Sprawozdanie z Laboratorium 7 - Pomiar czasu wyszukiwania losowego elementu w drzewie binarnym.

Kamil Kuczaj

8 maja 2016

1 Wstęp

Zadaniem na laboratorium był pomiar czasu wyszukiwania losowego elementu w drzewie czerwono-czarnym (*ang. Red Black Tree*. Drzewo to zostało opracowane przez Rudolfa Bayera w 1972 roku. Wg teorii algorytm wyszukiwania elementu w tej strukturze powinien mieć złożoność obliczeniową równą $O(log(n))$.

W naszym przypadku mieliśmy załadować kolejno 10^1, 10^3, 10^5, 10^6, 10^9 oraz sprawdzić czas wyszukiwania znanego elementu. Niestety komputer przy próbie alokacji miliarda elementów zapełniał cała pamięć fizyczną RAM komputera i dalsze ładowanie elementów następowało bardzo powoli. Odczyt z tych obszarów pamięci (pliku stronicowego *swap*) zajmował również więcej czasu, co wpłynęło by negatywnie na wyniki pomiarów. Dlatego zdecydowałem się robić pomiary 10^8 elementów zamiast 10^9. Ich alokacja kończyła się na zapełnieniu ok. $6,5GB$ pamięci RAM. W każdej z 50 prób wyszukiwane było to samo słowo, które zostało wylosowane podczas budowania drzewa korzystając z biblioteki *<cstdlib>*.

Obok tzw. *Red Black Tree* możemy znaleźć w literaturze inne konstrukcje (w nawiasach rok opracowania):

1. AVL tree (1962)
2. B-tree (1972)
3. Scapegoat tree (1989)
4. WAVL tree (2015)
5. Splay tree (1985)

Jednak drzewo czerwono-czarne jest tak świetnym pomysłem, że twórcy języka C++ postanowili je zaimplementować jako element biblioteki STL w postaci *std::set<>*. Pomiary zostały wykonane właśnie na tej konstrukcji ze względu na najpewniejszą dokładność pomiarów.

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

3 Pomiary oraz ich interpretacja

Tablica 1:

	Czasy dla znajdywania elementu w milisekundach				
Ilość elementów	**10**	**1000**	**100000**	**1000000**	**100000000**
Pomiar 1 [ms]	33	28	32	29	26
Pomiar 2 [ms]	30	30	31	29	28
Pomiar 3 [ms]	40	24	26	15	10
Pomiar 4 [ms]	18	28	10	14	7
Pomiar 5 [ms]	25	22	7	11	7
Pomiar 6 [ms]	17	24	7	13	7
Pomiar 7 [ms]	17	23	10	14	7
Pomiar 8 [ms]	16	22	7	12	7
Pomiar 9 [ms]	17	23	6	18	7
Pomiar 10 [ms]	17	24	6	12	7
Pomiar 11 [ms]	17	23	7	12	8
Pomiar 12 [ms]	19	23	7	13	7
Pomiar 13 [ms]	17	23	6	12	7
Pomiar 14 [ms]	18	33	6	14	7
Pomiar 15 [ms]	16	23	7	11	7
Pomiar 16 [ms]	16	23	7	12	7
Pomiar 17 [ms]	16	23	7	12	7
Pomiar 18 [ms]	16	28	7	10	6
Pomiar 19 [ms]	19	24	7	8	7
Pomiar 20 [ms]	16	20	7	7	7
Pomiar 21 [ms]	16	18	8	7	7
Pomiar 22 [ms]	17	19	11	7	7
Pomiar 23 [ms]	20	20	7	7	7
Pomiar 24 [ms]	16	18	7	7	7
Pomiar 25 [ms]	16	21	7	7	7
Pomiar 26 [ms]	16	18	7	7	7
Pomiar 27 [ms]	17	18	7	7	8
Pomiar 28 [ms]	16	18	7	7	7
Pomiar 29 [ms]	17	19	10	7	7
Pomiar 30 [ms]	16	18	6	7	7
Pomiar 31 [ms]	17	18	7	7	7
Pomiar 32 [ms]	17	19	7	7	8
Pomiar 33 [ms]	19	17	7	7	7
Pomiar 34 [ms]	17	18	7	7	9

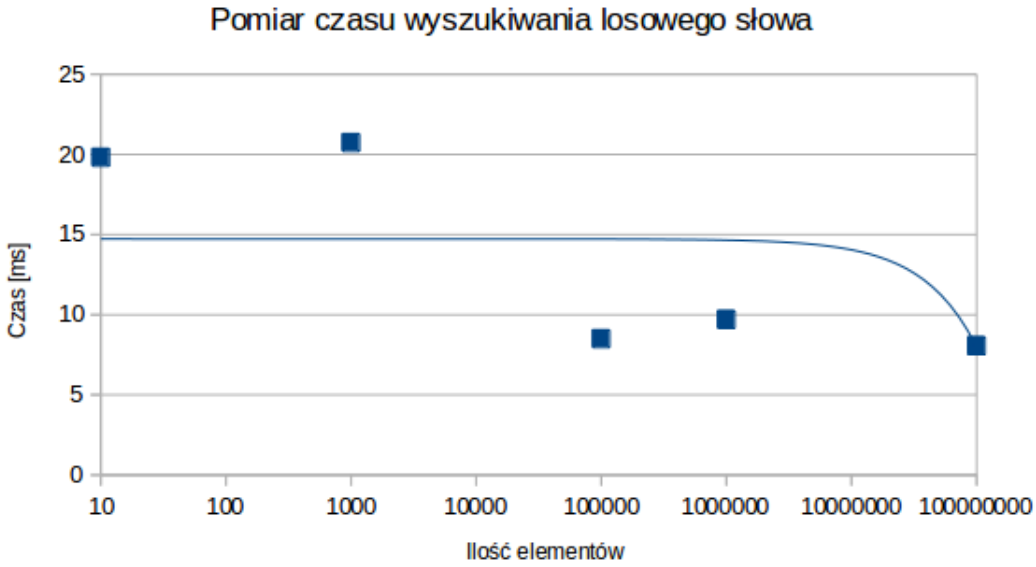
Tablica 2:

	Czasy dla znajdywania elementu w milisekundach				
Ilość elementów	**10**	**1000**	**100000**	**1000000**	**100000000**
Pomiar 35 [ms]	16	18	7	7	7
Pomiar 36 [ms]	20	18	8	6	8
Pomiar 37 [ms]	16	17	7	6	8
Pomiar 38 [ms]	17	17	6	7	7
Pomiar 39 [ms]	34	18	6	7	8
Pomiar 40 [ms]	27	18	6	7	7
Pomiar 41 [ms]	22	18	7	7	7
Pomiar 42 [ms]	22	18	7	7	8
Pomiar 43 [ms]	21	18	7	7	7
Pomiar 44 [ms]	23	18	7	7	8
Pomiar 45 [ms]	21	18	7	7	7
Pomiar 46 [ms]	22	18	7	6	7
Pomiar 47 [ms]	23	18	6	7	7
Pomiar 48 [ms]	25	18	7	7	8
Pomiar 49 [ms]	23	18	7	7	7
Pomiar 50 [ms]	22	19	8	7	7

Tablica 3:

Ilość elementów	**Średni czas na podstawie 50 pomiarów w milisekundach [ms]**
10	19,82
1000	20,74
100000	8,5
1000000	9,7
100000000	8,06

Wyraźnie widać, że algorytm wyszukiwania odbiega od teorii. Po głębszej analizie biblioteki STL okazało się, że *std::set<>* nie zawsze musi być implementowane jako drzewo czerwono-czarne. W moim przypadku jest to najprawdopodobniej odmiana tablicy asocjacyjnej, gdyż wyniki wskazują na to, iż algorytm wyszukiwania ma złożoność liczbową O(1). Być może miało tu miejsce również tzw. przewidywanie gałęzi (*ang. Branch Prediction.* Jest to zjawisko, które jest obecne w nowszych systemach jądrach systemu. System mając posortowane dane, lepiej i szybciej radzi sobie z ich przetwarzaniem.



Rysunek 1: Uśrednione wyniki pomiarów wraz z regresją liniową. Oś odciętych w skali logarytmicznej.

4 Wnioski

Wyniki nie są zgodne z teorią. Drzewo czerwono-czarne jest bardzo popularną implementacją drzewa binarnego, które nie wymaga rebalansowania. Jest jednak niezmiernie trudne w implementacji, czego przekonano się podczas próby zaprogramowania tej struktury. Nie zdecydowano się na pomiar zwykłego drzewa bez rebalansowania, gdyż najprawdopodobniej otrzymano by złożoność obliczeniową $O(n)$ z wiadomych względów.