Programowanie Rozproszone

Mateusz Bednarski

117194 grupa L3

Approximate string search

# Wstęp

Zadanie polegało na implementacji algorytmu przybliżonego porównywania stringów, mogącego służyć np. do sprawdzania pisowni.

# Słownik

Użyty słownik pochodzi ze strony sjp.pl i zawiera 2 721 111 polskich słów.

# Algorytm

Do oceny podobieństwa ciągów została wykorzystana odległość Levenshteina. W skrócie jest to minimalna liczba operacji elementarnych (wstawienie, usunięcie, zamiana pojedynczego znaku) za pomocą których można przekształcić jeden ciąg w drugi. Algorytm iteruje po słowniku i oblicza odległość każdego słowa, po czym wybiera te z minimalną. Złożoność wynosi O(m2n) gdzie m to długość słowa a n to wielkość słownika.

# Implementacja rozproszona

Algorytm zrównoleglenia jest następujący:

Master jest procesem o id = 0, pozostałe są slave.

Master dzieli słownik na n-1 części gdzie n jest liczbą procesów. Każdy slave dostaje część słownika, offset jego części w słowniku głównym oraz słowo do wyszukania (wiadomość REQUEST). Slave wykonuje algorytm sekwencyjny, po czym odsyła wynik do mastera (wiadomość RESPONSE). Master czeka na odpowiedź od wszystkich slave, po czym z otrzymanych wyników wybiera najlepszy.

Struktura wiadomości REQUEST:  
Offset : int – offset pierwszego słowa we fragmencie słownika względem całego słownika  
Term: string – słowo do wyszukania  
Dictionary: list of string - fragment słownika w który dany proces ma przeszukiwać.

Struktura wiadomości RESPONSE  
Word : string – znalezione słowo  
Index : int – indeks znalezionego słowa   
Distance : odległość znalezionego słowa od zadanego

C:\Users\bednarsm\Downloads\Untitled Diagram.png

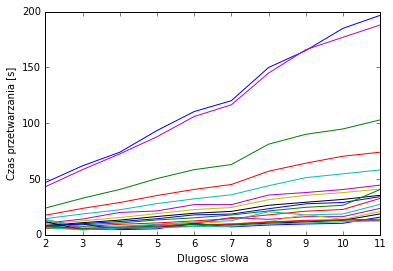
## Przykładowe wyniki

|  |  |
| --- | --- |
| xd | ad |
| fyć | być |
| miud | miód |
| kułka | bułka |
| wziąść | wziąć |
| wzglond | wzgląd |
| otfieracz | otwieracz |
| kąputerowy | komputerowy |
| wyrafnowany | wyrafinowany |
| metodologjia | metodologia |
| sgffihrgyuuiu | sapfirynie |
| nieposkromoiny | nieposkromiony |
| wykwaliifkowaną | wykwalifikowaną |

# Czasy wykonania

Uwaga, jako że w trakcie wykonywania pomiarów komputery w pracowniach wykonywały również inne zadania, wyniki mogą być lekko przekłamane

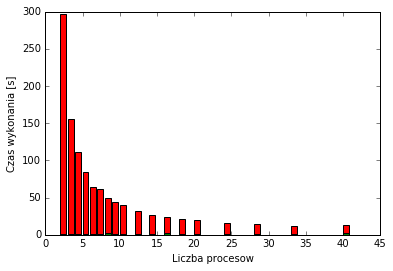
## Implementacja sekwencyjna vs rozproszona



Linia niebieska to algorytm sekwencyjny. Fioletowa – rozproszony z jednym procesem master i jednym slave.

Zgodnie z oczekiwaniami wersja sekwencyjna oraz 1+1 dają praktycznie identyczne wyniki gdyż w tym przypadku pojedynczy slave zachowuje się jak program sekwencyjny. Dodawanie kolejnych procesów zmniejsza czas wykonania.

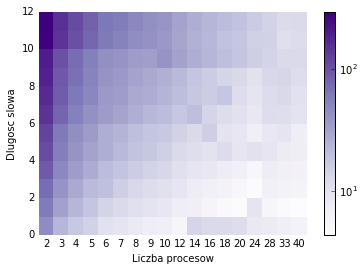
## Wpływ ilości procesów na czas przetwarzania



Wraz ze wzrostem liczby procesów spada czas wykonania. Teoretycznie czas powinien być odwrotnie proporcjonalny do liczby procesów (każdy proces ma do wykonania n/p pracy) co znajduje odzwierciedlenie w praktyce.

## Czas wykonania w zależności od liczby procesów i długości słowa.

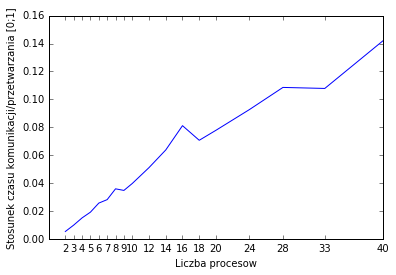
Uwaga, skala jest logarytmiczna!



Można zaobserwować, że większy wpływ na czas wykonania ma liczba procesów niż długość słowa.

## Czas obliczeń vs czas komunikacji.

Dodatkowo sprawdzono ile czasu zajmuje rozsyłanie słowników w stosunku do obliczeń. Wykres dla n=12



Zaobserwowana zależność jest zgoda z intuicją. Początkowa komunikacja (rozesłanie słowników) ma złożoność n-1 (bo wysyła się słownik do n-1 slave’ów). Późniejsze odbieranie wyników również wymaga n-1 komunikacji, co w sumie daje O(n-1+n-1)=O(2n)=O(n) czyli zależność liniową, widoczną na wykresie.