Krystian Matusiak 249460

Czwartek 9:00-11:00

Dr inż. Łukasz Jeleń

Projektowanie Algorytmów i Metody Sztucznej Inteligencji

Projekt 1 – Algorytmy sortowania

1. Opis projektu

Celem projektu była analiza trzech algorytmów sortowania pod względem efektywności. Test został przeprowadzony dla 100 tablic o następujących rozmiarach:

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

Elementy tablicy stanowiły liczby całkowite. Test dokładniej polegał na wyliczeniu czasu, w jakim program sortował tablice według danego algorytmu, tj. **quick sort, merge sort, introspective sort**. Ponadto, sortowanie dokonano dla poszczególnych przypadków:

- wszystkie elementy losowo rozmieszczone,
- 25%, 50%, 75%, 95%, 99%, 99,7% początkowych elementów już jest posortowane,
- wszystkie elementy są posortowane lecz w odwrotnej kolejności

Dla wszystkich przypadków ustawień elementów sortowanie sprawdzano na tych samych tablicach dla poszczególnych rozmiarów (tj. na jednej tablicy dokonano sortowania 3 algorytmów dla jednego rozmiaru lecz dla wszystkich ustawień elementów).

2. Badane algorytmy

a) Quick sort

Na początku wybierany jest tzw. element osiowy. Następnie tablica dzielona jest na dwie podtablice. Pierwsza z nich zawiera elementy mniejsze od elementu osiowego, druga elementy większe lub równe, element osiowy znajdzie się między nimi. 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 obliczeniowa:

- Najgorszy przypadek **O(n^2)**
- Średni przypadek **O(n log n)**

b) Merge sort

Sortowana tablica dzielona jest rekurencyjnie na dwie podtablice aż do uzyskania tablic jednoelementowych. Następnie podtablice te są scalane w odpowiedni sposób, dający w rezultacie tablicę posortowaną. Wykorzystana jest tu metoda podziału problemu na mniejsze, łatwiejsze do rozwiązania zadania („dziel i rządź”).

Złożoność obliczeniowa:

- Najgorszy przypadek **O(n log n)**
- Średni przypadek **O(n log n)**

c) Introspective sort

Jest to metoda hybrydowa, będąca połączeniem sortowania szybkiego i sortowania przez kopcowanie. Sortowanie introspektywne pozwala uniknąć najgorszego przypadku dla sortowania szybkiego (nierównomierny podział tablicy w przypadku, gdy jako element osiowy zostanie wybrany element najmniejszy lub największy).

Złożoność obliczeniowa:

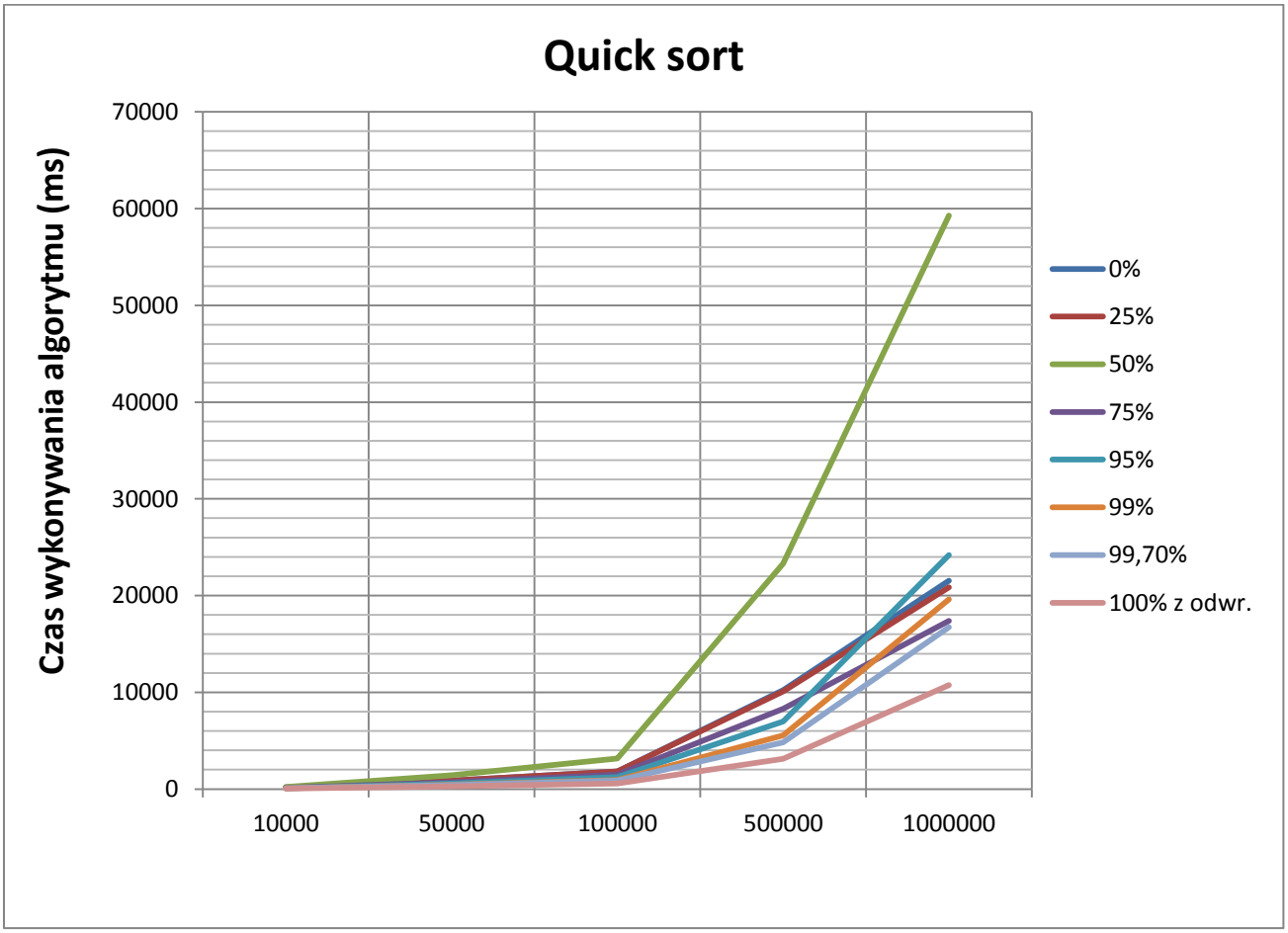
- Najgorszy przypadek **O(n log n)**
- Średni przypadek **O(n log n)**

3. Otrzymane wyniki testu

Czas trwania poszczególnych sortowań zapisano w milisekundach dla dokładniejszej analizy.

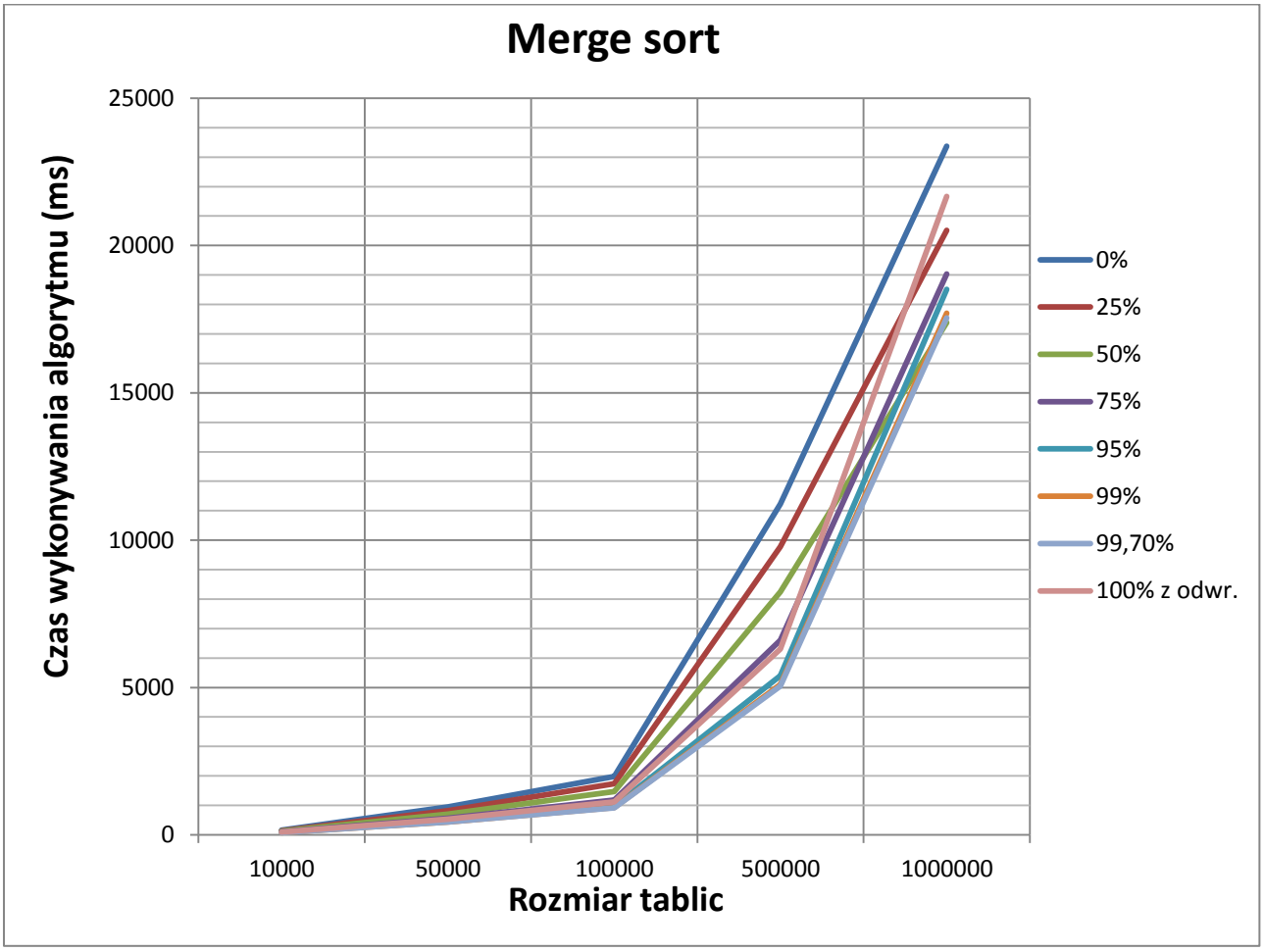
Quick sort:

Quick Sort	10000	50000	100000	500000	1000000
0%	152	868	1825	10219	21536
25%	167	861	1818	10085	20850
50%	202	1418	3169	23310	59269
75%	128	700	1478	8286	17376
95%	98	587	1223	6979	24177
99%	81	447	950	5533	19594
99,70%	75	414	857	4832	16725
100% z odwr.	49	268	563	3112	10732



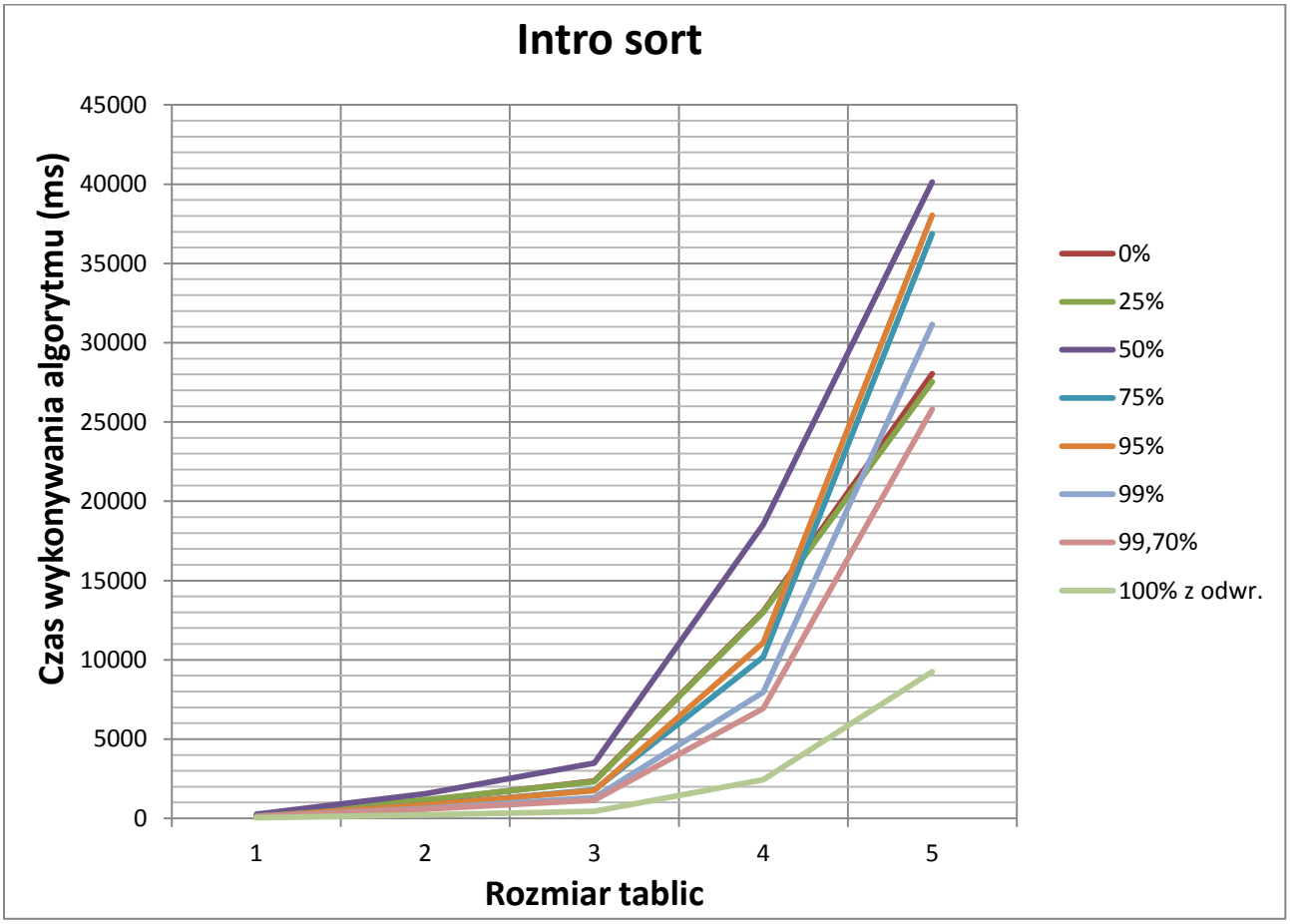
Merge sort:

Merge Sort	10000	50000	100000	500000	1000000
0%	158	943	1984	11225	23374
25%	143	825	1737	9775	20512
50%	124	699	1472	8241	17383
75%	106	562	1183	6603	19032
95%	84	462	965	5401	18508
99%	77	438	922	5101	17705
99,70%	76	439	912	5054	17545
100% z odwr.	92	530	1122	6304	21670



Intro sort:

Intro Sort	10000	50000	100000	500000	1000000
0%	185	1120	2349	13063	28042
25%	189	1164	2309	13000	27550
50%	263	1554	3489	18540	40139
75%	146	849	1786	10180	36878
95%	126	840	1757	11083	38032
99%	105	635	1312	7955	31146
99,70%	99	581	1150	6934	25801
100% z odwr.	43	218	451	2443	9253



4. Wnioski

Analizując powyższe wyniki testu można od razu zauważyć, że dla przypadku, gdy 50% początkowych elementów tablicy jest już posortowane, sortowanie według algorytmu szybkiego oraz introspektywnego trwa o wiele dłużej. Jest to spowodowane tym, że pivot wybierany jest jako element środkowy. W przypadku, gdy pierwsza połowa tablicy jest posortowana, wartość ta jest wysoka.

Podczas, gdy coraz to większy procent elementów jest już posortowanych czas trwania sortowania się skraca ze względu na mniejszą liczbę potrzebnych obliczeń i porównań

W przypadku sortowania przez scalanie można zauważyć, że wykresy są do siebie dosyć zbliżone. Różnica między najdłuższym a najkrótszym przypadkiem, dla miliona elementów, wynosi około 6 sekund, podczas gdy przy sortowaniu szybkim (pomijając przypadek 50%) różnica ta wynosi około 14 sekund. Można po tym wywnioskować, że quick sort oraz introspective sort są optymalne tylko w niektórych przypadkach.

Dla sytuacji, gdy 50% początkowych elementów jest posortowanych, można zauważyć, że złożoność obliczeniowa sortowania szybkiego odpowiada charakterystyce kwadratowej, co potwierdza założenie o najgorszym przypadku złożoności obliczeniowej tego sortowania, a mianowicie **O(n^2)**.

Dla zmodernizowanego algorytmu szybkiego widać, że zastosowanie hybrydowego algorytmu sortowania zniwelowały możliwość pojawienia się przypadku, gdy **O(n^2)**. Na wykresie sortowania introspektywnego można zauważyć, że przypadek 50% początkowo posortowanych elementów ma bardziej zbliżony czas sortowania do pozostałych przypadków.

5. Literatura

http://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie

http://pl.wikipedia.org/wiki/Sortowanie_szybkie

https://pl.wikipedia.org/wiki/Sortowanie_introspektywne

https://pl.wikipedia.org/wiki/Sortowanie_przez_kopcowanie

Jerzy Grębosz Symfonia C++ Standard. Programowanie w języku C++ orientowane obiektowo. Tom I