# Využitie Travisových čiastočných súm vo Wheelerových grafoch a nástroj Wheeler Graph Toolkit: Report za letný semester

Ondrej Škorňák

Jún 2024

## 1 Úvod

Tento semester som sa mal zamerať na štúdium a implementáciu metódy Travisových čiastočných súm (TPS) a jej využitie vo Wheelerových grafoch. Hlavným cieľom bolo pochopiť, ako TPS môže zlepšiť techniky kompresie dát, najmä v kontexte Burrows-Wheelerovej transformácie (BWT) a jej implementácie v blokových kompresoroch ako Bzip2. Pre tento semester sa upustilo od implementácie BWT do Bzip2 z dôvodu toho, že po finálnej konzultácii v zimnom semestri, nám prišlo zaujímavejšie skúsiť zlepšiť priamo BWT pomocou daných špeciálnych čiastočných súm.

## 2 Ciele projektu

Projekt mal dva hlavné ciele:

- Štúdium BWT a jeho využitie v kompresii dát.
- Implementácia tunelovacích metód v Bzip2 s cieľom vytvoriť plne funkčný kompresný algoritmus.

Počas letného semestra sa sústredím stále na prvý cieľ, so špecifickým zameraním na pochopenie TPS a ich aplikácie v BWT ako aj možné využitie aj v iných prípadoch.

## 3 Travisové čiastkové sumy

Travisové čiastkové sumy (TPS) sa využívajú v komprimovaných dátových štruktúrach a indexovaní na zlepšenie efektivity kompresie dát a rýchlejšie spracovania dotazov. TPS sú nevyhnutné pre:

- **Kompresia dát**: Zlepšenie kompresných algoritmov využitím dátových štruktúr na minimalizáciu redundancie a ich zrýchlenie.
- **Analýza sekvencií**: Efektívne reprezentovanie dátových štruktúr, ako sú de Bruijnové grafy, na spracovanie veľkých genomických dát.
- **Všeobecné dátové štruktúry**: Zlepšenie dátových štruktúr, ako sú tries a suffix trees, pre rýchlejšie spracovanie dotazov a zníženie požiadaviek na úložný priestor.

# 4 Využitie Travisových čiastočných súm vo Wheelerových grafoch

TPS sú obzvlášť užitočné vo Wheelerových grafoch pre:

- **Reprezentácia de Bruijnových grafov**: Efektívne ukladanie a dotazovanie sekvencií stupňov, čo je kľúčové pre úlohy ako skladanie genómov.
- **Dotazy na stupne**: Rýchle dotazy na in-degrees a out-degrees vrcholov.
- **Kompaktná reprezentácia grafov**: Kompaktné reprezentovanie grafových štruktúr a zlepšenie efektivity dotazov.
- **Praktické aplikácie**: Optimalizácia procesov skladania genómov a efektívne spracovanie veľkých dát.

# 5 Popis algoritmu pre konštrukciu indexu a spracovanie dotazov

## 5.1 Konštrukcia indexu

Konštrukcia indexu zahŕňa vytvorenie štyroch hlavných dátových štruktúr:

### 5.1.1 Dotazy na súčty out-stupňov ($D_{out}$)

- **Inicializácia sekvencie**: Začneme so sekvenciou out-stupňov $D_{out}$ pre všetky vrcholy vo Wheelerovom grafe.
- **Komprimácia sekvencie**: Použijeme kompresie založenej na empirickej entropii pre úsporu miesta.
- **Úprava sekvencie**: Vylepšenie komprimovanej sekvencie pomocnými štruktúrami na uľahčenie dotazov na súčty v konštantnom čase.

### 5.1.2 Dotazy na rank označení hrán ($L$)

- **Zoradenie označení hrán**: Vytvoríme zoznam $L$ označení hrán zoradených podľa ich východiskových bodov.
- **Vytvorenie štruktúry na podporu rank**: Použijeme dátovu štruktúru ako wavelet tree pre efektívne rank dotazy.
- **Pomocné informácie**: Uloženie pomocných polí alebo bitových vektorou na zrýchlenie výpočtov rank.

### 5.1.3 Dotazy na súčty frekvencií označení hrán ($C$)

- **Výpočet frekvencií**: Určíme frekvencie každého označenia hrany.
- **Kumulatívna suma**: Vytvorenie kumulatívnej sumy $C$, kde každá položka $C[a]$ predstavuje súčet frekvencií označení hrán lexikograficky menších ako $a$.
- **Efektívne uloženie**: Použitie kompresnej techniky na efektívne uloženie $C$.

#### 5.1.4 Dotazy na vyhľadávanie in-stupňov ($D_{in}$)

- **Inicializácia sekvencie**: Začneme so sekvenciou in-stupňov $D_{in}$ pre všetky vrcholy.
- **Komprimácia sekvencie**: Aplikujeme kompresiu založenú na empirickej entropii na $D_{in}$.
- **Augmentácia sekvencie**: Vylepšime $D_{in}$ štruktúrami, ktoré podporujú rýchle vyhľadávacie operácie.

### 5.2 Spracovanie dotazov

Pomocou skonštruovaného indexu nasledujúce kroky opisujú, ako efektívne spracovať rôzne typy dotazov:

#### 5.2.1 Dotazy na súčty out-stupňov

- **Vstup**: Index $i$.
- **Operácia**: Získajme $D_{out}.sum(i)$ pomocou upravenej dátovej štruktúry.
- **Výstup**: Súčet out-stupňov až po $i$-ty vrchol.

#### 5.2.2 Dotazy na rank označení hrán

- **Vstup**: Označenie hrany $a$ a index $i$.
- **Operácia**: Vypočítajme $L.rank_a(i)$ pomocou dátovej štruktúry podporujúcej rank.
- **Výstup**: Rank $a$ medzi prvými $i$ označeniami hrán.

#### 5.2.3 Dotazy na súčty frekvencií označení hrán

- **Vstup**: Označenie hrany $a$.
- **Operácia**: Prístup ku kumulatívnej sume $C$ na získanie $C.sum(a)$.
- **Výstup**: Súčet frekvencií označení hrán menších ako $a$.

#### 5.2.4 Dotazy na vyhľadávanie in-stupňov

- **Vstup**: Hodnota $j$.
- **Operácia**: Vykonajme binárne vyhľadávanie alebo priame vyhľadávanie v augmentovanom $D_{in}$ na nájdenie najväčšieho indexu $i$, kde $D_{in}.search(j) \leq j$.
- **Výstup**: Index $i$.

## 6 Nástroj Wheeler Graph Toolkit (WGT)

Nástroj Wheeler Graph Toolkit (WGT) je open-source balík (MIT licencia) určený na generovanie, rozpoznávanie a vizualizáciu Wheelerových grafov. Centrálnou súčasťou WGT je "Wheelie", rýchly algoritmus na rozpoznávanie Wheelerových grafov. Tu je prehľad príkazov a ich funkcií.

## 6.1 Moduly vo WGT

- **Recognizer**: Určuje, či je daný graf Wheelerovým grafom.
- **Visualizer**: Vizualizuje Wheelerové grafy v bipartitnej reprezentácii.
- **Generators**: Generuje rôzne typy grafov, ako sú tries, de Bruijnové grafy a reverzné deterministické grafy.

## 6.2 Generátory grafov

### 6.2.1 Generátor de Bruijnových grafov

```
Usage:
    DeBruijnGraph_generator.py
        [-h / --help] [-v/ --version] [-o / --ofile FILE]
        [-k / --kmer k-mer length] [-l / --seqLen sequence length]
        [-a / --alnNum alignment number] sequence FATA file
Optional arguments:
    -h / --help : Print the usage of "DeBruijnGraph_generator.py".
    -v / --version : Print the version of the program.
    -o / --ofile : The name of the output file. Default is "tmp.dot".
    -k / --kmer : The k-mer length. Default is "3".
    -l / --seqLen : Maximum sequence length for each entry in the FASTA file.
       Default is the full sequence length.
    -a / --alnNum : The number of alignments (sequences) used for creating the
       De Bruijn graph. Default is "3".
```

### 6.2.2 Generátor reverzných deterministických grafov

```
Usage:
    RevDetGraph_generator.py [-h / --help] [-v/ --version] [-o / --ofile FILE]
                            [-l / --seqLen sequence length] [-a / --alnNum
                               alignment number]
                            multiple sequence alignments (MSA) FASTA file
Optional arguments:
    -h / --help : Print the usage of "RevDetGraph_generator.py".
    -v / --version : Print the version of the program.
    -o / --ofile : The name of the output file. Default is "tmp.dot".
    -l / --seqLen : Maximum sequence length for each entry in the FASTA file.
       Default is the full sequence length.
    -a / --alnNum : The number of alignments (sequences) used for creating the
       De Bruijn graph. Default is "3".
```

### 6.2.3 Generátor tries

```
Usage:
    Trie_generator.py [-h / --help] [-v/ --version] [-o / --ofile FILE]
                     [-l / --seqLen sequence length] [-a / --alnNum alignment
                        number]
                     multiple sequence alignments (MSA) FASTA file
```

```
Optional arguments:
    -h / --help : Print the usage of "Trie_generator.py".
    -v / --version : Print the version of the program.
    -o / --ofile : The name of the output file. Default is "tmp.dot".
    -l / --seqLen : Maximum sequence length for each entry in the FASTA file.
       Default is the full sequence length.
    -a / --alnNum : The number of alignments (sequences) used for creating the
       De Bruijn graph. Default is "3".
```

## 6.3 WGT Recognizer

```
Usage:
    recognizer <in.dot> [--version] [-h / --help] [-v / --verbose] [-o / --
        outDir]
              [-s / --solver <smt / p>] [-w / --writeIOL] [-r / --writeRange]
              [-i / --label_is_int] [-b / --benchmark] [-e / --
                 exhaustive_search]
              [-f / --full_range_search]
Options:
   version : Print the current version (0.1.0).
   help : Print the usage message.
   verbose : Run in the verbose mode.
   outDir : The directory for the files to be outputted.
   solver : Specify which solver to use (‘SMT‘ or ‘p‘). Default is ‘SMT‘.
   writeIOL : Output the I, O, L three-bitarray data structure.
   writeRange : Output the range information produced by the renaming heuristic
       .
   label_is_int : Specify if the edge labels are integers. Default is strings.
   benchmark : Run in benchmark mode, outputting results in columns.
   exhaustive_search : Run in exhaustive search mode.
   full_range_search : Run in full range search mode, skipping the renaming
       heuristic.
```

## 6.4 WGT Visualizer

```
Usage:
    python3 visualizer.py [-h] [-o / --ofile output file] <Wheeler graph in DOT
         format>
Example:
    python3 visualizer.py ../data/example/out__example/graph.dot
```

# 7 Relevancia pre môj projekt

Štúdium a implementácia TPS sú kľúčové pre ciele môjho projektu, najmä pri zlepšovaní kompresnej efektivity algoritmu Bzip2. Pochopením a aplikáciou TPS sa snažím dosiahnuť:

- Zlepšenie kompresného pomeru Bzip2 prostredníctvom efektívneho spracovania sekvencií stupňov vo Wheelerových grafoch.

- Implementácia a optimalizácia tunelovacích metód na zníženie redundancie dát a zlepšenie celkového výkonu kompresie.
- Využitie TPS na rýchle a efektívne spracovanie dotazov, čo je nevyhnutné pre spracovanie veľkých dát v praktických aplikáciách.

Navyše, znalosti a nástroje z WGT mi poskytli praktické skúsenosti s prácou s Wheelerovými grafmi, rozpoznávaním ich vlastností a vizualizáciou ich štruktúry, čo je nevyhnutné pre implementáciu efektívnych kompresných techník.

# 8 Záver

Metodológia Travisových čiastočných súm poskytuje významné výhody v kontexte kompresie dát a analýzy sekvencií, najmä pri aplikácii vo Wheelerových grafoch. Zavedením TPS do môjho projektu sa budem snažíť dosiahnuť lepšie kompresné pomery a efektívnejšie spracovanie dát, čo prispieva k pokroku v kompresných algoritmoch ako Bzip2. Nástroj Wheeler Graph Toolkit ďalej zlepšil moje chápanie a schopnosti pri manipulácii a vizualizácii týchto komplexných grafových štruktúr.

# 9 Zdroje

Pri štúdiu som použil nasledujúce zdroje:


- https://oparu.uni-ulm.de/server/api/core/bitstreams/c881cc5a-80c3-40bc-a44c-4a80
content
- https://www.ncbi.nlm.nih.gov/tools/sviewer/seqtrackdata/
- https://ftp.ensembl.org/pub/release-112/fasta/
- https://github.com/Kuanhao-Chao/Wheeler_Graph_Toolkit
- https://www.sciencedirect.com/science/article/pii/S0304397517305285
- https://arxiv.org/pdf/2204.07916
- https://sourceware.org/bzip2/
- https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Mitarbeiter/baier/slides_tunneling.pdf
- https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.190/Intern/medien/slides_dcc_19.pdf
- https://github.com/waYne1337/tbwt
- https://www.youtube.com/playlist?list=UUrDmN9uRVJR7KM8aRE_58Zw