SPRAWOZDANIE - LISTA 1

Zuzanna Pawlik, 282230

23.10.2024

FRAGMENTY KODÓW

INSERTIONSORT

Fragment kodu INSERTIONSORT: Fragment kodu zmodyfikowanej wersji INSERTIONSORT wstawiającej dwa elementy naraz: W tej wersji algorytmu wykorzystujemy fakt, że dwa elementy wstawiane jednocześnie są już posortowane, zatem szukając miejsca na drugi, zaczynamy od pozycji, na którą wstawiono pierwszy z nich.

MERGESORT

Fragment algorytmu MERGESORT: Wykorzystujemy funkcję pomocniczą MERGE pozwalającą na połączenie dwóch posortowanych tablic (tutaj L i P) w jedną. Funkcja MERGESORT działa na zasadzie rekurencyjnych wykonań, dzięki czemu zaczynamy od małych posortowanych tablic i "budujemy" z nich na nowo początkową tablicę.
Fragment zmodyfikowanej funkcji dzielącej tablicę na tej samej zasadzie jak w MERGESORT, ale tym razem na 3 części:

HEAPSORT

Algorytm HEAPSORT bazuje na utworzeniu kopca z elementów listy do posortowania. Wykorzystuje on w tym celu funcje pomocnicze HEAPIFY oraz BUILDHEAP, które odpowiedznio pozwalają "naprawić" kopiec i "zbudować" kopiec z elementów listy. Następnie algorytm sortowania wypisuje elementy kopca w odpowiedniej kolejności. Poniższa wersja algorytmu działa na tej samej zasadzie co klasyczny HEAPSORT, ale elementy kopca mają po 3 "potomków", zamiast 2.

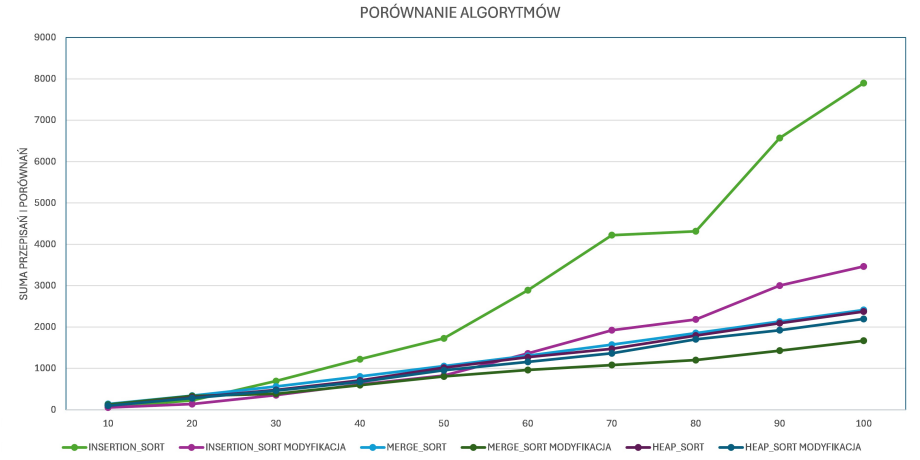
TABELE I WYKRESY

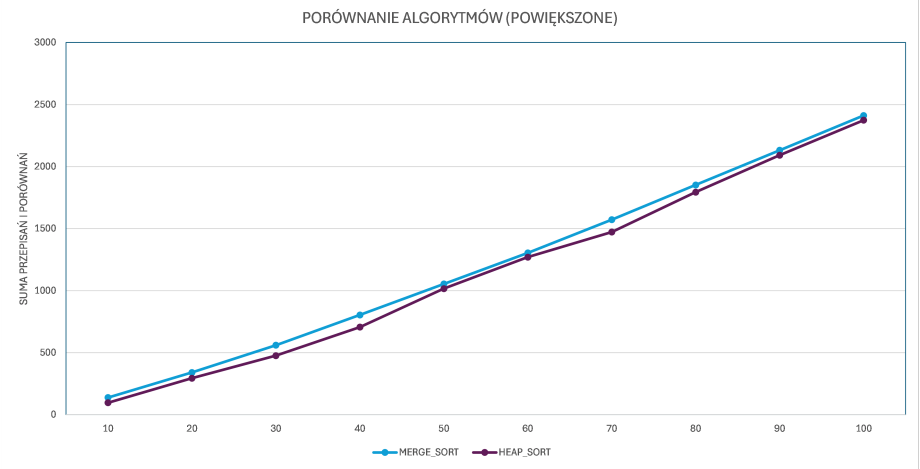
TABELA ZAWIERAJĄCA DANE DO WYKRESÓW

	INSERTION_SORT			INSERTION_SORT MODYFIKACJA			MERGE_SORT			MERGE_SORT MODYFIKACJA			HEAP_SORT			HEAP_SORT MODYFIKACJA		
Długość listy	Porównania	Przypisania	SUMA	Porównania	Przypisania	SUMA	Porównania	Przypisania	SUMA	Porównania	Przypisania	SUMA	Porównania	Przypisania	SUMA	Porównania	Przypisania	SUMA
10	19	41	60	12	40	52	34	104	138	76	52	128	52	44	96	63	36	99
20	68	153	221	43	93	136	88	252	340	192	138	330	156	138	294	174	104	278
30	221	471	692	146	216	362	148	412	560	192	192	384	252	224	476	288	174	462
40	394	827	1221	366	266	632	216	588	804	292	392	684	372	334	706	420	294	714
50	599	1367	1726	353	473	826	288	768	1056	392	412	804	532	484	1016	591	362	953
60	943	1965	2888	603	757	1360	368	948	1304	480	480	960	664	606	1270	717	440	1157
70	1384	2837	4221	875	1047	1922	432	1140	1572	560	528	1088	770	702	1472	846	518	1364
80	1412	2903	4315	997	1185	2182	512	1340	1852	640	560	1200	936	858	1794	1053	650	1703
90	2160	4409	6569	1392	1610	3002	680	1140	2132	756	672	1428	1090	1002	2092	1188	734	1922
100	2600	5299	7899	1698	1896	3464	672	1740	2412	876	792	1668	1236	1138	2374	1356	838	2194

W powyższej tabeli oraz poniższych wykresach wykorzystano wartości zwracane przez funkcje zliczające przypisania, oraz porównania w każdej z omawianych wyżej wersji algorytmów sortujących.

WYKRESY





WNIOSKI

Jak widać z powyższych wykresów oraz tabeli pomimo, że dla krótkich list algorytm INSERTIONSORT wydawał się wydajniejszy to jednak dla większej ilości sortowanych elementów algorymy MERGESORT i HEAPSORT okazały się działać znacznie szybciej. Ponadto można zauważyć, że wprowadzone do algorytmów modyfikacje znacznie przyspieszyły ich działanie co jest widoczne zwłaszcza w przypadku algorytmu INSERTIONSORT. Dodatkowo warto zaznaczyć, że pomimo podobnego czasu wykonywania się MERGESORT i HEAPSORT, ten drugi jest bardziej wydajny, ponieważ poza szybkim wykonaniem nie zapisuje on tablic tymczasowych (jak P i L w MERGU) zatem jest bardziej oszczędny na pamięci. Zależnie zatem od warunków sprzętowych najlepszymi z rozważanych algorytmów są modyfikacje algorytmów MERGESORT i HEAPSORT.