Politechnika
Wrocławska

Marcin Karasiewicz

Projektowanie algorytmów
i metody sztucznej inteligencji

Algorytmy sortowania

Spis treści

1 Algorytm Merge Sort **1**
1.1 Opis 1
1.2 Tabela pomiarów 1
1.3 Wykres 2

2 Algorytm Quick Sort **3**
2.1 Opis 3
2.2 Tabela pomiarów 3
2.3 Wykres 4

3 Algorytm Heap Sort **5**
3.1 Opis 5
3.2 Tabela pomiarów 5
3.3 Wykres 6

4 Algorytm Hybrid Sort **7**
4.1 Opis 7
4.2 Tabela pomiarów 7
4.3 Wykres 8

5 Wnioski **9**

1 Algorytm Merge Sort

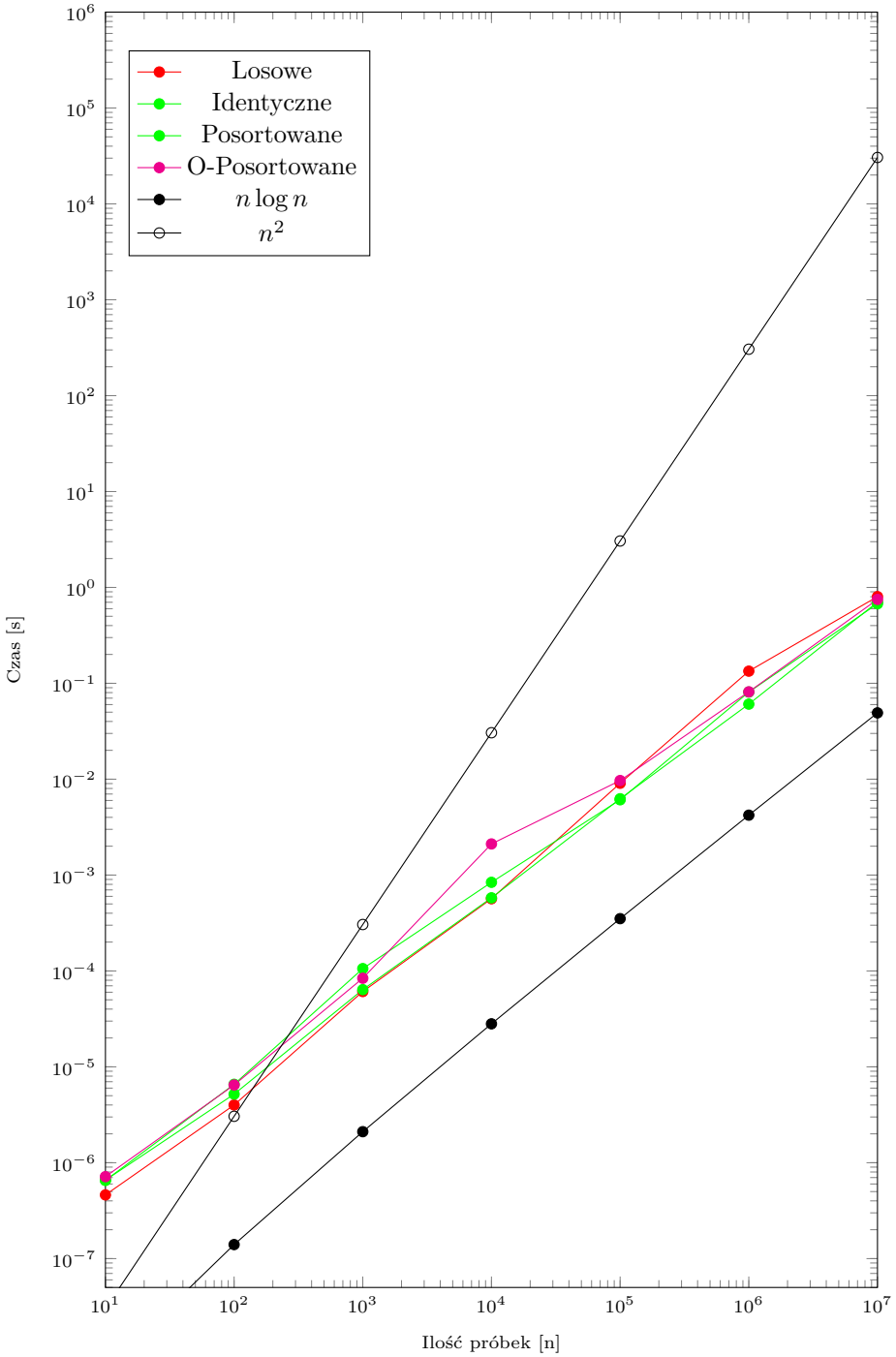
1.1 Opis

Sortowanie przez scalanie jest przykładem algorytmu rekurencyjnego o złożoności $O(n \log n)$. Algorytm wykorzystuje metodę "dziel i zwyciężaj". Nasza nieposortowana tablica zostaje podzielona na dwie części. Rozkład tablic następuje rekurencyjnie aż tablice staną się jednoelementowe. Następnie tablice zostają złączone w szeregu posortowanym.

1.2 Tabela pomiarów

Rozmiar[n]	Czas[s]			
	Losowe	Posortowane	O-Posortowane	Identyczne
10	0.0000004616	0.0000006574	0.0000007159	0.0000006544
100	0.0000040008	0.0000051883	0.0000064687	0.0000065753
1000	0.0000609885	0.0000638151	0.0000839588	0.0001058314
10000	0.0005665224	0.0005789379	0.0021117148	0.0008403904
100000	0.0090663026	0.0062508104	0.0096813096	0.0060969113
1000000	0.1338663745	0.0606525145	0.0814999795	0.0810095399
10000000	0.8017014327	0.7031141995	0.7495614558	0.6749413904

1.3 Wykres


Losowe
Identyczne
Posortowane
O-Posortowane
$n \log n$
n^2
Czas [s]
Ilość próbek [n]
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

2 Algorytm Quick Sort

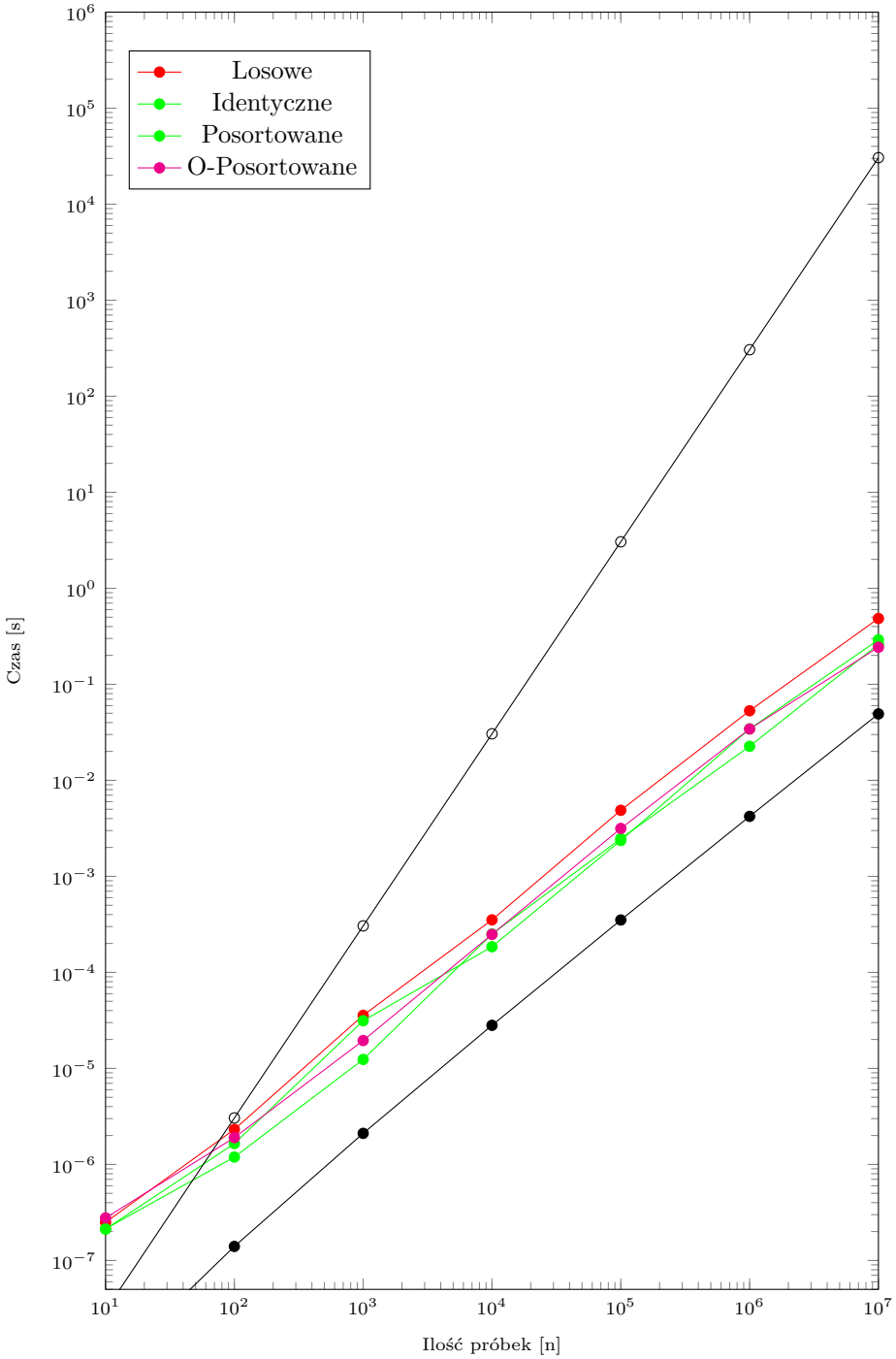
Algorytm wykorzystuje technikę "dziel i zwyciężaj". Według ustalonego schematu wybierany jest jeden element w sortowanej tablicy nazywany Pivotem. W algorytmie zaimplementowanym na potrzeby kursu użyto techniki "median of three". Następnie ustawiamy elementy nie większe na lewo tej wartości, natomiast nie mniejsze na prawo. W ten sposób powstaną nam dwie części tablicy, gdzie w pierwszej części znajdują się elementy nie większe od drugiej. Następnie każdą z tych podtablic sortujemy osobno według tego samego schematu. Przypadek optymistyczny ma złożoność $O(n \log n)$, natomisat pesymistyczny który dało się wyeliminować po przez technikę "median od three" ma złożoność $O(n^2)$

2.1 Opis

2.2 Tabela pomiarów

Rozmiar[n]	Czas[s]			
	Losowe	Posortowane	O-Posortowane	Identyczne
10	0.0000002496	0.0000002133	0.0000002772	0.0000002119
100	0.0000023181	0.0000011925	0.0000019025	0.0000016491
1000	0.0000357067	0.0000123955	0.0000195050	0.0000313642
10000	0.0003523159	0.0002506793	0.0002486864	0.0001850952
100000	0.0048754355	0.0024690919	0.0031532092	0.0023649113
1000000	0.0530444570	0.0226112096	0.0342633065	0.0343200958
10000000	0.4847639445	0.2607655615	0.2432170252	0.2907808971

2.3 Wykres


Losowe
Identyczne
Posortowane
O-Posortowane
Czas [s]
Ilość próbek [n]
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

3 Algorytm Heap Sort

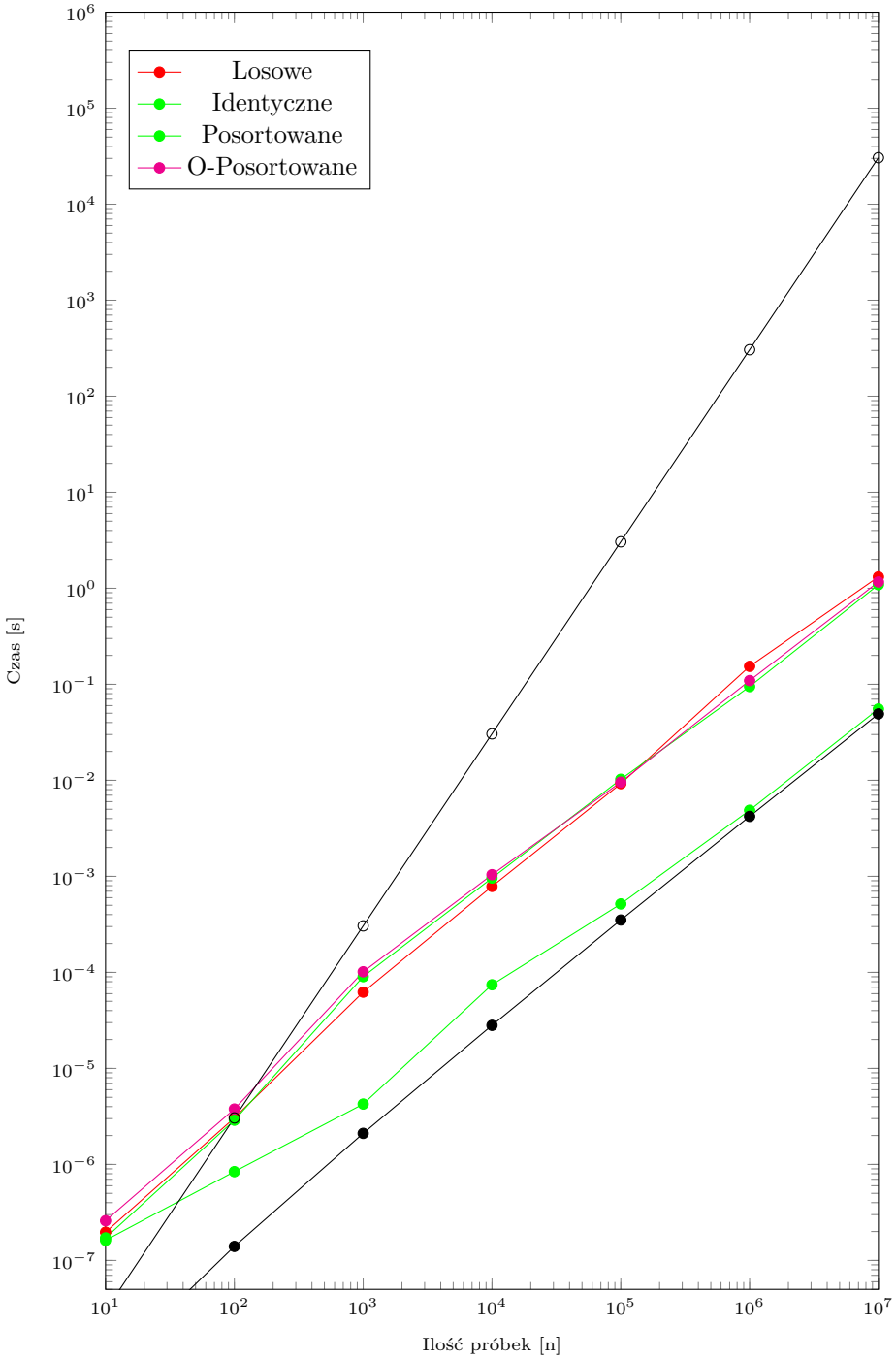
3.1 Opis

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. Sortowanie przez kopcowanie ma złożoność $O(n \log n)$

3.2 Tabela pomiarów

Rozmiar[n]	Czas[s]			
	Losowe	Posortowane	O-Posortowane	Identyczne
10	0.0000001969	0.0000001717	0.0000002591	0.0000001621
100	0.0000030504	0.0000028876	0.0000037742	0.0000008411
1000	0.0000622609	0.0000903414	0.0001014180	0.0000042528
10000	0.0007856392	0.0009601499	0.0010400160	0.0000743194
100000	0.0092356374	0.0102522893	0.0095718207	0.0005163060
1000000	0.1538449949	0.0947384417	0.1092921324	0.0048810180
10000000	1.3153185818	1.0889244169	1.1636245519	0.0553684626

3.3 Wykres


Losowe
Identyczne
Posortowane
O-Posortowane
Czas [s]
Ilość próbek [n]
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

4 Algorytm Hybrid Sort

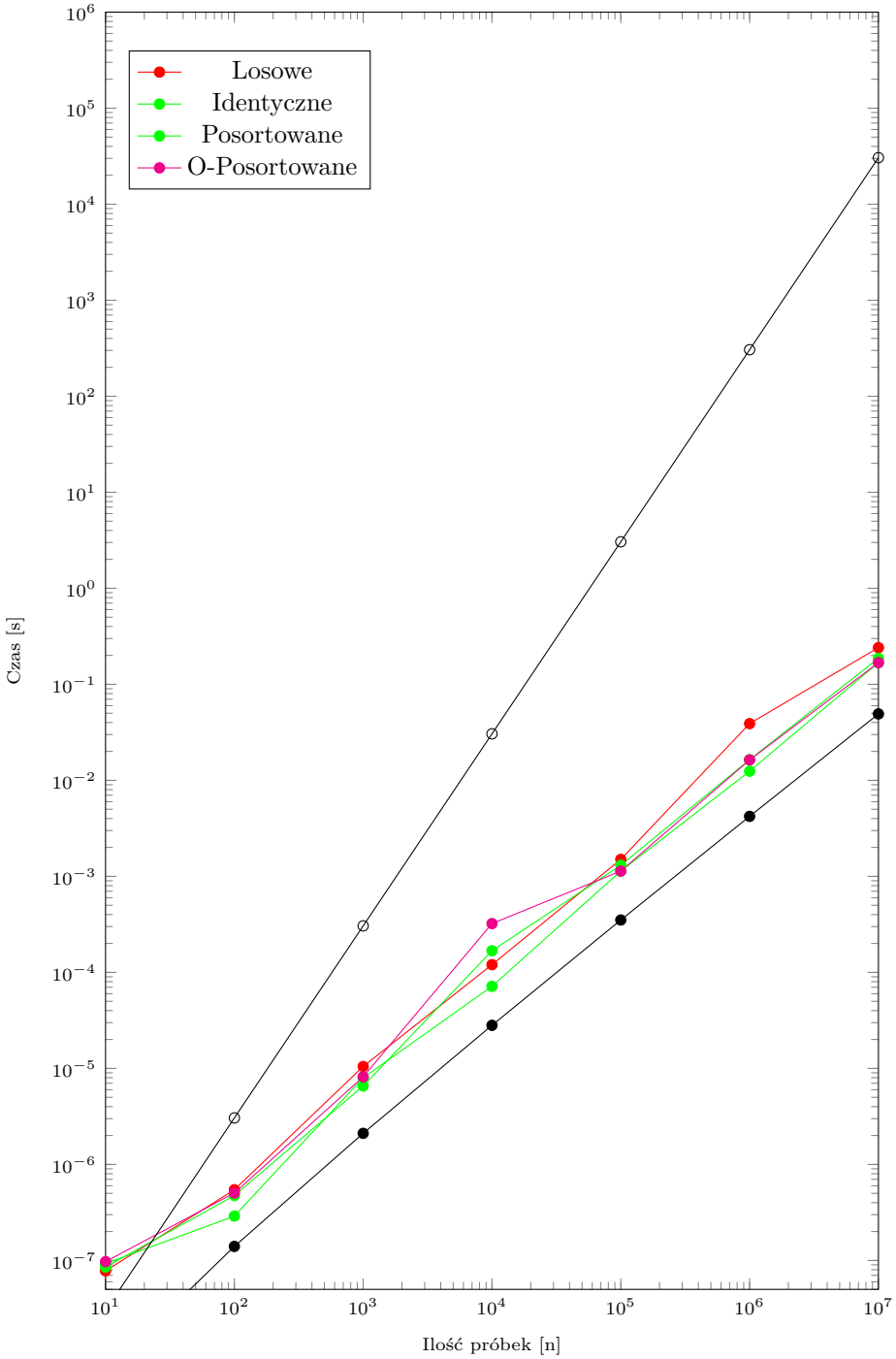
4.1 Opis

Jest to algorytm Quick Sort rozszerzony o sortowanie przy użyciu Insertion Sort podtablic gdy ich długość nie przekrasza 60 elementów.

4.2 Tabela pomiarów

Rozmiar[n]	Czas[s]			
	Losowe	Posortowane	O-Posortowane	Identyczne
10	0.0000000779	0.0000000928	0.0000000972	0.0000000854
100	0.0000005448	0.0000002899	0.0000005039	0.0000004741
1000	0.0000104756	0.0000079857	0.0000081701	0.0000065568
10000	0.0001202881	0.0000716271	0.0003220885	0.0001679542
100000	0.0015019490	0.0011353100	0.0013169724	0.0013053846
1000000	0.0389635356	0.0124492272	0.0163029062	0.0164278488
10000000	0.2409883819	0.1682131104	0.1682539896	0.1864922700

4.3 Wykres


Losowe
Identyczne
Posortowane
O-Posortowane
Czas [s]
Ilość próbek [n]
10^{6}
10^{5}
10^{4}
10^{3}
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{1}
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}

5 Wnioski

Pomiary zostały przeprowadzone na komputerze z procesorem i5-6300U o częstotliwości pracy 2.40Ghz.
Wielkość danych wejściowych zaczynała się od 10 do 10^7 Badane były 4 typy danych wejściowych:

- Liczbny losowe $x \in (1...n)$
- Liczby identyczne
- Liczby posortowane
- Liczby posortowane odwrotnie

Pomiary zostały uśrednione z 5 powtórzeń. Najszybszy okazał się Hybrid Sort. Quick Sort po przez odpowieni dobór Pivota nie ogiągnął pesymistycznego przypadku. Najwolniejszym algorytmem okazał się Heap Sort, jedynie dla danych identycznych był zdecydowanie najszybszy.