Piotr Józefek 272311
grupa 18 parzyste czwartek, 11:15 - 13:00
Dariusz Banasiak

Wprowadzenie

1. Sortowanie przez wstawianie (insertion sort)

Algorytm sortowania przez wstawianie będzie składał się z dwóch pętli. Pętla zewnętrzna symuluje pobieranie kart, czyli w tym wypadku elementów zbioru. Odpowiednikiem kart na ręce jest tzw. lista uporządkowana, którą sukcesywnie będziemy tworzyli na końcu zbioru (istnieje też odmiana algorytmu umieszczająca listę uporządkowaną na początku zbioru). Pętla wewnętrzna szuka dla pobranego elementu miejsca na liście uporządkowanej. Jeśli takie miejsce zostanie znalezione, to elementy listy są odpowiednio rozsuwane, aby stworzyć miejsce na nowy element i element wybrany przez pętlę główną trafia tam. W ten sposób lista uporządkowana rozrasta się. Jeśli na liście uporządkowanej nie ma elementu większego od wybranego, to element ten trafia na koniec listy. Sortowanie zakończymy, gdy pętla główna wybierze wszystkie elementy zbioru. Sortowanie odbywa się w miejscu.

Złożoność obliczeniowa w przypadku średnim wynosi: $O(n^2)$

Złożoność obliczeniowa w przypadku pesymistycznym wynosi: $O(n^2)$

2. Sortowanie przez kopcowanie (heapsort)

Podstawą algorytmu jest użycie kolejki priorytetowej zaimplementowanej w postaci binarnego kopca zupełnego. Zasadniczą zaletą kopców jest stały czas dostępu do elementu maksymalnego (lub minimalnego) oraz logarytmiczny czas wstawiania i usuwania elementów; ponadto łatwo można go zaimplementować w postaci tablicy.

Algorytm sortowania przez kopcowanie składa się z dwóch faz. W pierwszej sortowane elementy reorganizowane są w celu utworzenia kopca. W drugiej zaś dokonywane jest właściwe sortowanie.

Jest on algorytmem sortowania w miejscu, nie wymaga dodatkowych struktur danych.

Złożoność obliczeniowa w przypadku średnim wynosi: $O(n\ log\ n)$
Złożoność obliczeniowa w przypadku pesymistycznym wynosi: $O(n\ log\ n)$

3. Sortowanie Shella (Shellsort) z uwzględnieniem dwóch doborów odstępów

Algorytm sortowania działający w miejscu i korzystający z porównań elementów. Można go traktować jako uogólnienie sortowania przez wstawianie lub sortowania bąbelkowego, dopuszczające porównania i zamiany elementów położonych daleko od siebie. Na początku sortuje on elementy tablicy położone daleko od siebie, a następnie stopniowo zmniejsza odstępy między sortowanymi elementami. Dzięki temu może je przenieść w docelowe położenie szybciej niż zwykłe sortowanie przez wstawianie.

Pierwszą wersję tego algorytmu, której zawdzięcza on swoją nazwę, opublikował w 1959 roku Donald Shell. Złożoność czasowa sortowania Shella w dużej mierze zależy od użytego w nim ciągu odstępów. Wyznaczenie jej dla wielu stosowanych w praktyce wariantów tego algorytmu pozostaje problemem otwartym

Przyjęte w programie wyrazy ogólne to
wersja 1 $\frac{3^k-1}{2}$ i nie większe od $\lceil \frac{N}{3} \rceil$ - autor profesor Donald Knuth
wersja 2 $\lfloor N/2^k \rfloor$ - autor Donald Shell

Wersja Knutha

Złożoność obliczeniowa dla wersji w przypadku średnim wynosi: $O(n^{\frac{2}{3}})$

Złożoność obliczeniowa w przypadku pesymistycznym wynosi: $O(n^{\frac{2}{3}})$

Wersja Shella

Złożoność obliczeniowa dla wersji w przypadku średnim wynosi: $O(n^2)$

Złożoność obliczeniowa w przypadku pesymistycznym wynosi: $O(n^2)$

4. Sortowanie szybkie (quicksort) z uwzględnieniem 4 różnych pivotów: lewy, środkowy, prawy, losowy.

Z tablicy wybiera się element rozdzielający, po czym tablica jest dzielona na dwa fragmenty: do początkowego przenoszone są wszystkie elementy nie większe od rozdzielającego, do końcowego wszystkie większe. Potem sortuje się osobno początkową i końcową część tablic. Rekursja kończy się, gdy kolejny fragment uzyskany z podziału zawiera pojedynczy element, jako że jednoelementowa tablica nie wymaga sortowania.

Jest algorytmem niestabilnym, dla dużych tablic trzeba przydzielić odpowiednio większy stos.

Złożoność obliczeniowa w przypadku średnim wynosi: $O(n\ log\ n)$

Złożoność obliczeniowa w przypadku pesymistycznym wynosi: $O(n^2)$

Plan eksperymentu

Założeniem eksperymentu jest przeprowadzenie symulacji algorytmów sortujących dla wybranych rozmiarów tablic. Rozmiary tablic to: 10000, 20000, 30000, 40000, 50000, 60000, 70000 i 80000 elementów.

Dla każdego rozmiaru tablic zostały zmierzone czasy sortowań tablic zapełnionych w różny sposób
- Tablica losowa,
- Tablica posortowana rosnąco,
- Tablica posortowana malejąco,
- Tablica posortowana częściowo w 33%,
- Tablica posortowana częściowo w 66%.

Dla dokładności pomiarów każdy rodzaj tablicy został posortowany 100 razy. W tabeli z danymi zapisany jest podany średni wynik pomiarów.

Wszystkie pomiary zostały wykonane dla typu danych int. Dla sortowania shella zostały wykonane dodatkowo pomiary dla danych typu float.

Dla każdego pomiaru została wygenerowana nowa tablica danego typu oraz rozmiaru. Generowanie tablic zostało zrealizowane za pomocą biblioteki <random> oraz funkcji uniform_real_distribution. Poszczególne sortowania są testowane na podstawie kopii wygenerowanej tablicy.

Do mierzenia czasu została wykorzystana biblioteka <chrono> oraz funkcja high_resolution_clock. Dla każdego algorytmu został zapisany czas rozpoczęcia oraz zakończenia działania algorytmu. Na podstawie ich różnicy został obliczony czas wykonania sortowania.

Omówienie przebiegu eksperymentów

Jednostkami czasu pomiarów są ms.

Tabele:

Int	Sortowanie przez wstawianie				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	54.8034	189.523	196.133	4.11685	163.573
20000	210.555	706.865	731.101	210.616	612.761
30000	465.651	1592.66	1652.32	465.743	1376.37
40000	805.922	2755.08	2850.76	806.045	2389.6
50000	1249.59	4279.72	4425.35	1249.75	3719.79
60000	1801.68	6180.89	6396.26	1801.86	5366.77
70000	2454.68	8377.33	8667.01	2454.9	7280.37
80000	3176.23	10945.6	11317.9	3176.48	9512.34

Int	Sortowanie przez kopcowanie				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	2.11129	7.91758	9.95585	1.56665	5.80127
20000	4.59314	16.9955	21.2674	8.82359	12.4215
30000	7.26119	26.5176	33.0809	13.7067	19.3032
40000	9.91512	36.1498	45.1946	18.6313	26.349
50000	12.6618	45.8417	57.449	23.4642	33.2281
60000	15.5433	56.1765	70.2181	28.6114	40.6912
70000	18.315	66.059	82.5646	33.77	48.0144
80000	21.3588	77.0054	96.4424	39.6197	55.5803

Int	Sortowanie Shella Knutha				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.2929	2.90626	3.4656	1.43603	1.99017
20000	2.88343	6.40669	7.61632	3.44388	4.40428
30000	4.71817	10.3117	12.267	5.6424	7.08102
40000	6.47348	14.2637	16.986	7.71154	9.71526
50000	8.35938	18.2624	21.7492	9.94476	12.4752
60000	10.2452	22.5033	26.784	12.2283	15.3373
70000	12.5051	27.3169	32.3218	14.744	18.728
80000	15.1753	32.0715	37.9145	17.7122	22.1329

Float	Sortowanie Shella Knutha				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.52079	3.30448	3.9139	1.7801	2.2515
20000	3.35132	7.2499	8.57146	3.89349	4.95083
30000	5.43298	11.5785	13.7225	6.31178	7.86525
40000	7.48104	15.9296	18.8778	8.66737	10.8108
50000	9.69566	20.6089	24.4014	11.201	13.9639
60000	11.8194	25.1438	29.7935	13.6527	16.994
70000	14.1431	30.1469	35.6678	16.2669	20.4735
80000	16.4257	34.8495	41.2533	18.8643	23.6105

Int	Sortowanie Shella				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.17433	2.71461	3.23409	1.43603	1.88128
20000	2.89129	6.56053	7.78853	3.49578	4.55266
30000	4.25589	9.6421	11.465	5.1466	6.67968
40000	6.74472	15.0059	17.8671	8.08375	10.3158
50000	7.94343	18.064	21.5917	9.62268	12.4665
60000	10.473	23.2501	27.5925	12.531	15.1026
70000	15.835	34.6983	42.2407	15.2162	20.3023
80000	20.6304	47.2304	53.8252	18.3011	25.8544

Int	Sortowanie szybkie pivot lewy				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.25342	191.21	257.481	75.3815	149.15
20000	2.65757	757.797	1020.62	297.729	592.547
30000	4.10249	1698.17	2286.88	666.286	1328.44
40000	5.67117	3015.05	4061.12	1184.79	2360.01
50000	7.20961	4713.14	6345.32	1852.58	3689.22
60000	8.73033	6781.24	9132.47	2652.15	5303.45
70000	10.0841	9209.96	12407.1	3610.87	7208.89
80000	11.5949	12039.7	16212.6	4713.25	9411.17

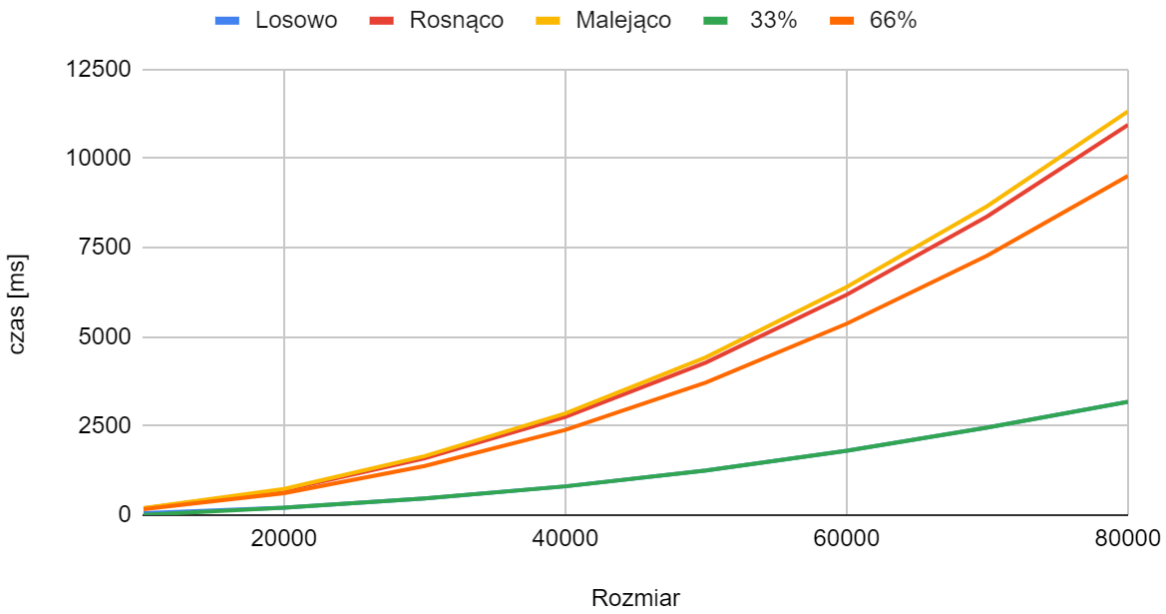
Int	Sortowanie szybkie pivot środkowy				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.22933	2.9941	3.65463	1.63238	2.05722
20000	2.60241	6.27565	7.65723	3.44133	4.30423
30000	4.00625	9.69334	11.829	5.28556	6.61635
40000	5.44082	13.1728	16.0521	7.17062	8.99256
50000	6.98304	16.6453	20.3446	9.21203	11.4331
60000	8.34537	20.1623	24.649	11.0165	13.8259
70000	9.81759	23.464	28.7479	12.9262	16.0476
80000	11.3076	27.4684	33.6185	14.8837	18.5278

Int	Sortowanie szybkie pivot prawy				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	1.25927	158.443	192.045	75.3462	149.226
20000	2.66631	626.94	759.586	297.645	592.178
30000	4.10326	1405.94	1702.61	666.242	1328.82
40000	5.56853	2495.02	3020.29	1182.86	2359.56
50000	7.09175	3900.56	4718.89	1854.02	3690.68
60000	8.59805	5596.25	6775.07	2649.44	5294.8
70000	10.0407	7624.75	9226.25	3613.01	7217.15
80000	11.5926	9944.84	12037.8	4706.56	9409.29

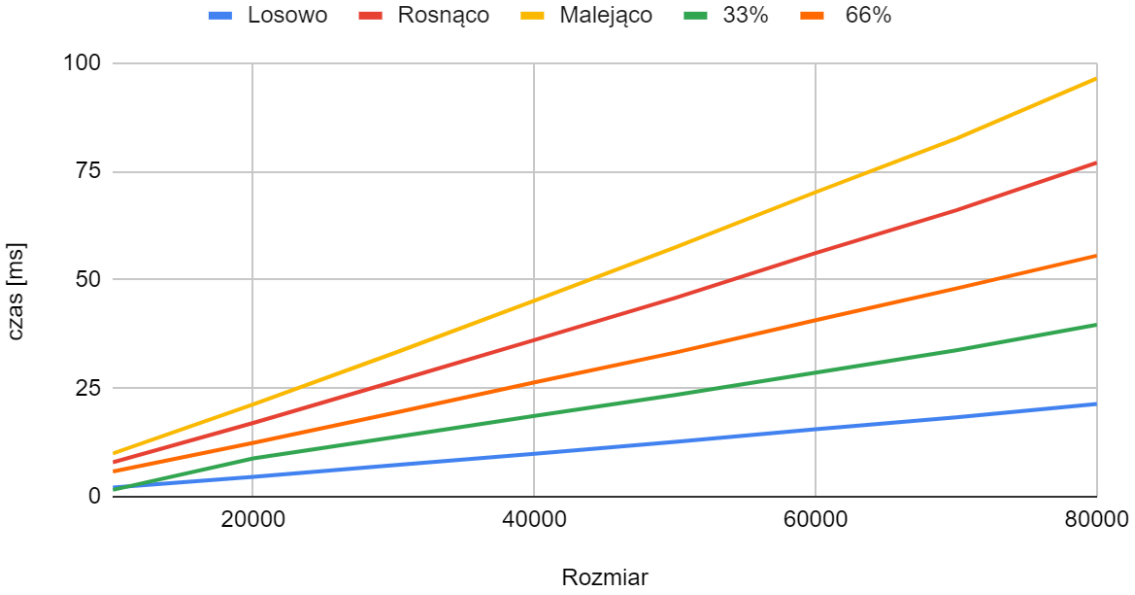
Int	Sortowanie szybkie pivot losowy				
Rozmiar	Losowo	Rosnąco	Malejąco	Posortowanie w 33%	Posortowanie w 66%
10000	2.36083	8.04728	9.99008	4.12415	5.88224
20000	4.88069	16.4067	20.3383	8.43428	12.0023
30000	7.41163	25.3076	31.2385	13.1939	18.6936
40000	9.9101	33.6585	41.7914	17.4613	24.7057
50000	12.7196	42.5216	53.1257	22.3321	31.2507
60000	15.4085	70.2297	84.6545	26.7819	44.7568
70000	17.947	90.4433	114.445	32.4798	61.863
80000	20.2503	112.3342	138.3969	37.811	79.4031

Wykresy:

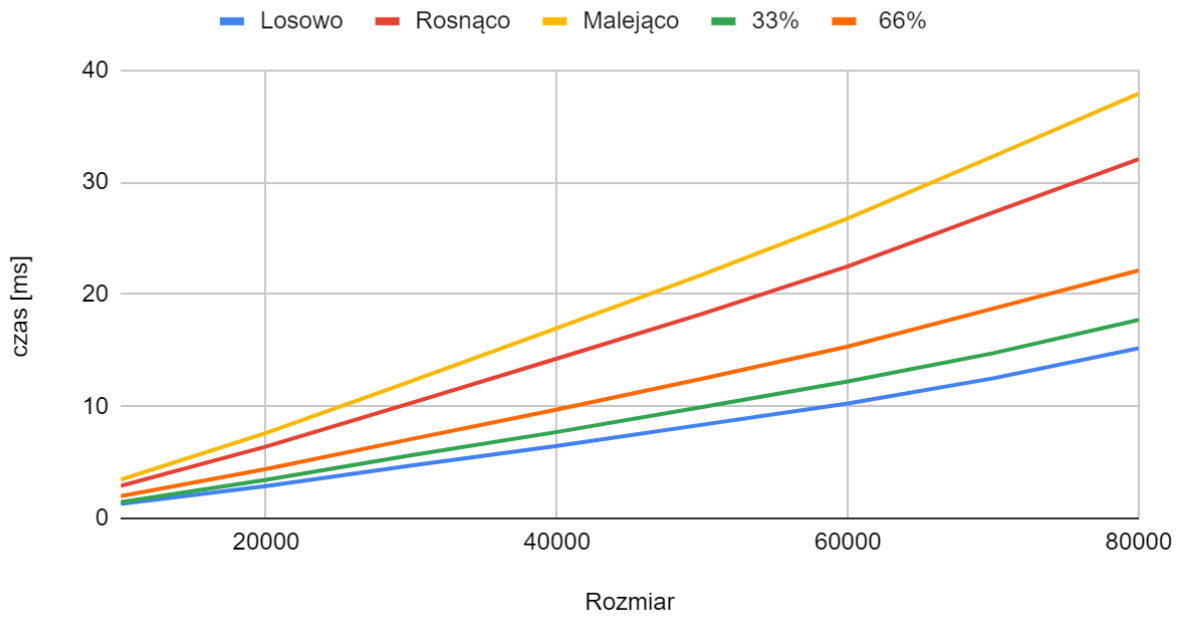
Int Sortowanie przez wstawianie



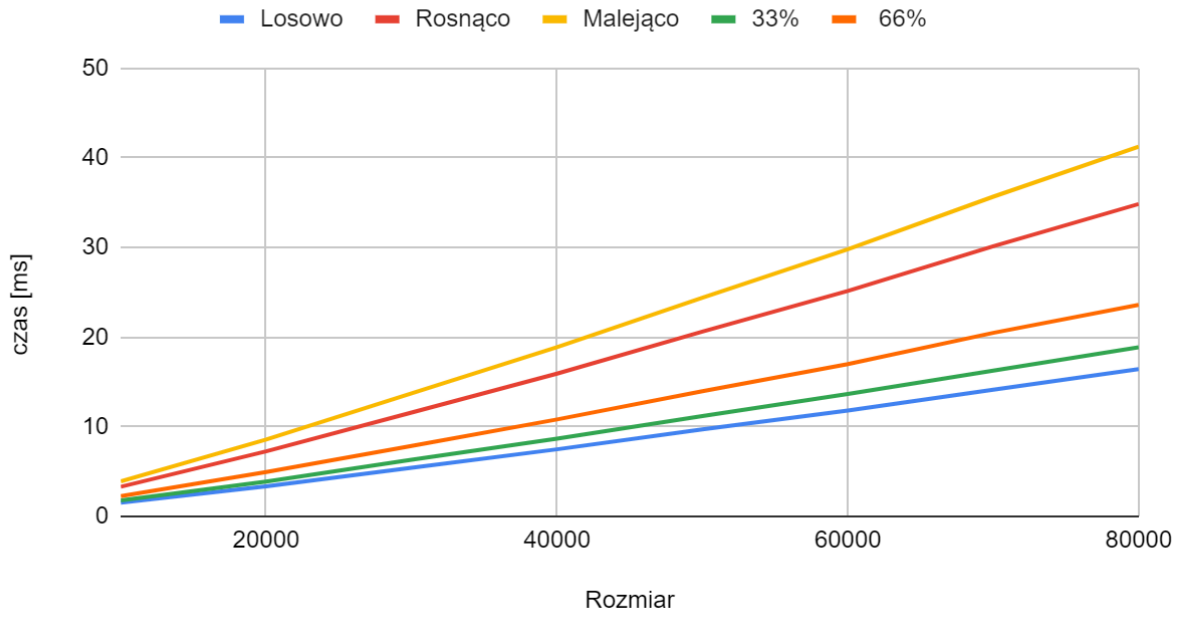
Int Sortowanie przez kopcowanie

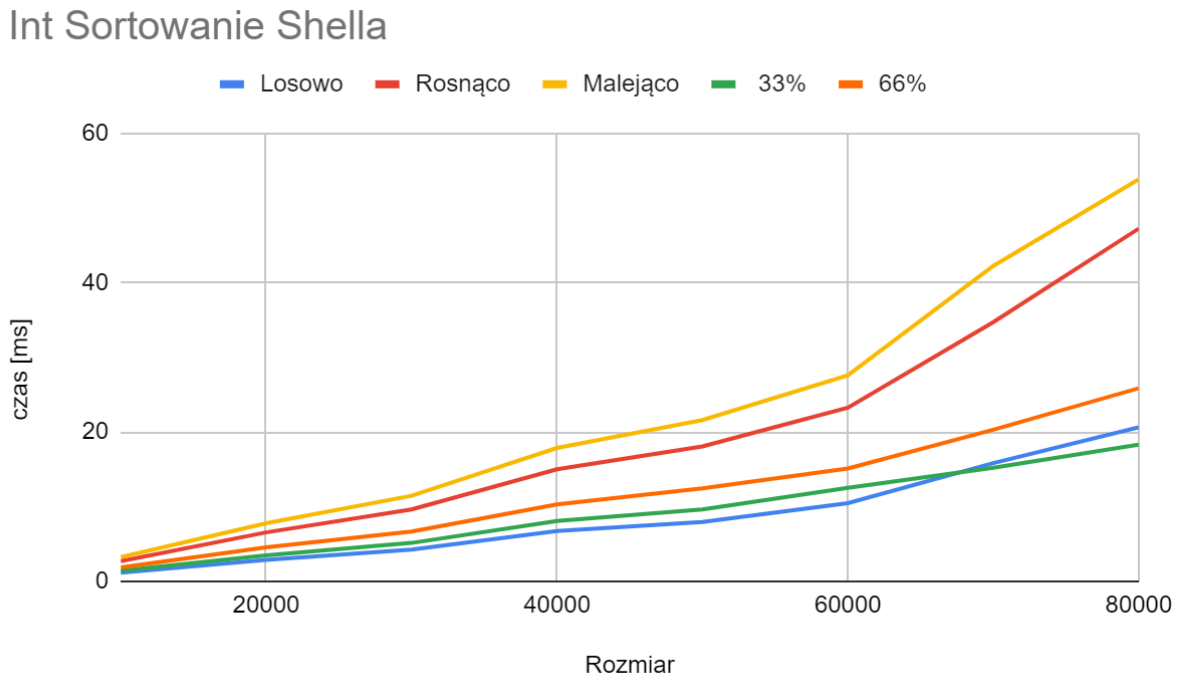


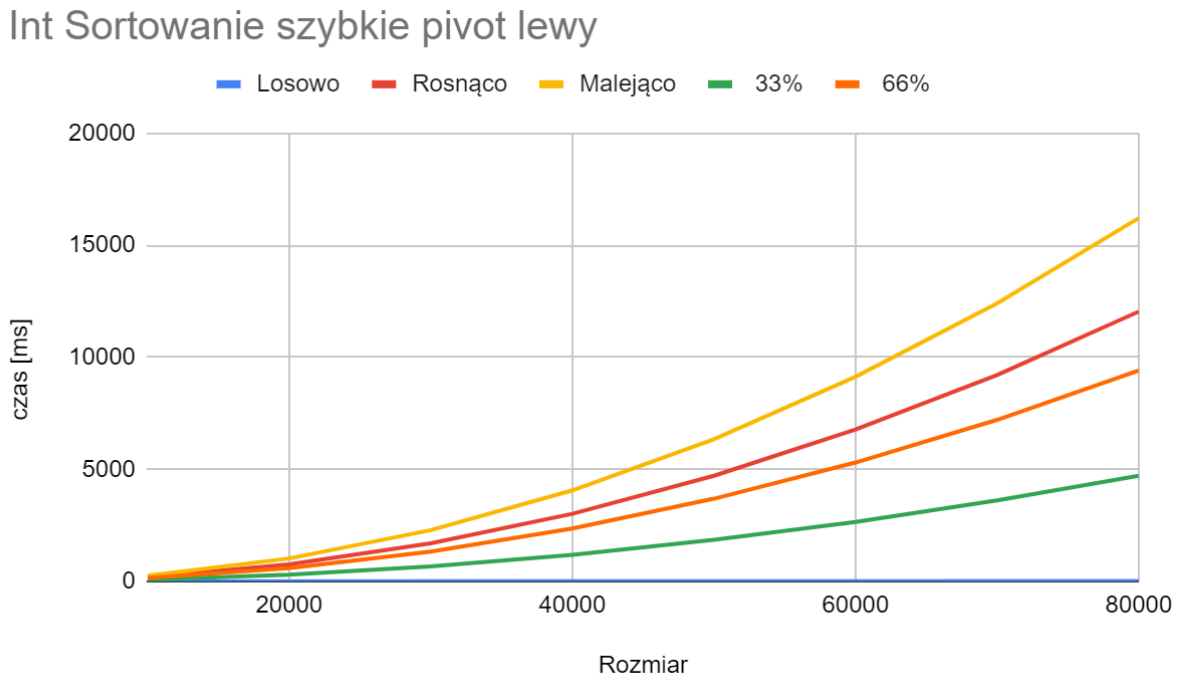
Int Sortowanie Shella Knutha

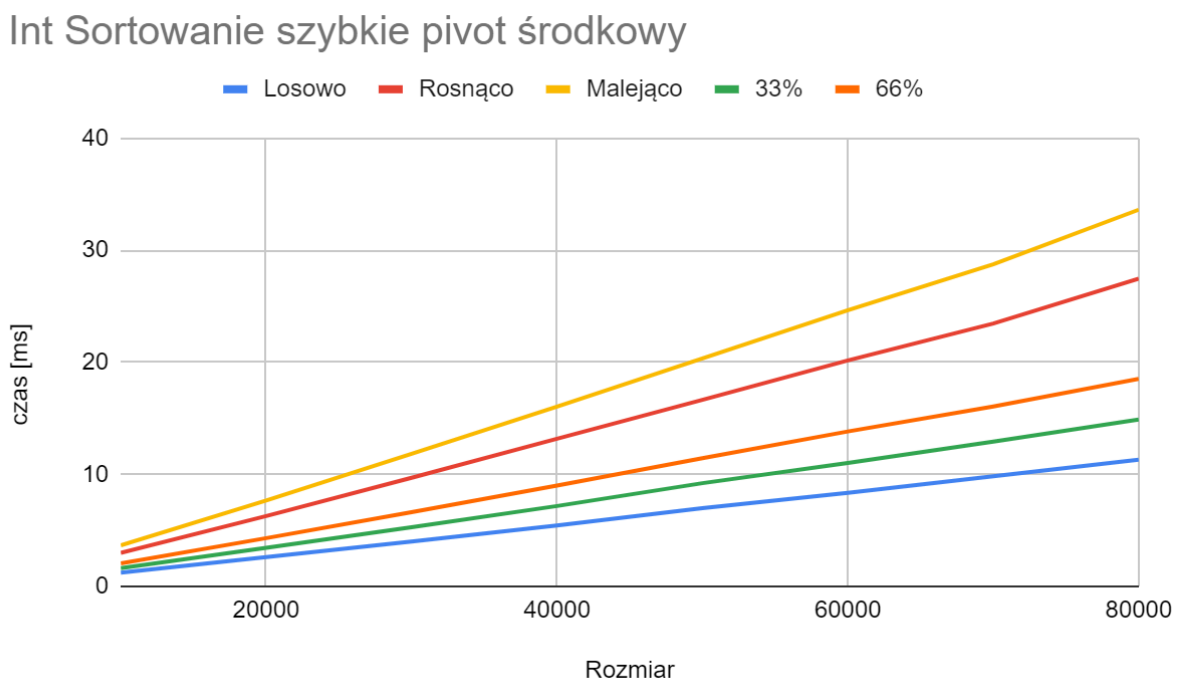


Float Sortowanie Shella Knutha

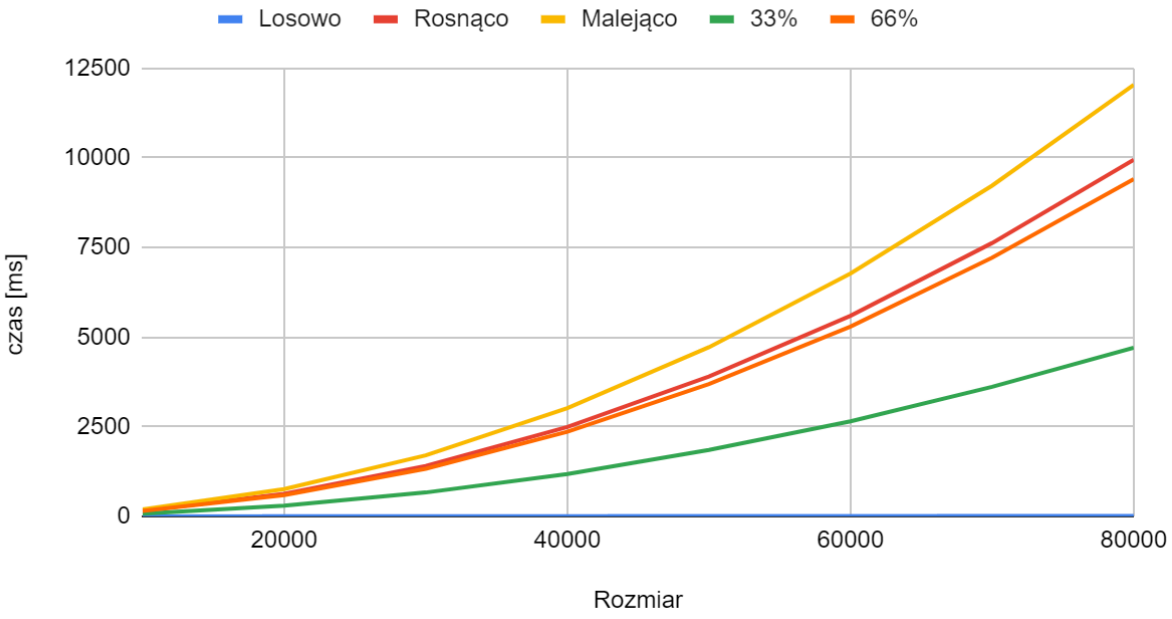



Int Sortowanie Shella
Losowo
Rosnąco
Malejąco
33%
66%
60
40
20
0
czas [ms]
20000
40000
60000
80000
Rozmiar

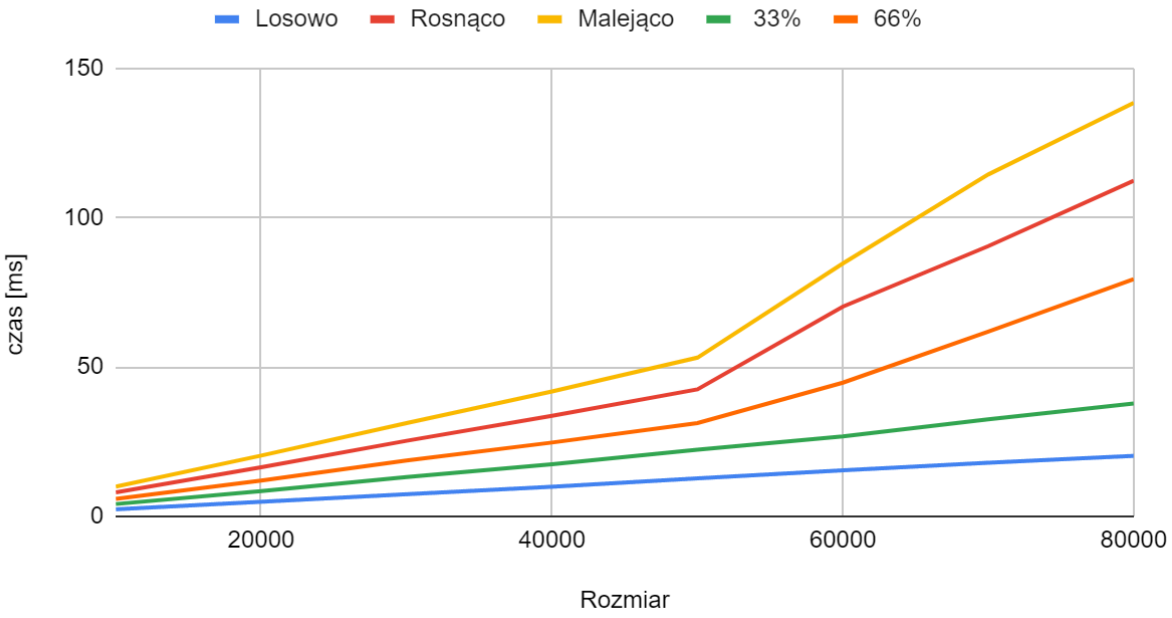

Int Sortowanie szybkie pivot lewy
Losowo
Rosnąco
Malejąco
33%
66%
20000
15000
10000
5000
0
czas [ms]
20000
40000
60000
80000
Rozmiar


Int Sortowanie szybkie pivot środkowy
Losowo
Rosnąco
Malejąco
33%
66%
40
30
20
10
0
czas [ms]
20000
40000
60000
80000
Rozmiar

Int Sortowanie szybkie pivot prawy



Int Sortowanie szybkie pivot losowy



Podsumowanie i wnioski

- Zastosowane algorytmy zachowały się w sposób zbliżony do oczekiwanego
- Pomimo tego że tablica posortowana w 66% wydała się że będzie szybsza do posortowania niż ta posortowana w 33% efekt był odwrotny
- Wykres sortowania shella jest dosyć zniekształcony
- Porównanie int i float dla sortowanie shella knutha pokazało że dla typu int jest ono szybsze
- dla sortowania szybkiego z lewym i prawym pivotem widać kwadratowy wykres funkcji dla pesymistycznych danych natomiast dla losowych danych są to bardzo efektywne sortowania
- sortowanie szybkie ze środkowym pivotem jest pewniejsze w użyciu niż pozostałe opcje pivota
- najszybszymi sortowaniami okazały się quicksort z środkowym pivotem oraz shellsort Knuth'a
- dla sortowania szybkiego z pivotem losowym od rozmiaru 60 tys wykres odchylił się w kierunku osi X

Literatura

- https://pl.wikipedia.org/wiki/Sortowanie
- https://eduinf.waw.pl/inf/alg/003_sort/index.php
- https://pl.wikipedia.org/wiki/Sortowanie_przez_wstawianie
- https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
- https://en.wikipedia.org/wiki/Shellsort
- https://pl.wikipedia.org/wiki/Sortowanie_szybkie
- https://eduinf.waw.pl/inf/alg/003_sort/0012.php
- https://eduinf.waw.pl/inf/alg/003_sort/0010.php