Politechnika Wrocławska	Projektowanie algorytmów i metody sztucznej inteligencji	Data ćwiczenia: 6.04.2017 Data sprawozdania: 12.04.2017
	Badanie czasu algorytmu sortowania szybkiego	Rafał Borysionek, 226262 Automatyka i Robotyka Wydział Elektroniki
Prowadzący: Mgr inż. Andrzej Wytyczak-Partyka		Grupa: E02-18o

1. Cel ćwiczenia

Celem ćwiczenia jest zbadanie złożoności obliczeniowej algorytmu sortowania szybkiego pod kątem wyboru osi (pivota) oraz danych wejściowych do sortowania.

2. Sposób badania

Zaimplementowany algorytm badano dla różnych danych wejściowych, którymi były:
-tablica wypełniona losowymi liczbami z przedziału 0-wielkość tablicy,
-tablica posortowana rosnąco,
-tablica posortowana malejąco.

Ponadto, badania przeprowadzono dla różnych doborów miejsca „pivotów" w algorytmie:
-pivot wybierany zawsze na pierwszym miejscu problemu,
-pivot zawsze na środku tablicy,
-pivot zawsze ostatni w badanej tablicy.

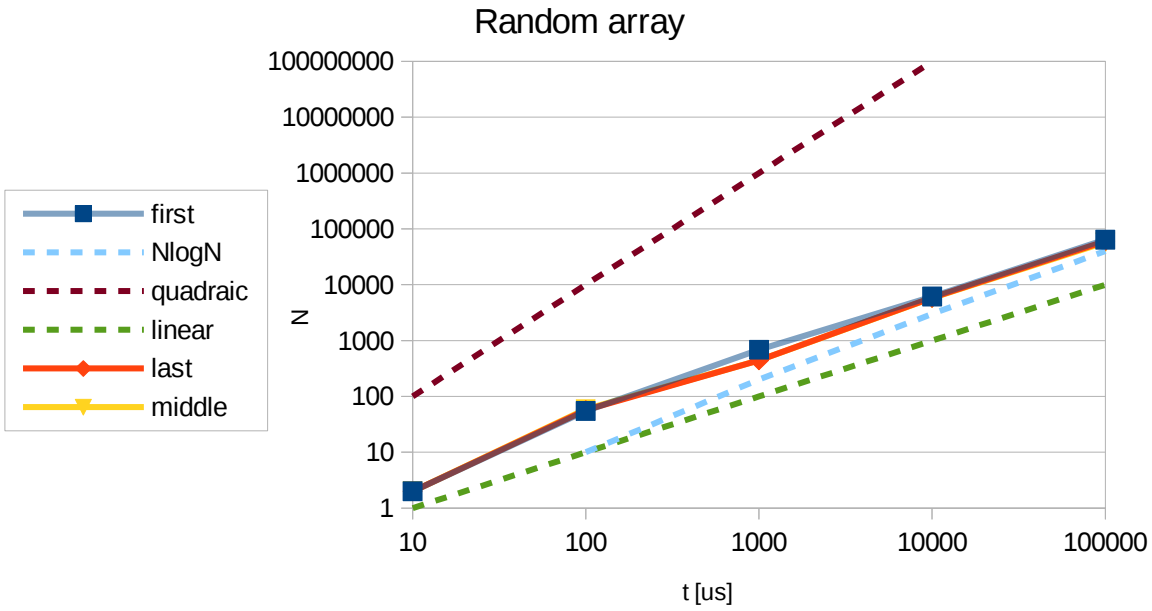
W celu kontroli błędów w algorytmie, zastosowano pętlę sprawdzającą poprawność mierzonego sortowania, co przedłużyło czas, lecz nie zmieniło złożoności obliczeniowej (sprawdzenie to O(n) a sam algorytm w najlepszym przypadku to O(n log n)). Wykonano po 20 pomiarów dla każdego N i wyciągnięto średnią arytmetyczną.

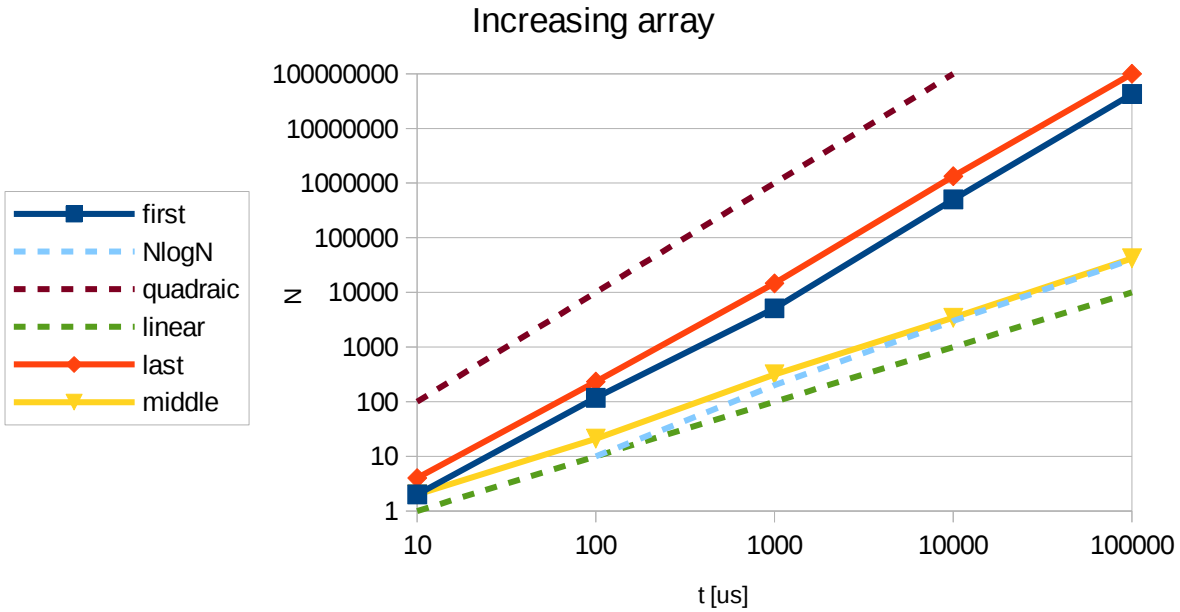
3. Wyniki pomiarów

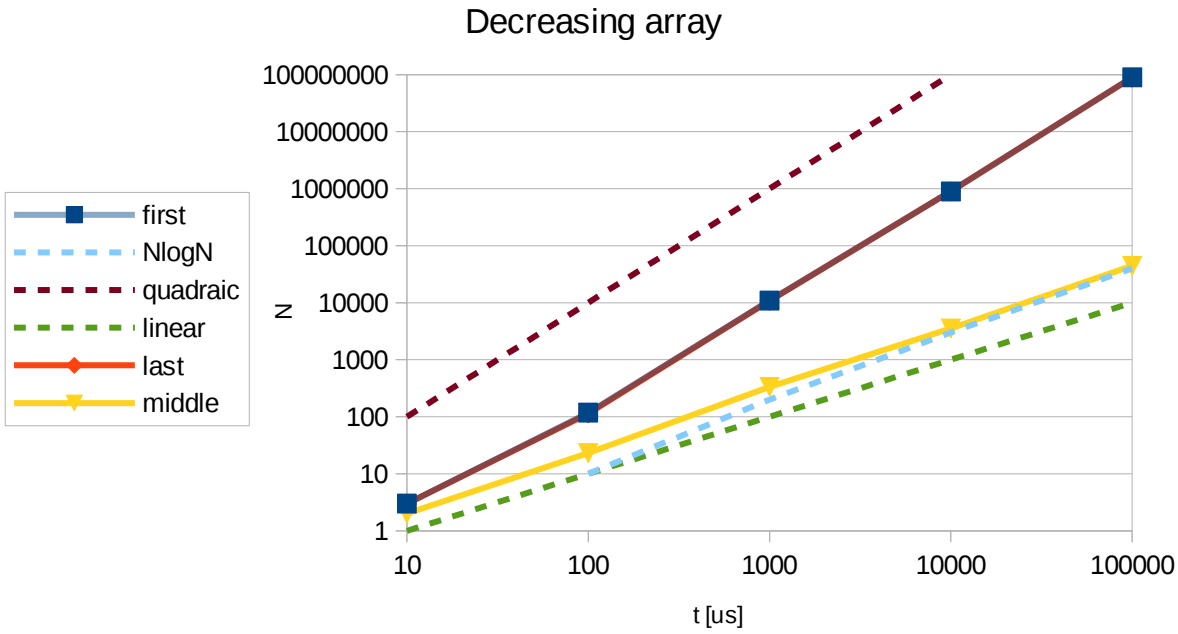
N- liczba elementów w tablicy
Popełniany błąd pomiaru programowego wynosi ±1 mikrosekunda
P- pivot jest zawsze pierwszy w tablicy
Ś- pivot jest wybierany na środku tablicy
O- pivot jest zawsze ostatni w tablicy
Czas jest podawany w mikrosekundach

Ilość elementów	Tablica losowa			Tablica rosnąca			Tablica malejąca		
N	P	Ś	O	P	Ś	O	P	Ś	O
10	2	2	2	2	4	2	3	2	3
100	55	59	57	118	21	233	118	21	114
1000	685	442	442	5104	314	14595	10973	335	10842
10000	6134	5780	5847	500550	3423	1332180	894219	3523	892582
100000	64236	57469	60436	42426434	41797	119964118	89862691	44294	89215520

4. Wykresy







5. Wnioski

1. Wykres tablicy o losowo ułożonych elementach do sortowania:

Mozna zauważyć, że dla tego przypadku wybór miejsca pivota praktycznie nie ma wpływu na złożoność obliczeniową. Z otrzymanej charakterystyki można stwierdzić że średni przypadek działania algorytmu quicksot to O(nlog(n)).

Nieliniowość charakterystyki można uzasadnić tym, że losowo wybrana tablica mogła być częściowo posortowana i wtedy czas wykonywania algorytmu zwiększał się. Ponadto przyczyną mogło być także nadmierne obciążenie procesora podczas wykonywania programu (w tle działały inne programy które także potrzebowały zasobów).

2. Wykres tablicy posortowanej rosnąco :

W tym przypadku dobór pivota miał duży wpływ na złożoność obliczeniową. Przy miejscu średnim, algorytm sprawdzał tylko równe wartości tablicy. Problem był dzielony na dwie rózne części (n/2 i n/2). Wówczas ten algorytm miał złożoność O(nlog(n)).

Podczas skrajnych doborów osi, algorytm wybierał najmniejsze, lub największe wartości pivotów. Wówczas podproblemy były dzielone na 1 i n-1 elementów. Wtedy złożoność obliczeniowa wyniosła O(n^2).

3. Wykres tablicy posortowanej malejąco:

W tym przypadku dobór pivota także miał duże znaczenie. Wybór pivota po środku okazał się najlepszym wyborem. Związane jest to z faktem, że w tak posortowanej tablicy podproblemy były dzielone równo na pół (n/2 i n/2) i do takich podproblemów były wykonywane kolejne rekurencje sortowania. W tym przypadku algorytm działał ze złożonością O(nlog(n)), czyli okazał się najlepszym z możliwych przypadków (przypadek średni dla tego algorytmu ma złożoność taką samą).

Skrajny dobór osi okazał się być nieefektywny. Jego złożoność obliczeniowa wyniosła O(n^2) i była najgorszym z możliwych przypadków. Powodem takiej złożoności jest to, że problem był dzielony na 1 oraz n-1, co jest definicją najgorszego przypadku dla algorytmu quicksort. Wartości pivotów są wtedy albo zawsze najmniejsze, albo największe. Wówczas ten algorytm staje się podobny do sortowania przez wstawianie.

4. Biorąc pod uwagę powyższe wnioski można stwierdzić, że pivot najbardziej opłaca się wybierać na środku badanej tablicy, żeby zabezpieczyć algorytm przed posortowanymi danymi. Lecz aby sortowanie szybkie działało najlepiej, należy się skupić nie na miejscu wyboru osi lecz na jej wartości. W najlepszym z mozliwych przypadków, algorytm powinien zawsze wybierać oś która ma wartość średnią dla danego podproblemu. Wtedy algorytm będzie działał najszybciej i jego złożoność obliczeniowa będzie wynosić O(nlog(n)).