Sprawozdanie z Laboratorium 6 - Pomiar czasu wyszukiwania klucza w tablicy asocjacyjnej.

Kamil Kuczaj

17 kwietnia 2016

1 Wstęp

Zadaniem na laboratorium był pomiar czasu wyszukiwania klucza w tablicy asocjacyjnej. Jest ona zbudowana z tablicy asocjacyjnej, która składa się z list niewielkich rozmiarów. Funkcja sortująca przyporządkowuje odpowiedni *bucket* po podaniu do niej klucza (*tj. elementu*. Wg teorii algorytm ten powinien mieć złożoność obliczeniową równą O(1).

W naszym przypadku mieliśmy załadować kolejno 10^1, 10^3, 10^5, 10^6, 10^9 oraz sprawdzić czas wyszukiwania znanego elementu. Konstrukcja tablicy asocjacyjnej przypominała w moim przypadku książkę telefoniczną, tj. na podstawie słowa zwracana była liczba złożona maksymalnie z 9 cyfr.

Na potrzeby zadania został napisany krótki program, który na podstawie zadanego słownika - pobrano angielski składający się z 349900 wyrażeń - generuje *książkę telefoniczną* przypisując każdemu wyrażeniu losową liczbę całkowitą mniejszą niż 10^10.

Ponieważ w tablicy asocjacyjnej należało posłużyć się listą, a mój poprzedni kod dla niej wykorzystywał węzły, a nie tablicę posłużyłem się biblioteką *STL* języka *C++* i szablonem *std::list<>*.

Jako *funkcję sortującą* (ang. *hash function*) został obrany algorytm, który oparty był o *Twierdzenie Bézouta*. Istnieje wiele innych, np. •

Obsługa kolizji w tablicy asocjacyjnej polega na tym, że nowy element jest po prostu dopisywany. Gdybym uwzględniał ten fakt, nigdy nie mógłbym dodać więcej elementów niż wspomniane 349900 elementów.

Tablica asocjacyjna zna swój rozmiar od momentu inicjalizacji - nie wystarczyło mi czasu na dodanie tej funkcji. Nie udało mi się również przeprowadzić testów dla miliarda elementów, gdyż zabrakło zasobów fizycznych

na komputerze na obsługę takiej listy. Nadmienię iż dla samej tablicy miliarda liczb całkowitych, było wymagane **5 GB pamięci fizycznej**. Tutaj dochodzi do tego zmienna typu std::string, która zajmuje dwa razy więcej pamięci. Dlatego, aby wykonać test dla miliarda elementów tego typu, komputer potrzebuje co najmniej **15 GB pamięci podręcznej**. W przeciwnym wypadku korzystając z pliku stronicowania, którego szybkość odczytu zależy od zastosowanego typu dysku twardego, proces ten będzie trwał bardzo długo.

2 Specyfikacja komputera

Wersja kompilatora *g++*	4.8.4
System	Ubuntu 14.04.4
Procesor	Intel Core i5 2510M 2.3 GHz
Pamięć RAM	8 GB DDR3 1600 MHz
Dysk twardy	HDD (5400 obr./min)
Rozmiar zmiennej *int*	4 bajty
Rozmiar zmiennej *std::string*	8 bajty

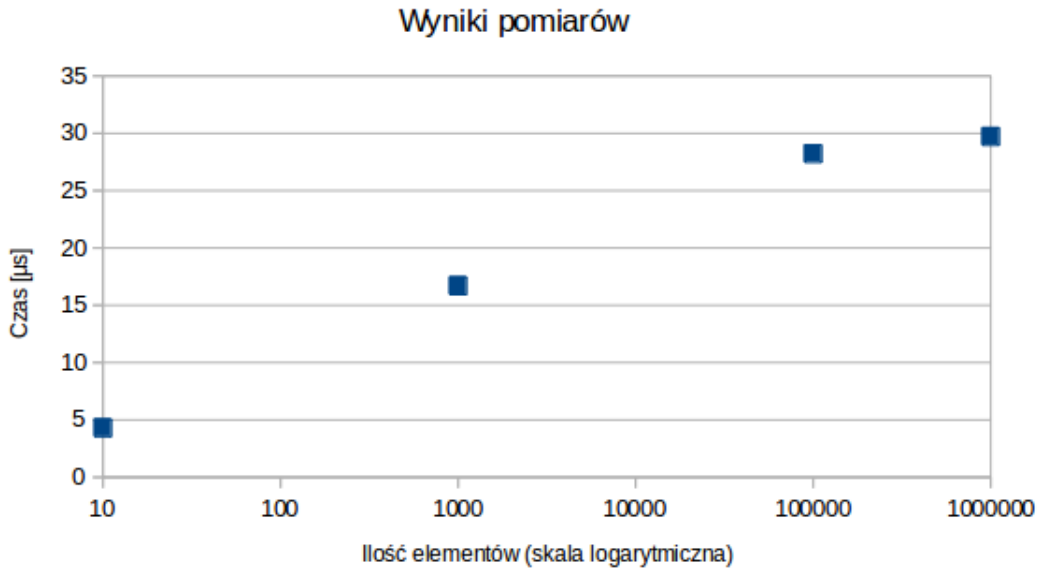
3 Pomiary oraz ich interpretacja

Tablica 1:

Tablica hashowa – wyniki w mikrosekundach μs			
10	**1000**	**100000**	**1000000**
30	8	24	30
5	14	27	38
4	26	25	32
4	18	26	30
3	19	26	28
4	19	27	27
4	21	27	29
4	22	28	27
4	21	27	26
4	17	26	31
4	16	35	31
4	17	31	26
3	22	28	26
3	15	26	29

Tablica 2:

Tablica hashowa – wyniki w mikrosekundach μs			
10	**1000**	**100000**	**1000000**
4	15	36	29
4	17	27	35
4	16	27	38
4	19	38	40
4	14	27	41
3	14	26	28
4	19	26	27
4	18	36	28
4	15	31	28
3	14	26	28
4	13	26	28
4	14	31	31
4	14	28	27
4	59	26	26
4	14	27	27
4	14	26	27
3	14	25	36
3	13	25	27
4	12	26	28
4	13	27	30
4	12	26	28
3	13	28	32
4	13	27	28
3	13	27	37
4	12	31	28
4	12	35	28
3	13	27	26
4	13	28	28
4	12	25	28
4	13	27	37
3	13	39	30
4	21	34	29
4	13	25	27
3	13	27	26
4	25	28	27
4	28	27	28
Średnie czasów [μs]			
4,3	16,7	28,22	29,72



4 Wnioski

Wyraźnie widać po wynikach pomiarów, że dostęp do każdego elementu jest błyskawiczny. Widać jednak, że w skali logarytmicznej punkty układają się mniej więcej na linii prostej, co implikuje to, że złożonośc obliczeniowa w czasie mojej listy asocjacyjnej to $O(log(n))$, a nie $O(1)$, tak jak mówi teoria. Być może odpowiedzialny jest za to algorytm haszujący.