Sprawozdanie 1

„Projektowanie algorytmów i metod sztucznej inteligencji”

3 kwietnia 2019

Temat projektu: Algorytmy sortowania oraz ich złożoność obliczeniowa.

Autor : Paweł Gajda

Termin zajęć: Środa 7.30-9.00

Prowadzący: Dr inż. Łukasz Jeleń

1. Wstęp teoretyczny

1.1 Opis projektu

Projekt polegał na przeanalizowaniu trzech algorytmów sortowania pod względem wydajności, tj. czasu w jakim są one w stanie posortować określone porcje danych. W projekcie analizowane będą następujące algorytmy:

- 2.1 Sortowanie szybkie
- 2.2 Sortowanie introspektywne
- 2.3 Sortowanie przez scalanie

Powyższe algorytmy zostały przetestowane na rożnych objętościach do sortowania ale również zostały poddane badaniom, w których na wejście algorytmy dostawały już w części posortowane dane.

Poszczególne wielkości:

- 10 000
- 50 000
- 100 000
- 500 000
- 1 000 000

Procent posortowania danych, które algorytmy dostawały do posortowania:

- 0% - wszystkie elementy losowe
- 25%
- 50%
- 75%
- 95%
- 99%
- 99,7%
- -100% - dane posortowane lecz w odwrotnej kolejności

2. Opis algorytmów

2.1 Sortowanie szybkie - QuickSort

Sortowanie szybkie należy do grupy algorytmów „dziel i zwyciężaj”. Według ustalonego schematu wybierany jest jeden element tablicy, który będzie nazywany pivotem. Następnie ustawiamy elementy nie większe na lewo od tej wartości, natomiast wartości nie mniejsze na prawo. W taki sposób tablica została podzielona na dwie mniejsze niekoniecznie równe tablice. Następnie obie te tablice sortujemy osobno według tego schematu. W przygotowanym przeze mnie programie, pivot jest środkowym elementem tablicy.

2.2 Sortowanie introspektywne - IntroSort

Jest to odmiana sortowania hybrydowego, wyeliminowany został tu problem kwadratowej złożoności obliczeniowej, która mogła się pojawić w najgorszym przypadku algorytmu sortowania szybkiego. Metoda tego sortowania polega na kontrolowaniu głębokości wywoływania rekurencyjnego dla algorytmu sortowania szybkiego. W przypadku, gdy pewna wartość x, określona jako 2*log2n, obliczona na początku działania programu oraz zmniejszana z każdym ponownym wywołaniem rekurencyjnym algorytmu sortowania szybkiego osiągnie zero, dla podproblemu którym obecnie się zajmujemy zostaje wywołana procedura sortowania poprzez kopcowanie, sortowanie to jest traktowane jako pomocnicze. Na końcu algorytmu wykorzystywane jest sortowanie poprzez wstawianie dla posortowanej wstępnie tablicy.

2.3 Sortowanie przez scalanie - MergeSort

Sortowanie przez scalanie należy do algorytmów, działających na metodzie dziel i zwyciężaj. Złożoność algorytmu wynosi O(n log n). Algorytm dzieli rekurencyjnie dane aż do momentu osiągniecia jednoelementowych podproblemów następnie scalając sortuje je.

2.4 Porównanie algorytmów

	QuickSort	MergeSort	IntroSort
Złożoność pamięciowa	O(log n)| O(n)	O(n)	O(log n)
Najgorszy przypadek złożoności czasowej	$O(n^2)$	O(nlog n)	O(nlog n)
Stabilność	Niestabilny	Stabilny	Niestabilny
Lepsza wydajność	Dla mniejszych zestawów danych	Równa dla każdej wielkości zestawu danych	Dla w znaczącym stopniu posortowanych wstępnie danych

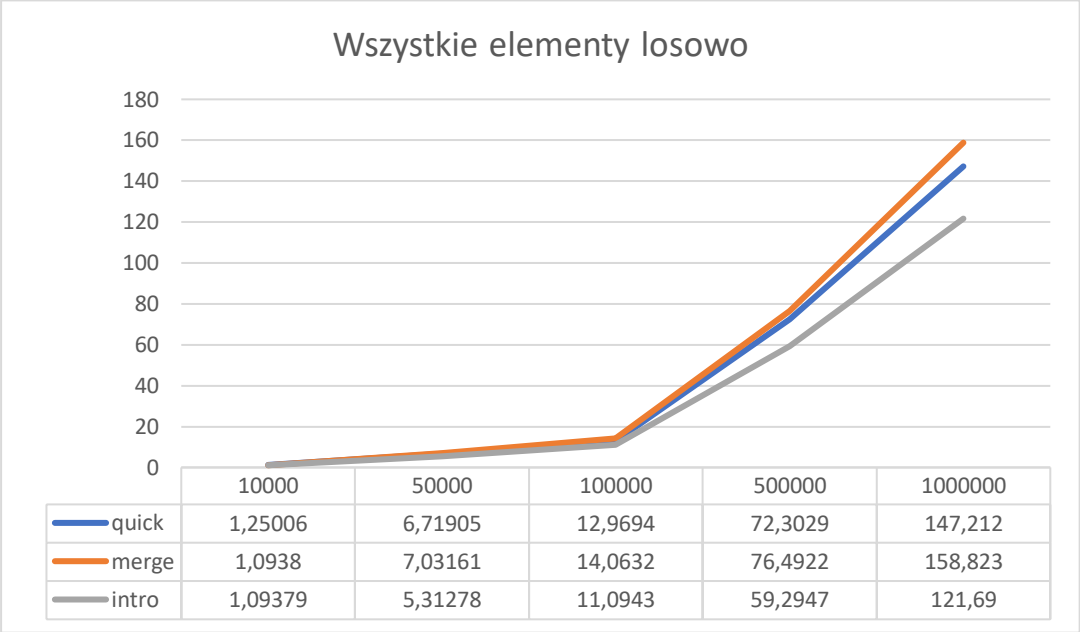
2.5 Przewidywane wyniki

Analizując algorytmy można stwierdzić, że najszybszy powinien być IntroSort a najwolniejszy MergeSort. Jednak dla dużych ilości danych oraz dla dużego stopnia posortowania wstępnego MergeSort powinien być szybszy od QuickSorta z uwagi na dużą głębokość rekurencji jaką QuickSort będzie musiał pokonać.

W programie użyto niestandardowej implementacji MergeSorta, która polega na stworzeniu dodatkowej struktury danych dzięki czemu algorytm nie musi jej tworzyć podczas każdej rekurencji. Działanie to powinno znacznie obniżyć czas działania tego algorytmu.

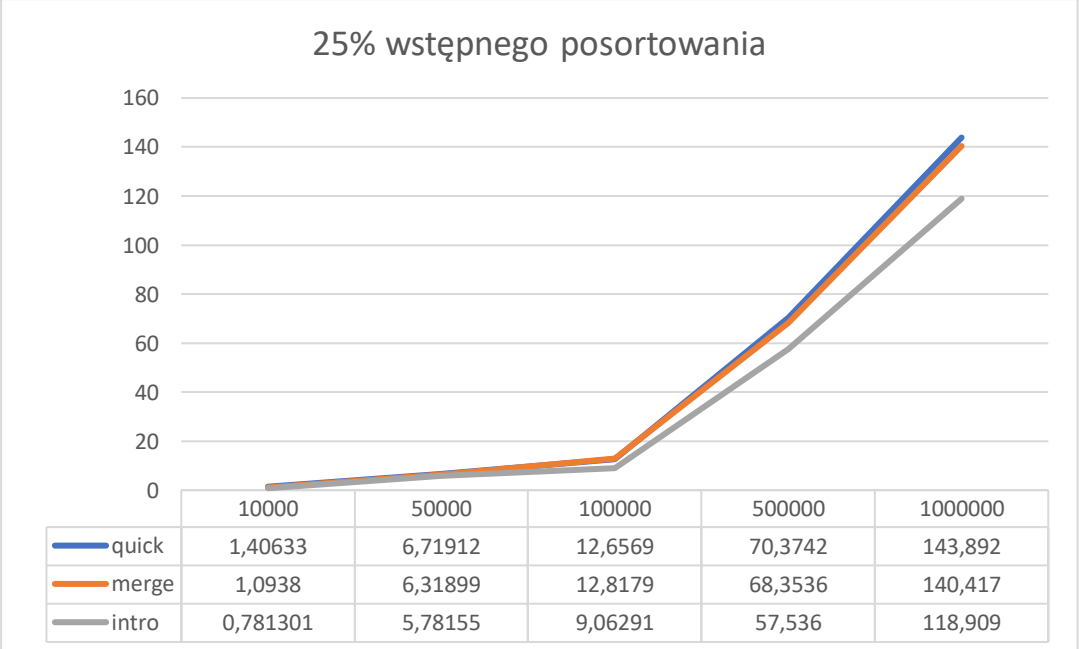
3 Przebieg testów

3.1 Wszystkie elementy losowo



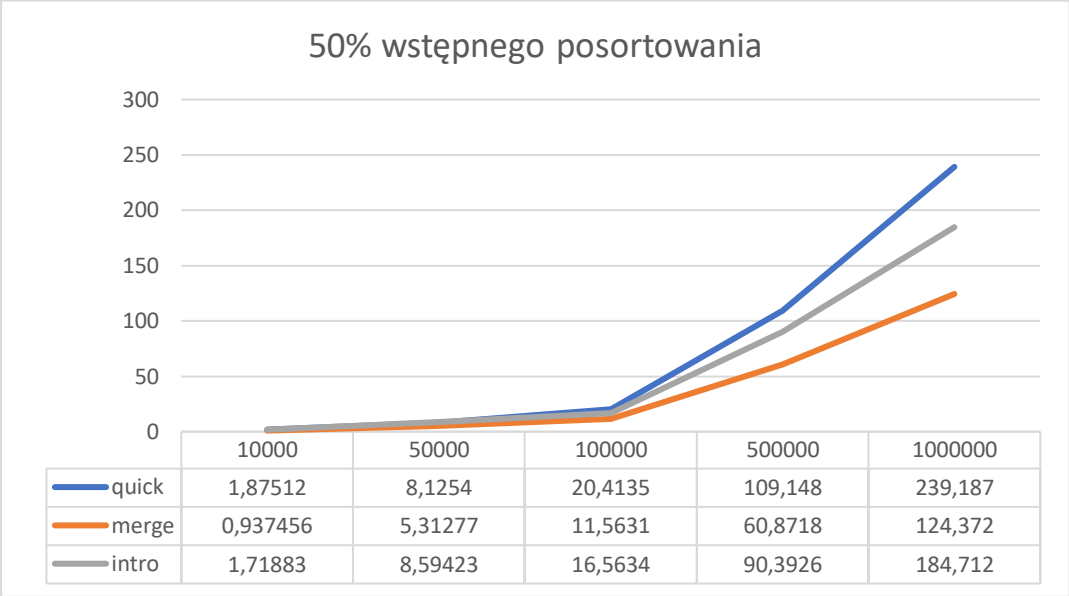
	10000	50000	100000	500000	1000000
quick	1,25006	6,71905	12,9694	72,3029	147,212
merge	1,0938	7,03161	14,0632	76,4922	158,823
intro	1,09379	5,31278	11,0943	59,2947	121,69

3.2 25% wstępnego posortowania



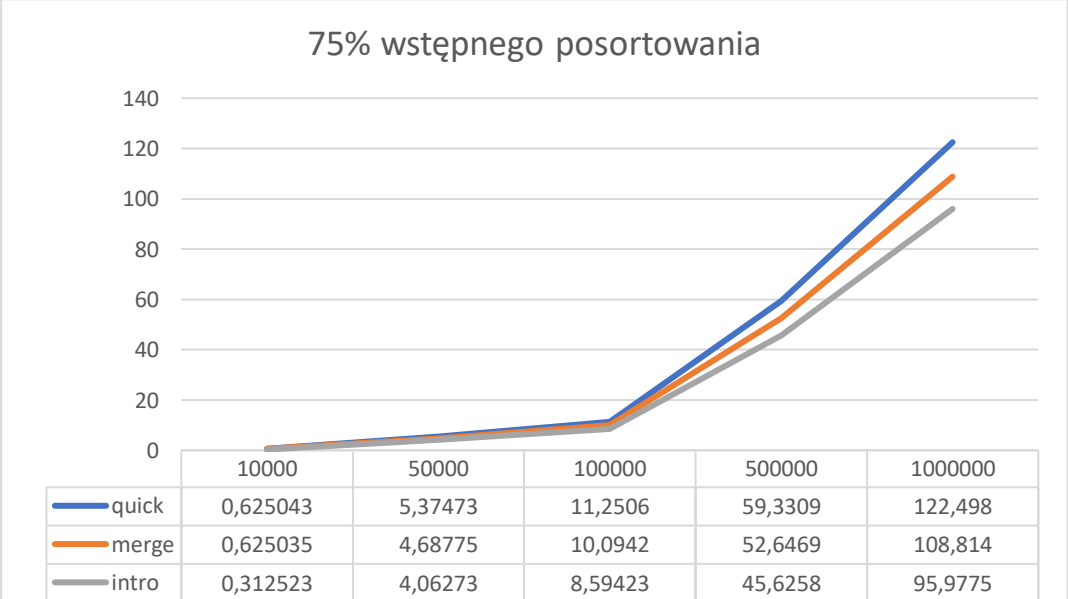
	10000	50000	100000	500000	1000000
quick	1,40633	6,71912	12,6569	70,3742	143,892
merge	1,0938	6,31899	12,8179	68,3536	140,417
intro	0,781301	5,78155	9,06291	57,536	118,909

3.3 50% wstępnego posortowania



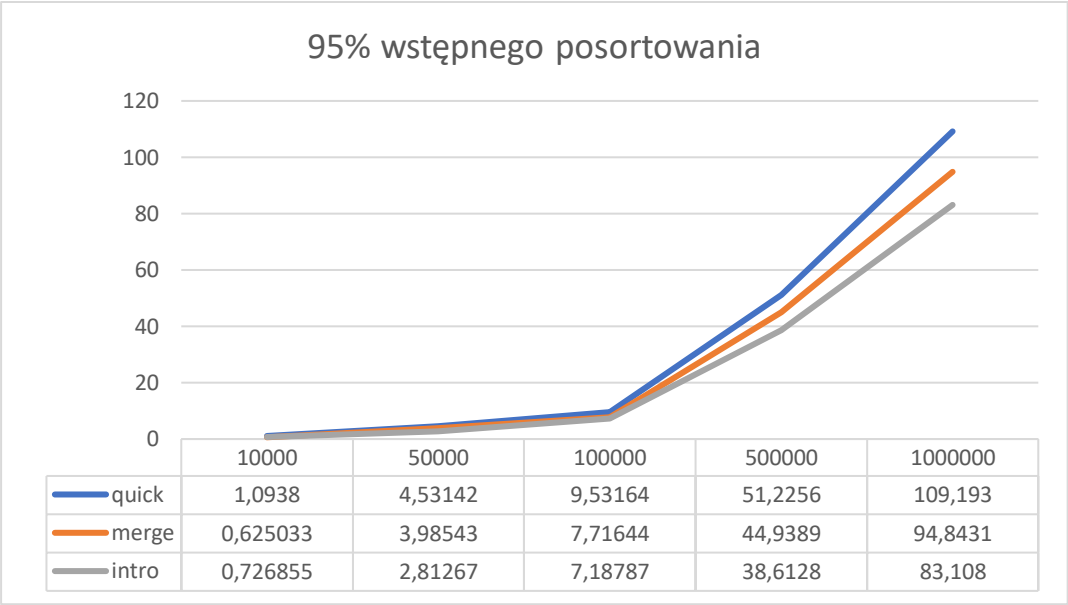
	10000	50000	100000	500000	1000000
quick	1,87512	8,1254	20,4135	109,148	239,187
merge	0,937456	5,31277	11,5631	60,8718	124,372
intro	1,71883	8,59423	16,5634	90,3926	184,712

3.4 75% wstępnego posortowania



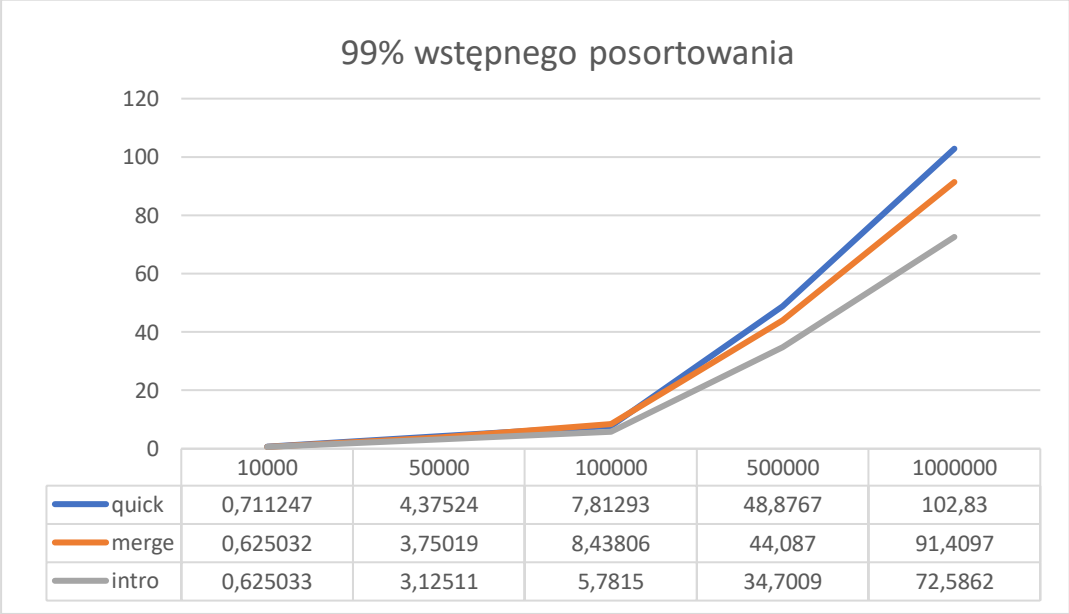
	10000	50000	100000	500000	1000000
quick	0,625043	5,37473	11,2506	59,3309	122,498
merge	0,625035	4,68775	10,0942	52,6469	108,814
intro	0,312523	4,06273	8,59423	45,6258	95,9775

3.5 95% wstępnego posortowania



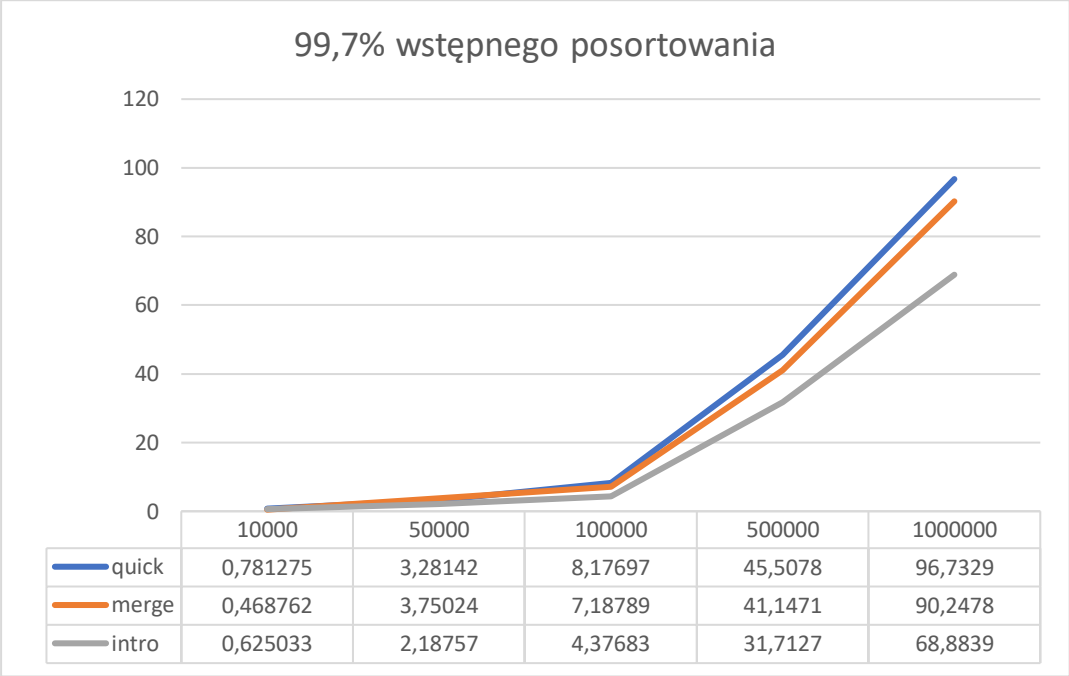
	10000	50000	100000	500000	1000000
quick	1,0938	4,53142	9,53164	51,2256	109,193
merge	0,625033	3,98543	7,71644	44,9389	94,8431
intro	0,726855	2,81267	7,18787	38,6128	83,108

3.6 99% wstępnego posortowania



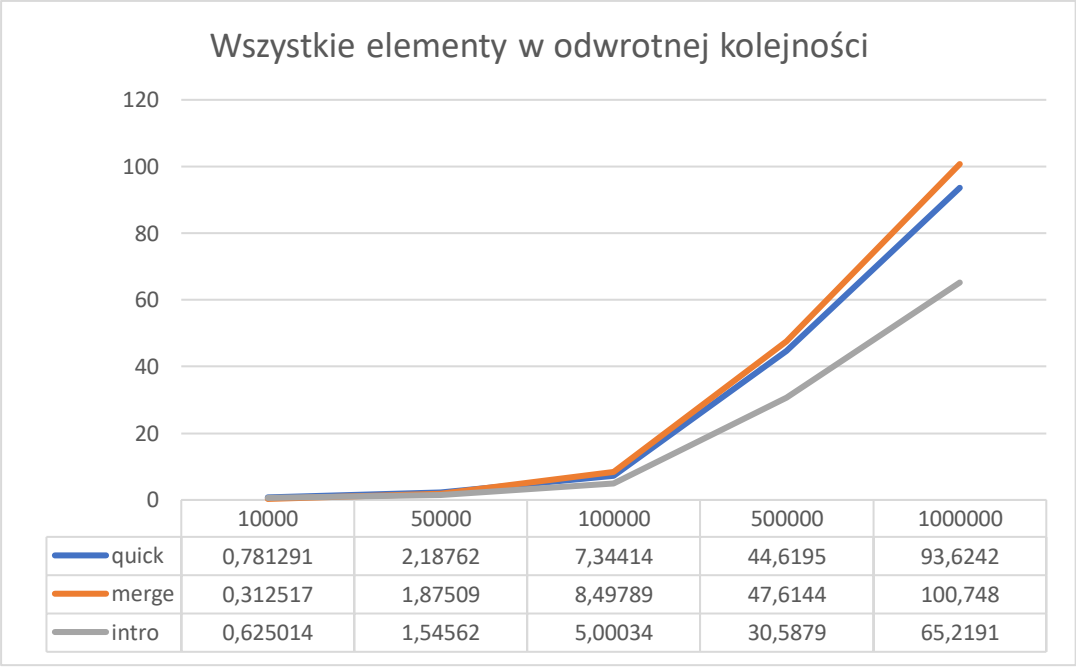
	10000	50000	100000	500000	1000000
quick	0,711247	4,37524	7,81293	48,8767	102,83
merge	0,625032	3,75019	8,43806	44,087	91,4097
intro	0,625033	3,12511	5,7815	34,7009	72,5862

3.7 99,7% wstępnego posortowania



	10000	50000	100000	500000	1000000
quick	0,781275	3,28142	8,17697	45,5078	96,7329
merge	0,468762	3,75024	7,18789	41,1471	90,2478
intro	0,625033	2,18757	4,37683	31,7127	68,8839

3.8 Wszystkie elementy w odwrotnej kolejności



	10000	50000	100000	500000	1000000
quick	0,781291	2,18762	7,34414	44,6195	93,6242
merge	0,312517	1,87509	8,49789	47,6144	100,748
intro	0,625014	1,54562	5,00034	30,5879	65,2191

4. Wnioski

- Przypadek posortowania wstępnego 50% jest odstępstwem od reszty, widać na nim drastyczny wzrost czasu wykonywania się algorytmu sortowania szybkiego i introspektywnego. Jest to prawdopodobnie spowodowane faktem, iż oba algorytmy zostały zaimplementowane w ten sposób, iż pivotem czyli elementem osiowym jest środkowy element.
- Poza przypadkiem 50% wstępnego posortowania danych IntroSort jest najszybszym z algorytmów a różnice przy większych ilościach danych zaczynają być znaczące.
- MergeSort dzięki zaimplementowanej tablicy pomocniczej nie odbiega zbytnio od pozostałych dwóch algorytmów pod względem czasu wykonania, a nawet w większości przypadków jest szybszy od QuickSorta.
- Wnioski końcowe w większości zgadzają się z analizą wstępną.

5. Literatura

- https://pl.wikipedia.org/wiki/Sortowanie
- https://www.youtube.com/watch?v=8RkE7MbqVl8&t=243s
- https://www.youtube.com/watch?v=iJyUFvvdfUg&t=328s
- https://www.youtube.com/watch?v=82XxdhRCMbI&t=336s
- http://www.algorytm.edu.pl/