

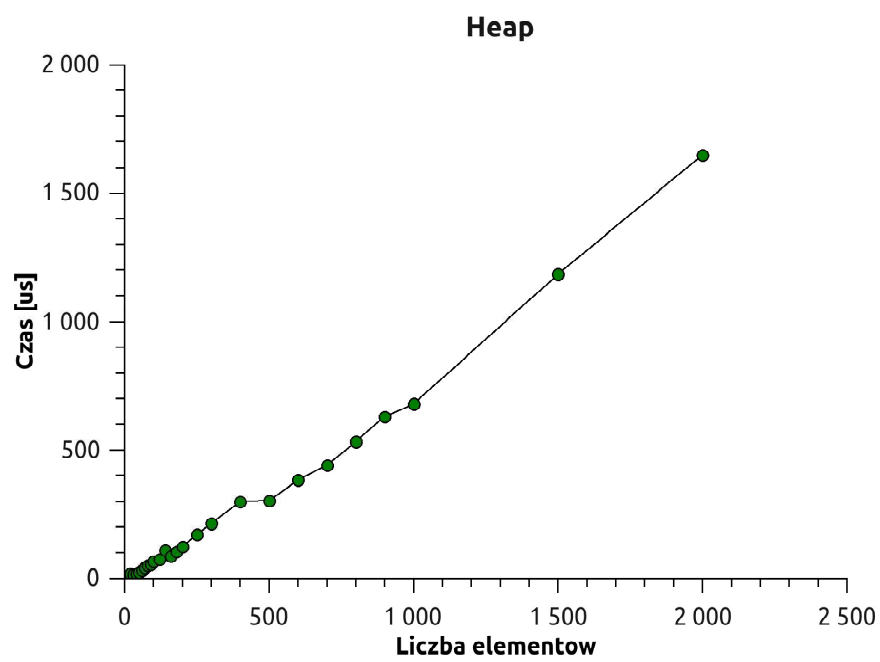
Sprawozdanie z laboratorium PAMSI

Damian Oleksak

Sortowanie heapsort polega na wykorzystaniu kopca binarnego. Na początku budowany jest z wszystkich elementów kopiec, którego korzeń jest elementem największym. Następnie kolejne największe elementy są wstawiane na odpowiednie miejsca w tablicy, natomiast elementy zajmujące te miejsca są wstawiane na szczyt kopca i jest on odbudowywany w dół za pomocą operacji Heapify.

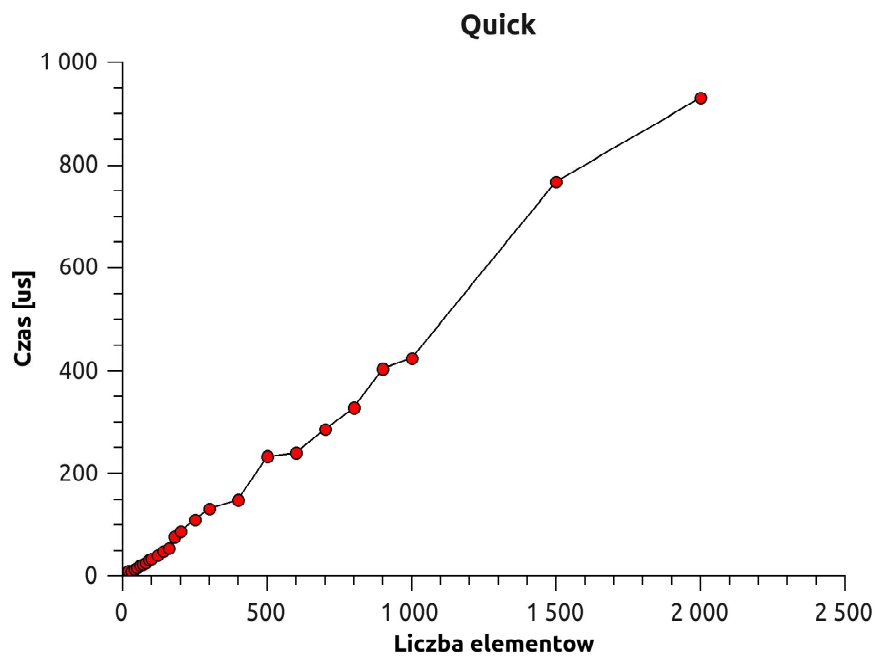
Algorytm budowy kopca działa w ten sposób, że buduje najpierw małe kopce, z elementów znajdujących się na końcu tablicy, i później - idąc "w górę" tablicy - kopce te łączone są w większe, aż do powstania całego kopca. Ponieważ elementy znajdujące się w "dolnej" połowie tablicy są liśćmi kopca - nie trzeba z nich budować małych kopców (są kopcami jednoelementowymi).

Algorytm sortowania przez kopcowanie zalicza się do algorytmów szybkich. Jego złożoność jest liniowo-logarytmiczna - $O(n \cdot \log(n))$. Jest około 3 razy wolniejszy od sortowania szybkiego.



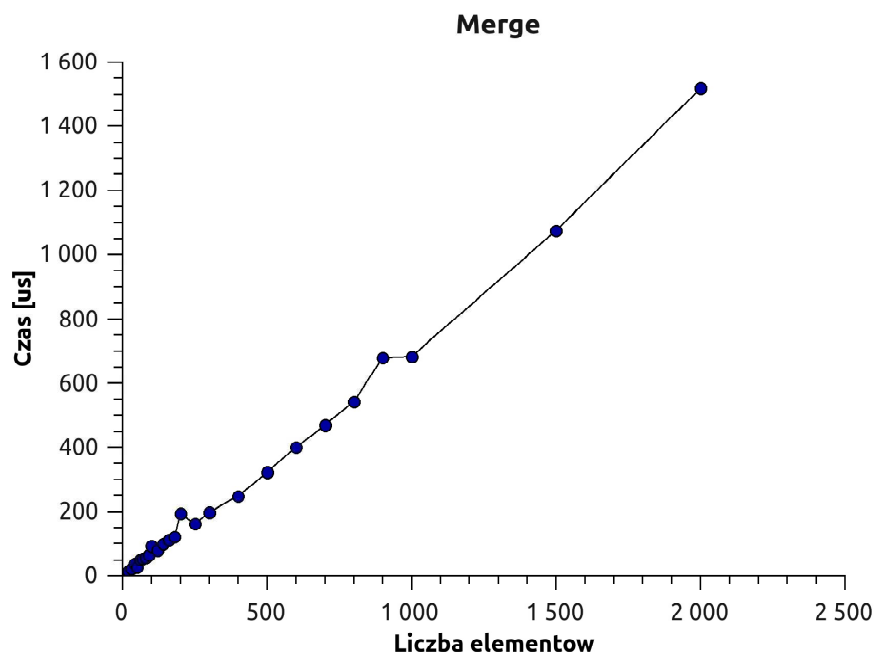
Wykres 1. Sortowanie przez kopcowanie

Sortowanie szybkie (quicksort) jest używane w sytuacjach, gdy sortuje się duże ilości danych – gdy danych jest niewiele, proste algorytmy sortowania są efektywniejsze. Jako granicę sensowności stosowania sortowania algorytmem quicksort przyjmuje się zwykle ilość sortowanych elementów, wynoszącą minimum 10. Złożoność obliczeniowa jest rzędu $O(n \cdot \log(n))$, lecz w przypadku pesymistycznym $O(n^2)$ - sortowanie przez kopcowanie okazuje się szybsze.



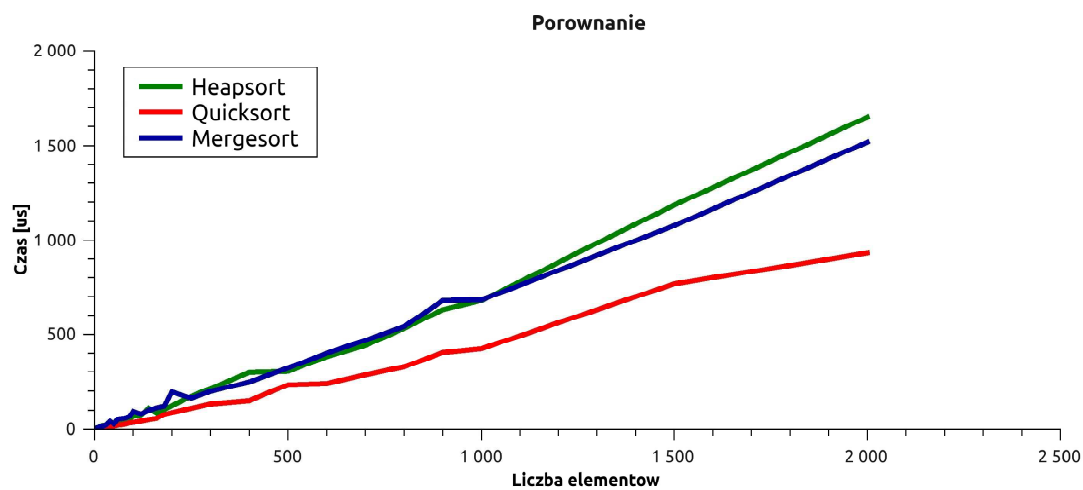
Wykres 2. Sortowanie szybkie

Sortowanie przez scalanie, podobnie jak algorytm quicksort, jest algorytmem typu "dziel i zwyciężaj". Jednak w odróżnieniu od sortowania szybkiego, algorytm ten w każdym przypadku osiąga złożoność $O(n \cdot \log(n))$. Niestety algorytm mergesort posiada większą złożoność pamięciową, potrzebuje do swojego działania dodatkowej, pomocniczej struktury danych. Ideą działania algorytmu jest dzielenie zbioru danych na mniejsze zbiory, aż do uzyskania n zbiorów jednoelementowych. Następnie zbiory te są łączone w coraz większe zbiory posortowane, aż do uzyskania jednego, posortowanego zbioru n -elementowego.



Wykres 3. Sortowanie przez scalanie

Wykresy zostały wygenerowane na podstawie pomiarów czasu sortowania losowo wygenerowanych elementów od ich liczby.



Wykres 4. Porównanie wszystkich algorytmów sortowania