Data : 26.03.2020
Imię i nazwisko : Kacper Kołodyński
Nr. indeksu : 249018
Nazwa przedmiotu : Projektowanie algorytmów i metody sztucznej inteligencji
Prowadzący : dr inż. Łukasz Jeleń
Zajęcia : Czwartek 9.15 – 11.00

Sprawozdanie

Sortowanie algorytmów

1. Wprowadzanie

Zadanie polega na zaimplementowaniu trzech wybranych przez siebie algorytmów i przeprowadzeniu analizy ich efektywności.

Wybrane przeze mnie algorytmy to:

- Sortowanie przez scalanie
- Sortowanie Szybkie
- Sortowanie Introspektywne

Dla 100 tablic (elementy typu całkowitoliczbowego) o następujących rozmiarach:
10 000, 50 000, 100 000, 500 000 i 1 000 000 wykonujemy eksperymenty z sortowaniem w następujących przypadkach:

- wszystkie elementy tablicy losowe
- 25%, 50%, 75%, 95%, 99%, 99,7% początkowych elementów tablicy jest już posortowanych
- wszystkie elementy tablicy już posortowane ale w odwrotnej kolejności

2. Opisy badanych algorytmów

Algorytm sortowania przez scalanie

Algorytm sortowania przez scalanie jest algorytmem rekurencyjnym. Wykorzystywana jest przy nim zasada „dziel i zwyciężaj”. Algorytm sortujący dzieli podporządkowany zbiór (tablice) na kolejne połowy dopóki taki podział jest możliwy czyli w tym przypadku do uzyskania tablic jednoelementowych. Następnie uzyskane w ten sposób części zbioru są sortowane rekurencyjnie tym samym algorytmem. Gdy sortowanie się zakończy, części są ze sobą łączone poprzez scalanie w taki sposób aby zbiór końcowy był posortowany.

Złożoność

Czasowa: $O(n \cdot \log(n))$ Pamięciowa: $O(n)$

Algorytm sortowania szybkiego

Na początku wybierany jest tzw. element osiowy (pivot). Następnie tablica dzielona jest na dwie podtablice w taki sposób aby elementy leżące w pierwszej części były mniejsze od wszystkich elementów drugiej części zbioru leżącej po prawej stronie od elementu osiowego. Proces dzielenia powtarzany jest aż do uzyskania tablic jednoelementowych, nie wymagających sortowania. Właściwe sortowanie jest tu jakby ukryte w procesie przygotowania do sortowania. Wybór elementu osiowego wpływa na równomierność podziału na podtablice (najprostszy wariant – wybór pierwszego elementu tablicy – nie sprawdza się w przypadku, gdy tablica jest już prawie uporządkowana).

Złożoność

Czasowa: $O(n \cdot \log(n))$
Pesymistycznie: $O(n^2)$

Pamięciowa: Zależnie od implementacji

Algorytm sortowania introspektywnego

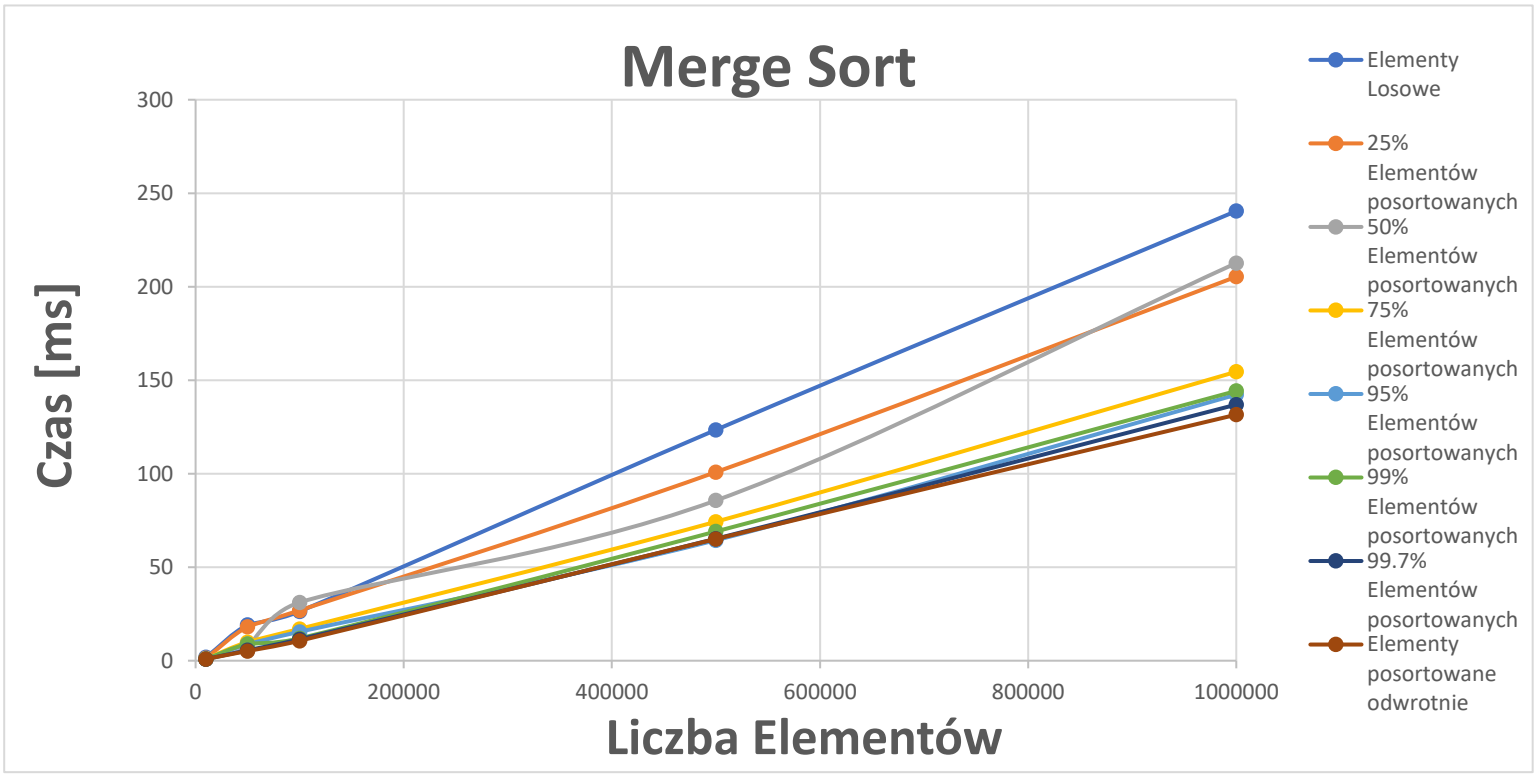
Jest to metoda hybrydowa, będąca połączeniem sortowania szybkiego , sortowania przez kopcowanie oraz sortowania przez wstawianie. Sortowanie introspektywne pozwala uniknąć najgorszego przypadku dla sortowania szybkiego (nierównomierny podział tablicy w przypadku, gdy jako element osiowy zostanie wybrany element najmniejszy lub największy). W sytuacji gdy parametr określający dozwoloną głębokość wywołań rekurencyjnych z poziomu, na którym się obecnie znajdujemy wynosi 0 to wtedy wywołania rekurencyjne są kończone i dla podproblemu, którym się obecnie znajdujemy używamy sortowania przez kopcowanie.
Natomiast gdy wartość tego parametru jest >0 to wtedy wywoływana jest funkcja partition używana w sortowaniu szybkim , która dzieli tablicę na dwa rozłączne podzbiory, gromadząc w pierwszej elementy posiadające klucze mniejsze lub równe od wartości pivota. W drugim podzbiorze gromadzone są elementy o wartościach kluczy większych lub równych pivotowi. Następnie wykonywana jest rekurencyjnie funkcja sortowania introspektywnego z parametrem określającym głębokość zmniejszonym o 1 po czym stosowane jest sortowanie przez wstawianie.

Złożoność

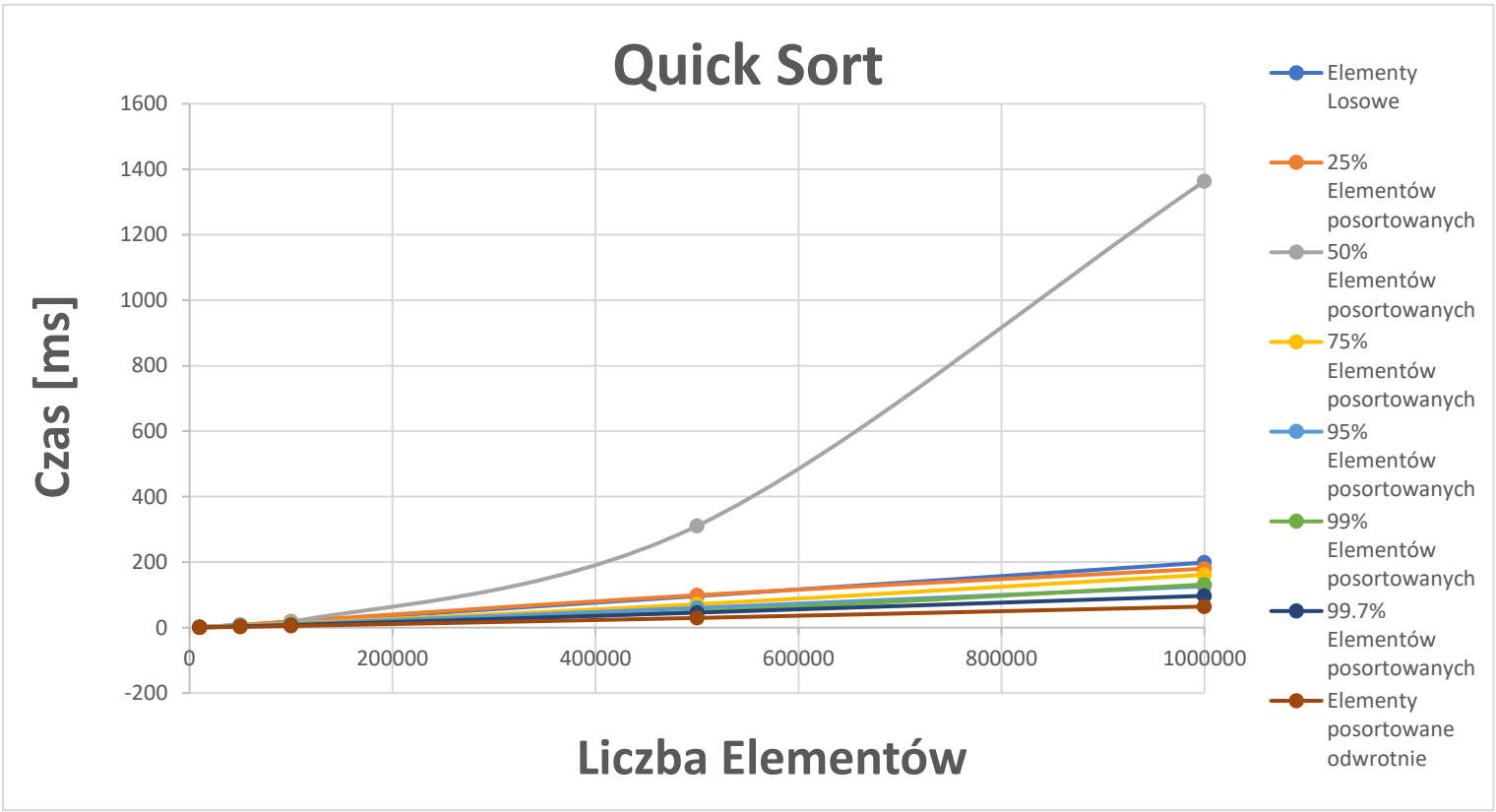
Czasowa: $O(n \cdot \log(n))$,

3. Pomiary

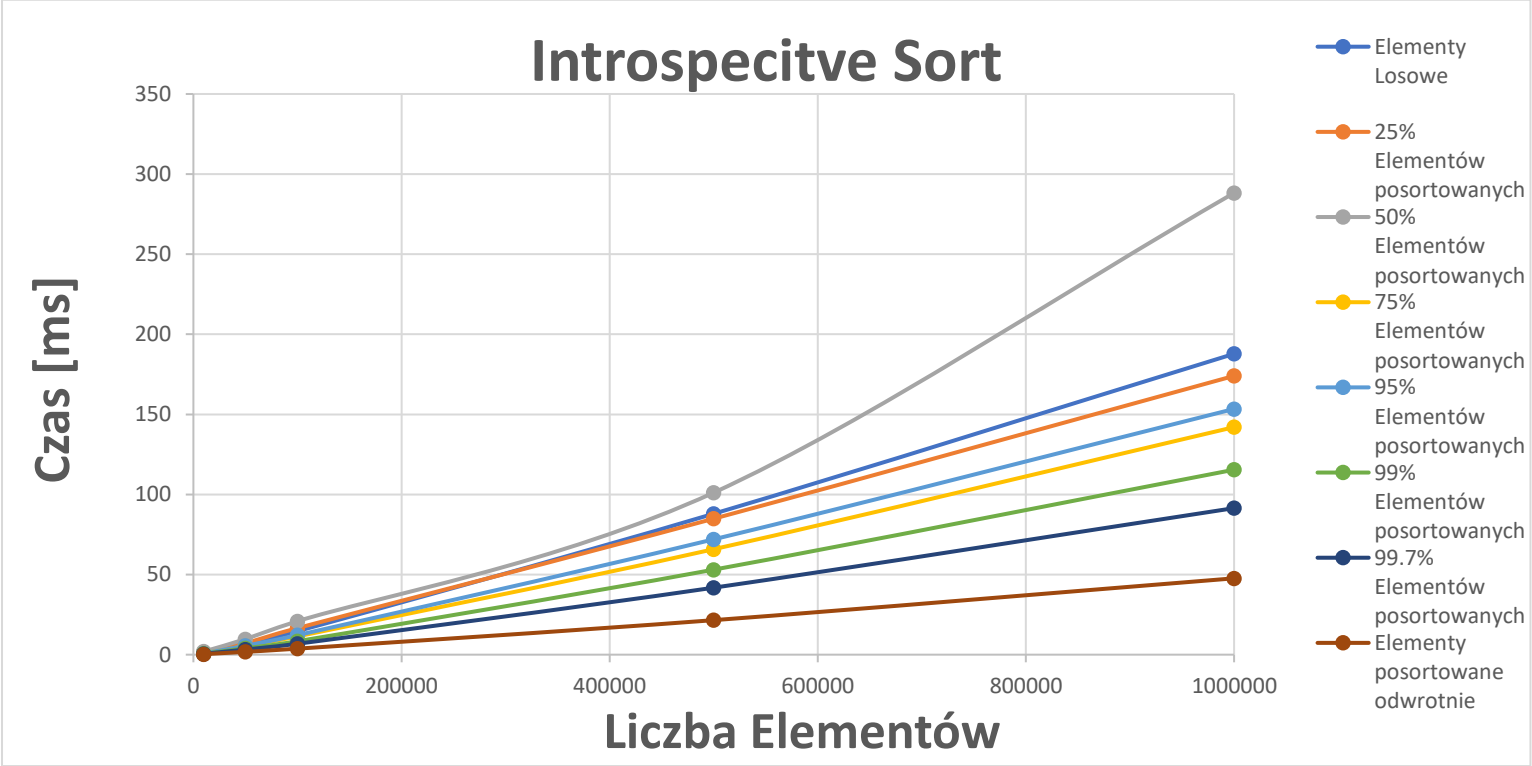
Sortowanie przez scalanie						
	Liczba Elementów	10 000	50 000	100 000	500 000	1 000 000
Posortowane		1.80367	19.1189	26.3663	123.46	240.464
Losowo		1.33885	18.1297	26.937	100.86	205.354
25%		1.25496	7.44574	31.1698	85.6937	212.647
50%		1.09033	9.99706	17.0241	74.3277	154.535
75%		0.854835	8.8508	15.4744	64.4441	142.277
95%		0.860225	8.91314	11.9484	69.1008	144.26
99.7%		0.854933	5.59254	11.4644	65.2364	136.965
Malejąco		0.905389	5.21346	10.67	65.0853	131.571



Sortowanie szybkie						
	Liczba Elementów	10 000	50 000	100 000	500 000	1 000 000
Posortowane		1.40812	8.1676	17.0879	96.2095	198.594
Losowo		1.38617	7.81581	19.2736	99.0074	179.963
25%		1.87292	8.29417	16.8	310.446	1363.47
50%		1.03272	6.06231	12.6814	71.3877	161.448
75%		1.04224	4.96796	10.8628	60.1259	126.477
95%		0.671474	3.85437	8.16929	48.5952	131.712
99.7%		0.639722	3.71122	7.64561	45.6926	97.0774
Malejąco		0.433625	2.31832	5.18083	29.4567	64.3281



Sortowanie introspektywne						
	Liczba Elementów	10 000	50 000	100 000	500 000	1 000 000
Posortowane		1.18032	7.14972	14.6908	87.9766	187.788
Losowo		1.15917	7.10612	16.5909	84.878	174.031
25%		1.95557	9.70964	20.8827	101.153	288.207
50%		0.880022	5.33635	11.4124	65.8535	142.03
75%		0.849031	5.5767	12.1761	72.015	153.387
95%		0.597774	4.02801	8.38917	53.0792	115.452
99.7%		0.516761	3.21179	6.81548	41.825	91.4801
Malejąco		0.320047	1.78057	3.74819	21.5561	47.6599



4. Wnioski

Eksperyment miał na celu zbadanie różnych algorytmów oraz sprawdzenie ich efektywności dla różnych rozmiarów tablic przy zmianie procentów posortowanych wcześniej w tych tablicach elementów.

Zgodnie z oczekiwaniami najgorzej wypadło sortowanie przez scalanie, zarówno dla tablic o mniejszych jak i większych rozmiarach. Najlepszym , najbardziej efektywnym algorytmem okazało się sortowanie introspektywne. Hybrydowa

natura eliminująca najgorszy przypadek sortowania szybkiego pozwoliła osiągnąć tak dobre wyniki temu właśnie algorytmowi.

5. Literatura

https://pl.wikipedia.org/wiki/Sortowanie_szybkie
https://pl.wikipedia.org/wiki/Sortowanie_introspektywne
https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie
https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie
https://www.geeksforgeeks.org/
http://informatyka.2ap.pl/ftp/3d/algorytmy/podr%C4%99cznik_algorytmy.pdf?fbclid=IwAR2EI47e40-RRPqG9z-RWGNPlQILecmffKhXp6QgxEdYGz8FP5YiHvzDays