Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Science
and Technology

Projekt dyplomowy
Implementacja drzew klasy trie

Jacek Oleś[1]

[1]Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej

ul. Mickiewicza 30
30-059 Kraków
Polska

28.02.2020

Cele pracy i ich motywacje

1. Cele:
   - Analiza algorytmów dotyczących drzewa *Trie* (tworzenie, przeszukiwanie)
   - Implementacja algorytmów dotyczących wybranego rodzaju drzewa (*Patricia*) w języku *Java*
2. Motywacja celów:
   - Rozjaśnienie i uściślenie pojęć związanych z pojęciem *Trie* (oraz *Patricia*), gdyż:
     - *Trie* to często pojawiające się pojęcie w literaturze oraz innych źródłach.
     - Wielokrotnie zostaje przedstawione powierzchniowo, pobieżnie.

Hipoteza pracy i motywacja za nią

1. Hipoteza
   - Jest możliwym modyfikacja 3 algorytmów Knuth'a - dotyczące drzewa *Patricia* - tak, aby:
     - Klucze mogły przyjmować postać pojedynczego słowa;
     - Pozwalając (jednocześnie) użytkownikowi na wybór strategii definiującej postać klucza.
2. Motywacja hipotezy
   - Chęć ujednolicenia pojęcia klucza;
   - Gdyż przyjmował on wyróżniającą się postać na tle pozostałych algorytmów zaproponowanych przez Knuth'a.

```
1 "THIS IS THE HOUSE THAT JACK BUILT;"
```

Plik źródłowy drzewa Patricia

```
1 "THIS IS THE HOUSE THAT JACK BUILT;"
2 "IS THE HOUSE THAT JACK BUILT;"
3 "THE HOUSE THAT JACK BUILT;"
4 "HOUSE THAT JACK BUILT;"
5 "THAT JACK BUILT;"
6 "JACK BUILT;"
7 "BUILT;"
```

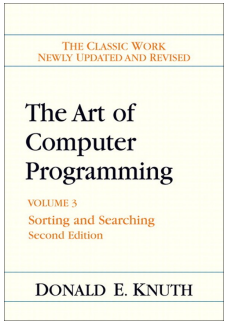
Postać klucza według algorytmów Knuth'a
Strategia 1: Start Position To End Of File

```
1 "THIS "
2 "IS "
3 "THE "
4 "HOUSE "
5 "THAT "
6 "JACK "
7 "BUILT;"
```

Klucz postaci pojedynczego słowa
Strategia 2: Single Word

Opracowanie teoretyczne

1. Opracowanie teoretyczne zawiera przedstawienie:
   1. historii terminu *Trie*,
   2. abstraktu struktury drzewa *Trie*,
   3. 10 operacji możliwych do wykonania na drzewie *Trie*,
   4. 5 wariacji struktury drzewa *Trie*,
   5. zagadnienia koniunkcyjnej postaci normalnej.
2. Wykorzystując:
   1. książkę „Sztuka programowania“ Donalda E. Knuth'a;
      Cały rozdział „Przeszukiwanie cyfrowe“ oraz ćwiczenia z nim powiązane;
   2. oraz dodatkowych 30 źródeł.



Drzewa *Trie*

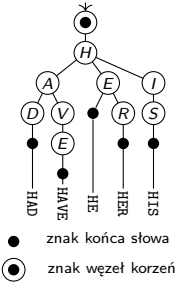
Przedstawienie zagadnień

Drzewo *Trie* - drzewo poszukiwań przechowujące w węzłach fragmenty kluczy

Drzewo stopnia M (M-*ary* lub M-*way*)

Drzewo, którego węzły posiadają co najwyżej M dzieci;

1. Każdy węzeł może być reprezentowany w postaci M-elementowych wektorów.
2. Komórki wektorów przyporządkowane są do poszczególnych M znaków, M elementowego alfabetu.
3. Każda z komórek (znak jej przyporządkowany) pozwala na przejście gałęzią z węzła na węzeł na następnym poziomie.

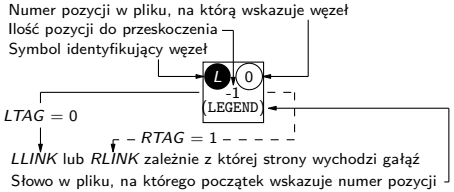


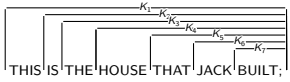
Drzewa *Patricia*

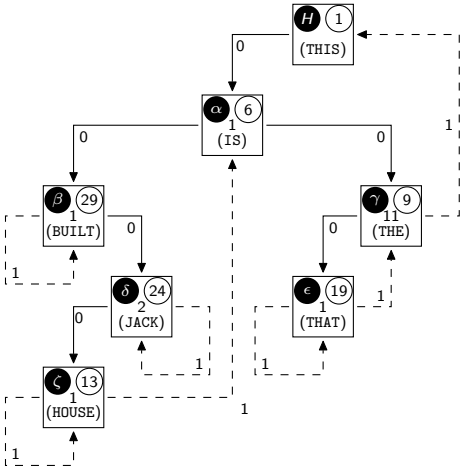
Przedstawienie zagadnień

Drzewo *Patricia* to wariacja drzewa *Trie*.

- Przechowuje klucze w postaci zawartości pliku.
- Sama struktura przechowuje odniesienia do pozycji w pliku.
- Wstępnie porównuje tylko bity rozróżniające klucze zawarte w drzewie.
- Na końcu porównuje zgodność wszystkich bitów wstępnie dopasowanego klucza z bitami argumentu przeszukiwania.







Drzewo *Patricia*

Implementacja na podstawie algorytmów Knuth'a

Implementacja mojego programu w języku *Java* zawiera:

1. Implementację funkcjonalności drzewa *Patricia*:
   1. Operacje:
      1. Wstawienie nowego **KLUCZA** (z istniejącego pliku źródłowego).
         - Insert **KEY** (From Existing File)
      2. Przeszukiwania względem **PREFIKSÓW**:
         1. Czy drzewo zawiera słowo *X* jako **PREFIKS**?
            - Look-up **PREFIX**
         2. Które węzły akceptują słowo *X* jako **PREFIKS**?
            - Search **PREFIX**

Na podstawie 3 algorytmów przeznaczonych na Maszynę MIX Knuth'a, której bajt ma 5 bitów

2. Symulacje Maszyny MIX
   1. o parametryzowanej ilości bitów w bajcie
      (od 5 wzwyż, zwykle bajt maszyny MIX ma 6 bitów)
   2. oraz tablicy kodowania znaków
      (aby zapewnić obsługę wymaganych znaków w odpowiednim zakresie liczbowym).

Drzewo *Patricia*

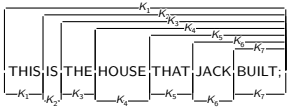
Autorskie rozszerzenie na podstawie wcześniejszej implementacji

Implementacja dodatkowych funkcjonalności autorskich drzewa *Patrcia*

1. Strategia klucza (2) definiująca go w postaci pojedynczego słowa - rozłącznych kluczy
2. Operacje:
   1. Wstawienie wszystkich **KLUCZY** (z istniejącego pliku źródłowego)
      - Insert All **KEYS** (From Existing File)
   2. Przeszukiwania względem **KLUCZY**:
      1. Czy drzewo zawiera słowo *X* jako **KLUCZ**?
         - Look-up **KEY**
      2. Który węzeł akceptuje słowo *X* jako **KLUCZ**?
         - Search **KEY**

Na podstawie wcześniej zaimplementowanych funkcjonalności

Implementacja klucza zgodnie z wymaganiami algorytmów Knuth'a
Obiekt klasy: *new FileOps(..., WordStrategy.START_POSITION_TO_EOF, ...);*
Pole klasy: *WordStartPositionToEOFStrategy extends FileOpsStrategy*



Implementacja autorska klucza postaci pojedynczego słowa
Obiekt klasy: *new FileOps(..., WordStrategy.SINGLE, ...);*
Pole klasy: *WordSingleStrategy extends FileOpsStrategy*

Konwerter DIMACS CNF

Autorska implementacja konwertera na pliki źródłowe implementacji drzewa *Patricia*

CNF – koniunkcyjna postać normalna – formuła logiczna postaci koniunkcji klauzul.

DIMACS CNF – format zapisu plików CNF.

Konwerter sortuje literały wewnątrz każdej z klauzul oraz zapisuje je w nowym pliku.

Nowy plik jest parametryzowany względem znaków rozdzielających literały, klauzule, koniec pliku.

Implementacja konwertera plików DIMACS CNF na pliki źródłowe drzewa *Patricia*.

Bez linii tekstu niezawierających kluczowych informacji

- O klauzulach w koniunkcyjnej postaci normalnej - (linii komentarzy i problemu),

O posortowanych literałach wewnątrz każdej z klauzul,

Ze zmodyfikowanymi parametryzowanymi znakami rozdzielającymi literały wewnątrz klauzul oraz same klauzule

i o dodanym parametryzowanym znaku końca pliku.

```
/***
// [...]
***/
v7272 ∧
v30549 ∧
v7270 ∧
v30550 ∧
v30551 ∧
v30552 ∧
v7314 ∧
v30553 ∧
v7268 ∧
v7267 ∧
// [...]
(¬v31503 ∨ ¬v442 ∨ ¬v24184)∧
(v31503 ∨ ¬v442 ∨ v24184)∧
(v31503 ∨ v442 ∨ ¬v24184)
// [...]
```

CNF

```
c 100 20
// [...]
p cnf 31503 94748
7272 0
30549 0
7270 0
30550 0
30551 0
30552 0
7314 0
30553 0
7268 0
7267 0
// [...]
-31503 -442 -24184 0
31503 -442 24184 0
31503 442 -24184 0
// NL at the end of file
```

DIMACS CNF

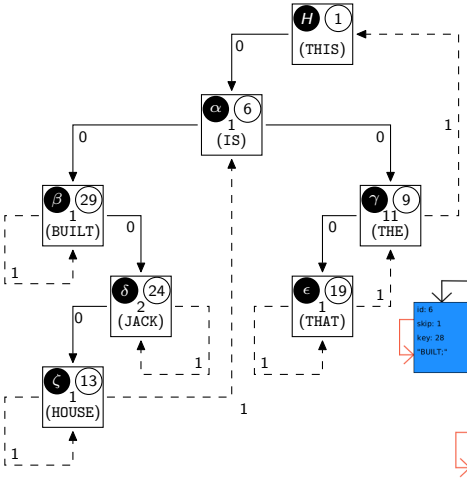
```
7272_0|30549_0|7270_0|30550_0|30551_0|30552_0|7314_0|30553_0|7268_0|7267_0|/***
    [...]
***/-31503_-24184_-442_0|-442_24184_31503_0|-24184_442_31503_0|;
```

Plik źródłowy drzewa *Patricia*

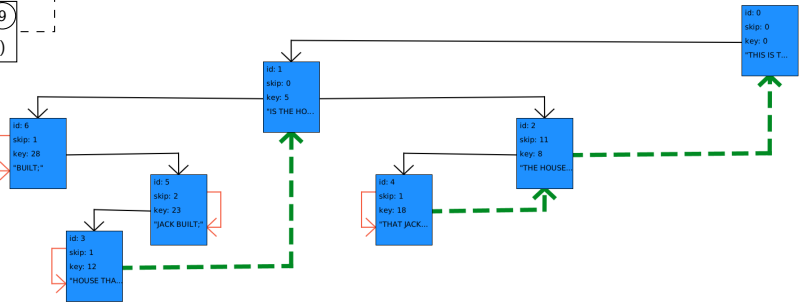
Reprezentacja graficzna 1.1

Autorska implementacja samo-skalującej się reprezentacji struktury drzewa *Patricia*

Rysunek: Model



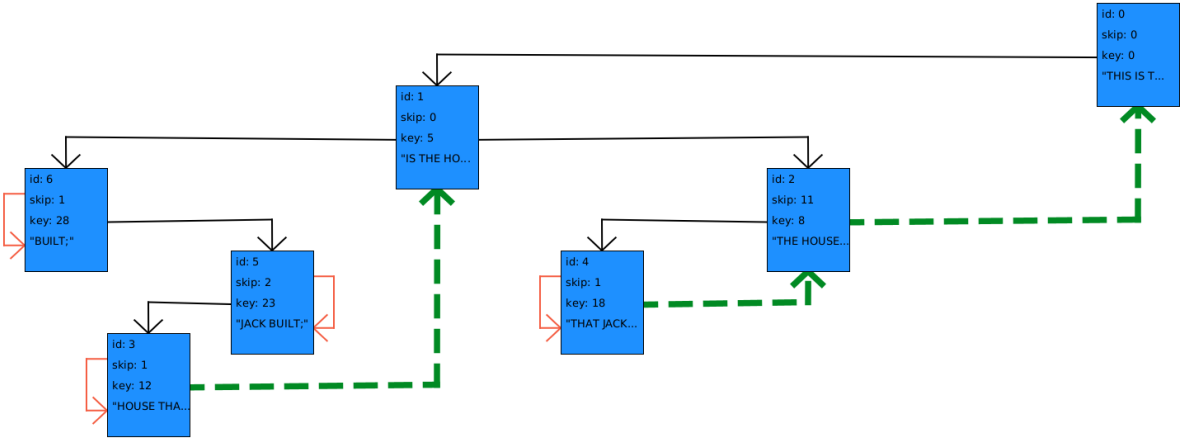
Rysunek: Reprezentacja 1



Reprezentacja graficzna 1.2

Autorska implementacja samo-skalującej się reprezentacji struktury drzewa *Patricia*

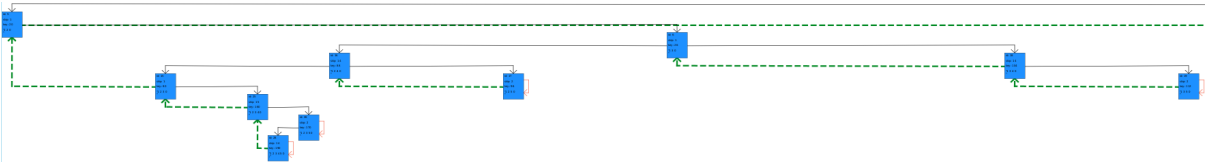
Rysunek: Powiększona reprezentacja 1



Reprezentacja graficzna 2

Autorska implementacja samo-skalującej się reprezentacji struktury drzewa *Patricia*

Rysunek: Reprezentacja 2 Powiększona



Rysunek: Reprezentacja 2 Pomniejszona

Testy

Testy jednostkowe i manualne gwarantujące poprawność implementacji

1. 570 złożonych testów sprawdzających funkcjonalność klas *PatriciaTree* (2 strategii definiujących formę kluczy) i *CNFConverter* i klas przez nie wykorzystywanych
2. Testy manualne 16 przykładów dostępnych do uruchomienia w programie w trybie tekstowym (9) oraz graficznym (7)

Uruchamianie programu

Plik wykonywalny, uruchomienie programu oraz argumenty wywołania

1. Uruchamianie programu w trybie wizualnym:

```
java -jar target/TrieTreeImplementations_complete_standalone.jar Visual 0 false
```

   Uruchamianie programu w trybie tekstowym:

```
java -jar target/TrieTreeImplementations_complete_standalone.jar Text 9 true
```

2. Numery przykładów: 0 - 9
3. Trzeci parametr (true lub false):

TRUE Konwertować plik CNF na źródłowy drzewa czy pominąć
FALSE Pominąć krok konwertowania pliku CNF

Podsumowanie

Odniesienie do hipotezy i celów oraz możliwości rozwoju projektu

Temat został zrealizowany w zakresie wyznaczonych założeń.

1. Cele:
   - Analiza algorytmów drzewa *Trie* ✓ Wprowadzenie teoretyczne części pisemnej projektu inżynierskiego
   - Implementacja drzewa *Patricia* ✓ Strategia klucza według Knuth'a
2. Hipoteza:
   - Modyfikacja algorytmów drzewa *Patricia* ✓ Strategia pojedynczego słowa
3. Dodatkowo:
   - Reprezentacja graficzna struktury drzewa *Patricia* ✓
   - Próba rozwiązania problemu (DIMACS) CNF ✓

Możliwe opcje dalszego rozwoju:

1. Zastosowanie implementacji drzewa *Patricia* do problemów dużych zbiorów
   (których elementy mogą być grupowane ze względu na początkowe bity reprezentacji).
2. Rozwój samej implementacji drzewa *Patricia* - implementacja pozostałych operacji
   (dodaj nowe słowo do pliku, usuń klucz ze struktury, usuń słowo z pliku, następca, poprzednik, minimum, maximum).
3. Zastosowania innego rozwiązania do praktycznego problemu DIMACS CNF
   (jako, że kolejność literałów wewnątrz klauzuli nie ma znaczenia).
   - Wykorzystanie *hash*-funkcji.
4. Rozwój reprezentacji graficznej drzewa *Patricia*, aby mogła wyświetlać drzewa o dowolnym rozmiarze.
   - Ograniczenie możliwości oddalenia obrazu oraz dynamiczne wczytywanie i usuwanie obecnie wyświetlanych węzłów.