Politechnika Wrocławska

Projektowanie i Analiza Algorytmów

Projekt I

Piotr Brajer 272538

W4 ISA

Wt 15:15-16:65

05-04-2024

1. Wybrane Algorytmy:

Link do GitHub

a) Sortowanie przez Scalanie(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").

b) Sortowanie kubełkowe (Bucket sort)

Sortowanie kubełkowe to algorytm sortowania, który działa na liczbach równomiernie rozłożonych w przedziale [0, 1). Polega na podziale tego przedziału na niewielką liczbę kubełków, do których umieszczane są elementy z tablicy na podstawie ich wartości. Następnie zawartość każdego kubełka jest sortowana (często innym algorytmem sortowania), a wyniki są łączone, tworząc posortowaną tablicę. Sortowanie kubełkowe jest efektywne dla równomiernie rozłożonych danych w określonym przedziale, ale wymaga dodatkowej pamięci na przechowywanie kubełków.

c) Sortowanie Szybkie (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).

d) Sortowanie Introspektywne (Intro 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).

2. Złożoność Algorytmów

przypadki	MergeSort	QuickSort	IntroSort	BucketSort
najgorszy	$O(n\ log\ n)$	$O(n^2)$	$O(n\ log\ n)$	$O(n\ log\ n)$
średni	$O(n\ log\ n)$	$O(n\ log\ n)$	$O(n\ log\ n)$	$O(n\ log\ n)$
optymalny	$O(n\ log\ n)$	$O(n\ log\ n)$	$O(n\ log\ n)$	$O(n\ log\ n)$

Tabela 1: Porównanie złożoności obliczeniowej algorytmów

3. wyniki

	MergeSort		
n	czas	avg	mediana
10k	21,6972	5,4603	5
100k	241,601	6,06428	6
500k	1386,84	6,65549	7
1m	3059,88	6,63432	7
all	3174,24	6,63661	7

Tabela 2: Wyniki pomiarów MergSort

	QuickSort		
n	czas	avg	mediana
10k	10,8524	5,4603	5
100k	111,328	6,06428	6
500k	598,155	6,65549	7
1m	1255,51	6,63432	7
all	1344,16	6,63661	7

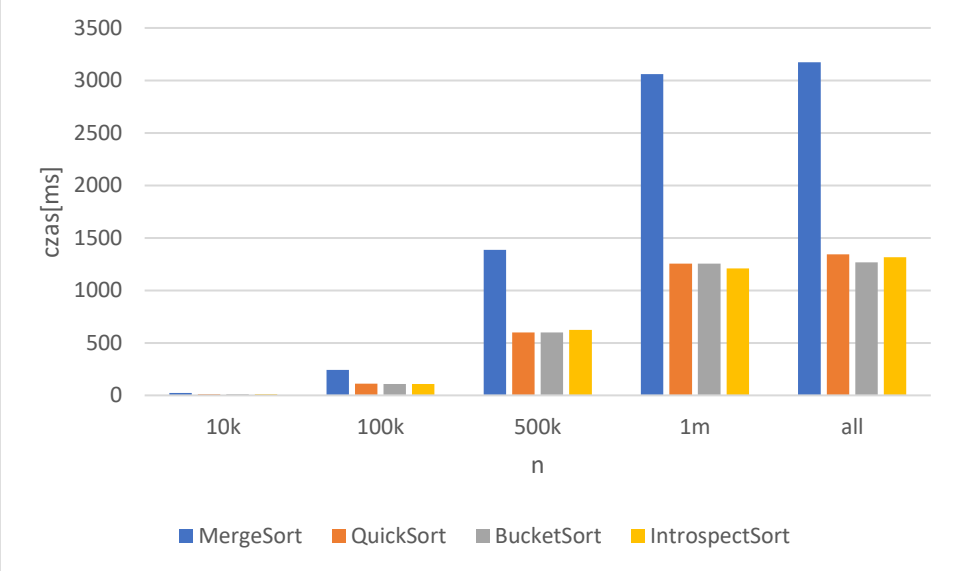
Tabela 3: Wyniki pomiarów QuickSort

	BucketSort		
n	czas	avg	mediana
10k	10,8124	5,4603	5
100k	109,637	6,06428	6
500k	600,871	6,65549	7
1m	1255,67	6,63432	7
all	1268,44	6,63661	7

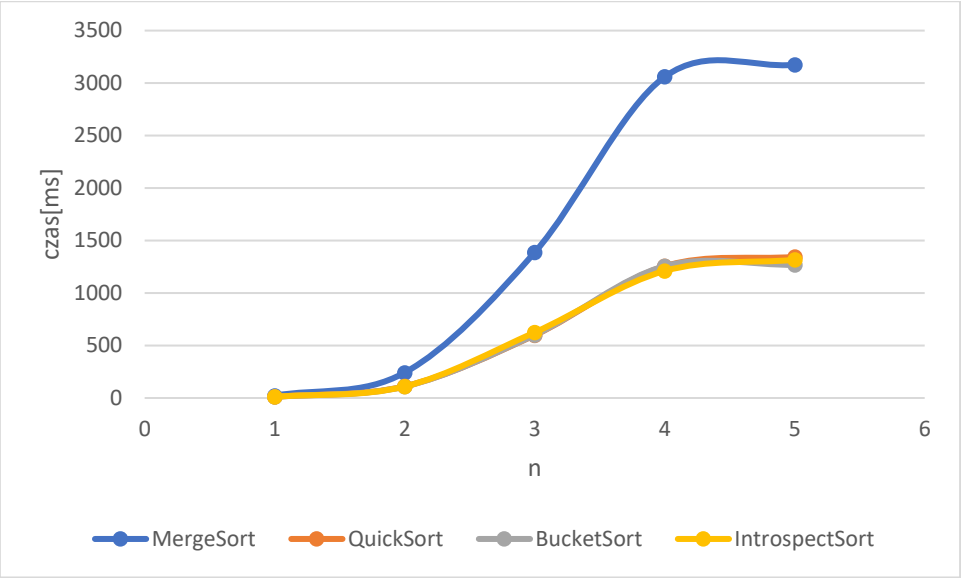
Tabela 4: Wyniki pomiarów BucketSort

	IntrospectSort		
n	czas	avg	mediana
10k	14,3723	5,4603	5
100k	147,259	6,06428	6
500k	751,654	6,65549	7
1m	1615,3	6,63432	7
all	1862,4	6,63661	7

Tabela 5: Wyniki pomiarów IntroSort



Rys.1: Czas sortowania na ilość danych



Rys.2: Wykres porównawczy czas sortowania algorytmów

4. Wnioski

Sortowanie introspektywne jest optymalnym wyborem dla losowych danych, umożliwiając skuteczne wykorzystanie sortowania szybkiego dla większych zbiorów i sortowania przez wstawianie dla mniejszych. Jest lepsze niż sortowanie szybkie dla małych tablic, gdyż może zastosować sortowanie przez wstawianie, ograniczając liczbę rekurencyjnych wywołań. Sortowanie przez scalanie jest najwolniejsze dla losowych danych, chociaż różnice między algorytmami są niewielkie. Wraz ze wzrostem stopnia uporządkowania danych, sortowanie przez scalanie staje się szybsze, a sortowanie szybkie trwa dłużej, choć różnica ta nie jest znacząca. Sortowanie introspektywne, choć bazuje na sortowaniu szybkim, może działać gorzej dla bardziej posortowanych danych, ale dla prawie uporządkowanych tablic nie jest to drastyczne. Dzięki ograniczeniu głębokości rekursji i wykorzystaniu sortowania przez kopcowanie, czas sortowania może się skrócić. Drobną różnicę w implementacji między sortowaniem szybkim a introspektywnym może prowadzić do nieco gorszej wydajności introsortu, szczególnie przy dużej liczbie rekurencyjnych wywołań. Podsumowując, wybór algorytmu zależy od charakterystyki danych, a sortowanie introspektywne często okazuje się być dobrą opcją ze względu na adaptacyjność i uniwersalność.