Treść zadania

Przetestuj funkcje haszujące:
-Bernsteina
-FNV
-modularną
dla ciągów znakowych. Porównaj jak działają wyżej wymienione funkcje przy użyciu standardowej implementacji unordered_set i własnej implementacji metody łańcuchowej.
Czy warto pomijać część znaków (np. co drugi, co trzeci) przy wyliczaniu funkcji haszującej (i dla jakiej długości ciągów).
Przetestuj różne rodzaje ciągów (np. losowe, wyrazy zdania).

Dodatkowe 5 punktów:
Zaimplementuj kopiec trójkowy.

Wstęp

Program testowy składa się z trzech struktur posiadających funkcje haszujące:

- Bernsteina(DJB2),

- Fowler–Noll–Vo(FNV) z użytymi stałymi dla 32 bitów tj. 2166136261 oraz 16777619

- Modularną

, które były wykorzystywane przy testowaniu standardowej implementacji unordered_set.

Własna metoda łańcuchowa opiera się o prostą klasę Lancuch posiadającą wbudowane funkcje ww. haszujące oraz funkcje dodaj, szukaj, usun.

Ciągi znaków użyte do testów generowane były przez dwie funkcje:

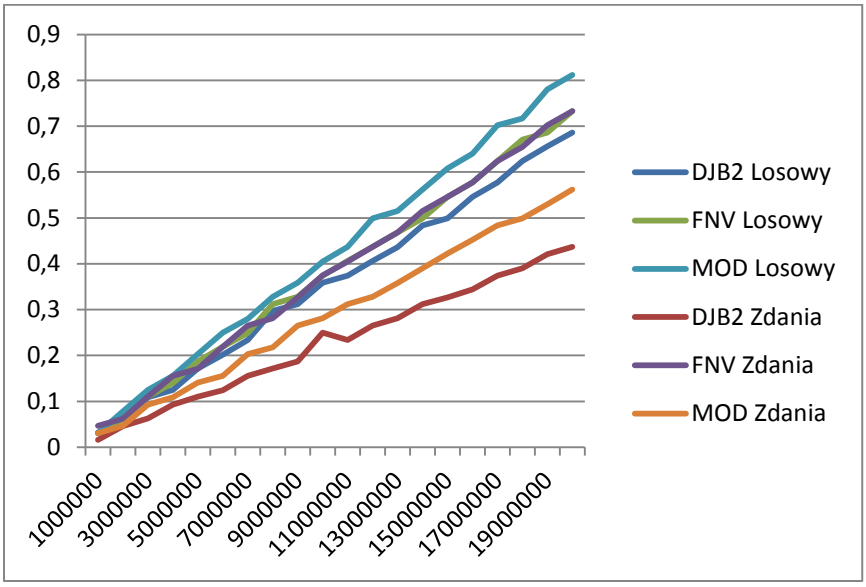
- jedną, generującą ciąg długości N z losowych liter alfabetu(a-z,A-Z)

- druga generującą ciag długości N z losowych 17 pierwszych słow tekstu tzw.” Lorem Ipsum”.

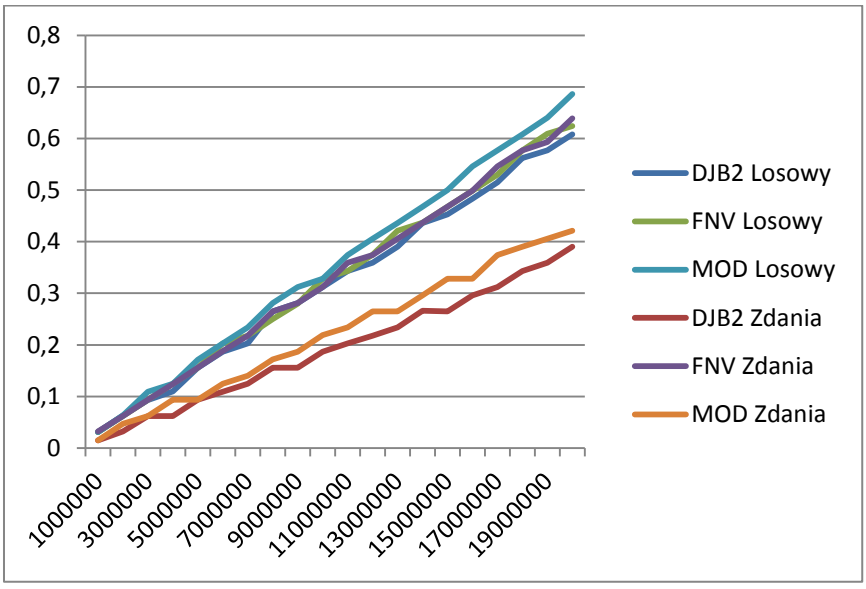
Wyniki

Wykresy przedstawiające czas w ms, przy zmiennej długości haszowanych ciągów, z podziałem na sposób generowania ciągu.

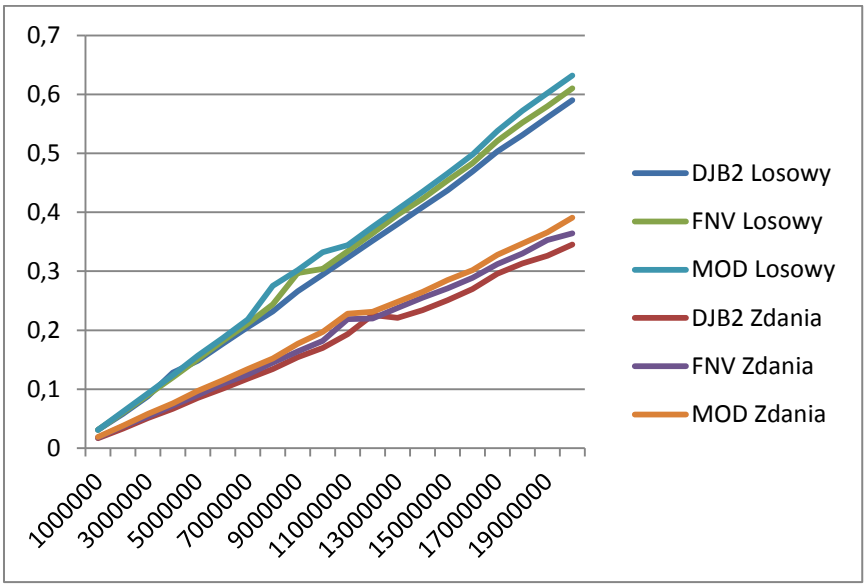
Dla skoku = 0. Wszystkie litery są haszowane.



Dla skoku = 1. Co druga litera jest haszowana.



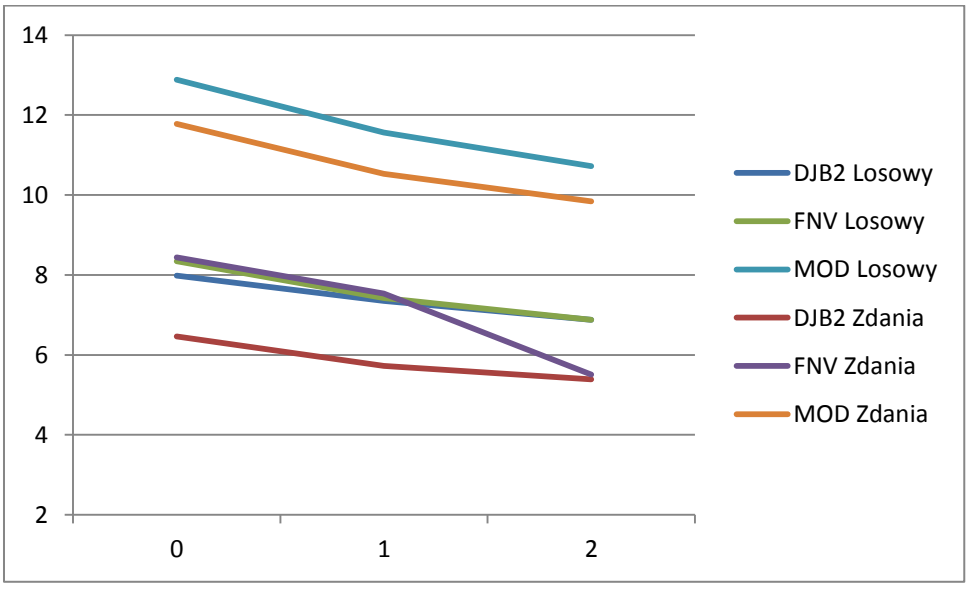
Dla skoku = 2. Co trzecia litera jest haszowana.



Jak możemy zauważyć haszowanie ciągów będących zdaniami , wypada nieznacznie wydajniej. Samą różnica po między poszczególnymi funkcjami jest naprawdę znikoma, ponieważ dla ciągu długości 20 000 000 wacha się w granicach około 0,1-0,3 sekundy . Przy dodatkowym ominięciu co drugiego, trzeciego znaku różnica ta, tym bardziej zanika.

Czas w ms, przy zmiennym skoku i przy stałej długości haszowanego ciągu

n = 1000.

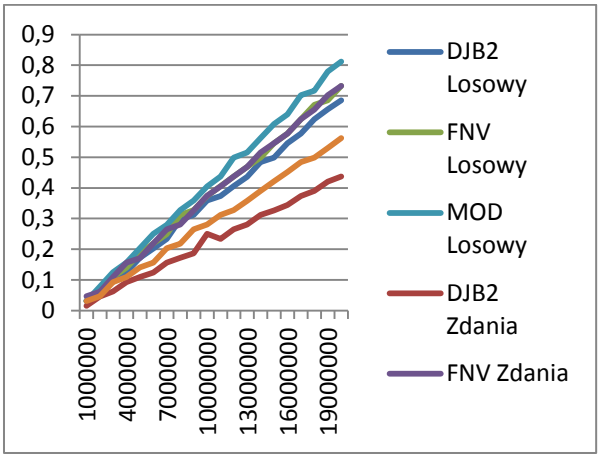


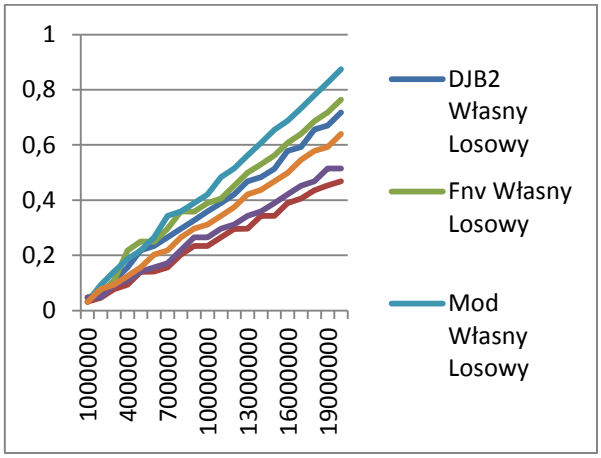
Wyniki dla własnej implementacji metody łańcuchowej

Wykresy przedstawiające czas w ms, przy zmiennej długości haszowanych ciągów, z podziałem na sposób generowania ciągu.

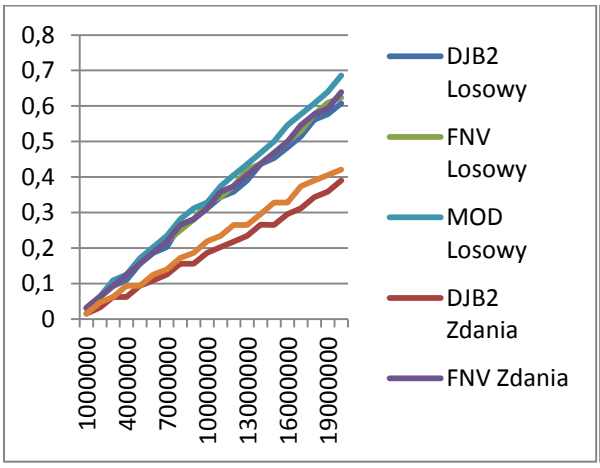
Po lewej stronie mamy wykresy standardowej implementacji unordered_set, a po prawej przedstawione są wykresy własnej implementacji metody łańcuchowej.

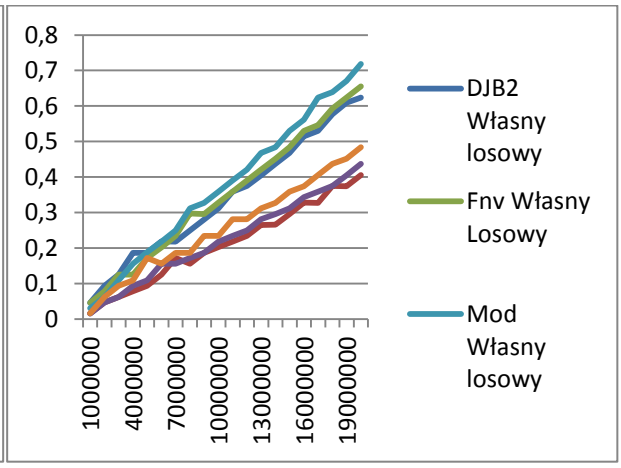
Dla skoku = 0. Wszystkie litery są haszowane.



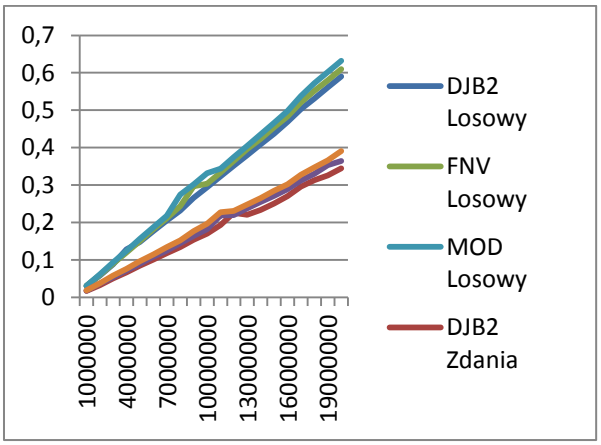


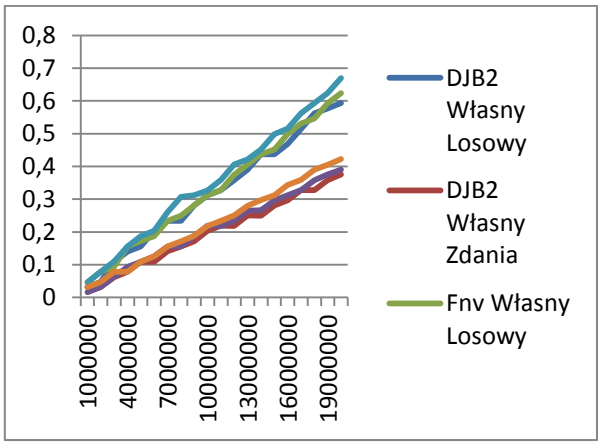
Dla skoku = 1. Co druga litera jest haszowana.





Dla skoku = 2. Co trzecia litera jest haszowana.





Jak widzimy na powyższych wykresach wyniki są bardzo zbliżone do standardowej implementacji.

Czas w ms, przy zmiennym skoku i przy stałej długości haszowanego ciągu

n = 1000.