

Analiza eksperymentalna nowych algorytmów sortowania w miejscu

Damian Baliński

Praca napisana pod kierunkiem
dra Zbigniewa Gołębiewskiego

2022, Wrocław

Motywacja

Cele tworzenia algorytmów hybrydowych

- *poprawa wydajności*
- *optymalizacja przypadku pesymistycznego*
- *działanie w miejscu*
- *minimalizacja liczby porównań*

Metodologia

Łączny koszt operacji

$$C = \alpha n_c + 3n_s + n_a$$

α – wartość współczynnika kosztu

n_c – liczba operacji porównania

n_s – liczba operacji zamiany miejsc

n_a – liczba operacji przypisania

Wady algorytmów podstawowych

Wady algorytmu Quick Sort

- $O(n^2)$ dla przypadku pesymistycznego
- duża liczba porównań

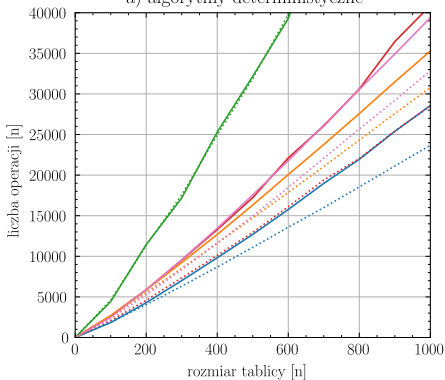
Wady algorytmu Merge Sort

- Konieczność posiadania dodatkowej pamięci o wielkości $O(n)$
- Dodatkowy nakład czasowy związany z alokacją oraz zwalnianiem pamięci

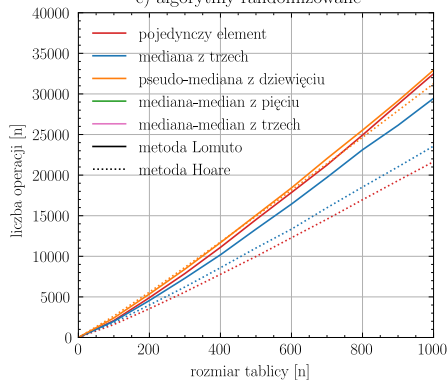
Rodzina algorytmów Quick Sort

Łączna liczba operacji wykonanych przez algorytmy z rodziny Quick Sort z podziałem na metody partycjonowania oraz polityki wyboru pivotu dla tablicy uporządkowanych danych

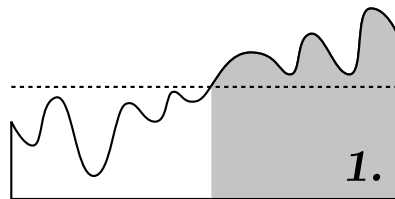
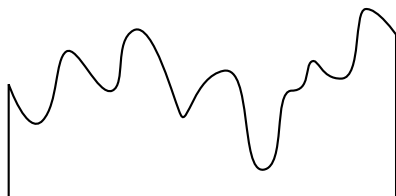
a) algorytmy deterministyczne



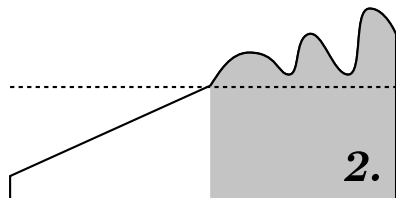
c) algorytmy randomizowane



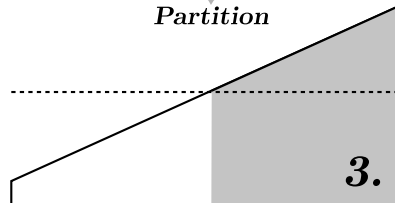
QuickMerge Sort



Partition



Merge Sort

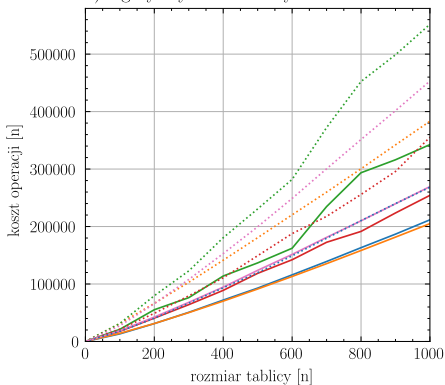


QuickMerge Sort

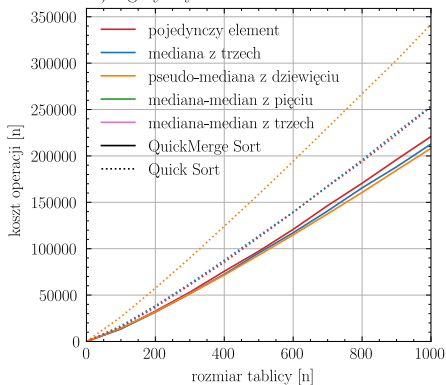
Rodzina algorytmów QuickMerge Sort

Łączny koszt operacji wykonanych przez algorytmy z rodziny QuickMerge Sort z podziałem na metody partycjonowania oraz polityki wyboru pivotu dla tablicy uporządkowanych danych przy współczynniku kosztu równym $\alpha = 10.0$

a) algorytmy deterministyczne - Lomuto



c) algorytmy randomizowane - Lomuto



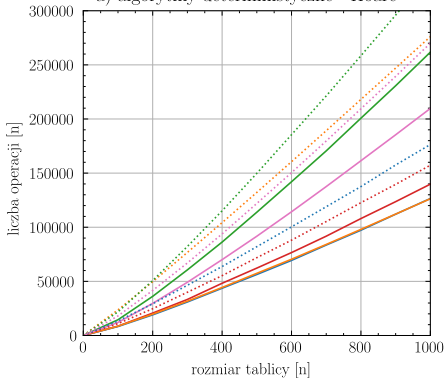
Intro Sort

```
1: procedure INTROSORT(ARR, DEPTH)
2:
3:   if  $\text{len}(\text{arr}) < \text{maxLength}$  then           ▷ sort. przez wstawianie
4:     InsertionSort(arr)
5:
6:   else if  $\text{depth} = 0$  then                       ▷ sort. przez kopcowanie
7:     HeapSort(arr)
8:
9:   else                                           ▷ szybkie sortowanie
10:     $\text{arr}_1, \text{arr}_2 \leftarrow \text{Partition}(\text{arr})$ 
11:    IntroSort(arr1, depth-1)
12:    IntroSort(arr2, depth-1)
```

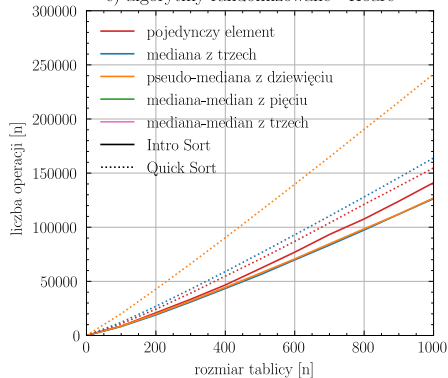

Rodzina algorytmów Intro Sort

Łączna liczba operacji wykonanych przez algorytmy
z rodziny Intro Sort z podziałem na polityki
wyboru pivotu dla tablicy losowych danych

a) algorytmy deterministyczne - Hoare



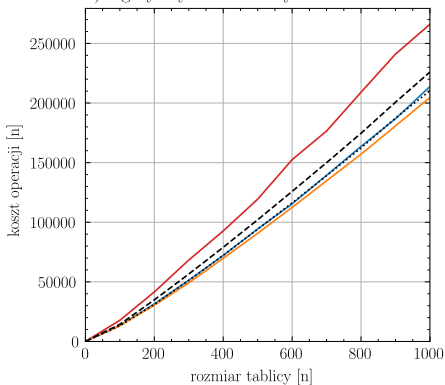
c) algorytmy randomizowane - Hoare



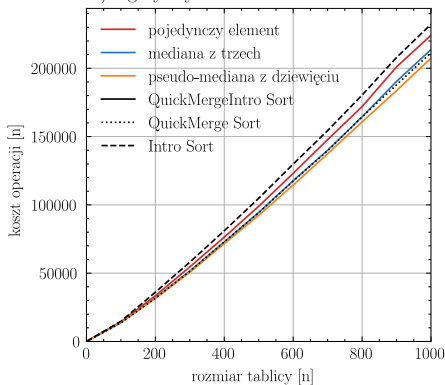
Rodzina algorytmów QuickMergeIntro Sort

Łączny koszt operacji wykonanych przez algorytmy z rodziny QuickMergeIntro Sort z podziałem na metody partycjonowania dla tablicy uporządkowanych danych przy współczynniku kosztu równym $\alpha = 10.0$

a) algorytmy deterministyczne - Lomuto



c) algorytmy randomizowane - Lomuto



Propozycje dalszego rozwoju aplikacji

- *wykorzystanie algorytmów wielopiwotowych, np. Dual Pivot Quick Sort*
- *aplikacja webowa do wizualizacji sortowania w czasie rzeczywistym*

Bibliografia

- [EW18a] Stefan Edelkamp and Armin Weiß.
Quickmergesort: Practically efficient constant-factor
optimal sorting.
2018.
- [EW18b] Stefan Edelkamp and Armin Weiß.
Worst-case efficient sorting with quickmergesort.
2018.
- [WE19] Sebastian Wild, Armin Weiß and Stefan Edelkamp.
Quickxsort – a fast sorting scheme in theory and
practice.
2019.

Dziękuję za uwagę.