa dá zbaviť jednoduchým opláchnutím matice. Na záver sú procesom denaturácie od matice oddelené oligonukleotidy s postupnosťou x. Alternatívny spôsob využíva magnetické guľôčky pripevnené ku komplementárnym postupnostiam. Po spojení stačí magnetom vytiahnuť magnetické guľôčky, s ktorými budú spojené hľadané oligonukleotidy.

2.6 Gélová elektroforéza

Dôležitá operácia, ktorá umožňuje triediť oligonukleotidy podľa ich dĺžky je gélová elektroforéza. Elektroforéza je proces pohybu molekúl, obsahujúcich nejaký elektrický náboj,

v elektrickom poli. Deoxyribonukleová kyselina nesie záporný náboj a preto má tendenciu pohybovať sa smerom ku kladnému pólu. Rýchlosť pohybu vo vodnom roztoku závisí na tvare a elektrickom náboji. Keďže DNA má rovnaký náboj na jednotku dĺžky, všetky oligonukleotidy sa vo vodnom roztoku pohybujú rovnako rýchlo. V takomto prípade sa s výhodou využíva gél (často agaróza, polyakrylamid, alebo ich kombinácia), ktorého štruktúra tvorí akúsi prekážku. Z tohoto dôvodu sa menšie molekuly dokážu gélom pohybovať rýchlejšie ako tie väčšie, čo vo výsledku spôsobí vytriedenie DNA oligonukleotidov do pásiem podľa ich dĺžky.

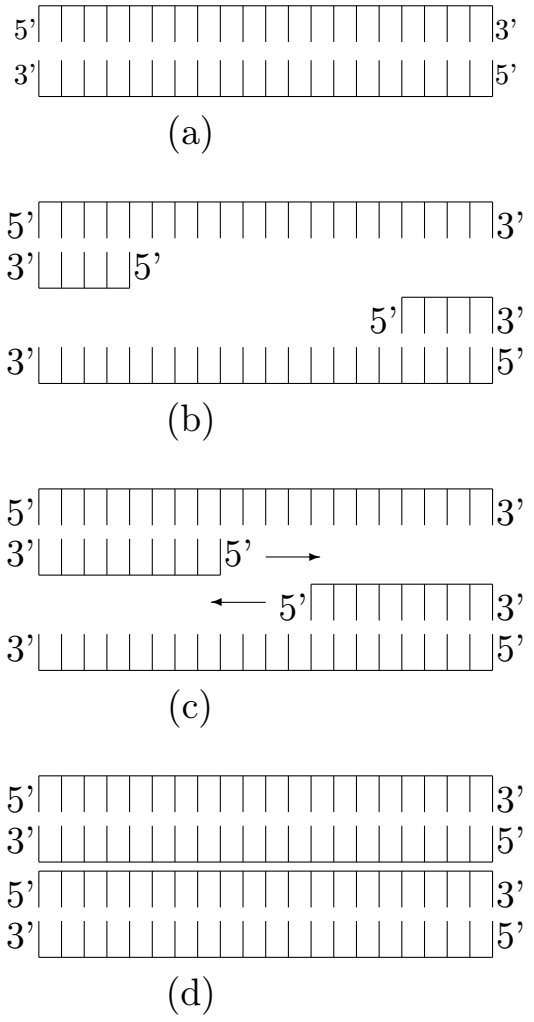
2.7 Polymerázová reťazová reakcia



Obrázok 2.4: Roztiahnutie primeru polymerázou; (a) primer p sa napojí na vzor v (b) polymeráza natiahne 3' koniec primeru

Je daný krátky primer oligonukleotidov p. Enzým polymerázy roztiahne p iba ak sa spojí s dlhším vzorovým nukleotidom v. Polymeráza roztiahne p doplnením komplementárnych bází smerom k 3' konci. Roztiahnutie pomocou polymeráze znázorňuje obrázok 2.4. Táto vlastnosť polymeráze sa využíva v kombinácii s metódou polymerázovej reťazovej reakcie. Polymerázová reťazová reakcia (angl. Polymerase Chain Reaction – PCR) je metóda, ktorá rýchlo znásobí množstvo DNA v danom roztoku. Každý cyklus reakcie zdvojnásobí počet vlákien v roztoku, čo predstavuje exponenciálny nárast množstva DNA vzhľadom na počet cyklov.

Pri tejto metóde sú najprv navrhnuté dva primery značiace začiatok a koniec úseku, ktorý má byť replikovaný. Ďalej je do roztoku pridané veľké množstvo týchto primerov a celý roztok sa zohreje, aby povolili vodíkové väzby medzi vláknami dvojzávitníc. Po ochladení sa primery napoja na komplementárne nukleotidy na jednotlivých vláknach. Pridaním polymerázy sú primery roztiahnuté po celej dĺžke pôvodného vlákna ako to ukazuje obrázok 2.5.



Obrázok 2.5: Duplikácia dvojzávitnice; (a) cieľová dvojzávitnica (b) dvojzávitnica je rozdelená na jednotlivé vlákna a pridané primery sú pripojené na odpovedajúce pozície vlákien (c) roztiahnutie primerov polymerázou (d) pôvodná dvojzávitnica je zduplikovaná

Kapitola 3

Problém hamiltonovskej cesty

Uvažujme graf zložený z vrcholov V a hrán H, spájajúcich vždy dva vrcholy. Z vrcholov si vyberieme jeden, ktorý bude počiatočný, nazvime ho V_p, a jeden, ktorý bude koncový, nazvime ho V_k. Hamiltonovská cesta týmto grafom je potom postupnosťou vrcholov V spojených hranami H, ktorá začína vo vrchole V_p, navštívi každý uzol, ale iba raz, a končí vrcholom V_k.

Problém hamiltonovskej cesty grafom je typickým zástupcom NP problémov, dokonca patrí medzi NP–kompletné problémy[1]. Na tomto probléme prezentoval v roku 1994 Leonard Adleman koncept DNA počítania.

3.1 Adlemanov experiment

Vo svojom článku [1] Adleman prezentoval nasledujúci nedeterministický algoritmus na zistenie existencie hamiltonovskej cesty grafom:

1. Generovanie veľkého množstva náhodných ciest grafom
2. Ponechanie ciest, ktoré začínajú vrcholom V_p a končia vrcholom V_k
3. Pri grafe obsahujúcom n vrcholov, ponechanie ciest, ktoré navštevujú práve n vrcholov
4. Ponechanie ciest, ktoré navštevujú všetky vrcholy aspoň raz
5. Ak nejaké cesty ostali, graf má práve tieto hamiltonovské cesty

Pri implementácií kroku 1 každému vrcholu V grafu bol pričlenený oligonukleotid obsahujúci jedinečnú náhodnú kombináciu dvadsiatich nukleotidov DNA (ďalej O_i). Každá hrana H spájajúca dva vrcholy (ďalej $O_{i\to j}$), bola zložená z desiatich nukleotidov 3' konca (posledných desať) oligonukleotidu O_i (okrem O_i odpovedajúcemu V_p, kedy bolo použitých všetkých dvadsať nukleotidov) nasledovaných desiatimi nukleotidmi 5' konca (prvých desať) oligonukleotidu O_j (okrem O_j odpovedajúcemu V_k, kedy bolo použitých všetkých dvadsať nukleotidov). Oligonukleotidy komplementárne k O_i boli označené ako $\overline{O_i}$. Bolo vytvorených 50 pmol každého oligonukleotidu $\overline{O_i}$ (okrem O_i reprezentujúcich V_p a V_k, ďalej O_p a $\overline{O_k}$) a 50 pmol každého oligonukleotidu $O_{i\to j}$. Tieto množstvá boli zamiešané spolu s enzýmom ligázy. Účelom oligonukleotidov $\overline{O_i}$ bolo pritiahnuť k sebe kompatibilné hrany a spojiť ich do náhodných ciest grafom za pomoci enzýmu ligázy.

[1] Cookov teorém

Filtrácia ciest v kroku 2 prebiehala zosilnením výsledku predošlého kroku polymerázovou reťazovou reakciou za použitia O_p a $\overline{O_k}$ ako primerov. V kroku 3 bol výsledok kroku 2 vypustený na gél agarózy, a za pomoci gélovej elektorforézy boli vybrané práve také cesty, ktoré odpovedali požadovanej dĺžke. Tieto cesty boli nasledovne umyté v demineralizovanej vode, aby sa očistili od zbytkov gélu. Výsledok tohto procesu bol ešte niekoľkokrát zosilnený PCR a elektroforézne prefiltrovaný na zaistenie jeho čistoty. Pre krok 4 boli využité magnetické guľôčky pripevnené na jednotlivé oligonukleotidy $\overline{O_i}$. Výsledné dvojzávitnice z kroku 3 boli rozdelené na osobitné vlákna, pričom vlákna reprezentujúce cesty grafom boli vyliate na oligonukleotidy s magnetickými guľôčkami zastupujúce jeden konkrétny vrchol grafu. Tie cesty, ktoré sa spárovali s oligonukleotidmi ostali magneticky prichytené, a ostatné sa zmyli. Tento proces bol zopakovaný s oligonukleotidmi každého vrcholu grafu. V kroku 5 bol celkový výsledok znovu zosilnený PCR a elektroforézne prefiltrovaný.

3.2 Výhody a problémy DNA počítačov

Teoreticky majú DNA výpočty potenciál prevýšiť výkon elektronických počítačov. Adleman vo svojom článku uvádza, že DNA výpočty môžu dosahovať až 10^{20} a viac operácií za sekundu a spotreba energie na 2×10^{19} operácií odpovedá približne 1 J. Ukladanie informácií prostredníctvom DNA je tiež veľmi efektívne. Hustota informácií môže dosahovať 1 bit na kubický nanometer. Napriek tomu praktické prekážky pri tvorbe počítača založeného na DNA stále pretrvávajú. Sú to problémy spôsobené komplexnosťou prípravy experimentov a chybovosťou takýchto počítačov. Taktiež, pre jednoduché aritmetické alebo logické operácie sú z hľadiska rýchlosti výpočtu elektronické počítače vhodnejšie. [3]

3.3 Formalizácia DNA výpočtov

V tejto kapitole bol popísaný Adlemanov postup pri riešení problému Hamiltonovskej cesty grafom a biologické detaily jeho riešenia. Na zavedenie simulácie DNA počítania do prostredia číslicových počítačov je však potrebné stanoviť formálne definície pre jednotlivé operácie. Adlemanov model výpočtu je možné zaradiť do kategórie filtračných modelov. Vo filtračných modeloch pozostáva výpočet z postupnosti operácií nad konečnou multimnožinou reťazcov. Multimnožiny sú množiny, ktoré môžu obsahovať viac ako jeden výskyt rovnakého prvku. Na začiatku výpočtu je teda nejaká multimnožina, ktorá je v priebehu výpočtu upravovaná definovanými operáciami. Výsledkom výpočtu je potom nejaká ďalšia multimnožina. Počiatočná multimnožina by mala byť taká, aby jej podmnožina obsahovala všetky možné riešenia problému. Vo výpočte sú z počiatočnej multimnožiny filtrované reťazce, ktoré nemôžu byť výsledkom. Operácie použité v Adlemanovom modeli, ako to uvádza [2], môžu byť formalizované nasledujúcim spôsobom:

- $separate(T, S)$. Z danej množiny T a podreťazcu S vytvor dve nové množiny $+(T, S)$ a $-(T, S)$, kde $+(T, S)$ sú všetky reťazce z T obsahujúce S a $-(T, S)$ sú všetky reťazce z T neobsahujúce S.
- $merge(T_1, T_2, \ldots, T_n)$. Z daných množín $T_1, T_2, \ldots, T_n$ vytvor $\cup(T_1, T_2, \ldots, T_n) = T_1 \cup T_2 \cup \ldots T_n$.
- $detect(T)$. Z danej množiny T vráť $true$ ak je T neprázdna, inak vráť $false$.

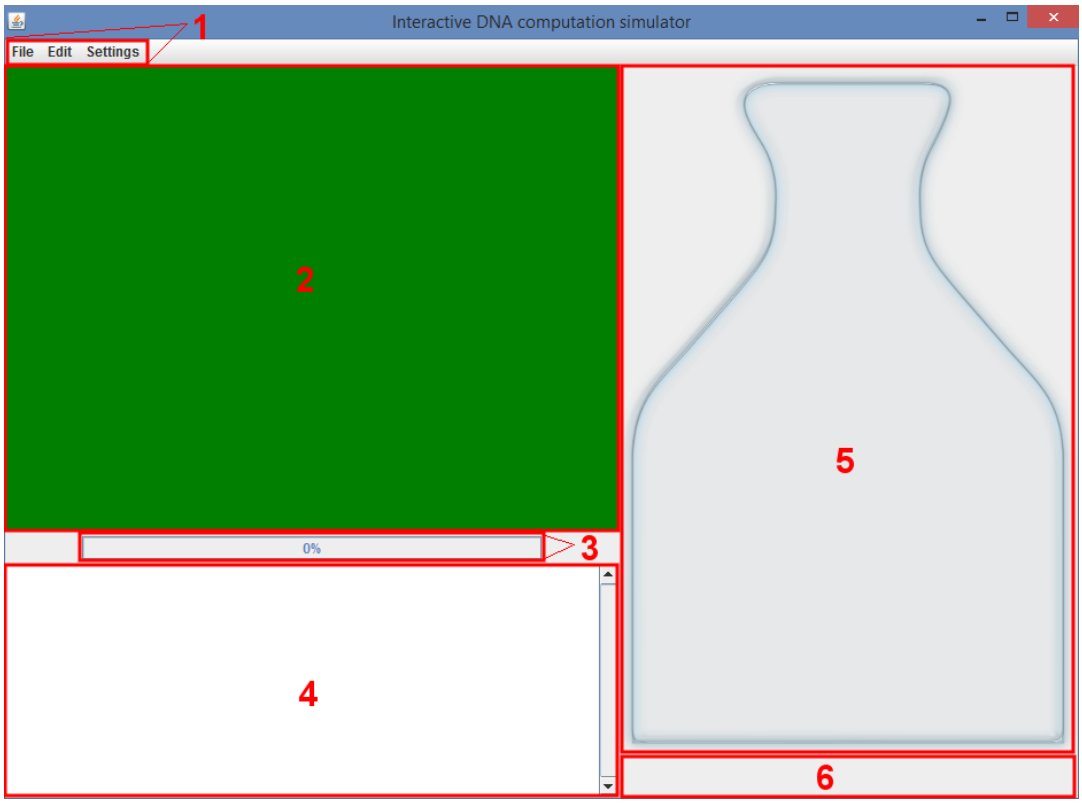
Prvá operácia je formalizmom procesu výberu vlákien obsahujúcich hľadaný oligonukleotid (pomocou ukotvenej matice, prípadne magnetických guľôčok). Druhá operácia je akýmsi opakom predchádzajúcej a odpovedá zmiešaniu vytvorených kópií oligonukleotidov v prvej fázi experimentu. Posledná uvedená operácia je kontrola existencie výsledku a môže byť interpretovaná ako overovanie množstva DNA pre ďalšie použitie po jednotlivých filtračných krokoch.

Kapitola 4

DNA Simulátor

Táto kapitola sa zaoberá implementačnou časťou práce. Prezentuje vlastnosti vytvoreného programu a detailne objasňuje dôležité programovacie štruktúry a postupy. Program je napísaný v jazyku `Java` a využíva štandardné grafické knižnice `awt` a `swing`. Programovacie techniky použité pri tvorbe grafického rozhrania programu sú uvedené v [6].

4.1 Vlastnosti programu



Obrázok 4.1: Grafické užívateľské rozhranie programu; 1 Hlavné menu programu 2 Priestor pre graf 3 „Progress bar“ 4 Textová oblasť 5 Banka 6 Priestor pre vykreslenie oligonukleotidu predstavujúceho Hamiltonovskú cestu grafom

Vytvorený program slúži ako demonštračná pomôcka pre pochopenie Adlemanovho postupu pri riešení problému Hamiltonovskej cesty grafom. Z tohoto dôvodu je výsledný program prehľadný a ponúka užívateľovi niekoľko druhov spätnej väzby. Prehľad grafického rozloženia programu je zobrazený na obrázku 4.1.

Hlavné menu programu umožnuje užívateľovi ovládať priebeh simulácie a nastavovať si vlastné parametre. V prvku `File` je možné vytvoriť novú inštanciu programu, načítať graf zo súboru, uložiť graf do súboru a vypnúť program. Prvok `Edit` obsahuje všetky funkcie na prípravu a riadenie simulácie. Týmto prvkom je ovládané krokovanie simulácie, prezentácia výsledkov, prípadne opätovné spustenie simulácie. Posledným prvkom menu je `Settings`. Tu môže užívateľ zmeniť vzhľad aplikácie a parametry simulácie. Tento prvok tiež obsahuje vysvetlivky a informácie o programe.

Priestor pre graf slúži na prípravu grafu, v ktorom bude hľadaná Hamiltonovská cesta. Graf je možné vytvoriť načítaním zo súboru, alebo ručne. Vrcholy sú vytvorené po kliknutí myšou. Hrany môžu byť vytvorené iba medzi dvomi vrcholmi metódou „drag-and-drop“.

Ostatné časti grafického rozhrania informujú užívateľa o priebehu simulácie. Tvoria spätnú väzbu programu. „Progress bar“ znázorňuje postup vykonávaného simulačného kroku v percentách. Textová oblasť hlási ukončenie každého simulačného kroku. V banke sú počas každého simulačného kroku vykresľované aktuálne spracúvané oligonukleotidy reprezentujúce rôzne cesty grafom. Pri zdarnom ukončení simulácie je možné vykresliť nájdené riešenia do grafu. Oligonukleotidy reprezentujúce tieto riešenia sú vykreslené do priestoru pod bankou.

4.2 Implementácia prípravnej fáze simulácie

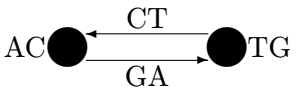
Po spustení programu sú ako prvé vytvorené inštancie tried zodpovedné za spracovanie užívateľových príkazov a riadenie chodu simulácie. Nasleduje vytvorenie grafického rozhrania, ktoré sa skladá z inštancií tried pre hlavné menu, priestor pre graf, „progress bar“, textovú oblasť a banku s priestorom pre vykreslenie oligonukleotidov predstavujúcich riešenie.

Po vytvorení všetkých zmienených komponentov má užívateľ možnosť vytvoriť svoj graf. Vnútorná štruktúra grafu je zložená zo zoznamu vrcholov a zoznamu hrán. Každý vrchol si uchováva informáciu o svojej polohe, prípadne o tom či je počiatočný alebo koncový z hľadiska problému Hamiltonovskej cesty. Každá hrana nesie informáciu o dvoch vrcholoch, ktoré spája. Vrchol aj hrana majú v programe vlastné tiedy.

Dokončením grafu začína prípravná fáza simulácie. Najprv je užívateľ vyzvaný aby vybral počiatočný a koncový vrchol. Tieto informácie sú uložené do odpovedajúcich inštancií. Ďalej je každý vrchol ohodnotený jedinečnou postupnosťou oligonukleotidov a postupnosť oligonukleotidov každej hrany je dotvorená tak, aby odpovedala Adlemanovmu modelu. Program sa od modelu líši tým, že každý oligonukleotid má rovnaký počet nukleotidov. Adleman vo svojom modele zakódoval všetky hrany vychádzajúce z počiatočného vrcholu alebo vchádzajúce do koncového vrcholu väčším počtom nukleotidov ako ostatné hrany a vrcholy. Z dôvodu prehľadnosti program používa zjednodušenie chemických vlstností pri spájaní oligonukleotidov DNA. Program vytvára náhodné cesty grafom tak, že vždy spája 3' koncovú polovicu jedného oligonukleotidu s 5' koncovou polovicou druhého oligonukleotidu. Kvôli tomuto zjednodušeniu je možná nasledujúca úvaha.

Pre zaistenie jedinečnosti 5' koncovej polovice aj 3' koncovej polovice každého oligonukleotidu, považujme oba tieto konce za samostatné oligonukleotidy, ktoré neskôr spojíme do jedného. Keďže každý nukleotid môže niesť štyri rôzne informácie, pre jedinečné zakódovanie jedného oligonukleotidu vzniká vzťah $|o| = \log_4 x$, kde $|o|$ je dĺžka oligonukleotidu a x

je dvojnásobok počtu vrcholov grafu. Vďaka vyššie zmienenému zjednodušeniu nie je potrebné venovať zakódovaniu hrán špeciálnu pozornosť, pretože nemôže nastať kolízia medzi zakódovaním žiadneho z vrcholov a žiadnou z hrán. Program generuje jednínečné zakódovanie vrcholov tak, aby komplementárne polovice oligonukleotidov jednotlivých vrcholov boli na rovnakom konci čo, pri uvažovanom zjednodušení, zamedzuje vytvoreniu spojenia medzi dvomi vrhcolmi. Pri najjednoduchšom úplnom orientovanom grafe, ktorý obsahuje Hamiltonovskú cestu (dva vrcholy spojené dvomi hranami), prvky nadobudnú nasledujúce zakódovanie. V grafe sú dva uzly, z čoho vzniká potreba štyroch jedinečných oligonukleotidov. Z predošlého vzťahu vyplýva, že každý oligonukleotid bude mať dĺžku jeden nukleotid. V dôsledku bude teda každý vrchol ohodnotený oligonuleotidom, ktorý vznikne spojením dvoch jedinečných nukleotidov (pre zaistenie jedinečnosti oboch koncov) a bude mať dĺžku dva nukleotidy. Túto situáciu znázorňuje obrázok 4.2.



Obrázok 4.2: Ohodnotenie prvkov jednoduchého grafu

Aby bol výpočet pri tvorbe ciest grafom rýchlejší, nukleotidy sú pri výpočte kódované číselne. Adenín je reprezentovaný nulou, cytozín jednotkou, tymín dvojkou a guanín trojkou. Keďže komplementárne nukleotidy sa od seba líšia hodnotou duhého bitu (binárne: C – 01, G – 11), pre zistenie schopnosti spojenia dvoch nukleotidov stačí na dvoch bitoch k jednému z nich pripočítať dvojku. Po vytvorení každého nového ohodnotenia nasleduje fáza generovania klonov oligonukleotidu s týmto ohodnotením. Z dôvodu nedostatku pamäte pri generovaní veľkého počtu klonov nedochádza k vytvoreniu každého klonu, ale iba jedného referenčného, pričom počet dostupných klonov k tomuto referenčnému je uložený do samostatného poľa.

4.3 Implementácia generovania náhodných ciest grafom

Generovanie náhodných ciest grafom (krok 1 z Adlemanovho modelu) je riadené algoritmom 1. Aby bola cesta grafom Hamiltonovská, jej počet hrán musí byť o jednu menší ako jej počet vrcholov. Každá z týchto ciest musí byť na oboch koncoch spojená so správnym vrcholom. Preto je počet spojení potrebný pre vytvorenie jednej Hamiltonovskej cesty vyjadrený vzťahom $s = 2 \times (vrcholy - 1)$.

Algoritmus 1 má optimistický prístup a ponúka šancu každému klonu každého vrcholu, byť súčasťou Hamiltonovskej cesty (počet spojení je násobený počtom klonov). Simulácie ukázali, že samotná táto hodnota nestačí, pretože čím viac spojení bolo vytvorených, tým menej potenciálnych spojení bolo k dispozícii pri ďalšom prechode cyklu. Čím menej potenciálnych spojení existuje tým ťažšie je náhodne nájsť dva také prvky, ktoré môžu spojenie vytvoriť, čo pri konci tohto kroku simuláciu výrazne spomaľovalo. Preto bola zavedená druhá hraničná hodnota (premenná *missmatches*), ktorá bola po niekoľkých rôznych simuláciach určená ako štvornásobok počtu očakávaných spojení.

Algoritmus 1: Generovanie náhodných ciest grafom

```
1  matches = 2 × (vertices − 1) × clones
2  missmatches = 4 × matches
3  while match < matches and missmatch < missmatches do
4      path1 = pickPath()
5      path2 = pickPath()
6      if complementary(path1, path2) or complementary(path2, path1) then
7          match++
8      else
9          missmatch++
10     end if
11 end while
```

Cesty grafom sú v programe implementované jednosmerne viazaným zoznamom celých čísel. Celé čísla predstavujú zakódovanie jednotlivých prvkov (hrán alebo vrcholov) grafu. Hamiltonovská cesta grafom so štyrmi vrcholmi by teda bola reprezentovaná zoznamom so siedmimi celými číslami.

Náhodný výber dvoch oligonukleotidov[1] k vytvoreniu spojenia komplikuje existencia pomocného poľa s informáciou o počte dostupných klonov jednotlivých prvkov. Na základe čísla vygenerovaného generátorom pseudonáhodných čísiel program rozhodne, či vziať už existujúcu cestu grafom, alebo použiť jeden z referenčných oligonukleotidov. Cesty grafom a referenčné oligonukleotidy sú tiež uložené v samostatnom poli. Generátor vygeneruje hodnotu od nuly do hodnoty získanej sčítaním všetkých dostupných klonov v pomocnom poli so všetkými už vytvorenými cestami[2]. Ak je toto číslo menšie ako počet prvkov v pomocnom poli, pre pokus o spojenie sa vyberie jeden z referenčných oligonukleotidov (zase pseudonáhodným spôsobom). Informácia o počte dostupných klonov daného referenčného oligonukleotidu je v zápätí dekrementovaná. V opačnom prípade sa pseudonáhodným spôsobom vyberie jedna z už existujúcich ciest.

Po výbere dvoch oligonukleotidov je na rade vytvorenie spojenia medzi nimi. Program najprv vytvorí správny komplement k 3' polovici posledného prvku zoznamu, ktorým je zakódovaná prvá cesta. Tento komplement potom porovná s 5' polovicou prvého prvku zoznmamu, ktorým je zakódovaná druhá cesta. Túto situáciu znázorňuje algoritmus 2.

Algoritmus 2 si najprv pripraví požadované konce oboch spájaných oligonukleotidov (o_1 a o_2). Premenná *partial* uchováva komplementárne nukleotidy k jednotlivým nukleotidom cesty o_1. Premenná *complement* bude po skončení cyklu obsahovať hodnotu komplementu oligonukleotidu o_1. V tele cyklu je najprv výsledný komplement posunutý o dva bity (dvomi bitmi je reprezentovaný jeden nukleotid). Potom je do pomocnej premennej *partial* uložená hodnota aktuálne spracúvaného nukleotidu o_1, ku ktorej je pripočítaná hodnota reprezentujúca rozdiel komplementárnych nukleotidov. Posledná úprava premennej *partial* spočíva v kontrole pretečenia ponechaním prvých dvoch bitov a vynulovaním všetkých ostatných. Nakoniec je hodnota komplementu aktuálneho nukleotidu v premennej *partial* pripočítaná k celkovému komplementu v premennej *complement*. Návratová hodnota určuje, či vytvorený komplementárny oligonukleotid je zhodný s o_2, teda či je možné spojiť dve nájdené cesty.

[1] Termíny oligonukleotid a cesta grafom sú v tomto kontexte zameniteľné

[2] Okrem referenčných oligonukoletidov

Algoritmus 2: Overenie komplementarity

```
1  o1 = secondHalf(o1)
2  o2 = firstHalf(o2)
3  partial = 0
4  complement = 0
5  for i = |o1|; i > 0; i = i − 2 do
6      complement = complement << 2
7      partial = o1 >> (i − 2)
8      partial = partial + 2
9      partial = partial & 3
10     complement = complement + partial
11 end for
12 return complement == o2
```

V prípade úspechu je druhá cesta pripojená na koniec zoznamu reprezentujúceho prvú cestu. Ak je niektorá z ciest referenčným oligonukleotidom, je pred pripojením potrebné vytvoriť kópiu tohto oligonukleotidu a pridať ju do poľa s cestami a referenčnými oligonukleotidami. Táto kópia je pridaná do zoznamu ciest ako samostatný prvok. V prípade neúspechu sa algoritmus 1 pokúsi spojiť vybrané cesty opačne.

4.4 Implementácia filtrovania neplatných ciest

Adleman vo svojom modele rozdelil filtráciu na tri časti:

- Ponechanie ciest, ktoré začínajú počiatočným a končia koncovým vrcholom
- Ponechanie ciest, ktoré navštevujú správny počet vrcholov
- Ponechanie ciest, ktoré navštevujú všetky vrcholy aspoň raz

Filtrovanie ciest grafom, ktoré nie sú hamiltonovské je v programe rozdelené do štyroch krokov. Poradie filtračných krokov bolo zvolené z praktických dôvodov. Ako prvý krok bola zvolená filtrácia ciest, ktoré nenavštevujú požadovaný počet vrcholov. Takéto poradie uľahčí implementáciu vykresľovania oligonukleotidov v banke (viac v 4.5). Z poľa ciest grafom sú odstránené tie cesty, ktoré nenavštevujú správny počet vrcholov. V cykle je kontorolovaná každá cesta v poli ciest grafom. Tá, ktorej zoznam má viac alebo menej prvkov ako je vzhľadom ku konkrétnemu grafu očakávané, je z poľa odstránená. Ďalší filtračný krok zmaže všetky cesty, ktoré nezačínajú počiatočným vrcholom. Program opäť cyklicky prechádza zvyšné platné cesty a porovnáva hodnotu prvého prvku ich zoznamu s hodnotou vrcholu, ktorý užívateľ označil za počiatočný počas prípravy simulácie. Tretí krok je vo svojej podstate rovnaký ako druhý. Rozdiel spočíva v porovnaní hodnoty posledného prvku zoznamu cesty s hodnotou vrcholu, ktorý užívateľ označil ako koncový. Posledný filtračný krok kontorluje, či daná cesta nenavštevuje niektorý z uzlov viac ako raz. Implementáciu tohto kroku znázorňuje algoritmus 3.

Parametrom funkcie v algoritme 3 je zoznam, ktorý reprezentuje cestu grafom. Keďže je táto funkcia rekurzívna je pamäťovo náročnejšia ako funkcie v predchádzajúcich krokoch. Preto bol krok, v ktorom je implementovaná zaradený na koniec. Ako prvá je vo funkcii uvedená ukončovacia podmienka. Cesta, ktorú funkcia dostala ako parameter navštevuje

každý vrchol iba raz, ak pri najhlbšom zanorení dostane ako parameter posledný prvok zoznamu (prvý a posledný prvok zoznamu sú jeden a ten istý). V takom prípade funkcia skončí úspešne. Ak parametrom funkcie nie je posledný prvok zoznamu, využije sa premenná *current*, ktorá sa v cykle posunie vždy o dva prvky ďalej. Pri posune je nevyhnutné vynechať každý párny prvok, pretože ten označuje hranu grafu. Ak funkcia zistí, že hodnota premennej *current* je zhodná s hodnotou jej parametru, znamená to, že cesta, ktorú zoznam predstavuje, navštívila niektorý z vrcholov viac ako raz. V takom prípade funkcia skončí neúspešne. Ak sa pri priechode celého zoznamu nenájde jediná zhoda s hodnotou parametru funkcie, funkcia sa zanorí hlbšie, ale ako parameter dostane zoznam začínajúci ďalším vrcholom v poradí.

Algoritmus 3: Overenie viacnásobnej návštevy vrcholu

```
1  containsAllVertices(List inputList)
2    last = lastItem(inputList)
3    if last == inputList then
4        return true
5    end if
6    current = inputList
7    while current != last do
8        current = nextItem(nextItem(current))
9        if value(inputList) == value(current) then
10           return false
11       end if
12   end while
13   return containsAllVertices(nextItem(nextItem(current)))
```

Aby táto funkcia fungovala správne je nevyhnutné, aby boli najprv odstránené cesty, ktoré nenavštevujú správny počet vrcholov. V opačnom prípade by sa mohlo stať, že zoznam skúmanej cesty má párny počet prvkov a pri posune vždy o dva prvky ďalej by došlo k neoprávnenému prístupu do pamäte.

Každý krok simulácie je implementovaný v samostatnej triede. Tieto triedy sú za účelom rýchlo reagujúceho grafického rozhrania založené na štandardnej triede `SwingWorker`. Trieda `SwingWorker` obsahuje funkciu `doInBackground`, ktorej telo využíva vlastné vlákno na realizáciu príkazov. Výpočtovo náročné operácie sú preto vhodne obsiahnuté práve v tele tejto funkcie, čo má za následok odľahčenie hlavného vlákna zodpovedného za komunikáciu medzi jednotlivými objektmi programu.

4.5 Implementácia iných častí programu

Obrázok 4.3: Implementácia prvkov grafu; (a) ukážka spojenia medzi vrcholmi a hranou (b) znázornenie troch potrebných bodov pre vykreslenie šípky

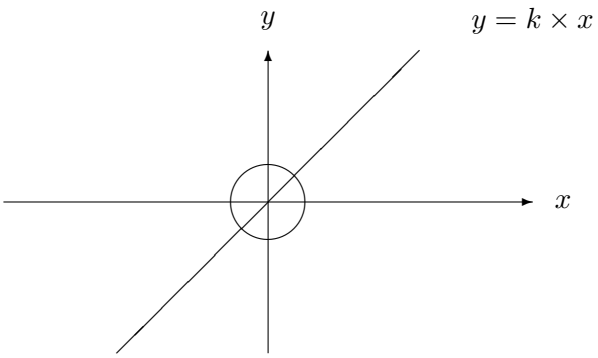
Pre jednoduchšiu identifikáciu prvkov grafu, program používa pre začiatok aj koniec hrany stredy vrcholov, ktoré spája. Pre vykreslenie šípky na znázornenie orientácie grafu je potrebné vypočítať pozície troch bodov, hrotu šípky a jej dvoch koncov. Tieto body zobrazuje obrázok 4.3.

Pri rôznych sklonoch hrán môže byť obtiažne tieto body identifikovať bez potrebných výpočtov. Nasledujúca sústava rovníc obsahuje dve rovnice. Prvou je rovnica kružnice s polomerom r a stredom v počiatku súradnicového systému. Táto kružnica predstavuje vrchol grafu. Druhou je rovnica priamky, ktorá prechádza stredom súradnicového systému a teda aj stredom kružnice z prvej rovnice. Priamka má sklon k a predstavuje hranu vchádzajúcu do vrcholu v grafe.

$$\begin{aligned} x^2 + y^2 &= r^2 \\ y &= k \times x \end{aligned}$$

$$\begin{aligned} x^2 + (k \times x)^2 &= r^2 \\ x^2 \times (1 + k^2) &= r^2 \\ x &= \pm \frac{r}{\sqrt{1 + k^2}} \end{aligned}$$

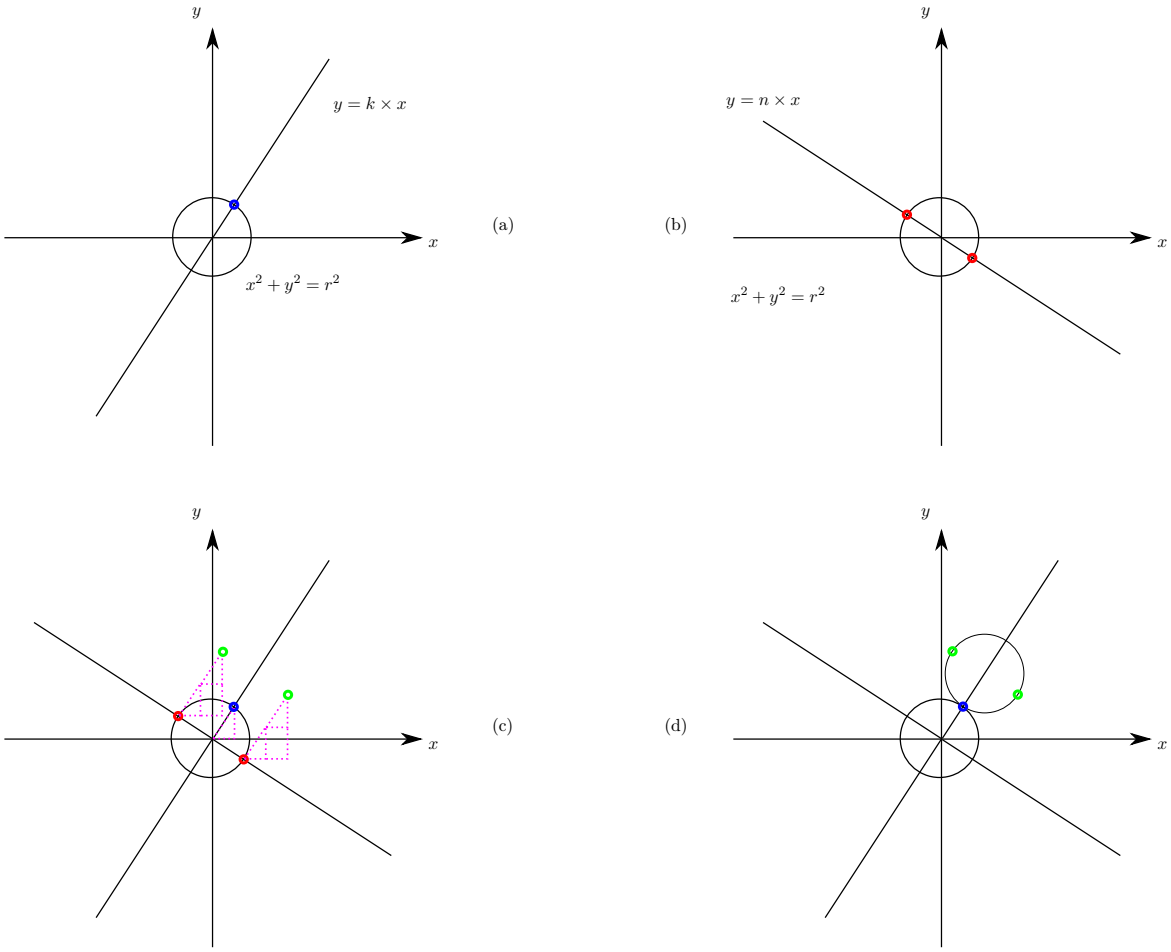
Znázornené riešenie tejto sústavy má dva výsledky. Tieto dva výsledky predstavujú dva body, v ktorých priamka pretína kružnicu (obrázok 4.4). Program rozhodne, ktorý z týchto dvoch bodov bude predstavovať hrot šípky na základe polohy východzieho vrcholu hrany. Ak je východzí vrchol vpravo od cieľového, vyberie sa bod s kladnou x-ovou súradnicou. V opačnom prípade sa vyberie ten so zápornou.



Obrázok 4.4: Zobrazenie prieniku priamky a kružnice v súradnicovom systéme

Pozíciu koncov šípky je možné vypočítať podobne ako pozíciu hrotu. Rozdiel je v rovnici priamky, kde je premenná k nahradená premennou n. Vzťah medzi týmito premennými, $n = -\frac{1}{k}$, spôsobí, že nová priamka je kolmá na pôvodnú priamku. Po dosadení do rovnice kruhu, opäť získavame dve riešenia. V tomto prípade budú obe použité ako konce šípky. Aby vytvorená šípka opticky ukazovala smerom k cieľovému vrcholu je potrebné dve

nájdené riešenia posunúť. Posun na správnu pozíciu využíva vetu o podobnosti trojuholníkov. K súradniciam nájdených riešení sa pripočíta dvojnásobok súradníc hrotu šípky, čo vo výsledku posunie konce šípky na pomyselnú kružnicu so stredom vzdialeným dve dĺžky polomeru od počiatku súradnicového systému v smere pôvodnej priamky (priamka so smernicou k). Pre zúženie šípky stačí pri výpočte jej koncov použiť v sústave rovníc kružnicu s menším polomerom. Celý výpočet zobrazuje obrázok 4.5



Obrázok 4.5: Výpočet hraničných bodov šípky; (a) hrot šípky (b) konce šípky pred posunom (c) posun koncov (d) posunuté konce a hrot na pomyselnej kružnici

K trom získaným bodom sú pripočítané súradnice stredu cieľového vrcholu, čím je zaručený ich posun na správnu pozíciu v grafe.

Za vykreslovanie oligonukleotidov v banke je zodpovedná trieda `AnimationPanel`. Pri vzniku novej inštancie tejto triedy sú, ako jej súčasť, vytvorené dve inštancie triedy `BufferedImage`. Táto trieda umožnuje tvorbu nových, prípadne manipuláciu existujúcich, obrázkov. Jedna z týchto inštancií obsahuje zo súboru načítaný obrázok s bankou, ktorý slúži ako pozadie. Druhá inštancia (ďalej výplň banky) má priesvitné pozadie, na ktoré budú vykreslované jednotlivé oligonukleotidy. Jej veľkosť a pozícia sú nastavené tak, aby sa opticky vošla do vnútra banky. Na začiatku každého kroku simulácie je jej obsah zmazaný, ponechávajúc opäť iba priesvitné pozadie. Pre vykreslenie nového prvku na exis-

tujúce plátno je v `Jave` volaná funkcia `repaint`, ktorá prekreslí obsah svojho objektu, ale popritom zmaže jeho predchádzajúci obsah.

Toto správanie je z pohľadu efektivy programu veľmi nežiadúce, pretože pri každom novom oligonukleotide je potrebné vykresliť aj všetky doteraz nájdené oligonukleotidy. To znamená, že pri k volaniach prekreslovacej funkcie by bolo vykreslených $k!$ oligonukleotidov. Program tento problém rieši použitím výplne banky. Keďže výplň banky je uložená v samostatnej premennej, pri prekreslení objektu ostáva nedotknutá. Pri zavolaní prekreslovacej funkcie je objekt, v tomto prípade `AnimationPanel`, zmazaný. Vykreslovaný oligonukleotid je vykreslený do výplne banky, výplň banky je vykreslená na objekt a na záver je na objekt vykreslené pozadie s bankou. Toto riešenie síce vykoná pri k volaniach prekreslovacej funkcie $3 \times k$ prekreslení, ale vykreslených oligonukleotidov je iba k, nie $k!$.

Je relatívne zložité vykresliť medzi dvojzávitnice predstavujúce cesty grafom, oligonukleotidy predstavujúce jednotlivé prvky grafu. V `Jave` je pre vykreslenie dvojrozmerných primitívnych útvarov potrebné udať počiatočnú x-ovú a y-ovú súradnicu a šírku a výšku útvaru. Keďže dvojzávitnica má šírku dvoch vlákien, a oligonukleotid iba jedného, ich kombinácia pri vykreslení môže mať tieto dva následky. Rozdiel šírok buď vytvorí prázdne biele miesto vo vnútri banky, alebo posunie ostanté dvojzávitnice o výšku jedného vlákna nižšie a spôsobí tak prekrytie už vykreslených dvojzávitníc. Program sa tomuto problému vyhýba tak, že neposkytuje príležitosť pre takúto kombináciu. V každom kroku simulácie sú vykreslované prvky s rovnakou výškou.

Pred každým krokom simulácie je v triede `AnimationPanel` vypočítaný maximálny počet prkov vykreslovaných na jeden riadok. Tento počet je vypočítaný ako podiel súčinu počtu prvkov grafu, počtu generovaných klonov pre jeden prvok a výšky útvaru vykreslovaného v danom kroku so súčinom výšky výplne banky a počtu prvkov, z ktorých bude vykreslený jeden (sto, ak má byť vykreslený jeden zo sta, tisíc ak jeden z tisíc, atď.).

Kapitola 5

Experimenty

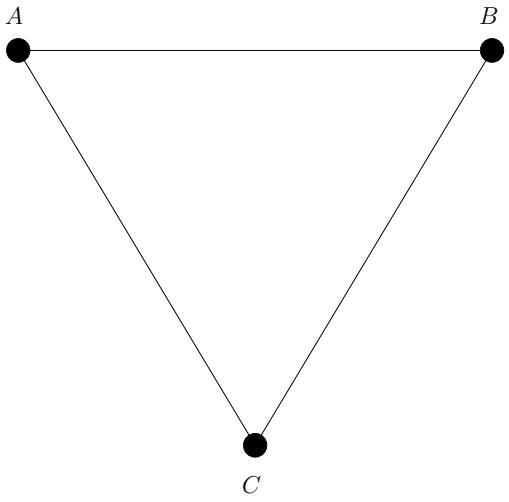
V tejto kapitole sú uvedené experimenty pri rôznych nastaveniach simulátoru. Každý experiment začína špecifikáciou parametrov simulácie. Graf, nad ktorým je experiment spustený, má popísané vrcholy kvôli určeniu počiatočných a koncových bodov simulácie. Súčasťou experimentov je aj tabuľka obsahujúca simulačné štatistiky vytvorené zo spätnej väzby programu. Program bol navrhnutý tak, aby neorientované grafy interpretoval ako grafy s hranami orientovanými oboma smermi. Spotrebu operačnej pamäte je možné odhadnúť pomocou celkového počtu oligonukleotidov vo všetkých vygenerovaných cestách grafom. Každá cesta je objektom obsahujúcim hodnotu oligonukleotidu (premenná typu `Integer`) a referenciu na ďalšiu časť cesty, ktorá je tiež objektom. Zo špecifikácie jazyku `Java` [5] vyplýva, že pre každú časť cesty je potrebných 32 B (16 B pre vytvorenie objektu, 8 B pre referenciu objektu, 4 B pre premennú s oligonukleotidom a 4 B pre zarovnanie na najbližší násobok 8 B). Keďže počet častí ciest je daný celkovým počtom oligonukleotidov v nich, pre celkovú spotrebu pamäte je potrebné vynásobiť vyššie uvedenú hodnotu týmto počtom. Informácie o spotrebe pamäte uvedené v tejto kapitole boli vypočítané priamo z údajov o stave operačnej pamäte (príkaz `free -m`). Na konci kapitoly sú vyvodené závery z experimentov a zhodnotenie výkonu programu.

5.1 Experiment nad trojuholníkovým grafom

Prvý experiment bol vykonaný nad trojuholníkovým grafom z obrázku 5.1. Vrchol A bol stanovený za počiatocný vrchol a vrchol C za koncový. V prvom behu experimentu bolo pre každý prvok grafu vygenerovaných milión kópií. V druhom behu experimentu bolo pre každý prvok grafu vygenerovaných desať miliónov kópií.

Celkový počet kópií	9 000 000	90 000 000
Generovaných ciest	2 219 768	22 200 507
Zmazaných na základe:		
dĺžky	2 094 436	20 998 389
počiatočného vrcholu	108 827	1 088 958
koncového vrcholu	8 124	79 686
opakovanej návštevy vrcholu	0	0
Počet hamiltonovských ciest	3 390	33 483

Tabuľka 5.1: Štatistiky simulácie s trojuholníkovým grafom



Obrázok 5.1: Trojuholníkový graf

V trojuholníkovom grafe je pomer vrcholov k hranám 1:2. Z tabuľky 5.1 možno stanoviť úspešnosť vytvorenia hamiltonovskej cesty grafom na 0,1518 %. Túto hodnotu možno získať spriemerovaním pomerov vytvorených hamiltonovských ciest k ostatným v oboch behoch simulácie. Maximálna spotreba pamäte pri jednotlivých behoch bola 308 MB, respektíve 1 995 MB.

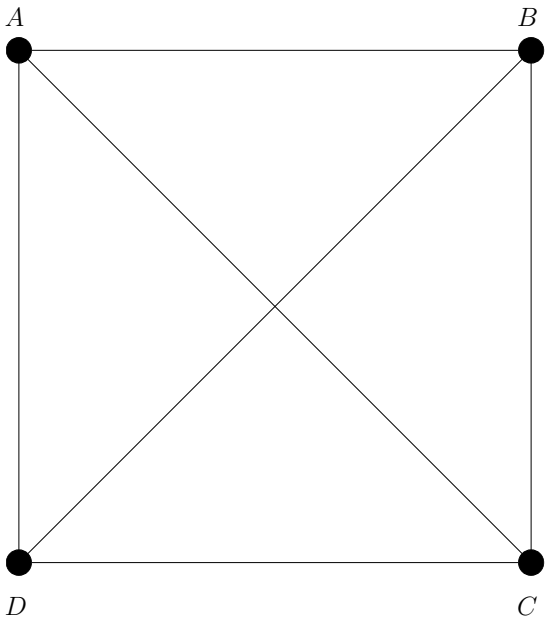
5.2 Experiment nad štvorcovým grafom

Ďalším z vykonaných experimentov bol experiment nad štvorcovým grafom (obrázok 5.3). Vrchol A bol stanovený za počiatocný vrchol a vrchol D za koncový. Rovnako ako v predošlom experimente bolo v prvom behu experimentu pre každý prvok grafu vygenerovaných milión kópií. V druhom behu experimentu to bolo desať miliónov kópií.

Celkový počet kópií	16 000 000	160 000 000
Generovaných ciest	3 308 272	33 077 250
Zmazaných na základe:		
dĺžky	3 288 050	32 874 260
počiatočného vrcholu	19 521	195 697
koncového vrcholu	539	5 412
opakovanej návštevy vrcholu	116	1 267
Počet hamiltonovských ciest	58	626

Tabuľka 5.2: Štatistiky simulácie so štvorcovým grafom

V štvorcovom grafe je pomer vrcholov k hranám 1:3. Tabuľka 5.2 prezrádza, že úspešnosť vytvorenia hamiltonovskej cesty takýmto grafom je $1,8341 \times 10^{-3}$ %. Maximálna spotreba pamäte programom pri jednotlivých behoch dosahovala 387 MB a 2 662 MB.



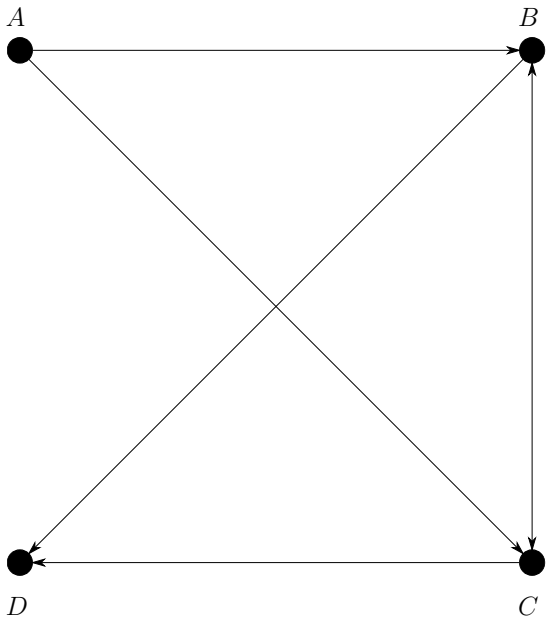
Obrázok 5.2: Štvorcový graf

5.3 Experiment s pridávaním hrán do grafu

Na grafe so štyrmi vrcholmi bol vykonaný ešte jeden experiment za účelom odvodenia vzťahu medzi počtom hrán v grafe a pravdepodobnosťou generovanie hamiltonovskej cesty grafom. Ako počiatočný a koncový vrchol grafu boli použité rovnaké vrcholy ako v predošlom experimente. Simulácia začala na šiestich vhodne zvolených hranách. Tieto hrany boli zvolené tak, aby bolo možné generovať obe hamiltonovské cesty daným grafom. Po ukončení prvého behu simulácie, bola do grafu náhodne pridaná nová hrana a simulácia bola spúštaná znovu, až kým nebol v grafe dovŕšený maximálny počet hrán. V orientovanom grafe so štytmi vrcholmi, môžu z každého vrcholu viesť až tri hrany, čo udáva maximálny počet dvanástich hrán. Výsledky simulácie vplyvu pridávania hrán do grafu bez možnosti vytvorenia novej hamiltonovskej cesty ukazuje tabuľka 5.3.

Počet hrán	**Generovaných ciest**	**Hamiltonovských ciest**	**Pomer** $[10^{-3}\%]$
6	2 569 605	18 374	715,05
7	2 935 074	7 562	257,64
8	2 961 689	2 261	76,34
9	3 036 682	922	30,36
10	3 138 099	346	11,00
11	3 225 902	114	3,53
12	3 307 251	74	2,23

Tabuľka 5.3: Štatistiky simulácie pri zvyšovaní počtu hrán



Obrázok 5.3: Štvorcový graf so šiestimi hranami

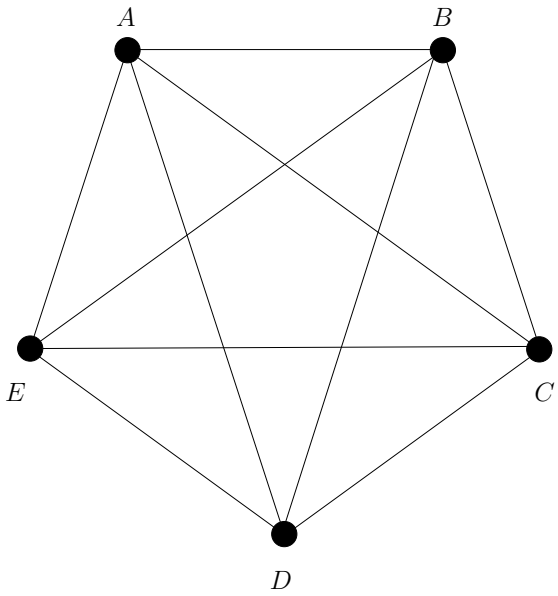
5.4 Experiment nad päťuholníkovým grafom

Kvôli priveľkým nárokom na pamäť, nebol experiment nad päťuholníkovým grafom spustený na osobnom počítači. Pre nájdenie aspoň jednej hamiltonovskej cesty grafom bolo vytvorených desať miliónov kópií každého prvku grafu a pri generovaní ciest grafom bolo spotrebovaných viac ako 3500 MB operačnej pamäte. Z tohto dôvodu bol experiment vykonaný na výkonnejšom školskom serveri. Vrchol A bol označený za počiatočný vrchol a vrchol E za vrchol koncový. V prvom behu simulácie program počítal s dvadsiatimi miliónmi kópií každého prvku. V druhom kole to bolo päťdesiat miliónov.

Celkový počet kópií	500 000 000	1 250 000 000
Generovaných ciest	76 383 010	190 978 135
Zmazaných na základe:		
dĺžky	76 380 886	190 972 548
počiatočného vrcholu	2 076	5 422
koncového vrcholu	54	149
opakovanej návštevy vrcholu	17	36
Počet hamiltonovských ciest	2	5

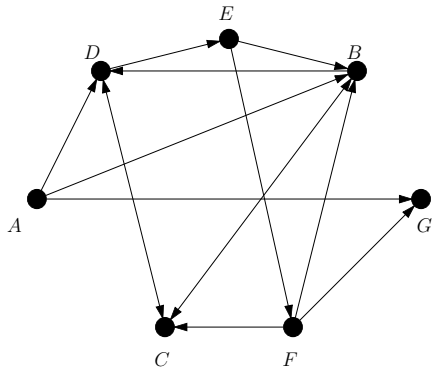
Tabuľka 5.4: Štatistiky simulácie s päťuholníkovým grafom

V päťuholníkovom grafe je pomer vrcholov k hranám 1:4. Údaje v tabuľke 5.4 stanovujú úspešnosť vytvorenia hamiltonovskej cesty päťuholníkovým grafom na $2,6 \times 10^{-6}$ %. Pri jednotlivých behoch programu dosahovala spotrebovaná operačná pamäť 6 204 MB, respektíve 14 813 MB.



Obrázok 5.4: Päťuholníkový graf

5.5 Experiment nad Adlemanovým grafom



Obrázok 5.5: Adlemanov graf

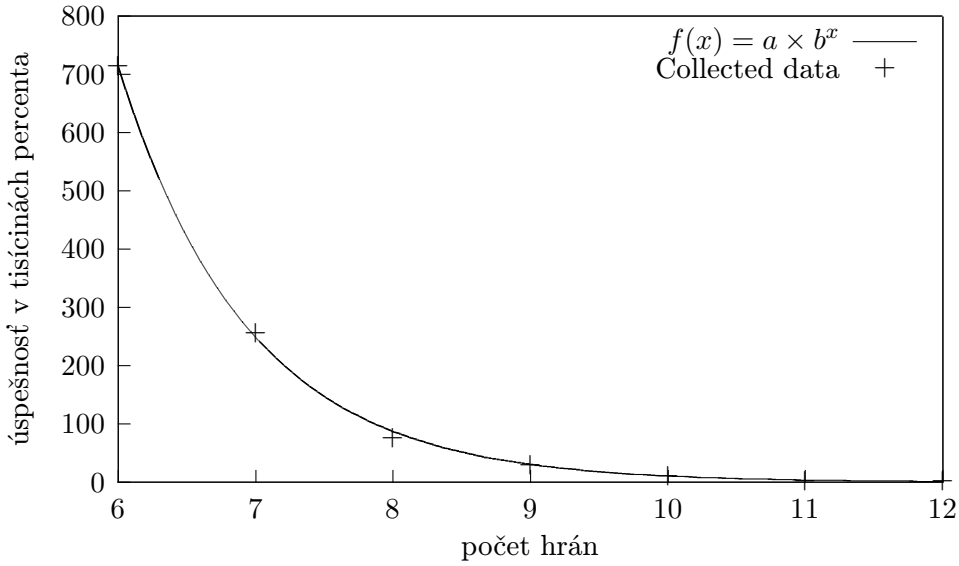
Rovnako ako pri päťuholníkovom grafe, aj Adlemanov graf musel byť odsimulovaný na školskom serveri, aby bol vygenerovaný dostatočný počet ciest na nájdenie Hamiltonovskej cesty grafom. V grafe 5.5 bol vrchol A označený za počiatočný a vrchol G za koncový. Simulácia počítala najprv s dvadsiatimi miliónmi kópií každého prvku grafu, a potom so štyridsiatimi miliónmi.

Adlemanov graf má pomer vrcholov a hrán 1:2. Kvôli veľkému počtu uzlov je však úspešnosť vytvorenia hamiltonovskej cesty grafom iba $1,4 \times 10^{-6}$ % ako to udáva tabuľka 5.5. Pri prvom behu simulácie bolo spotrebovaných 8 945 MB operačnej pamäte. Pri druhom behu jej bolo spotrebovanej 16 693 MB.

Celkový počet kópií	420 000 000	840 000 000
Generovaných ciest	91 588 287	183 185 735
Zmazaných na základe:		
dĺžky	91 588 043	183 185 169
počiatočného vrcholu	240	540
koncového vrcholu	20	40
opakovanej návštevy vrcholu	1	1
Počet hamiltonovských ciest	1	3

Tabuľka 5.5: Štatistiky simulácie s Adlemanovým grafom

5.6 Závery vyvodené z experimentov



Obrázok 5.6: Pokles úspešnosti generovania hamiltonovskej cesty grafom

Vo svojej práci Adleman uviedol, že pre nájdenie hamiltonovskej cesty grafom, ktorý vytvoril, použil 50 pmol oligonukleotidov pre každý prvok grafu. Táto hodnota v prepočte predstavuje 30 110 707 500 000 klonov každého prvku grafu. Celkový počet klonov použitých pri generovaní ciest bol dvadsaťjeden násobkom tohto čísla. Z tabuľky 5.5 je možné odvodiť, že počet ciest, ktorý program vytvorí je približne 22 % z celkového počtu klonov, čo predstavuje 139 111 468 650 000. Z údajou o spotrebe pamäte programu pri rôznych nastaveniach vychádza, že pre simuláciu Adlemanovho experimentu v mierke akú popisuje vo svojej práci, by bolo spotrebovaných približne 1 173 TB operačnej pamäte.

Vo vykonaných simuláciách je možné pozorovať trend exponenciálneho klesania efektivity generovania hamiltonovskej cesty grafom s nárastom pomeru počtu hrán k počtu vrcholov v grafe. Tento pokles je ešte drastickejší ak sa pri náraste počtu hrán zvyšuje aj komplexnosť grafu, čo potvrdzuje aj simulácia nad Adlemanovým grafom. Pomer počtu hrán k počtu vrcholov Adlemanovho grafu je menší ako pomer v päťuholníkovom grafe, no napriek tomu je percento úspešného generovania hamiltonovskej cesty o mnoho menšie.

Štatistiky z tabuľky 5.3 boli zakreslené do grafu na obrázku 5.6 a preložené exponenciálnou krivkou, s koeficientmi $a = 396141$ a $b = 0.349045$.

Vytvorený program úspešne nachádza hamiltonovské cesty v grafoch s tromi a štyrmi vrcholmi. Pri takýchto grafoch je možné simulácie spúšťať na bežne dostupných počítačoch. Na simuláciu úplných päťvrcholových grafov je potrebné použiť výkonnejší systém s aspoň 15 GB operačnej pamäte. Neúplné grafy s viac ako piatimi vrcholmi je tiež možné simulovať na takýchto systémoch. Pre vykonanie simulácie nad ich úplnymi alternatívami by však ani takýto systém nebol dostačujúci. Časová zložitosť nebola pri experimentoch uvádzaná, pretože je viac závislá na vykreslovaní nájdených ciest do banky, ako na samotnej simulácii. Celkové trvanie simulácie dosahuje 4 až 8 minút pre jednoduché grafy (troj a štvorvrcholové) a rastie s počtom kópií na prvok, so zložitosťou grafu a s počtom vykreslovaných ciest v banke.

Kapitola 6

Záver

V práci boli spracované výhody a úskalia využívania DNA pri matematických výpočtoch. Boli popísané operácie nad DNA potrebné k takýmto výpočtom. Práca demonštrovala postup pri riešení NP problémov za použitia DNA na konkrétnom príklade experimentu Leonarda Adlemana a stanovila formalizmy nevyhnutné pre zavedenie DNA simulácií do prostredia číslicových počítačov. Ďalej bol v práci popísaný program, vytvorený na simuláciu DNA počítania v prostredí číslicových počítačov. Práca detailne vysvetlila jeho kľúčové a zaujímavé prvky. Na koniec boli v práci uvedené experimenty vykonané s pomocou vytvoreného programu.

Vytvorený program je jednoduchý na ovládanie a jeho spätná väzba umožňuje užívateľovi dôsledne sledovať jednotlivé simulačné kroky výpočtu. Flexibilita programu ponúka užívateľovi možnosť simulovať priebeh DNA výpočtov nad rôznymi grafmi, s rôznymi vstupnými parametrami. Prehľadnosť programu slúži ako výborný základ pre nasadenie do výučby. Nevýhodou vytvoreného programu sú extrémne nároky na kapacitu operačnej pamäte, prípadne dlhé trvanie simulácie na menej výkonnom procesore. Toto obmedzenie môže brániť simuláciám pri úspešnom generovaní hamiltonovských ciest v mnohovrcholových grafoch.

Program aj táto práca vznikli za účelom nasadenia vo výučbe pre lepšie pochopenie konceptu DNA počítania a operácií, ktoré sú k tomu vyžadované. Práca by sa mohla uplatniť ako výućbová pomôcka v magisterskom kurze Biológiou inšpirovaných počítačov. Možné rozšírenie práce zahŕňa implementáciu simulácie SAT problému, ktorý je ďalším z množiny NP problémov, na ktorom bol uplatnený koncept DNA počítania.

Literatúra

[1] Adleman, L.: Molecular computation of solutions to combinatorial problems. *Science*, ročník 266, č. 5187, 1994-11-11: s. 1021–1024, doi:10.1126/science.7973651.

[2] Amos, M.: *Theoretical and Experimental DNA Computation*. Berlin: Springer, 2005, ISBN 35-406-5773-8, 172 s.

[3] Baskiyar, S.: Simulating DNA Computing. *High Performance Computing – HiPC 2002*, 2002-12-18: s. 411–419, doi:10.1007/3-540-36265-7˙39.

[4] Clay Mathematics Institute: P vs NP Problem. `http://www.claymath.org/millenium-problems/p-vs-np-problem`.

[5] Oracle: The Java®Language Specification. `http://docs.oracle.com/javase/specs/jls/se7/html/index.html`.

[6] Oracle: Trail: Creating a GUI With JFC/Swing. `http://docs.oracle.com/javase/tutorial/uiswing/index.html`.

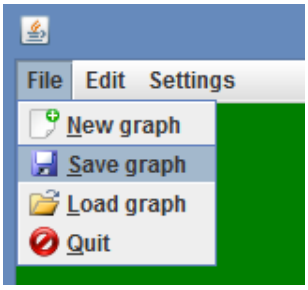
[7] Promega: Ligases. `https://www.promega.com/~/media/files/resources/product%20guides/cloning%20enzymes/ligases.pdf?la=en+`.

[8] Wikipedia, the free encyclopedia: An overview of the structure of DNA. `http://upload.wikimedia.org/wikipedia/commons/e/e4/DNA_Overview.png`.

Dodatok A

Manuál k vytvorenému programu

Po spustení programu sa od užívateľa očakáva vytvorenie grafu, nad ktorým prebehne simulácia. Graf môže užívateľ vytvoriť ručne alebo načítať za súboru. Kliknutím ľavým tlačidlom myši užívateľ pridá vrchol do grafu na aktuálnu pozíciu kurzoru. Opätovným kliknutím na vytvorený vrchol ho užívateľ zmaže. Metódou „drag-and-drop“ je do grafu pridaná nová hrana, ak je vedená medzi dvomi vrcholmi. Pri správnom pridávaní hrany, program napovedá užívateľovi sivou líniou začínajúcou v počiatočnom vrchole hrany a sledujúcou kurzor myši. Rovnakým spôsobom je možné hranu z grafu odstrániť. Pri odstraňovaní vrcholu, z/do ktorého vedú nejaké hrany, sú tieto hrany automaticky zmazané. Vytvorený graf si užívateľ môže uložiť do súboru pre opätovné použitie (obrázok A.1). Keďže program využíva generátor pseudonáhodných čísiel, uloženie prvotnej hodnoty generátoru umožňuje opakovaním simulácie dospieť k rovnakému výsledku.

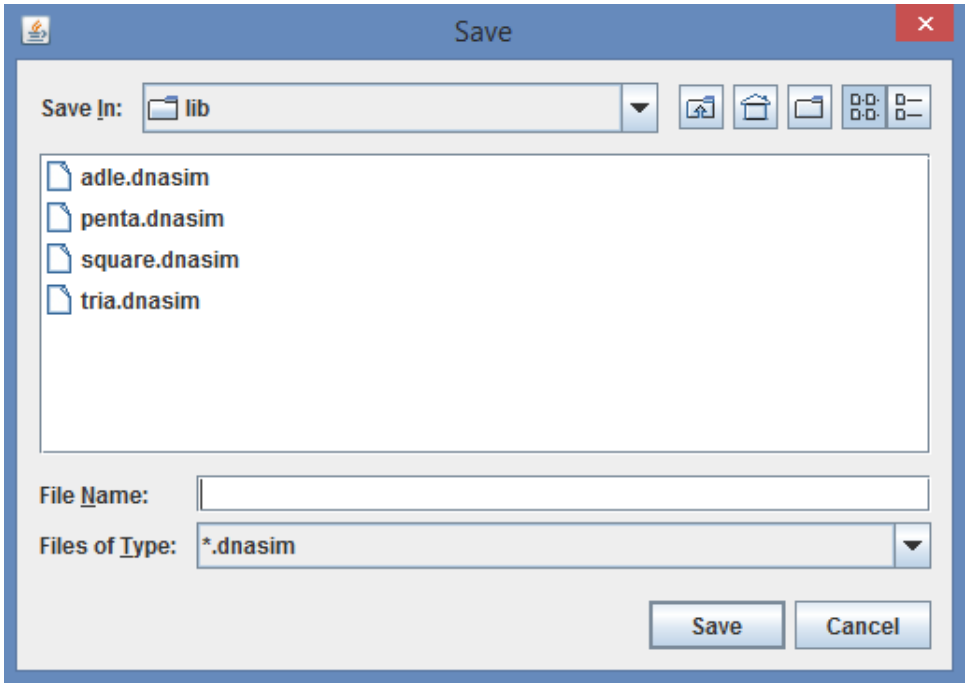


Obrázok A.1: Uloženie grafu do súboru pomocou menu programu

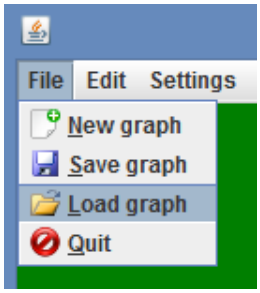
Po voľbe uloženia grafu sa užívateľovi zobrazí navigátor súborovým systémom, do ktorého vyplní názov ukladneného grafu, prípadne špecifikuje adresár na uloženie súboru (obrázok A.2).

Ďalšou možnosťou vytvorenia grafu je načítanie grafu zo súboru (obrázok A.3). Užívateľ má možnosť vybrať si z predpripravených súborov, prípadne sa dostať k svojim vlastným obdobným spôsobom ako pri ukladaní grafu do súboru. Program ukladá svoje súbory s príponou `.dnasim`.

Obsah súboru s uloženým grafom ukazuje obrázok A.4. V tomto súbore sa nachádzajú dôležité informácie z hľadiska simulácie. Súbor je znakom mriežky rozdelený na tri celky. V hornom celku sú uvedené súradnice vrcholov grafu. Každý riadok nasleduje formu: x-ová súradnica vrcholu, medzera, y-ová súradnica vrcholu, koniec riadku. Stredný celok obsahuje informácie o hranách grafu. Forma riadku stredného celku je nasledovná: x-ová



Obrázok A.2: Navigátor súborovým sytémom



Obrázok A.3: Načítanie grafu zo súboru pomocou menu programu

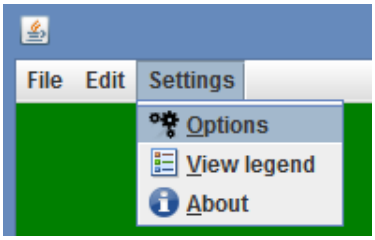
súradnica vrcholu, v ktorom hrana začína, medzera, y-ová súradnica vrcholu, v ktorom hrana začína, dvojbodka, x-ová súradnica vrcholu, v ktorom hrana končí, medzera, y-ová súradnica vrcholu, v ktorom hrana končí, koniec riadku. Posledný celok obsahuje doplňujúce informácie. Na prvom riadku tohto celku sa nachádza inicializačná hodnota pseudonáhodného generátoru čísel (seed) nasledovaná koncom riadku. Na druhom riadku je uvedená orientácia grafu. Keďže graf môže byť buď orientovaný, alebo neorientovaný, tento riadok nadobúda iba dve hodnoty, konkrétne jednotku ak je graf orientovaný, alebo nulu ak je neorientovaný. Ako každý riadok s nejakou informáciou v súbore, aj tento je ukončený znakom konca riadku. Obrázok A.4 bol vystavený nad súborom `square.dnasim`

Užívateľ má tiež možnosť nastaviť si niektoré parametre simulácie a pozmeniť vzhľad programu podľa seba. K nastaveniam sa dá pristúpiť cez položku `Settings` v hlavnom menu (obrázok A.5).

Po zobrazení okna s nastaveniami (obrázok A.6) je možné zmeniť orientáciu grafu, farbu

```
150 100
350 100
350 300
150 300
#
150 100:350 100
350 100:150 100
350 100:350 300
350 300:350 100
350 300:150 300
150 300:350 300
150 300:150 100
150 100:150 300
150 100:350 300
350 300:150 100
350 100:150 300
150 300:350 100
#
1393263385025
1
```

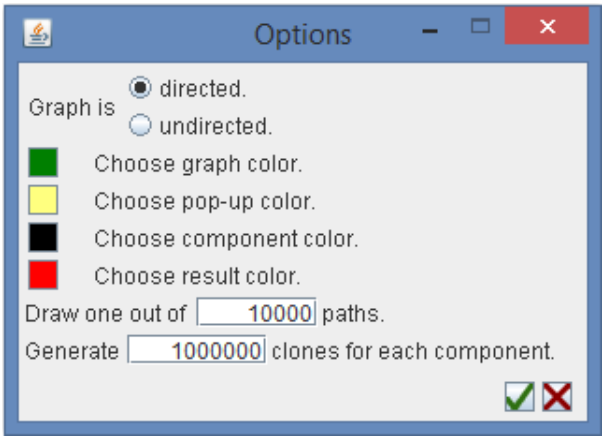
Obrázok A.4: Obsah uloženého súboru



Obrázok A.5: Nastavenia programu

pozadia grafu, farbu nápovedy k prvku grafu (uvedené neskôr), farbu vrcholov a hrán grafu a farbu prvkov grafu, ktoré budú súčasťou nájdenej Hamiltonovskej cesty grafom. Ďalej je možné nastaviť parametre simulácie, ktoré ovplyvňujú pomer v akom budú oligonukleotidy a dvojzávitnice vykreslované v jednotlivých krokoch simulácie, respektíve počet klonov každého prvku grafu. Počas priebehu simulácie nie je možné ďalej meniť orientáciu grafu a počet klonov, pretože také zásahy by mohli viesť k nesprávnosti jednotlivých simulačných krokov.

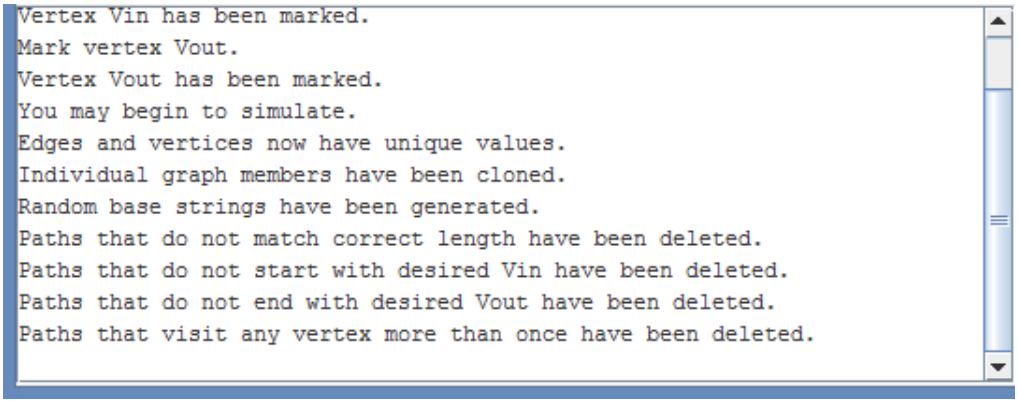
Keď je užívateľ spokojný s vytvoreným grafom a chcel by nad ním spustiť simuláciu, môže graf „dokončiť“, čo zamedzí ďalšej manipulácii s grafom. Dokončenie grafu je prístupné cez položku `Finnish graph` v časti `Edit` hlavného menu programu. O dokončení grafu je užívateľ informovaný v textovej oblasti programu výpisom `Graph finnished.`, a v zápätí je vyzvaný k určeniu počiatočného vrcholu grafu. Po kliknutí na vrchol grafu, ktorý užívateľ vyberie za počiatočný, je tento vrchol ďalej v simulácii sprevádzaný šípkou. V textovej oblasti sa opäť objaví informácia pre užívateľa, `Vertex Vin has been marked.`,



Obrázok A.6: Okno s nastaveniami

a užívateľ je vyzvaný k označeniu koncového vrcholu grafu. Po jeho označení je ďalej v simulácii tento vrchol zakrúžkovaný. Tentokrát užívateľ obdrží v textovej oblasti dve správy: `Vertex Vout has been marked.` a `You may begin to simulate.`.

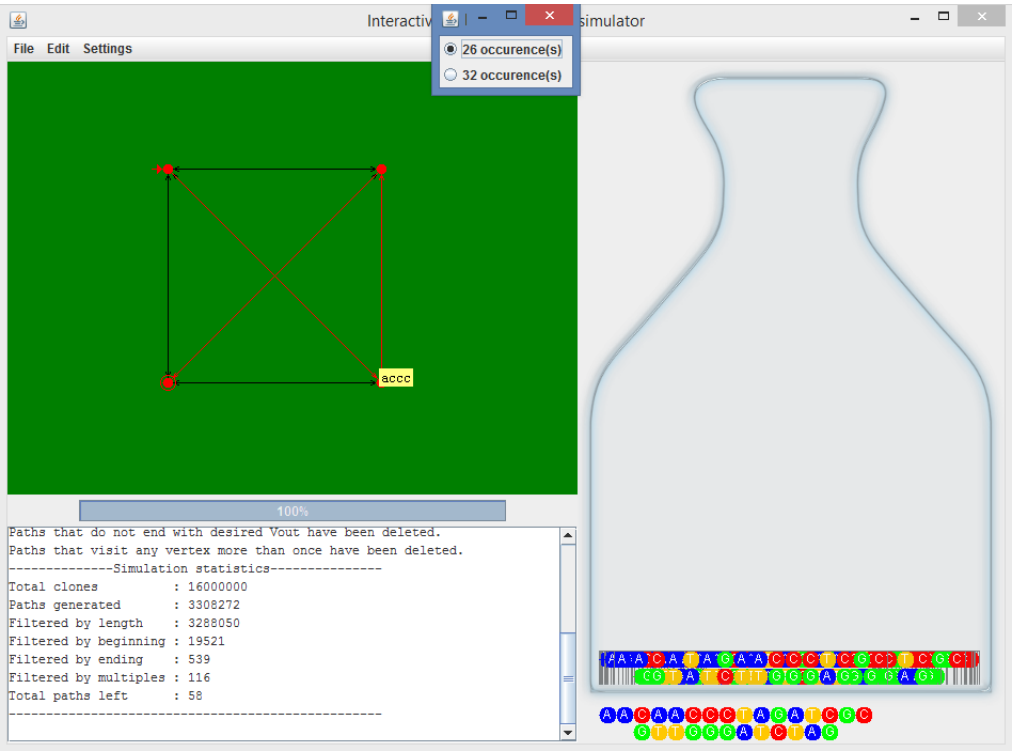
V tomto bode je simulácia pripravená na vykonanie prvého kroku, teda generovania klonov prvkov grafu. Spustenie simulácie je prístupné cez položku `[Begin|Continue] simulation` v časti `Edit` hlavného menu programu. Užívateľ znovu obdrží sprievodný výpis. Ďalšie kroky simulácie sú spúšťané rovnakým spôsobom. Výpis všetkých sprievodných správ simulácie ukazuje obrázok A.7.



Obrázok A.7: Sprievodné správy simulácie

Po dokončení simulácie si užívateľ môže zobraziť výsledky. Zobrazenie výsledkov otvorí nové okno, v ktorom sú nájdené Hamiltonovské cesty grafom uložené do zoznamu, a výberom hociktorej z nich sú prvky grafu, ktoré cesta obsahuje zvýraznené. Výsledky sú zobrazené položkou `Present results` v časti `Edit` hlavného menu programu. Do priestoru pod bankou je takisto vykreslený zjednodušený model dvojzávitnice reprezentujúci danú Hamiltonovskú cestu. Ak by si chcel užívateľ overiť koreláciu dvojzávitnice s jednotlivými prvkami grafu, môže využiť nápovedu k prvku grafu. Nápoveda obsahuje oligonukleotid, ktorým bol daný prvok vo výpočte reprezentovaný. Nápoveda sa zobrazí nad daným prvkom keď ho užívateľ označí (kliknutie ľavým tlačidlom myši na vrchol zobrazí nápovedu k da-

nému vrcholu, „drag-and-drop“ od jedného vrcholu k druhému zobrazí nápovedu k danej hrane za predpokladu, že taká hrana v grafe je). Užívateľ môže nápovedu skryť kliknutím niekam inam do priestoru grafu. Popri grafickej forme výsledku simulácie, užívateľ dostáva aj výsledok vo forme textu. V textovej oblasti programu sú zobrazené štatistiky simulácie. Všetky popísané prvky sú znázornené na obrázku A.8.



Obrázok A.8: Výsledky simulácie

Užívateľ má možnosť pripraviť a vykonať novú simuláciu prostredníctvom položky `Restart simulation` v časti `Edit` hlavného menu, čo spôsobí nastavenie programu na hodnoty tesne po spustení. Ďalšie položky časti `Settings` hlavného menu sú iba informatívneho charakteru. Každá položka hlavného menu môže byť zvolená klávesou, ktorá je v jej názve podčiarknutá. Všetky okná programu, okrem hlavného, je možné zavrieť klávesou `Escape`. Pre používanie programu je požadovaný 64-bit JVM a aplikácia ANT.