SPRAWOZDANIE

PAMSI Lab
pn 9:15-11:00
Krystian Lema 218453

Temat: Sortowanie szybkie, sortowanie przez scalanie

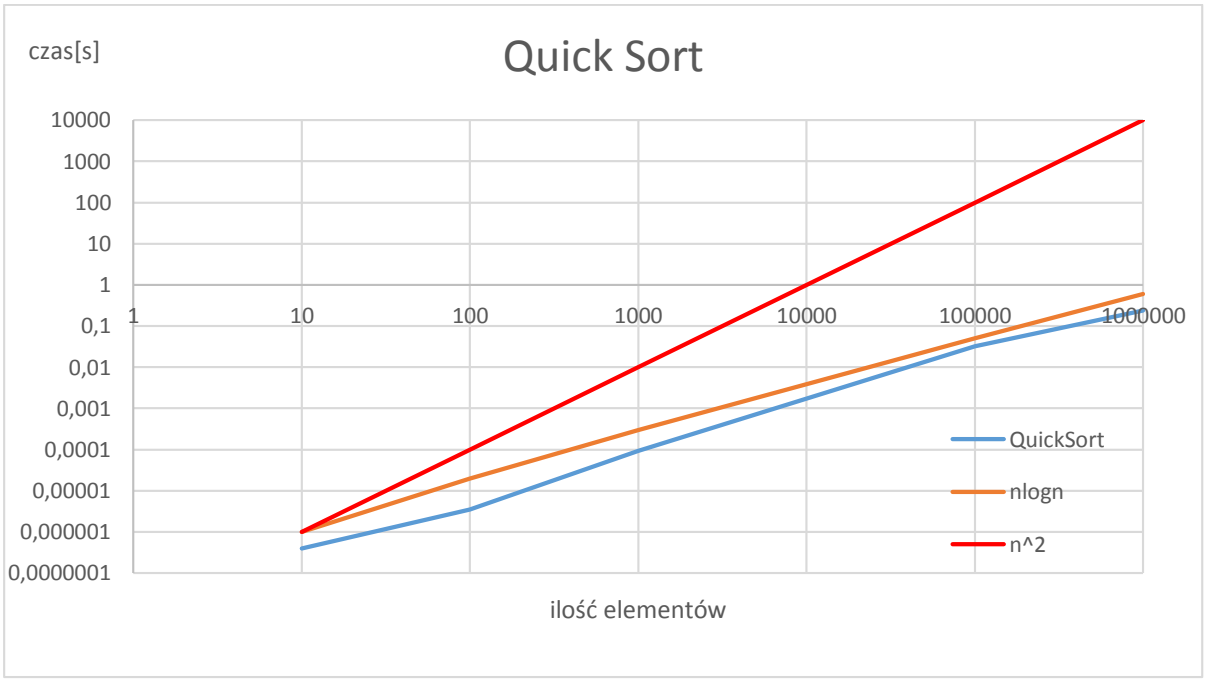
1. Wybór struktury danych do przechowywaniu rozmiarów problemów

Na strukturę danych przechowująca rozmiary problemów wybrałem Listę. W pierwszej kolejności odrzuciłem stos, ponieważ na stos trzeba by było wkładać elementy w odwrotnej kolejności aby przy ściąganiu ze stosu otrzymać rozmiary w poprawnej kolejności. Drugim powodem było to, że pozostałe struktury danych (kolejka i stos) były zaimplementowane na podstawie wcześniej zrobionej listy. Tak więc były stworzone na jej bazie. Wybór którejkolwiek z tych dwóch struktur wiązałby się z tym, ze dodatkowo trzeba sprawdzić poprawność działania samej listy. Ponadto lista daje możliwość w łatwy sposób na dodanie elementów i pobranie dowolnego z nich.
Dodatkowym problemem okazało się być to, że lista była zaimplementowana nie jako szablon a jako typ string. Aby nie zmieniać implementacji listy jako rozmiary problemów dodałem na listę liczby w postaci ciągu znaków string, a następnie przy wczytywaniu z listy funkcją atoi() parsowałem typ string na typ int.

2. Pomiary czasu działania sortowania szybkiego w różnych przypadkach

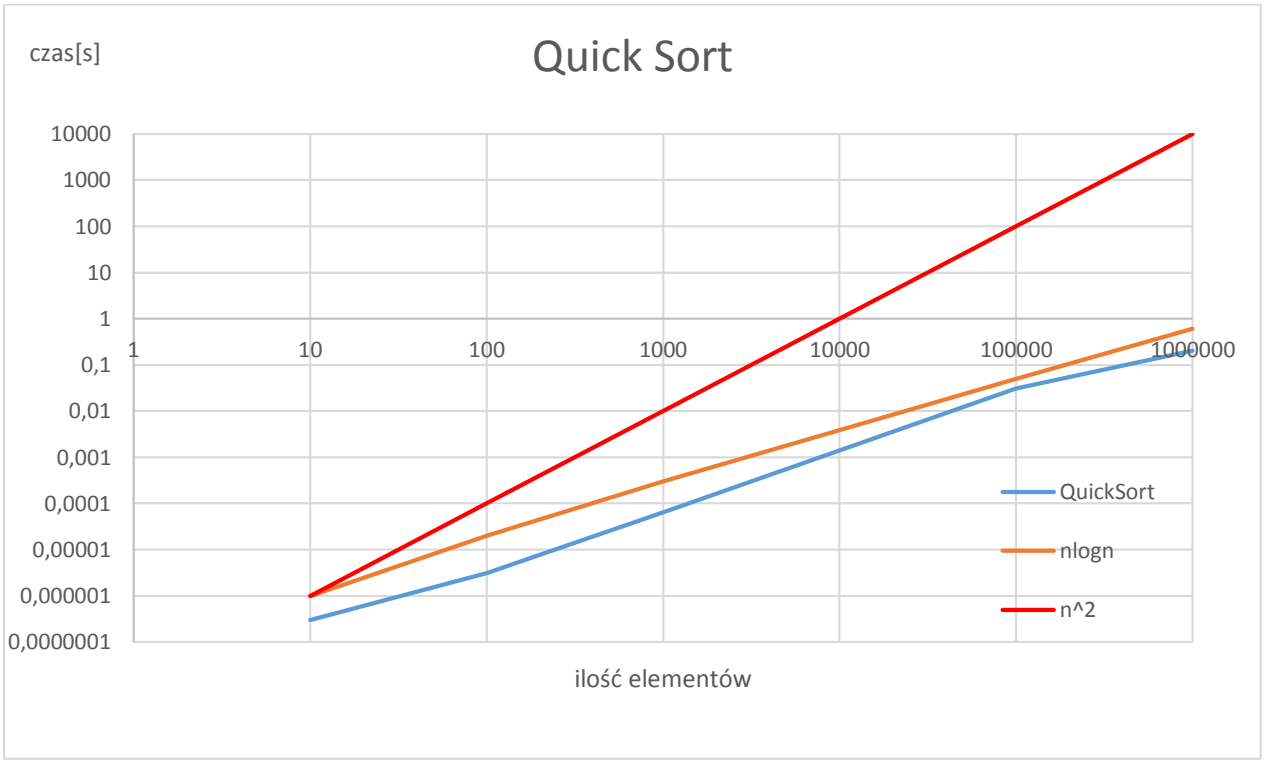
a) tablica z losowymi wartościami, piwot na lewej stronie

pomiar/n	10	100	1000	100000	1000000
1	0,000001s	0,000007s	0,000120s	0,030951s	0,237448s
2	0,000001s	0,000005s	0,000090s	0,034628s	0,279900s
3	0s	0,000004s	0,000087s	0,033155s	0,298774s
4	0s	0,000003s	0,000103s	0,030662s	0,215829s
5	0s	0,000002s	0,000085s	0,036885s	0,320881s
6	0s	0,000003s	0,000084s	0,029335s	0,180748s
7	0,000001s	0,000003s	0,000085s	0,033442s	0,177909s
8	0,000001s	0,000003s	0,000108s	0,034415s	0,196897s
9	0s	0,000003s	0,000084s	0,025352s	0,179320s
10	0s	0,000002s	0,000083s	0,030689s	0,308671s
średnia	0,0000004s	0,000004s	0,000093s	0,031951s	0,239638s



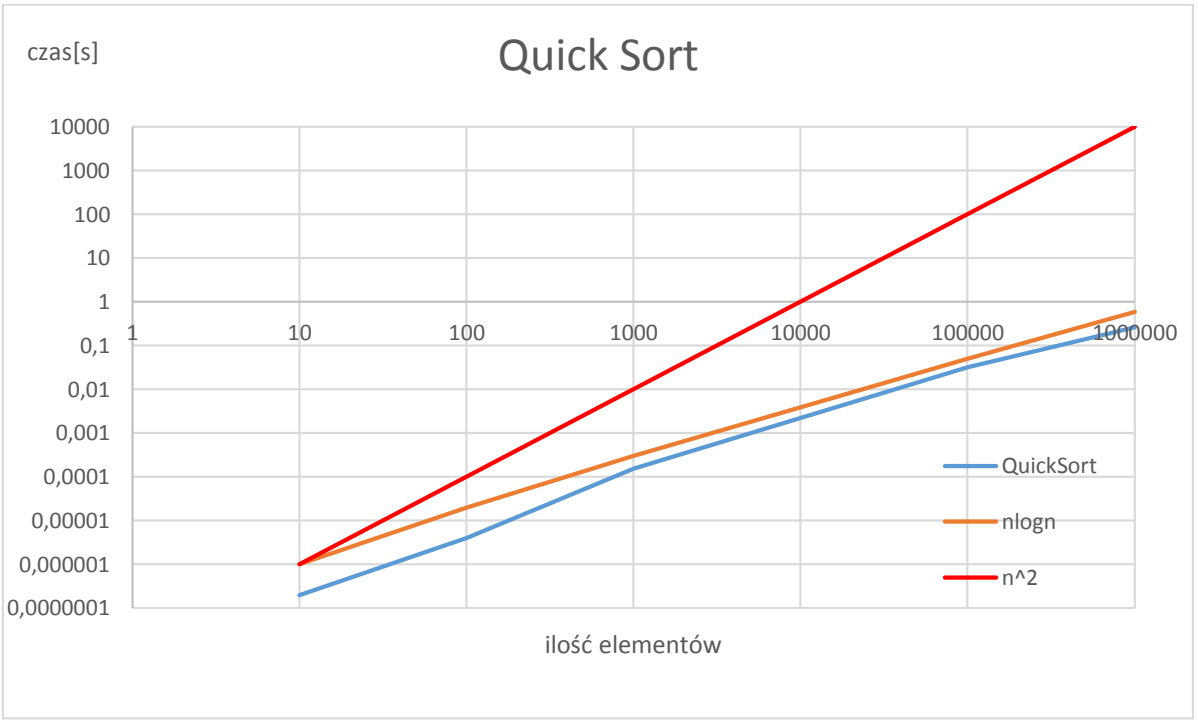
b) tablica z losowymi wartościami, piwot na środku

pomiar/n	10	100	1000	100000	1000000
1	0s	0,000006s	0,000080s	0,029550s	0,006983s
2	0s	0,000005s	0,000040s	0,032187s	0,228173s
3	0s	0,000003s	0,000030s	0,031190s	0,042462s
4	0,000001s	0,000003s	0,000020s	0,025578s	0,268304s
5	0s	0,000003s	0,000080s	0,026882s	0,241685s
6	0,000001s	0,000002s	0,000050s	0,029417s	0,265729s
7	0s	0,000002s	0,000020s	0,034669s	0,238367s
8	0s	0,000003s	0,000148s	0,030782s	0,269019s
9	0s	0,000002s	0,000141s	0,027833s	0,239470s
10	0,000001s	0,000002s	0,000040s	0,041373s	0,256581s
średnia	0,0000003s	0,000003s	0,000065s	0,030946s	0,205677s



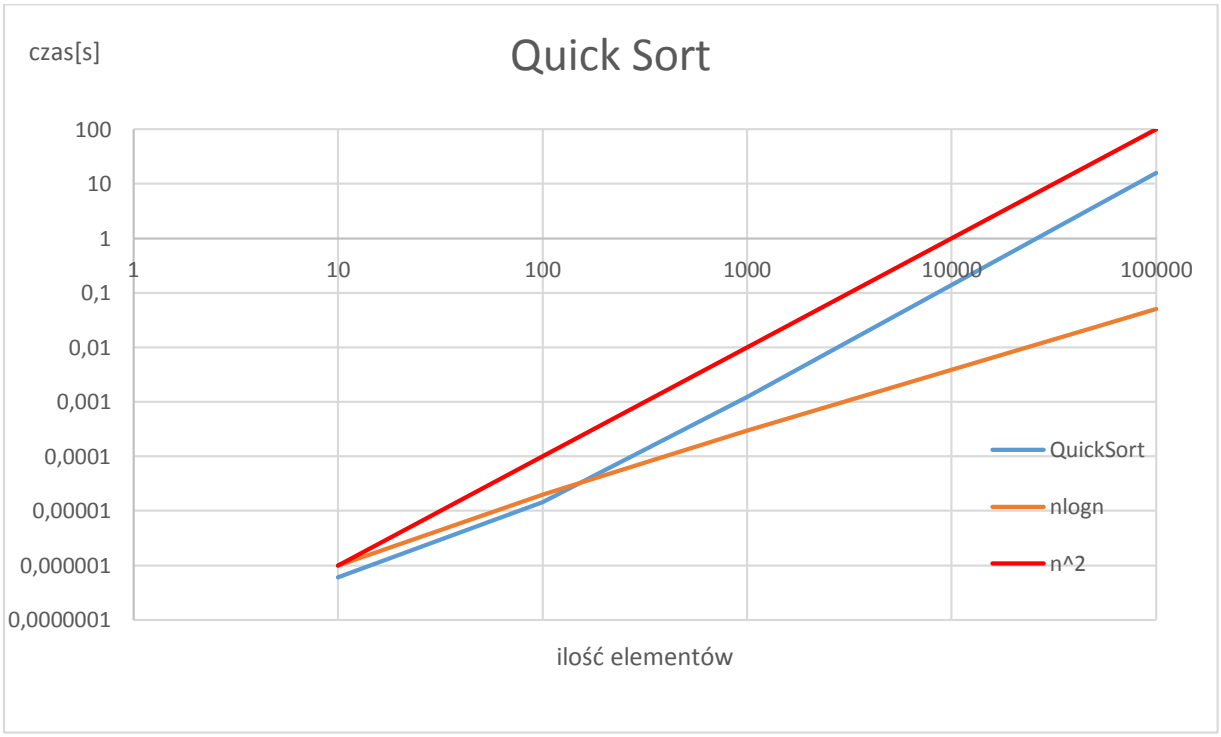
c) tablica z losowymi wartościami, piwot na prawej stronie

pomiar/n	10	100	1000	100000	1000000
1	0s	0,000008s	0,000093s	0,030100s	0,237192s
2	0,000001s	0,000006s	0,000088s	0,033305s	0,274481s
3	0s	0,000004s	0,000782s	0,031393s	0,251325s
4	0s	0,000004s	0,000086s	0,029246s	0,284805s
5	0s	0,000003s	0,000084s	0,032652s	0,259959s
6	0s	0,000003s	0,000083s	0,027782s	0,281542s
7	0s	0,000003s	0,000082s	0,031639s	0,255163s
8	0s	0,000003s	0,000082s	0,035259s	0,271613s
9	0s	0,000003s	0,000082s	0,028701s	0,250332s
10	0,000001s	0,000003s	0,000082s	0,041285s	0,243303s
średnia	0,0000002s	0,000004s	0,000154s	0,032136s	0,260972s



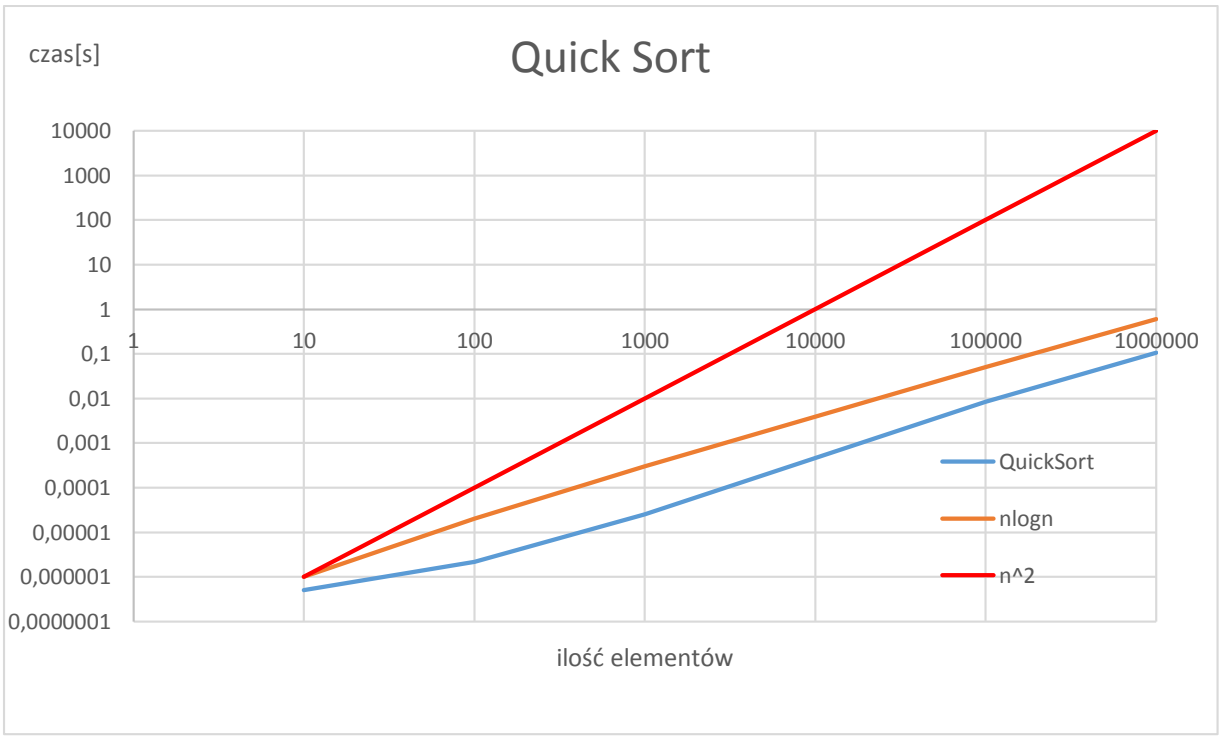
d) tablica z posortowanymi w odwrotnej kolejności wartościami, piwot na lewej stronie

pomiar/n	10	100	1000	100000
1	0,000002s	0,000018s	0,001257s	12,4794s
2	0s	0,000015s	0,001213s	12,6598s
3	0,000001s	0,000014s	0,001299s	14,9836s
4	0s	0,000014s	0,001288s	17,0400s
5	0s	0,000014s	0,001211s	17,0030s
6	0,000001s	0,000014s	0,001218s	17,1164s
7	0s	0,000014s	0,001300s	17,2128s
8	0s	0,000014s	0,001243s	17,0276s
9	0,000001s	0,000013s	0,001080s	17,0300s
10	0,000001s	0,000014s	0,001197s	17,2230s
średnia	0,0000006s	0,000014s	0,001231s	15,977560s



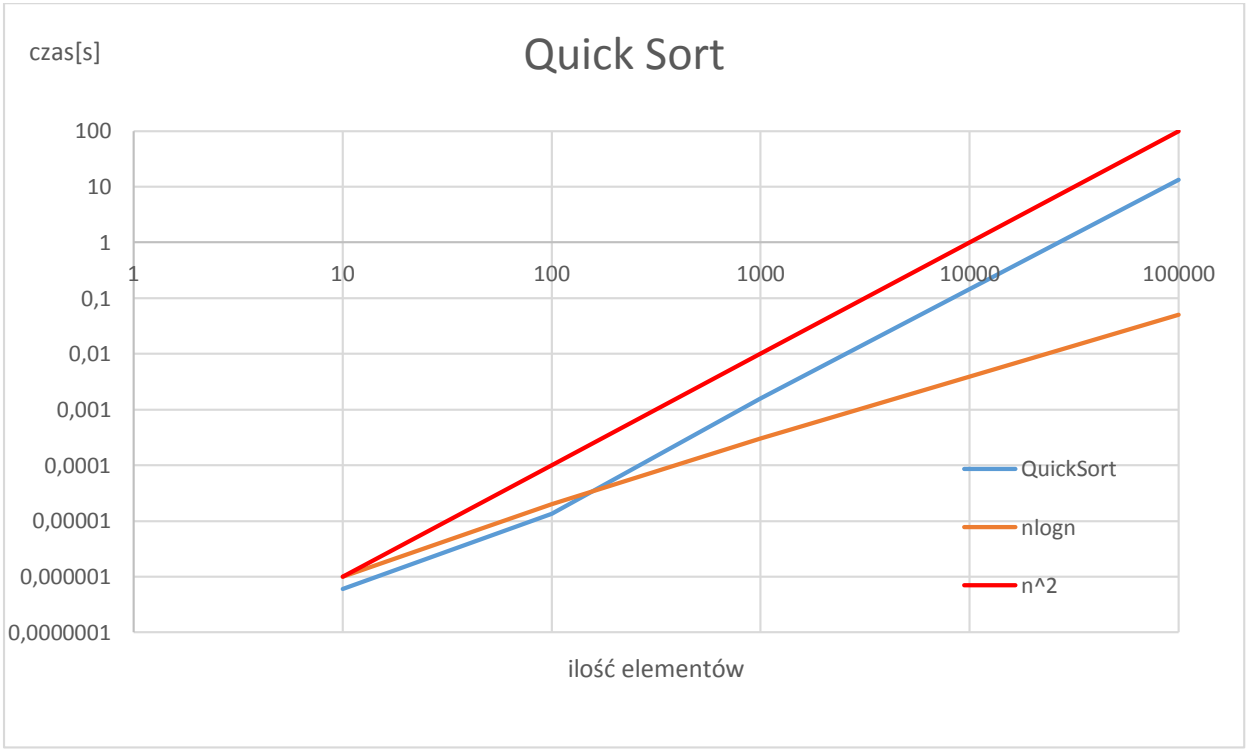
e) tablica z posortowanymi w odwrotnej kolejności wartościami, piwot na środku

pomiar/n	10	100	1000	100000	1000000
1	0,000002s	0,000003s	0,000050s	0,014857s	0,115418s
2	0,000001s	0,000003s	0,000025s	0,010795s	0,108662s
3	0,000001s	0,000002s	0,000025s	0,007650s	0,104843s
4	0s	0,000002s	0,000025s	0,004418s	0,100679s
5	0s	0,000002s	0,000024s	0,004037s	0,106924s
6	0s	0,000002s	0,000035s	0,008682s	0,104664s
7	0,000001s	0,000002s	0,000042s	0,012363s	0,111819s
8	0s	0,000002s	0,000025s	0,011934s	0,105802s
9	0s	0,000002s	0,000025s	0,004287s	0,107090s
10	0s	0,000002s	0,000025s	0,004344s	0,107622s
średnia	0,0000005s	0,000002s	0,000025s	0,008337s	0,107352s



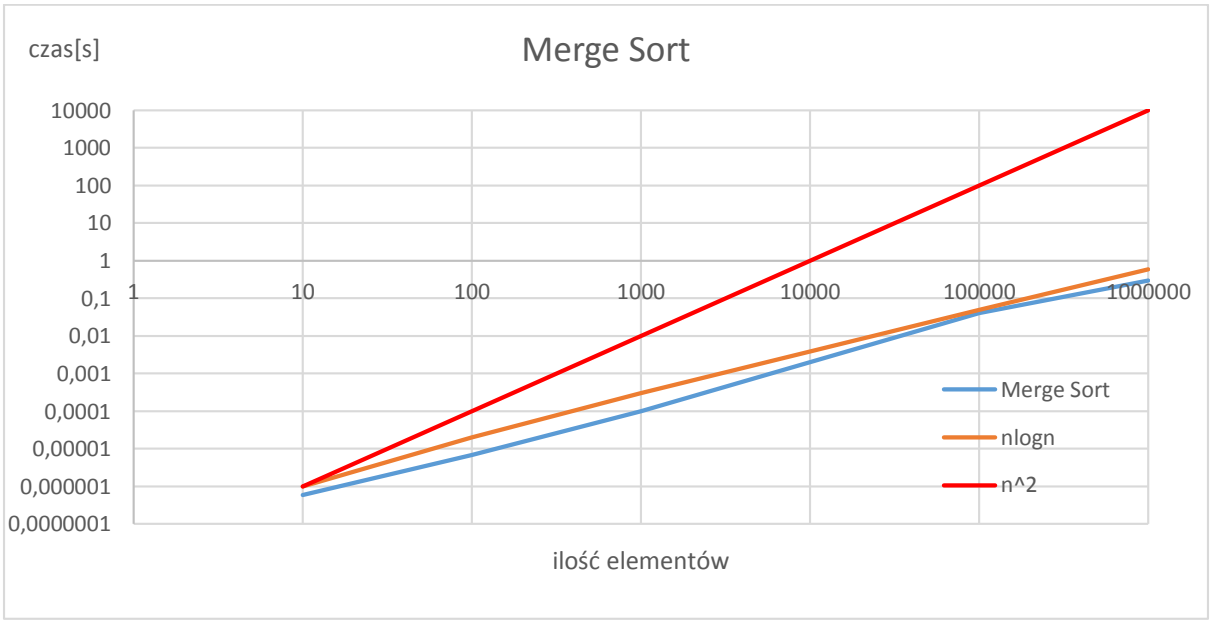
f) tablica z posortowanymi wartościami, piwot na prawej stronie

pomiar/n	10	100	1000	100000
1	0,000003s	0,000015s	0,001176s	12,4177s
2	0,000001s	0,000014s	0,001136s	12,7184s
3	0,000001s	0,000014s	0,001140s	12,5446s
4	0,000001s	0,000013s	0,001269s	12,5260s
5	0s	0,000013s	0,001248s	12,5240s
6	0s	0,000013s	0,001137s	12,7188s
7	0s	0,000013s	0,001401s	12,5808s
8	0s	0,000013s	0,004716s	12,5443s
9	0s	0,000013s	0,001215s	15,9658s
10	0s	0,000013s	0,001185s	17,0002s
średnia	0,0000006s	0,000013s	0,001562s	13,354060s



3. Pomiary czasu działania sortowania przez scalanie

pomiar/n	10	100	1000	100000	1000000
1	0,000001s	0,000018s	0,000113s	0,044591s	0,293536s
2	0,000001s	0,000007s	0,000130s	0,040027s	0,324848s
3	0s	0,000006s	0,000117s	0,045736s	0,292858s
4	0,000001s	0,000006s	0,000118s	0,043625s	0,296666s
5	0s	0,000005s	0,000104s	0,033495s	0,290761s
6	0s	0,000005s	0,000104s	0,039882s	0,293196s
7	0,000001s	0,000006s	0,000104s	0,038292s	0,294378s
8	0s	0,000005s	0,000104s	0,046437s	0,292317s
9	0,000001s	0,000005s	0,000102s	0,039981s	0,326653s
10	0,000001s	0,000005s	0,000125s	0,034486s	0,278182s
średnia	0,0000006s	0,000007s	0,000101s	0,040655s	0,298340s



4. Analiza wyników i wnioski

4.1. Quick Sort

Przy sortowaniu szybkim, kiedy do posortowania mamy tablicę z losowymi nieposortowanymi elementami, których rozkład jest przynajmniej w przybliżeniu równomierny ustawienie piwota praktycznie nie ma znaczenia. Jak wynika z wykresów czas takiego sortowania w przypadku piwota po lewej i prawej stronie jak i na środku zawsze był podobny, a złożoność O(n) była w przybliżeniu nlogn.
W przypadku skrajnym kiedy do posortowania jest już posortowana tablica, ale np. w odwrotnej kolejności kiedy piwot ustawiony jest na środku złożoność obliczeniowa jest taka sama jak w przypadku losowych wartości. Sprawa zmienia się drastycznie kiedy sortujemy posortowaną tablicę ale piwot znajduję się na jednym bądź drugim końcu. Wtedy mamy do czynienia z przypadkiem pesymistycznym, w którym teoretycznie złożoność tego algorytmu wynosi n^2. Z pomiarów i wykresów w tych przypadkach również wyszło zgodnie z założeniem $O(n) = n^2$.
Analizując wszystkie te przypadku można wnioskować, że najlepszym sposobem jest wybór piwota zawsze na środku. Wtedy zawsze mamy do czynienia z przypadkiem przeciętnym którego złożoność wynosi nlogn i jest to jeden z lepszych czasów uzyskiwany przez algorytmy sortujące.

4.2 Merge Sort

Po przeprowadzeniu testów sortowania przez scalanie na losowym zestawie wartości wynika, że złożoność obliczeniowa tego algorytmu wynosi zgodnie z teorią nlogn. Jest to dobry czas jak na algorytm sortowania i ciężki do pobicia, a sam algorytm jest prosty do implementacji i wykorzystuje podstawowy typ danych jakim jest tablica.