Badanie efektywności operacji dodawania (wstawiania), usuwania oraz wyszukiwania elementów w podstawowych strukturach danych

1. Plan pomiarów

a. Generowanie danych

Wszystkie struktury na których przeprowadzano pomiary zapełniono losowymi liczbami z przedziału (-2147483648, 2147483647) używając funkcji rand() za wyjątkiem badania wpływy wartości w strukturze na czas wykonywania w niej operacji

b. Pomiar czasu wstawianie i usuwanie elementów

Ponieważ jednokrotny pomiar operacji obciążony jest dużą niedokładnością, powtórzono ją kilkukrotnie, z nadzieją na poprawienie dokładności(więcej o tym w konkluzjach).
Ogólny algorytm prezentuje się następująco

```
rozpocznij pomiar czasu
for(i=0;i<ilosc testow;i++)
{
        utwórz strukturę
        wykonaj operację
        usuń strukturę
}
wynik = zmierzony czas/ilość testów
```

Następnie pomiar dla takich samych danych powtórzono wykonując jedynie instrukcje tworzenia i usuwania struktury, różnica czasu uzyskanego w tych dwóch pomiarów jest faktycznym czasem wstawiania/usuwania elementu.

c. Pomiar czasu wyszukiwanie elementów

Wielokrotność pomiary czasu operacji nie wymagało realokacji struktury, w algorytmie w podpunkcie b. w pętli zostało samo wykonywanie właściwe operacji.
Aby uzyskać przypadek pesymistyczny przy wyszukiwaniu, szukany element(unikatowy) umieszczono na końcu struktury, zmuszając program do przeszukania jej w całości za każdym razem.

2. Przebieg pomiarów

a. Czy wartość ma znaczenie?

Teoria mówi że skopiowanie wartości zapisanej na stałej ilości bitów zajmuje tyle samo(jeśli rozpatrujemy strukturę nieporządkowaną), więc na początek upewniłem się że nie ma sensu wykonywać pomiarów dla różnych zestawów danych.

Miejsce\wartości	-(2^32)/2;(2^32)/2	różnica	10;100
Przód	0,852957	0,001882	0,851074
Środek	0,849547	0,011257	0,838290
Koniec	0,847970	-0,022509	0,870479
Wynik średni 50 tyś pomiarów wstawienia elementu do tablicy 500 tyś-elementowej, czas w [ms]			

Pomiary zdają się potwierdzać tezę, różnica wstawiania do tablicy o małych wartościach stanowi 0,2 – 2,65% czasu operacji i nie jest jednoznacznie na korzyść którejś z tablic, co może być powodem błędów pomiarowych.

b. Wstawianie do tablicy i listy

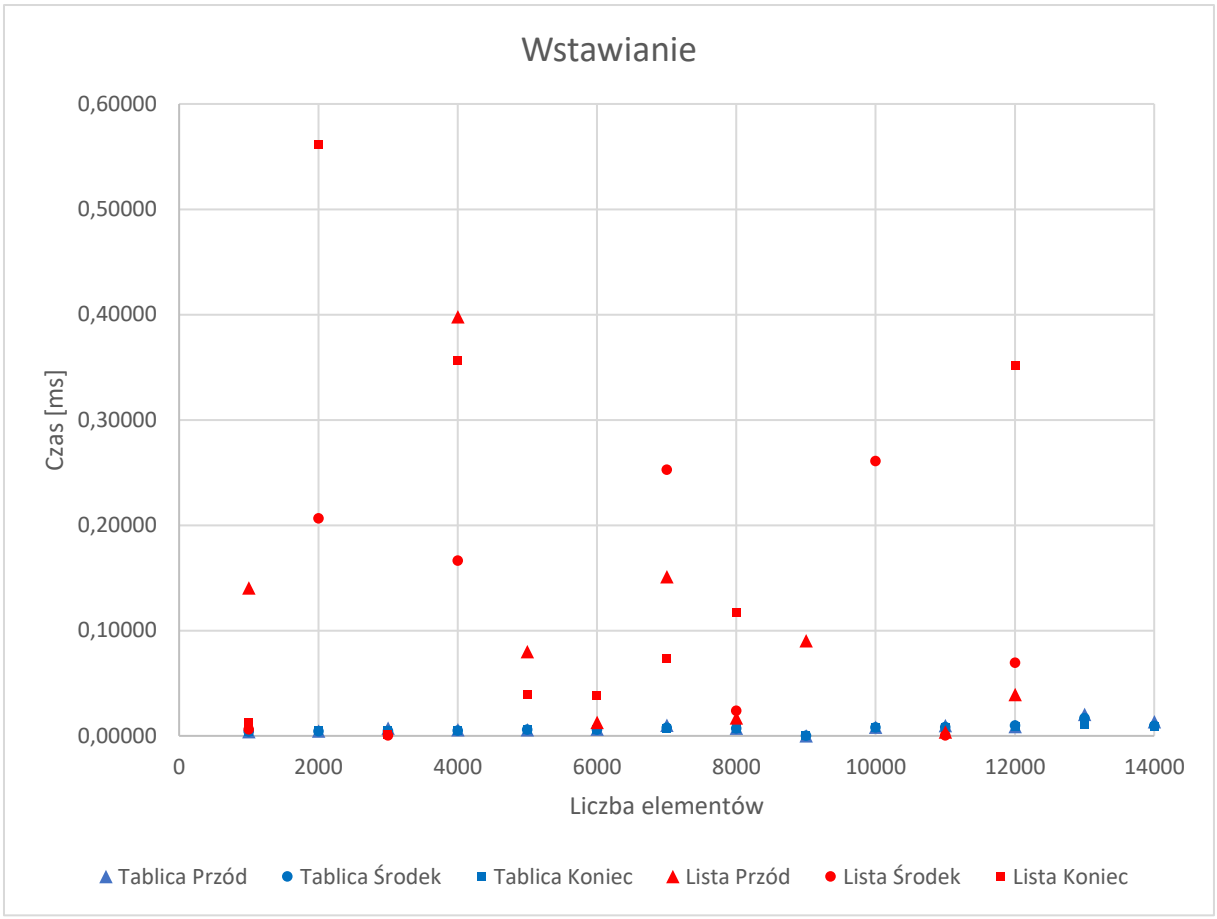
Teoretycznie wstawianie w dowolne miejsce tablicy powinno zająć tyle samo czasu, sumarycznie wykonujemy tyle samo operacji(kopiujemy n komórek pamięci, ewentualnie trochę rozsunięte) i wstawiamy dodatkową wartość.
Złożoność obliczeniowa, O(n).

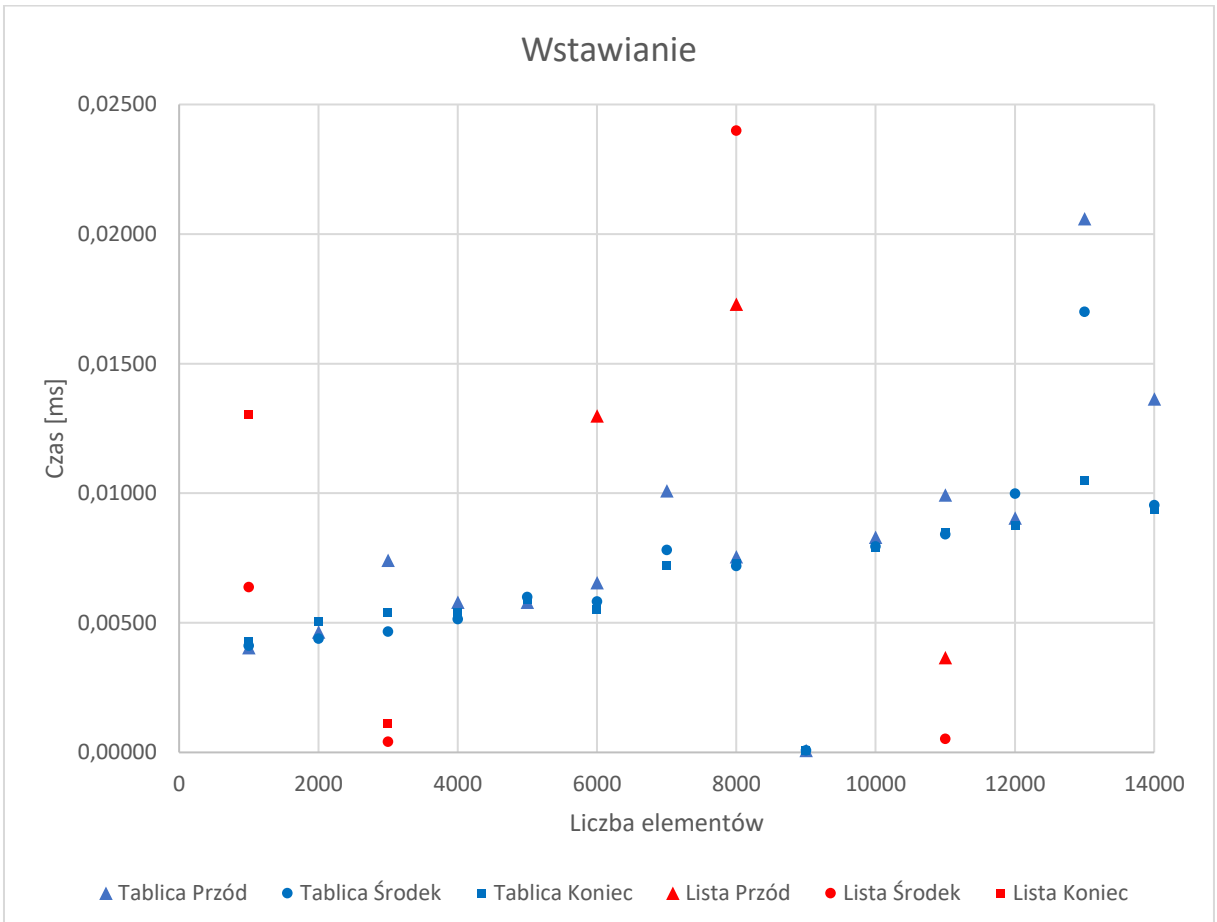
Dla listy, teoretycznie, to miejsce wstawienia powinno mieć wpływ na czas operacji, ponieważ wstawienia na początku i na końcu wymaga zmiany powiązań pomiędzy tylko dwoma węzłami, a wstawienie gdziekolwiek w środku dostanie się do elementu o wskazanym indeksie po wskaźnikach(przez brak dostępu swobodnego) a następnie zmiana powiązań między trzema kolejnymi węzłami.
Złożoność wstawiania na początku i końcu, O(1), w środku, O(n)

Jak widać poniżej, wyniki dla listy zdają się być zupełnie losowe, możne je uznać za nieważne, więcej na ten temat w konkluzjach.

Badanie tablic potwierdza teorię, długość poszczególnych operacji miesza się ze sobą na wykresie mieszcząc się w błędach pomiaru, ale układają się w funkcję liniową o takim samym nachyleniu.

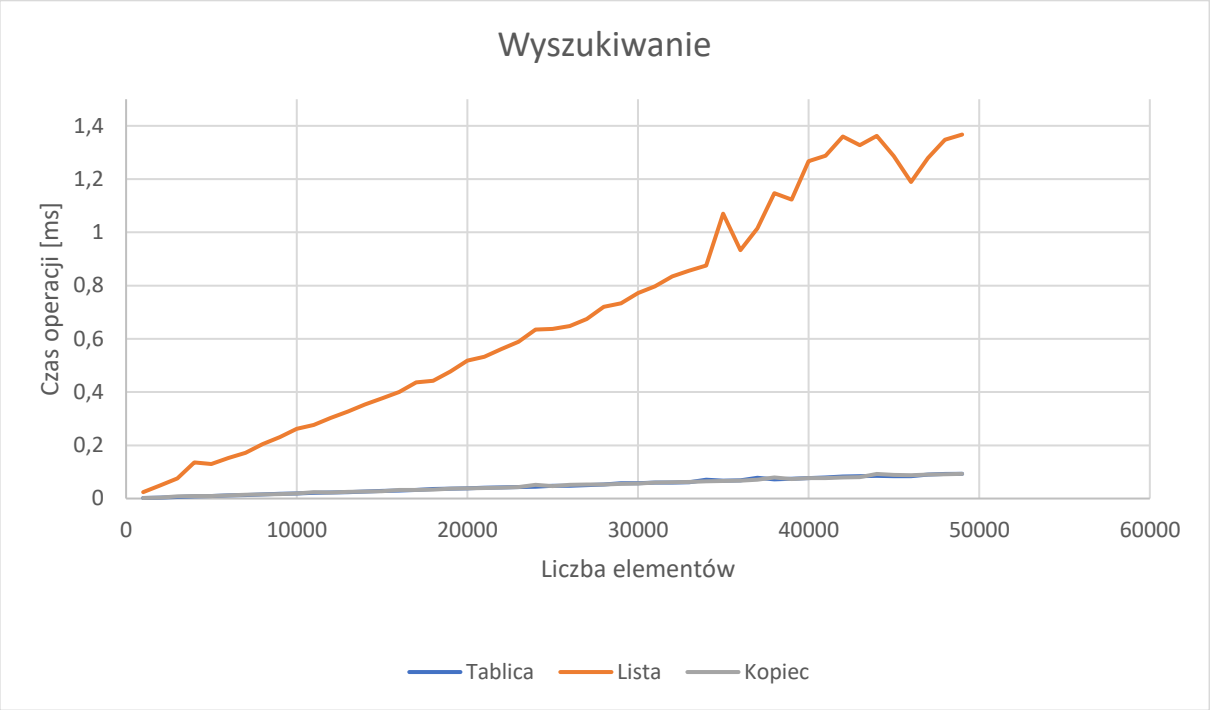
Wyniki średnie 500 – 50000 operacji, , czas w [ms]						
Liczba elementów	Tablica			Lista		
	Przód	Środek	Koniec	Przód	Środek	Koniec
1000	0,00404	0,00412	0,00427	0,14065	0,00637	0,01305
2000	0,00464	0,00438	0,00504	-0,14631	0,20660	0,56176
3000	0,00741	0,00465	0,00541	-0,00029	0,00041	0,00112
4000	0,00579	0,00514	0,00538	0,39803	0,16652	0,35614
5000	0,00579	0,00600	0,00589	0,08002	-0,06422	0,03959
6000	0,00654	0,00582	0,00550	0,01298	-0,02632	0,03789
7000	0,01009	0,00781	0,00718	0,15119	0,25280	0,07366
8000	0,00755	0,00719	0,00730	0,01729	0,02398	0,11736
9000	0,00008	0,00007	0,00007	0,09043	-0,53273	1,46069
10000	0,00830	0,00794	0,00790	1,83047	0,26083	-0,05493
11000	0,00993	0,00841	0,00846	0,00366	0,00052	-0,00011
12000	0,00904	0,00998	0,00876	0,03946	0,06948	0,35129
13000	0,02059	0,01700	0,01047	-0,62796	-1,18077	-0,88213
14000	0,01364	0,00953	0,00936	1,01248	0,85903	-0,50446

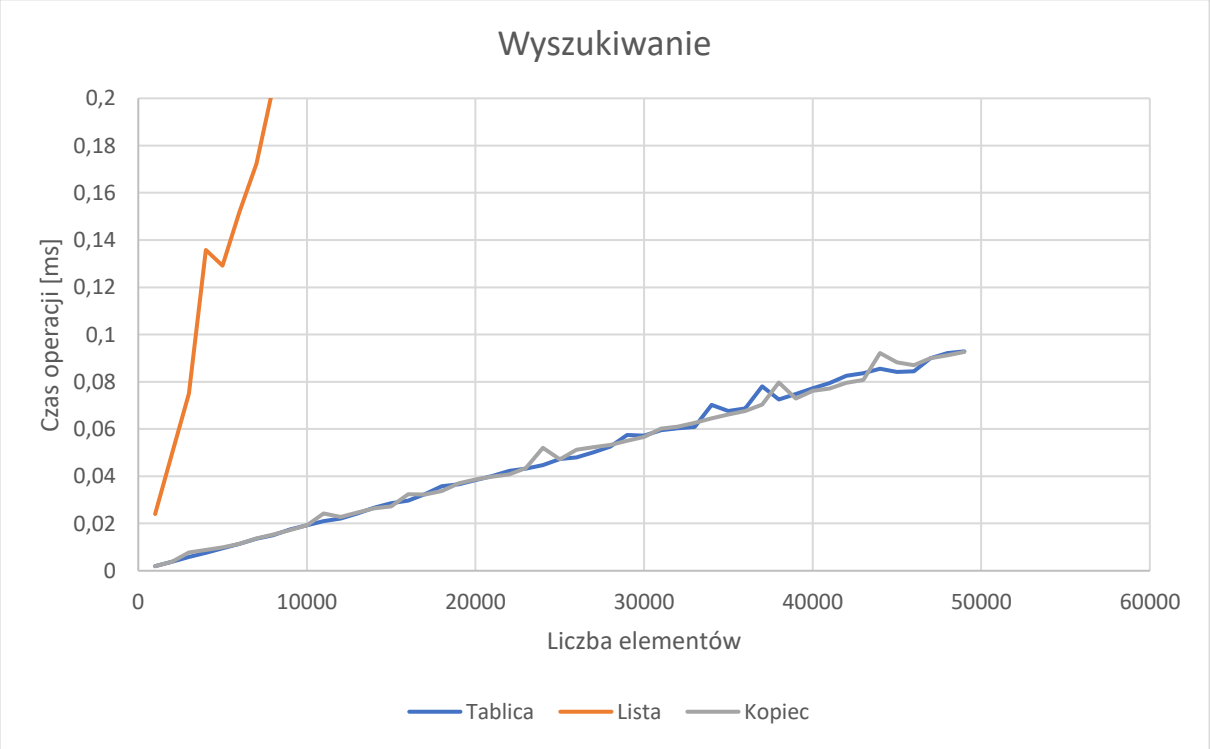

Wstawianie
0,60000
0,50000
0,40000
0,30000
0,20000
0,10000
0,00000
Czas [ms]
0
2000
4000
6000
8000
10000
12000
14000
Liczba elementów
Tablica Przód
Tablica Środek
Tablica Koniec
Lista Przód
Lista Środek
Lista Koniec


Wstawianie
0,02500
0,02000
0,01500
0,01000
0,00500
0,00000
Czas [ms]
0
2000
4000
6000
8000
10000
12000
14000
Liczba elementów
Tablica Przód
Tablica Środek
Tablica Koniec
Lista Przód
Lista Środek
Lista Koniec

c. Wyszukiwanie w liście, tablicy i kopcu

Dla wszystkich struktur wyszukiwanie powinno mieć złożoność O(n) ponieważ nie są w żaden sposób posortowane, i jeśli szukany element znajduje się w ostatniej komórce, trzeba sprawdzić wszystkie, lista powinna być znowu wolniejsza ponieważ skaczemy po referencjach.
Badania potwierdzają teorię. Błędy pomiarowe znowu trochę psują wykres ale generalnie widać liniowość.





3. Konkluzje

Wiele wyników zdaje się być obarczonych dużymi błędami, mogącymi wynikać z: zmiennego taktowania procesora, wahającego się między 3,3 a 3,6 GHz; oraz pamięci cache która przy jednym obiegu mogła zachować całą strukturę a za drugim nie. Z tego powodu nie da się zaobserwować np. dużo mniejszego czasu dodawania elementu na koniec listy w porównaniu do innych struktur.

a. Pytania dodatkowe

Q: Czy w tablicy mogą występować puste, czyli bez klucza podanego przez użytkownika i do czego może służyć ich istnienie?
A: W tablicy mogą występować puste(komórki), przykładem zastosowania są tablice mieszające, w której każdy element który może się znaleźć ma wyliczoną ze wzoru pozycję, więc podczas wyszukiwania elementu w tablicy mieszającej pusta komórka z wyliczonej pozycji oznacza że szukanego elementu tam nie ma, co zmniejsza złożoność wyszukiwania do O(1).

Q: Czy w pozostałych strukturach mogą występować miejsca puste? Odpowiedź uzasadnić?
A: Ciężko znaleźć autentyczny przypadek ale przyjmijmy hipotetycznie że przechowujemy dane w liście, do której będziemy chcieli w krótkim czasie wprowadzić dane. Możemy najpierw stworzyć wymaganą ilość węzłów bez żadnych konkretnych wartości, a potem, gdy otrzymamy dane zapisywać je we wcześniej przygotowanych węzłach. Sumarycznie czas operacji się wydłuży, ale rozbijemy go na dwa mniejsze okresy, co może mieć znaczenie jeśli np. mamy ograniczony czas poboru danych z nośnika.

Q: Czy samodzielnie budowane struktury są/mogą być efektywniejsze od implementowanych w dostępnych bibliotekach? Odpowiedź uzasadnić.
A: Mogą być minimalnie efektywniejsze, wystarczy pisać kod mniej odporny na błędy, np. nie sprawdzający czy próbujemy dodać element o ujemnym indeksie. Niemniej, nie zmienimy ich złożoności obliczeniowej(notacja dużego O), jeśli uzyskamy lepszą to w większości przypadków powinniśmy się zgłosić po nagrodę Nobla.

4. Literatura

http://eduinf.waw.pl/inf/alg/001_search/index.php