Sprawozdanie

Lab6(Tablica haszująca)

Zadanie polegało na zaimplementowaniu struktury danych zwaną Tablicą asocjacyjną, której jednym z rozwiązań jest tablica z haszowaniem. Po stworzeniu struktury należało sprawdzić jaką złożoność obliczeniową ma jedna z operacji słownikowych: wyszukiwanie.

Dla tablicy z haszowaniem operacji słownikowe typu wstawianie, wyszukiwanie i pobieranie powinny mieć, wg źródeł literaturowych, złożoność rzędu O(1) dla przypadku średniego oraz przy pesymistycznym doborze parametrów – O(n).

W swojej implementacji stworzyłem 2 funkcje haszujące, których realizację zaczerpnąłem z książki „Wprowadzenie do algorytmów" T. H. Cormen'a. Haszowanie modularne oraz haszowanie przez mnożenie są jednymi z prostszych funkcji haszujących nieidealnych.

Funkcja haszująca pracowała na kluczu, który był liczbą całkowitą ustaloną według kodu ASCII pierwszych liter słowa, które podawano oryginalnie.

Dla haszowania modularnego:

Symbolicznie można ten sposób wyrazić przez wzór zaczerpnięty z wyżej wspomnianej książki:

h(k)=k mod m, gdzie k jest kluczem przekonwertowanym uprzednio dzięki kodowi ASCII, a m rozmiarem tablicy

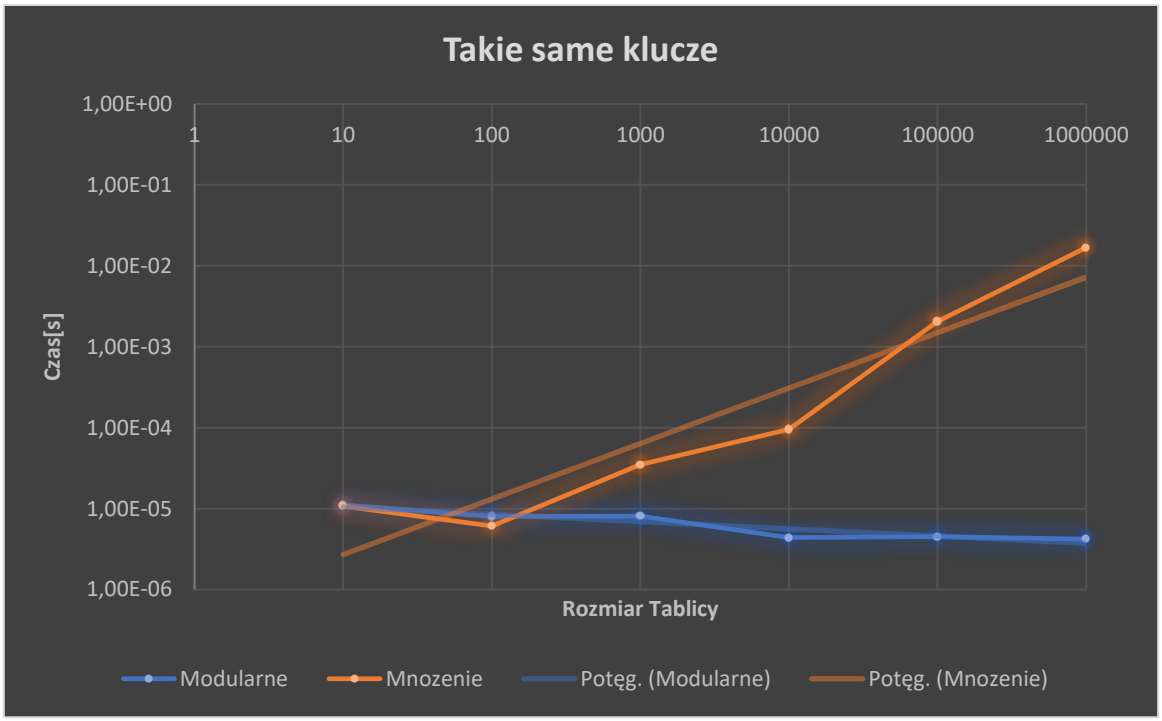
Dla haszowania przez mnożenie:

Analogicznie do poprzedniego jeśli chodzi o symbolikę jedynie wzór jest bardziej rozbudowany:

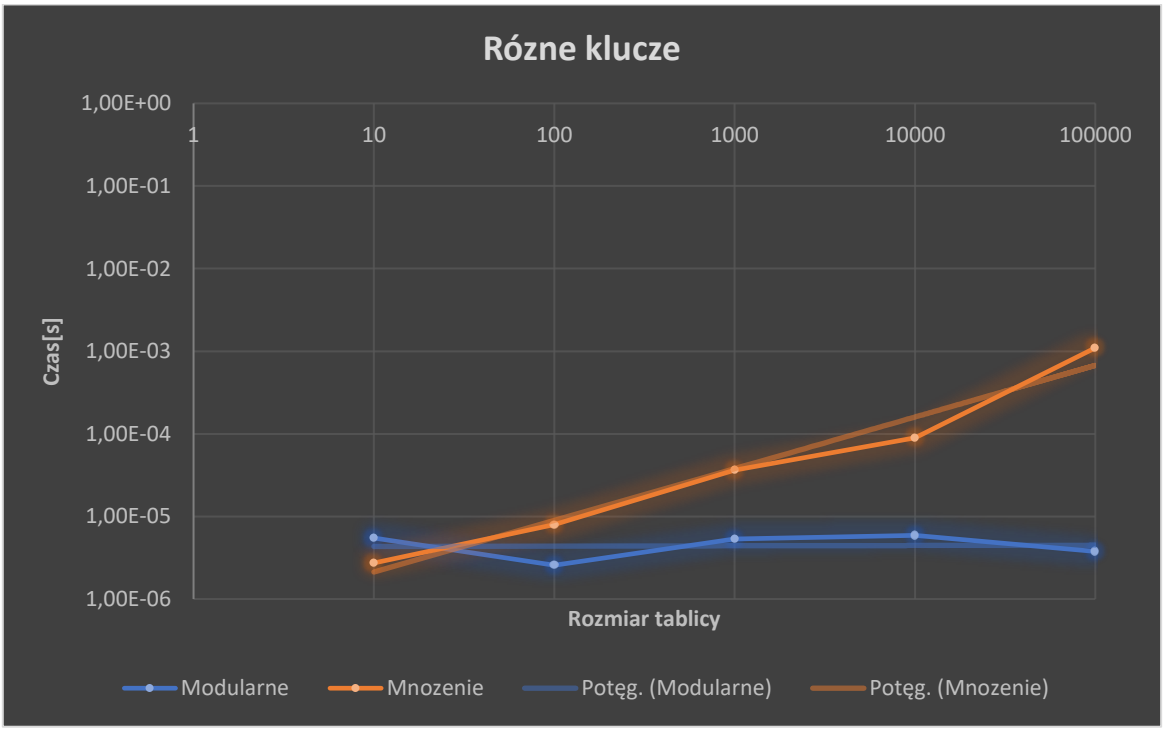
h(k)=$\lfloor m(kA \ mod \ 1) \rfloor$, gdzie A przyjąłem za 0,226256 czyli nr indeksu, mimo, że w książce Knutha zalecane jest dobieranie liczby A≈($\sqrt{5}$-1)/2=0,6180339887

Na następnej stronie wykresy dla średniego czasu wyszukiwania elementu.

Oczekiwane jest, aby przypominały one funkcję stałą.



Rysunek 1 Złożoność obliczeniowa dla przypadku gdzie klucze są identyczne jedynie szukany jest inny



Rysunek 2 Złożoność obliczeniowa dla przypadku, gdy klucze są różne, ale w postaci liczb całkowitych mogą się powtarzać.

Wnioski

- Widoczne jest, że dla haszowania modularnego zyskuje się lepsze wyniki niż z haszowaniem przez mnożenie. W drugim przypadku okazuje się, że trafiony jest przypadek bardziej pesymistyczny dlatego wykres kieruje się raczej w stronę funkcji liniowej, a nie zgodnie z oczekiwaniami w stronę stałej.
- Mimo takiego wyniku, podczas testowania tablicy zauważyłem, że budowanie tablicy dla większej ilości elementów przebiega szybciej dla haszowania przez mnożenie, co wynika najprawdopodobniej z szerszej gamy dobieranych kluczy typu całkowitego przez co występuje mniej kolizji. Dla tej implementacji problem kolizji został rozwiązany przez adresowanie liniowe, czyli gdy następowała kolizja było szukane następne wolne miejsce w tablicy. Tak więc jeśli mniej razy powtarzały się zhaszowane klucze tym rzadziej występowała kolizja co skutkuje szybszym budowaniem tablicy.
- Na obu wykresach widać, że pesymistyczny przypadek występuje, gdy uniwersum kluczy jest małe, a adresowanie jest rozwiązane w sposób liniowy, więc dodając ostatni element do tablicy algorytm musiał przeiterować wszystkie elementy.
- Dla haszowania modularnego w obu przypadkach wykres złożoności jest zbliżony do O(1), a w przypadku obu haszowań przez mnożenie zdecydowanie widoczna jest złożoność O(n). Jest to najprawdopodobniej spowodowane tym, że liczenie klucza w drugim przypadku jest bardziej rozbudowane, a przy każdej operacji porównania kluczy należy każdy kolejny klucz obliczyć.
- Złożoność obliczeniową dla haszowania przez mnożenie można zmniejszyć dobierając odpowiedni współczynnik A. Donald E. Knuth w „Sztuce programowania” doradza aby współczynnik był bliski $A\approx(\sqrt{5}-1)/2=0{,}6180339887$, ponieważ wtedy algorytm jest najbardziej efektywny a uniwersum kluczy największe.