Sabina Miśkiewicz200577
Wt 18.00

Sprawozdanie z laboratorium projektowanie algorytmów i metod sztucznej inteligencji

Sortowanie.

Wybrałam trzy sortowania: bąbelkowe, szybkie i przez scalanie.
Każdy algorytm został przetestowany dla pięciu różnych rozmiarów danych: 2^{12},2^{14},2^{16},2^{18},2^{20} oraz charakterystycznych warunkach: typowy, korzystny i niekorzystny.

Doświadczenia zostały przeprowadzone na komputerze o następujących parametrach:

- System: 64-bitowy Windows 7
- Procesor: Pentium® Dual-Core CPU T4500 @ 2.30GHz 2.30GHz
- Pamięć RAM: 3,00 GB

Omówienie algorytmów:

Bąbelkowe

Sortowanie to polega na porównywaniu dwóch elementów stojących obok siebie i zamienianiu ich gdy lewy jest większy od prawego oraz pozostawieniu bez zmian w przeciwnym przypadku. Sortowanie kończy się przy podczas kolejnego przejścia przez tablicę, nie zostaną wprowadzone żadne zmiany. Złożoność szacuje się na $O(n^2)$ dla przypadku typowego i jest to ta sama jak dla pesymistycznej. Dla optymistycznej jednak będziemy mieli $O(n)$-$O(n^2)$.

Szybkie

Najpierw wybieramy element osiowy(w moim przypadku jest to środkowy element tablicy). Względem tego elementu dzielimy tablicę na dwie podtablice. Do pierwszej (lewej) trafiają elementy mniejsze od elementu osiowego, a do drugiej (prawej) trafiają elementy większe od elementu osiowego. Ten proces(od wyboru elementu osiowego do podziału na podtablice i wrzuceniu do nich odpowiednich elementów) jest powtarzany, aż do momenty, gdy mamy tablice jednoelementowe. Złożoność obliczeniową szacuje się na O(n*log n) w przypadku typowym. Jednak przy niekorzystnym ułożeniu danych będzie to $O(n^2)$.

Przez scalanie

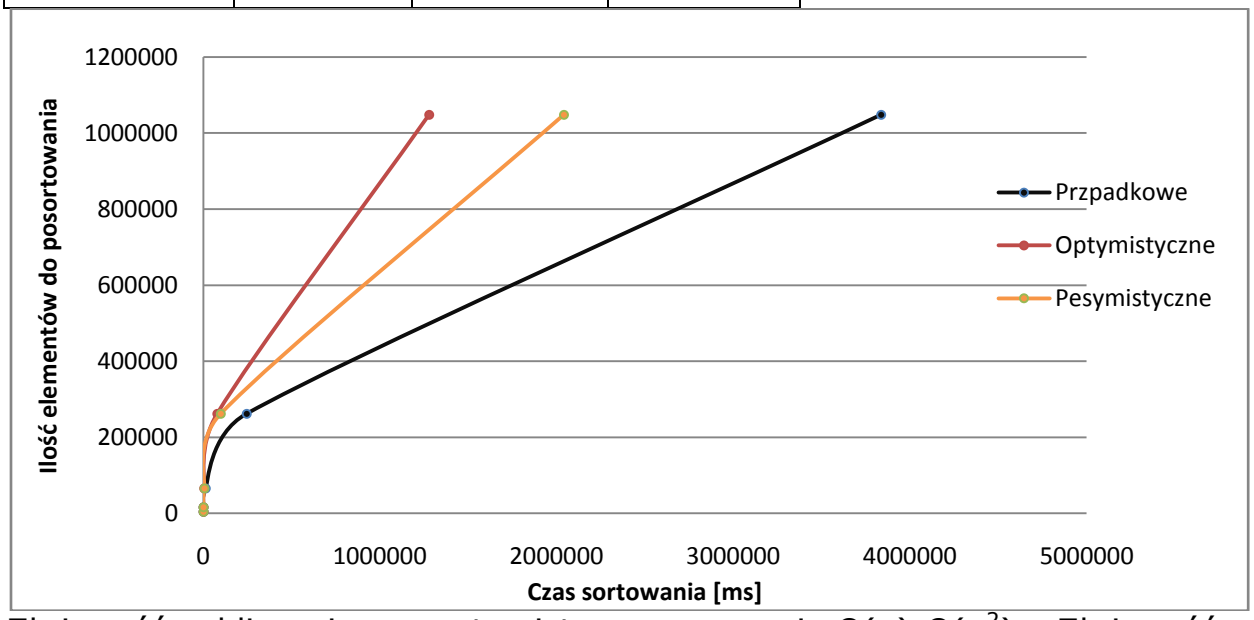
Otrzymaną tablicę, którą należy posortować dzielimy rekurencyjnie na dwie podtablice. Proces powtarzamy aż do otrzymania tablic jednoelementowych. Dopiero teraz podtablice scalamy, dzięki czemu otrzymujemy posortowaną tablicę. Działamy zgodnie z zasadą: „Dziel i zwyciężaj”. Złożoność obliczeniową szacuje się na O(n*log n).

Omówienie przebiegu eksperymentu z przedstawieniem wykresów i tabel

Każdy wynik w tabeli(za wyjątkiem sortowania bąbelkowego gdzie przy dużej ilości danych do posortowania ilość pomiarów została zmniejszona ze względu na długi czas oczekiwania na wynik programu) jest średnią arytmetyczną wyliczoną w arkuszu kalkulacyjnym ze stu pomiarów. Czasy te podane się w [ms].

Sortowanie bąbelkowe

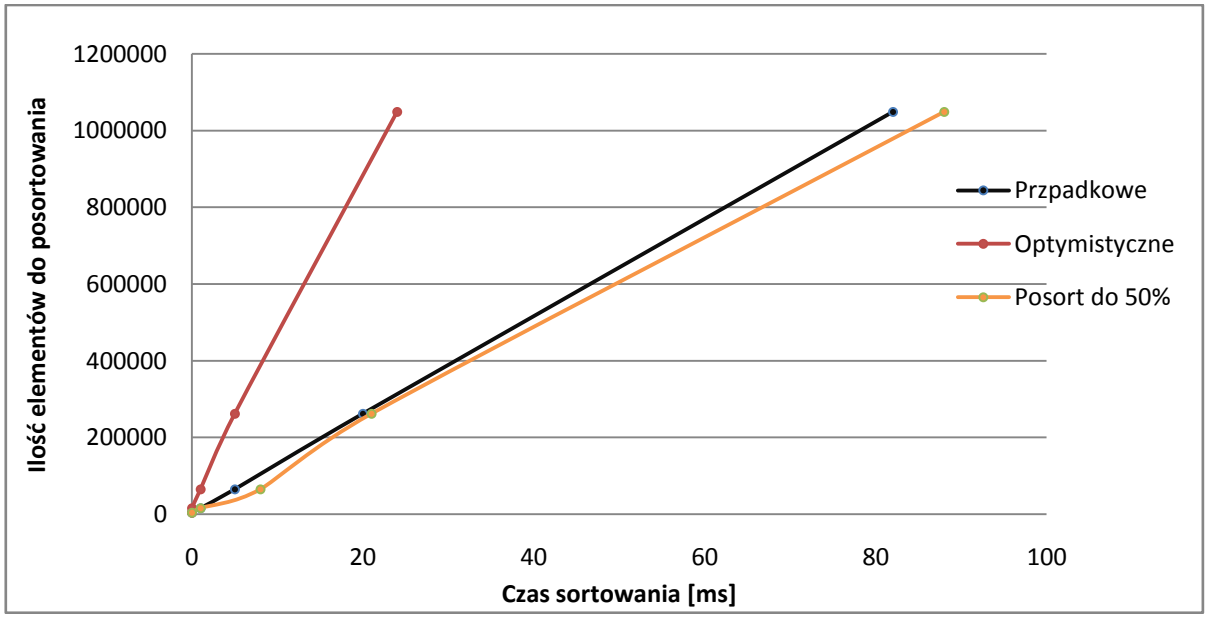
ile danych	Przypadkowe	Optymistyczne	Pesymistyczne
4096	45	15	19
16384	737	257	365
65536	13114	4148	5864
262144	244902	78324	98504
1048576	3835352	1277516	2040234



Złożoność obliczeniowa optymistyczna wynosi O(n)-O(n^2). Złożoność obliczeniowa typowa i pesymistyczna są sobie równe i wynoszą O(n^2). Przewidywany przebieg eksperymentu zakładał, że przypadek typowy znajdzie się pomiędzy optymistycznym a pesymistycznym. Mogło to zostać zaburzone przez zmniejszanie ilości wykonywanych pomiarów ze względu na zbyt duży czas oczekiwania na odpowiedź programu.

Sortowanie szybkie

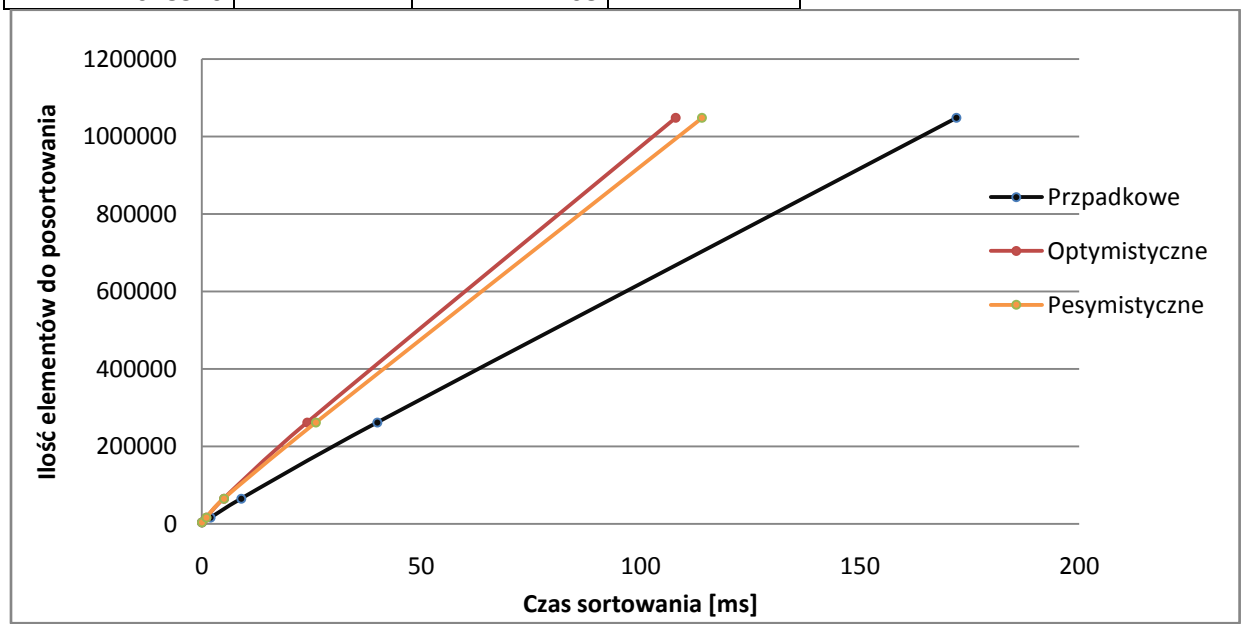
ile danych	Przypadkowe	Optymistyczne	Posort do 50%
4096	0	0	0
16384	1	0	1
65536	5	1	8
262144	20	5	21
1048576	82	24	88



Złożoność obliczeniową szacuje się na O(n log n) dla przypadku typowego i optymistycznego. Natomiast dla pesymistycznego wynosi ona $O(n^2)$. Dla sortowania szybkiego wszystkie czasu zostały uśrednione ze 100 pomiarów. W tabeli pojawiają się nawet zerowe czasy. Jest to spowodowane niewątpliwie tym, że czas wykonywania sortowania jest tak krótki, że wybiega poza zakres mierzalny. Nie został przetestowany w tym przypadku przypadek pesymistyczny, ponieważ był trudny do wygenerowania.

Sortowanie przez scalanie

ile danych	Przypadkowe	Optymistyczne	Pesymistyczne
4096	0	0	0
16384	2	1	1
65536	9	5	5
262144	40	24	26
1048576	172	108	114



Złożoność obliczeniową szacujemy na O(n log n) dla każdego z przypadku(typowego, optymistycznego oraz pesymistycznego). Ponownie w tabeli zaobserwowaliśmy zera, co tłumaczymy w ten sam sposób, jak przy sortowaniu szybkim.

Wady i zalety

Bąbelkowe

Zalety: Nieskomplikowany algorytm, jeden z najprostszych. Krótki kod.
Wady: Najwolniejszy z algorytmów jakie testowałam. Gdy element najmniejszy znajdzie się na końcu tablicy musi zostać przesunięty element po elemencie i z każdym zostać porównany. Nawet gdy tablica jest posortowana dokona się porównanie każdego poprzedniego elementu z kolejnym. Niewydajny dla dużej ilości danych.

Szybkie

Zalety: Niewątpliwie najszybszy algorytm sortujący. Co wykazały eksperymenty. Stosowany dla dużego rozmiaru tablic nie zwiększa czasu tak drastycznie, jak np. sortowanie bąbelkowe. Nie jest skomplikowany.
Wady: Wolniejszy w przypadku wybieraniu ciągle jako elementu osiowego elementu najmniejszego lub największego.

Przez scalanie

Zalety: Wydajny algorytm sortujący. Typu „Dziel i zwyciężaj”.
Wady: Do swojego działania potrzebuje dodatkowej pomocniczej struktury danych. Słabiej radzi sobie z większą ilością danych.

Wioski i posumowanie

Każdy algorytm ma swoje wady i zalety. Nie można jednak wybrać bezwzględnie najlepszego algorytmu, a jego wybór zależy od danych wejściowych.
W przypadku moich testów najszybszy okazał się algorytm sortowania szybkiego. Wybierając z tym przypadku za element osiowy element środkowy jest zagwarantowaniem, że unikniemy przypadku pesymistycznego. Najwolniejsze okazało się sortowanie bąbelkowe. Jego prostota przekłada się na duży czas działania.
Dla sortowania bąbelkowego najszybciej posortowana została tablica już uporządkowana, a najwolniej przypadkowa.
Dla sortowania szybkiego najszybciej posortowana została tablica również już uporządkowana, a najwolniej posortowana do połowy.
Dla sortowania przez scalanie najszybciej posortowana została (identycznie jak wcześniej) tablica uporządkowana, a najwolniej wypełniona przypadkowymi danym.

Wykorzystane materiały i źródła:

- Thomas H. Cormes, Charles E. Leiserson, Ronald L. Rivest “Wprowadzenie do algorytmów” wydanie czwarte
- Adam Drozdek “C++ algorytmy I struktury danych” wydawnistwa Helion