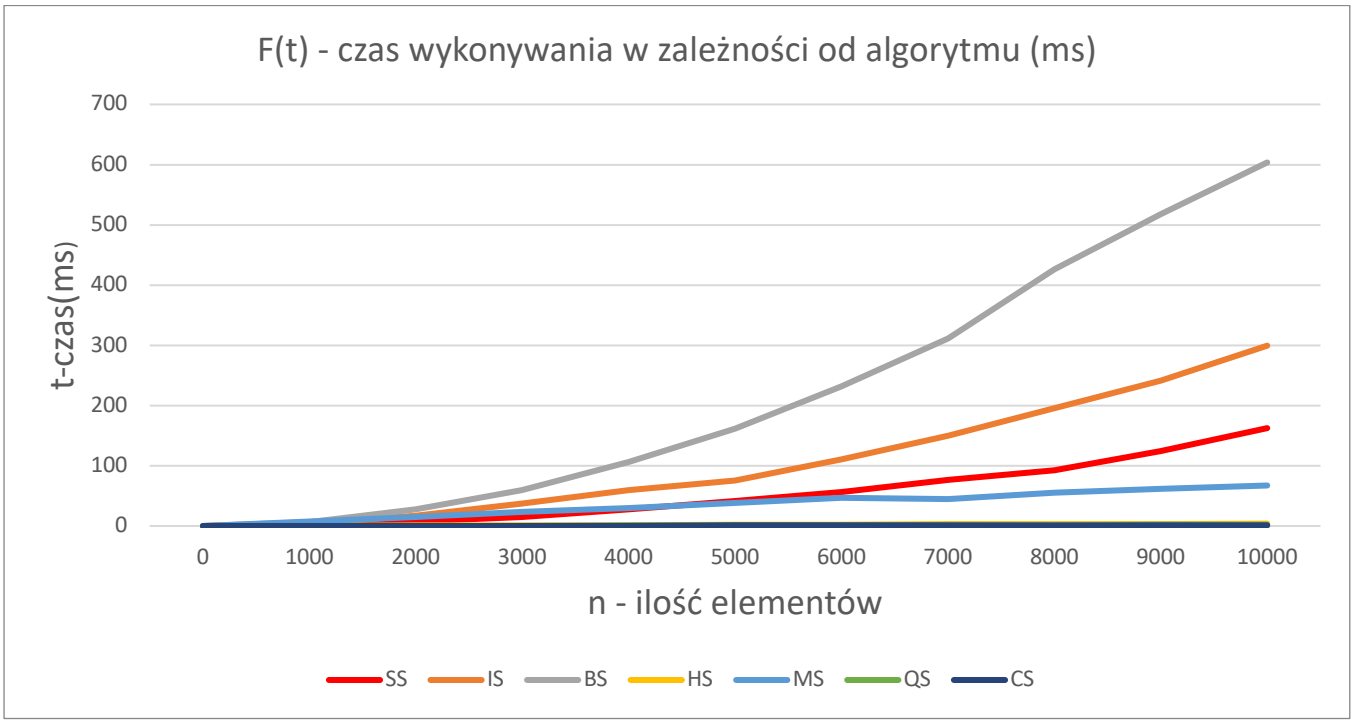
Sprawozdanie nr 1
18.03.2022, Ćwiczenia nr3.
Proste i zaawansowane algorytmy sortowania.

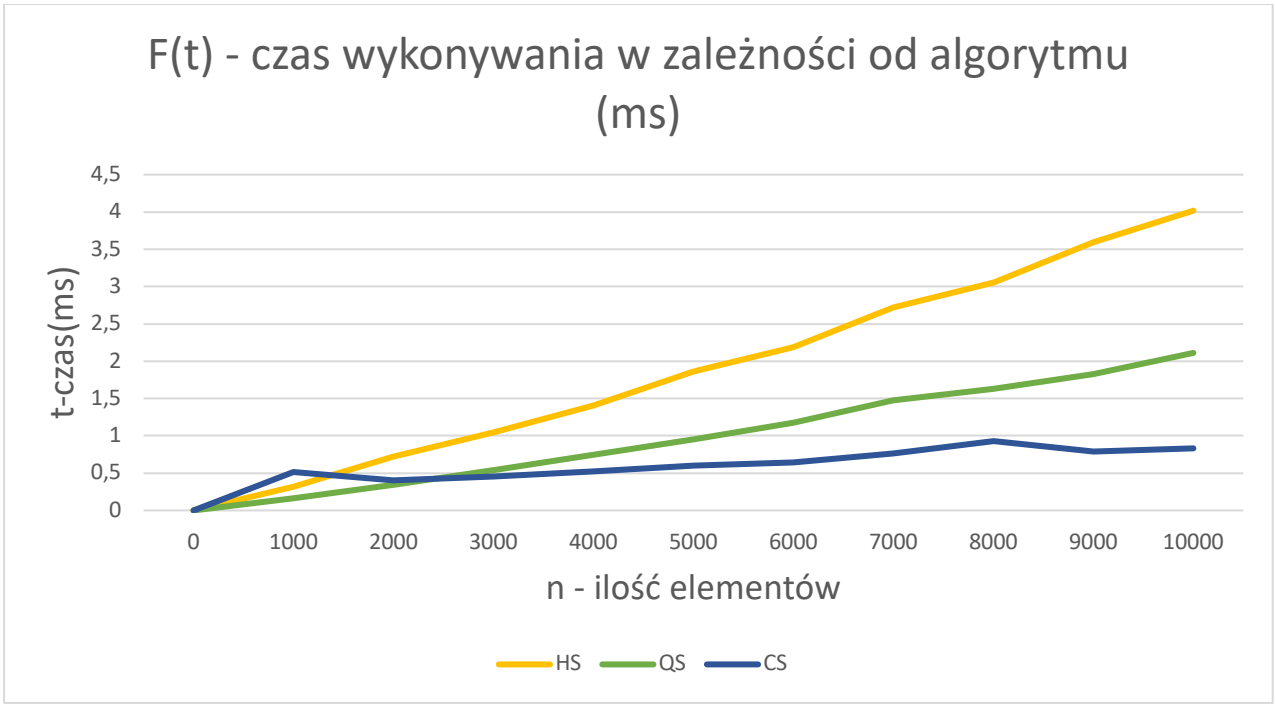
ALGORYTMY I STRUKTURY DANYCH
semestr letni, rok akademicki 2021/2022
piątek 9:45-11:15

1) Andrei Kartavik, Indeks: 153925
2) Ivan Kaliadzich, Indeks: 153936

Zadanie 2

	F(t) - czas wykonywania w zależności od algorytmu (ms)						
n-ilość elementów	SS	IS	BS	HS	MS	QS	CS
0	0	0	0	0	0	0	0
1000	1,837	4,217	6,802	0,317	7,475	0,163	0,511
2000	6,979	16,83	27,245	0,723	14,508	0,345	0,403
3000	15,465	36,897	59,878	1,046	23,007	0,543	0,453
4000	27,902	59,294	106,294	1,411	30,271	0,745	0,52
5000	41,829	75,431	161,748	1,863	38,551	0,956	0,598
6000	56,7	110,6	232,205	2,184	46,793	1,177	0,645
7000	76,51	149,616	311,946	2,722	44,628	1,472	0,76
8000	93,05	195,213	426,779	3,055	55,268	1,628	0,929
9000	124,514	241,811	518,118	3,591	61,865	1,831	0,787
10000	162,529	299,435	603,671	4,017	67,256	2,111	0,833




F(t) - czas wykonywania w zależności od algorytmu (ms)
t-czas(ms)
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
n - ilość elementów
HS
QS
CS

Zadanie 2

1) Efektywne są - HS, QS, CS, MS.
2) Złożoność czasowa(Przypadek najgorszy, najlepszy i średni, Wrażliwość na dane), Złożoność pamięciowa, Zachowanie naturalne, Stabilność.
3) Grupy:

1. Działają w miejscu nie działają w miejscu

    - Działają w miejscu
      SS,IS,BS,QS

    - Nie działają w miejscu
      MS,HS,CS

2. Wrażliwe na dane nie wrażliwy na dane

    - Wrażliwe na dane
      IS, QS, HS, CS

    - Nie wrażliwe na dane
      SS, BS, MS

3. Stabilne nie stabilne

    - Stabilne
      IS, BS, MS, CS

    - Nie stabilne
      SS, QS, HS

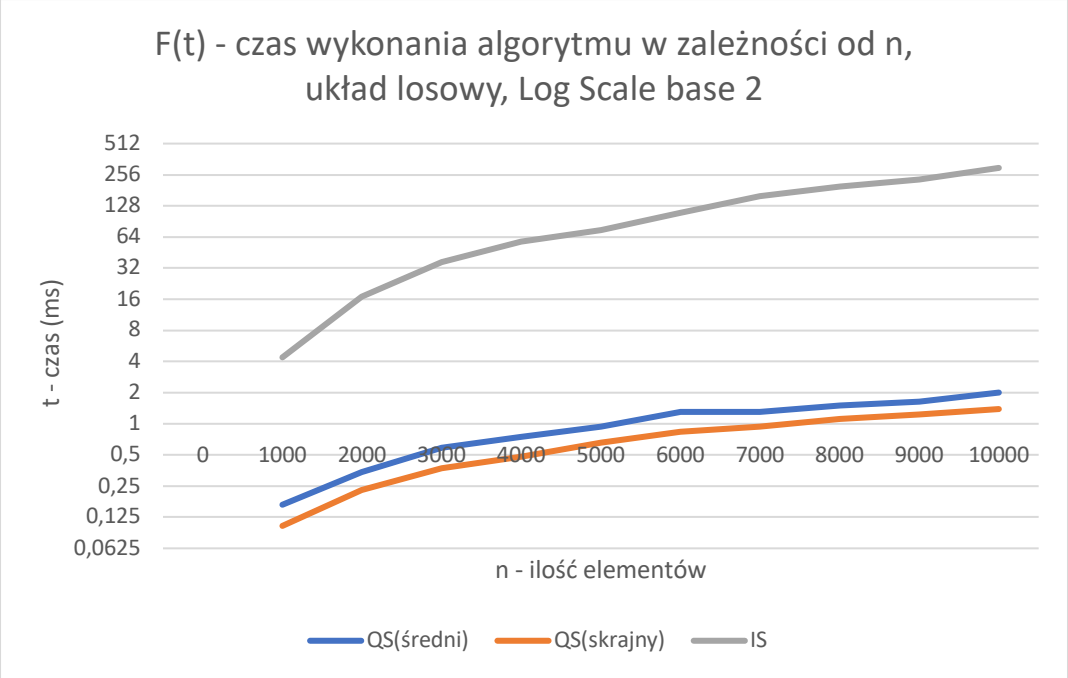
4. Rekurencyjne nie rekurencyjne

    - Rekurencyjne
      QS,HS,MS

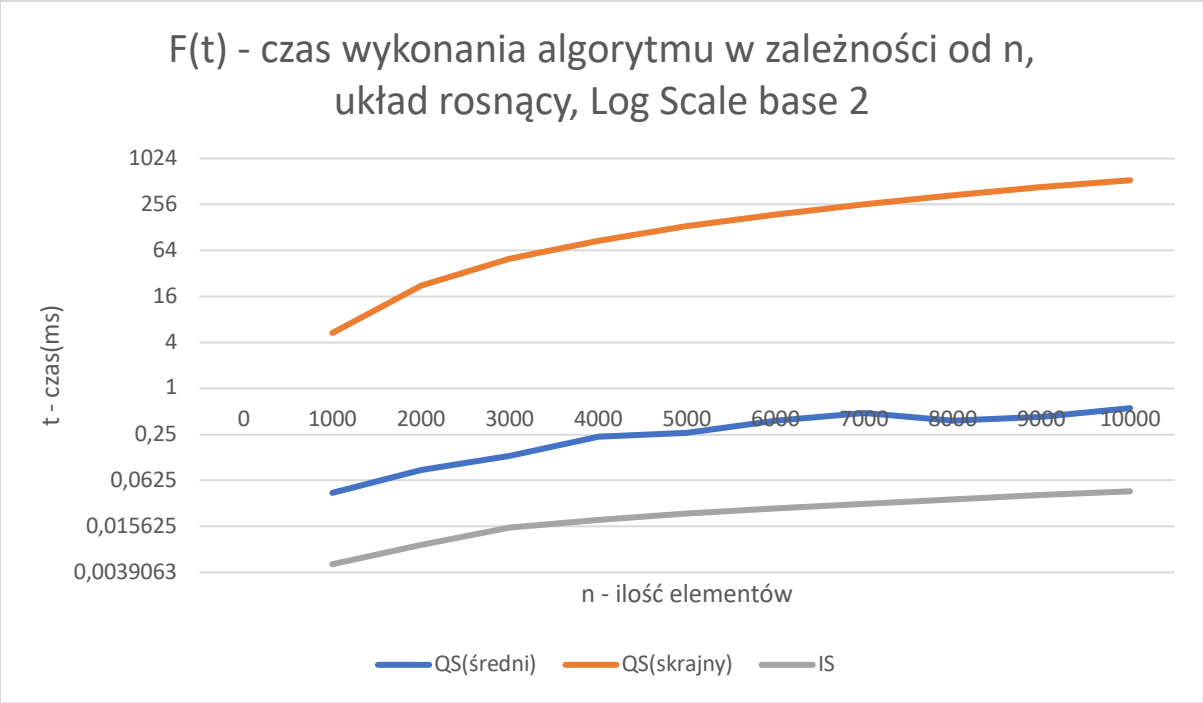
    - Nie rekurencyjne
      IS,SS,BS,CS

Zadanie 3

F(t) - czas wykonania algorytmu w zależności od n układ losowy (ms)			
n - ilość elementów	QS(średni)	QS(skrajny)	IS
0	0	0	0
1000	0,165	0,103	4,387
2000	0,343	0,229	16,834
3000	0,585	0,371	36,444
4000	0,744	0,479	58,003
5000	0,944	0,659	74,495
6000	1,302	0,832	109,735
7000	1,31	0,935	157,943
8000	1,498	1,116	196,366
9000	1,642	1,233	230,938
10000	2,005	1,389	297,981



F(t) - czas wykonania algorytmu w zależności od n układ rosnący (ms)			
n - ilość elementów	QS(średni)	QS(skrajny)	IS
0	0	0	0
1000	0,043	5,33	0,005
2000	0,085	22,139	0,009
3000	0,13	49,771	0,015
4000	0,235	85,046	0,019
5000	0,262	132,792	0,023
6000	0,376	190,612	0,027
7000	0,473	258,926	0,031
8000	0,382	337,012	0,035
9000	0,423	431,229	0,04
10000	0,552	531,703	0,045



Zadanie 3

1. Dla układu losowego QS(skrajny) był trochę szybszy niż QS(średni) z tego powodu, że dla QS(skrajny) tablica liczb w całkiem losowym porządku jest przypadkiem najlepszym. Dla układu rosnącego to miało dużo znaczenie. Z tego powodu, że QS(skrajny) bierze ostatni element, który, w przypadku układu rosnącego jest największym elementem w całej tablice, to dlatego algorytm podzieli tablicę na dwie, gdzie z lewej strony będzie ta cała tablica, ale rozmiaru n-1. Tak będzie dla każdego następnego podziału tablicy, co sprawia, że to jest przypadek najgorszy danego algorytmu - O(n^2).

2. Dla QS(Średni) przypadek najlepszy to taki przypadek, gdy pivot jest medianą (np. Układ posortowany), a przypadek najgorszy - gdy pivot element najmniejszy lub największy. Dla QS(Skrajny) przypadek najlepszy to tablica w całkiem losowym porządku, a przypadek najgorszy to tablica posortowana(rosnąco lub malejąco).

3. Podział przez medianę. Zaleta: bardzo przyspieszy algorytm, Wada: najgorszy przypadek poszukiwania mediany - O(n^2). Podział przez medianę trzech. (Bierze pierwszy, średni i ostatni element, sortuje ich i pivot w tym przypadku - to element średni). Zaleta: jest szansa przyspieszyć algorytm i nie wybrać element skrajni, Wada: jest szansa, że każdy z trzech elementów będzie elementem skrajnym.

4. O(logn)

5. QS w ogóle jest lepszy niż IS, dlatego że mają taki samy przypadek najgorszy, ale QS ma lepszy przypadek średni. Ale to jeszcze bardzo zależy od sposobu wybrania pivotu. Bo jak możemy zobaczyć, QS(średni) jest szybszy dla dwóch układów, ale QS(skrajny) bardzo przygrywa w przypadku, gdy zachodzi o "zachowaniu naturalnym".

6. *IS* - Zalety: działa w miejscu, dobry przypadek najlepszy, ma zachowanie naturalne, jest stabilny, prosty w implementacji. Wady: przypadek najgorszy - O(n^2), jest wrażliwy na dane.

SS - Zalety: działa w miejscu, nie jest wrażliwy na układ danych, prosty w implementacji. Wady: przypadek najlepszy = przypadek najgorszy = przypadek średni = O(n^2), bardzo słabe zachowanie naturalne, nie jest stabilny, nie ma zachowania naturalnego.

BS - Zalety: działa w miejscu, jest stabilny, nie jest wrażliwy na układ danych, bardzo prosty w implementacji. Wady: przypadek najlepszy = przypadek najgorszy = przypadek średni = O(n^2), nie ma zachowania naturalnego.

QS - Zalety: działa w miejscu, dobry przypadek optymistyczny (nlogn), dobry przypadek średni (nlogn), ma zachowanie naturalne (zależy). Wady: nie jest stabilny, potrzebna jest pamięć na rekurencję, jest bardzo wrażliwy na dane, przypadek najgorszy (O(n^2)), skuteczność zależy od wybrania punktu podziału.

MS - Zalety: Przypadek najgorszy = przypadek średni = przypadek najlepszy = O(nlogn), jest stabilny, nie jest wrażliwy na dane. Wady: nie działa w miejscu O(n), nie ma zachowania naturalnego.

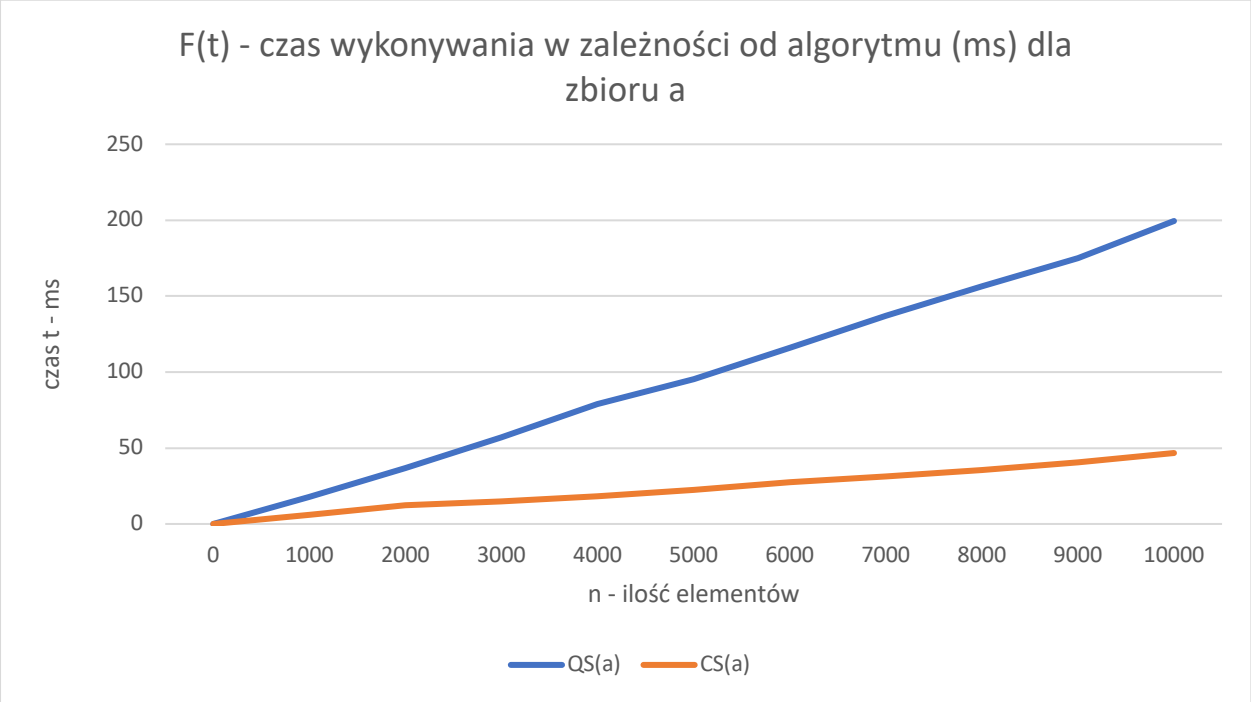
HS - Zalety: działa w miejscu, przypadek najgorszy = przypadek średni = O(nlogn), przypadek najlepszy O(n). Wady: jest wrażliwy na dane, nie jest stabilny, budowa zajmuje czas (przypadek najgorszy - O(logn), przypadek najlepszy O(1)), jest nie prosty w implementacji.

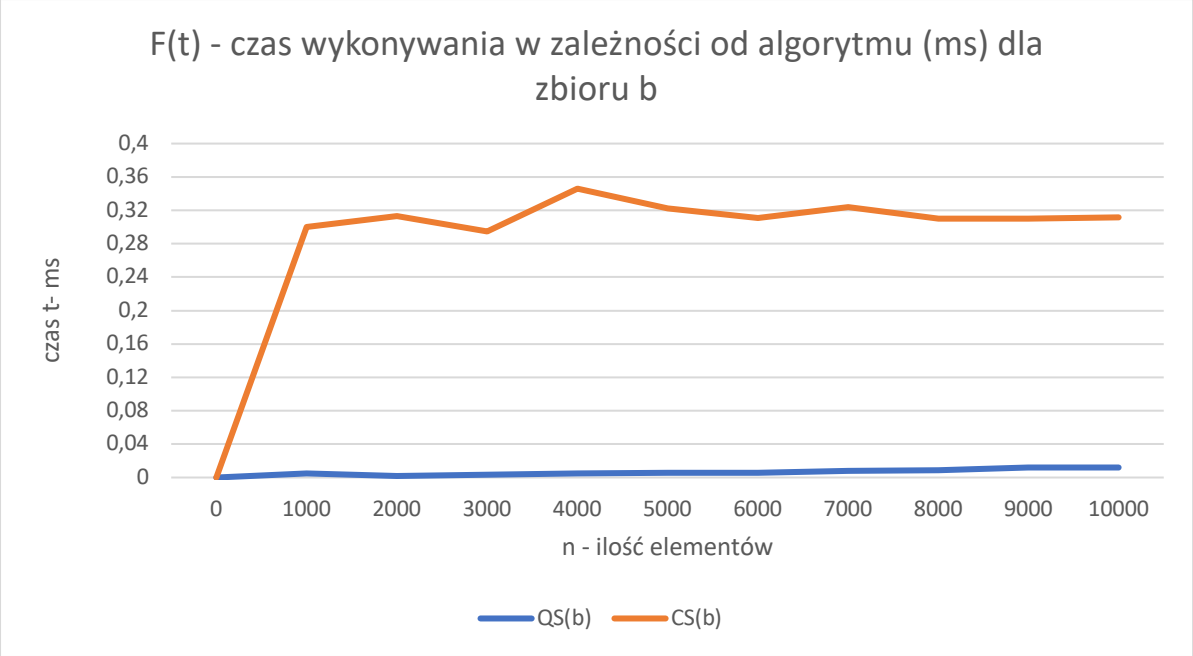
CS - Zalety: przypadek najlepszy = przypadek średni = przypadek najgorszy = O(n+k), jest stabilny. Wady: sortuje liczby z wąskiego zakresu, nie działa w miejscu, jest bardzo wrażliwy na dane.

7. Żeby wybrać metodę sortowania trzeba zwrócić uwagę na czas wykonywania algorytmu sortowania, pamięć, którą algorytm sortowania używa i liczby, które trzeba posortować (liczby dodatnie, liczby ujemne, ilość liczb, maksymalne i minimalne możliwe liczby). Na przykład:
Najszybszymi metodami sortowania są QS i CS. W takim razie najlepiej używać ich.
Ale CS działa tylko na wąskim zakresie liczb i jest lepszy, gdy różnica między największym i najmniejszym elementami jest niezbyt duża.
QS jest uniwersalną metodą sortowania, która jest zawsze szybka i nie używa wielu pamięci, ale jest wrażliwa na punk podziału tablicy.

Zadanie 4

F(t) - czas wykonywania w zależności od algorytmu (ms)				
n - ilość elementów	QS(a)	CS(a)	QS(b)	CS(b)
0	0	0	0	0
1000	17,669	5,984	0,005	0,3
2000	36,76	12,273	0,002	0,313
3000	56,976	14,696	0,003	0,295
4000	78,798	18,074	0,005	0,346
5000	95,176	22,378	0,006	0,322
6000	116,143	27,662	0,006	0,311
7000	137,11	31,201	0,008	0,324
8000	156,508	35,709	0,009	0,31
9000	175,229	40,449	0,012	0,31
10000	199,475	46,714	0,012	0,312





Zadanie 4

1) Dla układu a ([1:100*n]) CS był szybszy z tego powodu, że jego złożoność czasowa - O(n+k), gdy dla QS(średni) przypadek najgorszy to O(n^2), przypadek najlepszy i średni - O(nlogn), co jest większe niż w przypadku CS.

2) Dla układu b ([1:0.01*n]) QS był szybszy, z powodu wrażliwości na układ danych dla CS. Chociaż elementów do sortowania jest niewiele, złożoność czasowa CS - O(n+k), zależy od k – największego elementu w tablice. Z tego powodu, że układ danych jest całkiem losowy, to największy element może być 10000, co może bardzo wpłynąć na czas wykonywania algorytmu.