Sprawozdanie z Laboratorium 3 - Pomiar czasu znajdywania losowego słowa z listy.

Kamil Kuczaj

11 maja 2016

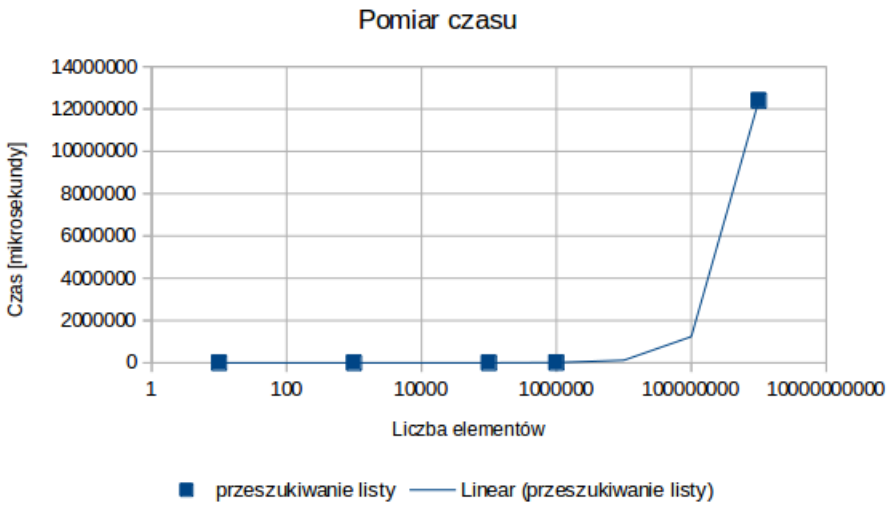
1 Wstęp

Podanym zadaniem był pomiar czasu znajdywania losowego elementu listy typu*int*. Należało wykonać pomiary zapisu: 10^1, 10^3, 10^5, 10^6 oraz 10^9. Wykorzystano bibliotekę *<cstdlib>* do generacji losowych liczba. Za każdym razem gdy generowana była lista określonego rozmiaru, losowane był indeks, o który oprą się pomiary. Następnie uśredniono wyniki pięćdziesięciu pomiarów każdego rozmiaru oraz obliczono predykcyjnie ile wyniósłby czas przeszukania całej listy, gdyby komputer utrzymał podane tempo przeszukiwanie listy

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
Rozmiar zmiennej *int*	4 bajty

3 Pomiary oraz ich interpretacja



Rysunek 1: Zobrazowanie wyników pomiaru oraz regresja liniowa wykonana w programie *LibreOffice Calc*.

	Pomiary czasu przeszukiwania listy n-elementowej				
Ilość elementów	10	10^3	10^5	10^6	10^9
Pomiar 1 [us]	27	14	801	10848	5118450
Pomiar 2 [us]	12	14	731	10299	5157010
Pomiar 3 [us]	8	13	727	10598	5119420
Pomiar 4 [us]	7	15	654	11445	5010000
Pomiar 5 [us]	6	15	598	10950	4807910
Pomiar 6 [us]	7	15	618	10011	4667350
Pomiar 7 [us]	8	14	858	10036	4824640
Pomiar 8 [us]	6	15	880	11255	4808720
Pomiar 9 [us]	8	14	856	10754	5056860
Pomiar 10 [us]	6	15	842	10385	5362980
Pomiar 11 [us]	6	15	925	11173	5631260
Pomiar 12 [us]	6	15	860	12431	4804130
Pomiar 13 [us]	8	15	875	10471	5053090
Pomiar 14 [us]	7	14	860	11501	5456830
Pomiar 15 [us]	6	15	635	10718	5152060
Pomiar 16 [us]	8	14	568	10621	4728930
Pomiar 17 [us]	6	14	566	10085	4709380
Pomiar 18 [us]	7	15	576	10755	4858750
Pomiar 19 [us]	7	15	566	10625	4746280
Pomiar 20 [us]	6	15	566	10638	5122880
Pomiar 21 [us]	7	14	768	10338	5021030
Pomiar 22 [us]	7	14	776	10423	5198310
Pomiar 23 [us]	7	15	803	10671	4938670
Pomiar 24 [us]	6	15	568	10110	5447080
Pomiar 25 [us]	7	14	557	9994	4762510
Pomiar 26 [us]	8	14	567	10343	4828810
Pomiar 27 [us]	6	15	598	10312	5148940
Pomiar 28 [us]	6	15	589	10330	5413450
Pomiar 29 [us]	7	13	598	10281	6023600
Pomiar 30 [us]	7	14	583	10351	5554910
Pomiar 31 [us]	7	13	579	10339	4938560
Pomiar 32 [us]	7	13	550	10541	5009070
Pomiar 33 [us]	6	13	556	10269	4969610
Pomiar 34 [us]	8	13	557	9980	5448210
Pomiar 35 [us]	7	14	731	10766	5188510
Pomiar 36 [us]	6	13	851	10066	5341730
Pomiar 37 [us]	8	13	769	10103	4893010
Pomiar 38 [us]	7	14	577	10728	4950110
Pomiar 39 [us]	7	14	565	9956	5033890
Pomiar 40 [us]	8	13	558	10198	5106000
Pomiar 41 [us]	7	14	557	10881	4836780
Pomiar 42 [us]	8	14	721	9890	4740970
Pomiar 43 [us]	6	13	827	9964	4676980
Pomiar 44 [us]	7	14	568	10792	4591480
Pomiar 45 [us]	7	13	551	9892	4593950
Pomiar 46 [us]	7	13	653	12217	4604570
Pomiar 47 [us]	7	14	562	11086	4696340
Pomiar 48 [us]	7	21	615	10414	4782500
Pomiar 49 [us]	8	13	612	11306	4792440
Pomiar 50 [us]	7	13	551	10386	4709200

Tablica 1:

	Uśredniony czas przeszukiwania listy				
Ilość elementów	10	10^3	10^5	10^6	10^9
Szukany indeks	5	180	47 180	847 180	403 878 424
Szukany indeks (procentowo)	50,00%	18,00%	47,18%	84,72%	40,39%
Średnia [us]	7,44	14,18	669,58	10 570,52	5 008 763

Tablica 2:

	Predykcja – ile zajęłoby czasu przeszukanie całej listy				
Ilość elementów	10	10^3	10^5	10^6	10^9
Średnia [us]	14,88	78,78	1 419,21	12 477,31	12 401 660,26

Po ostatniej tabeli wyraźnie widać, że wyszukiwanie elementów do tablicy ma złożoność obliczeniową rowną O(n). Najlepiej można to zaobserwować porównując predykcję dla miliona i miliarda elementów. Dzięki dużej ilości danych, bardzo zmalał błąd pomiaru czasu i można zaobserwować proporcjonalny wzrost czasu wyszukiwania - 1000 razy większy dla 1000 razy większej próbki.

4 Wnioski

Dysponując podanymi danymi oraz stojącą za nimi teorią zakwalifikowałbym przeszukanie tablicy jako algorytm rzędu O(n). Podczas implementacji algorytmu dowiedziałem się dlaczego implementacja tablicowa jest dużo lepsza. Pozwala na szybszy zapis ze względu na to, że alokacja pamięci następuje w złożoności obliczeniowej O($nlog(n)$) a nie O(n^2).