Porównanie wydajności algorytmów sortowania

Mikołaj Kikolski, 261830

13 maja 2021 r.

1 Wstęp

W zadaniu zostaliśmy poproszeni o zmodyfikowanie algorytmu sortowania szybkiego przedstawionego na wykładzie, przez wstawienie do niego sprawdzenia, zmieniającego algorytm sortowania na zaimplementoway przez nas na poprzedniej liście algorytm sortowania bąbelkowego lub sortowania przez wstawianie dla krótkich list. Następnie, należało porównać skuteczność algorytmu z wykładu, zmodyfikowanego quicksorta oraz bubblesorta przy sortowaniu losowo wygenerowanych list. Zadanie podzieliłem na kilka etapów, które opiszę w następnej sekcji.

2 Kroki rozwiązywania problemu

Na początku, określiłem trudności, które mogę napotkać podczas rozwiązywania zadania oraz dodatkowe zależności, które mógłbym zbadać jako rozwinięcie rozwiązania problemu.

Pojawiły się następujące pytania:

- *Jak długa jest krótka lista?*
- *Jakich długości list użyć?*
- *Z jak dużego zakresu liczb wybierać liczby do sortowania?*
- *Jak to zrobić żeby się nie narobić, czyli jak zbierać dane żey wygodnie je analizować?*

3 Początkowe ustalenia

Po przeprowadzeniu kilku prób, doszedłem do następujących wniosków i odpowiedzi na pytania:

Jak długa jest krótka lista?

Początkowo chciałem rozwiązać sprawę matematycznie i na stronie `wolframalpha.com`, sprawdziłem jakie rozwiązanie ma nierówność pomiędzy średnimi złożonościami obliczeniowymi danych algorytmów. Dla quicksorta, jest to $nlog(n)$, a dla bubblesorta - n^2. Okazuje się, że nierówność $nlog(n) > n^2$, nie ma rozwiązań rzeczywistych. Oznacza to, że teoretycznie, niezależnie od długości listy, quicksort radzi sobie lepiej. Teoria jednak nie zawsze ma odzwierciedlenie w praktyce, więc na potrzeby testów przyjąłem, że krótka lista, to taka, gdzie $n < 10$. Dodatkowo, jako rozwinięcie eksperymentu, można sprawdzić jak zmieniają się różnice w wydajności w zależności od granicznego n

Jakich długości list użyć?

Po pierwszych testach, w których użyłem zaproponowanych w treści zadania długości: 10, 10^2, 10^3 i 10^4, zauważyłem, że obie wersje quicksorta dają dla wszystkich prób na liście dziesięcioelementowej, czas równy 0s. Dlatego zdecydowałem się użyć długości 10^2, 10^3, 10^4 i 10^5.

Z jakiego zakresu wybierać liczby?

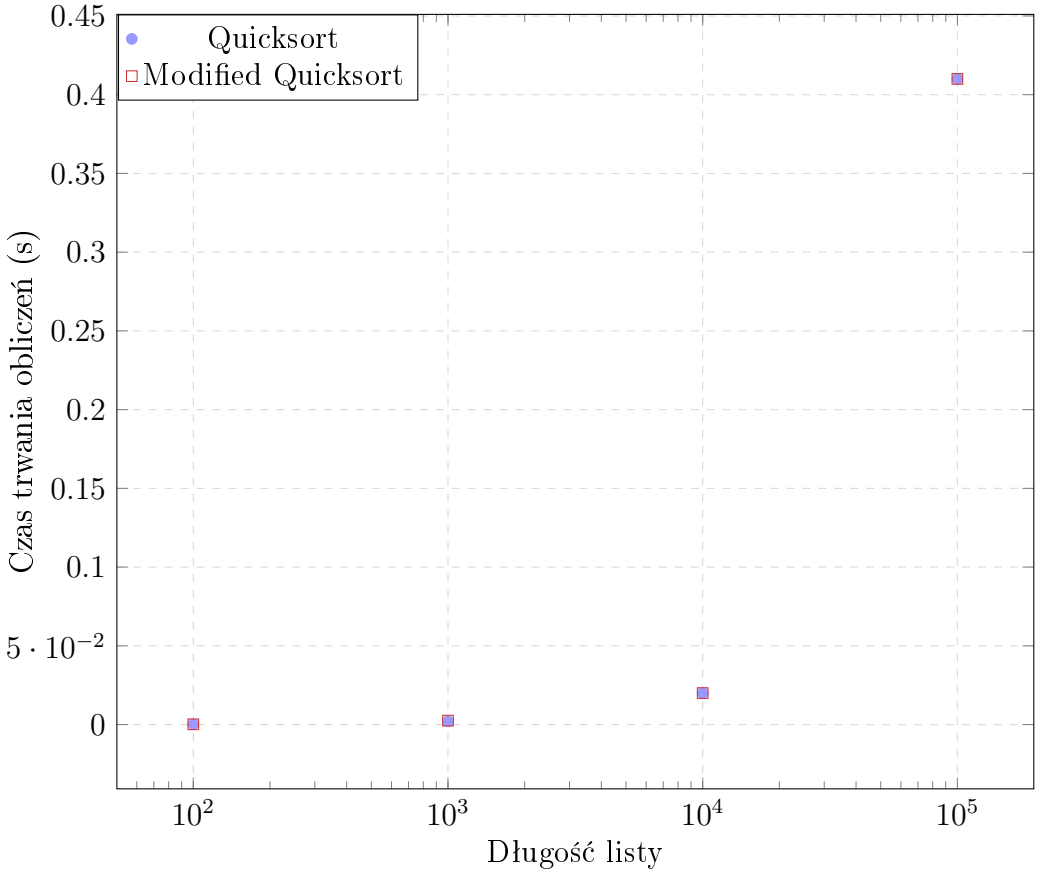
Szczerze mówiąc, nadal nie wiem, ale zdecydowałem się zmienić ją z proponowanego 10^6, na 10^9, ponieważ przesunąłem również zakres długości list.

Jak gromadzić dane?

Po zobaczeniu jak dużo danych mam do zgromadzenia, jako osoba leniwa, zdecydowałem się skorzystać z pomocy biblioteki `csv`, dzięki której byłem w stanie zapisywać wyniki pomiarów do plików `.csv`, które z kolei łatwo zaimportować i dalej opracowywać w programie MS Excel (oraz wykorzystywać do wykonywania wykresów w LaTeX).

4 Wykres zależności czasu obliczeń od długości listy

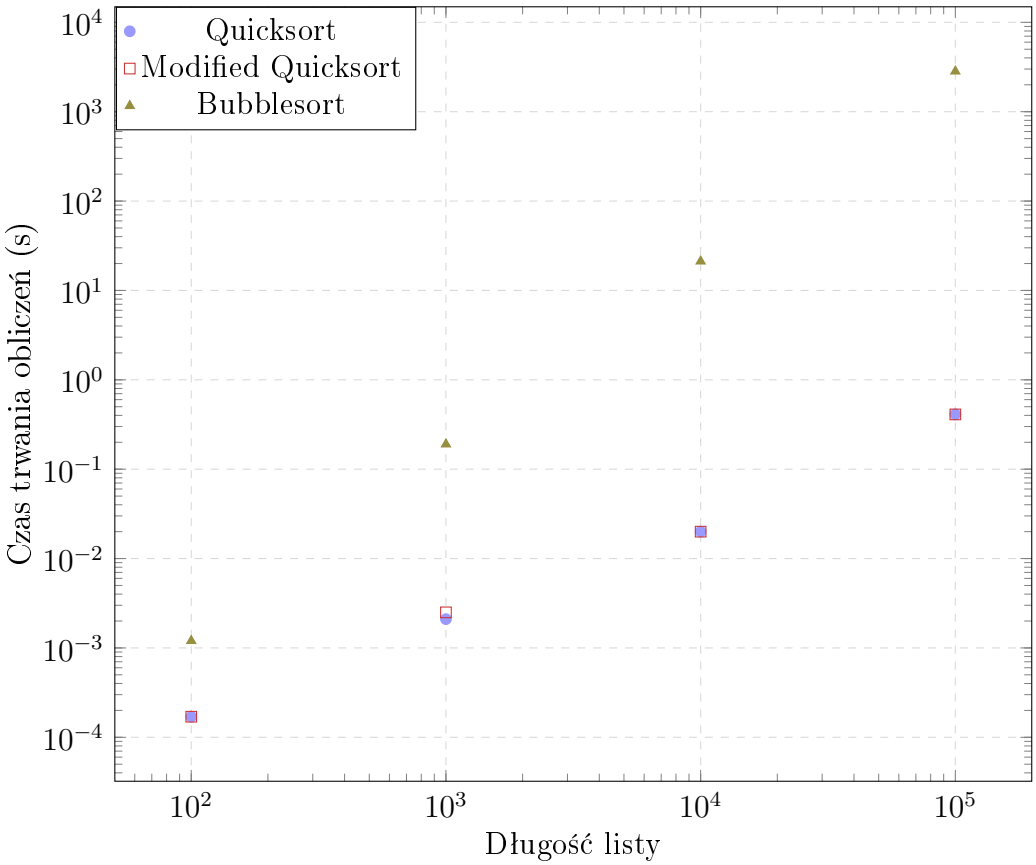
Porównanie dwóch wersji algorytmu quicksort



Zauważmy, że uśrednione wyniki dla obu wersji algorytmu, są praktycznie identyczne, co pozwala przypuszczać, że optymalizacja nic nie wnosi. Zdziwiłem się tymi wynikami, bo kilka dni wcześniej, tłumacząc jak zrobić to zadanie kolegom z roku, dla $n = 6$, zanotowałem poprawę wydajności o około 20%. Czy możliwe żeby taka różnica była powodowana tylko tym jakie tablice zostaną wylosowane?

Porównanie wszystkich algorytmów sortujących

Wzbraniałem się przed przedstawieniem wszystkich wyników pomiarów na jednym wykresie, ponieważ dla długości listy 10^4, bubblesort był ponad 1000 razy, a dla listy długości 10^5 - prawie 7000 razy wolniejszy. Poniższe wykresy na obu osiach wykorzystują skalę logarytmiczną. Zależność pokazuje,



że im dłuższa jest tablica, tym mniej opłacalne jest stosowanie algorytmu sortowania bąbelkowego.

Wnioski wynikające z tej części, sprawiły, że zdecydowałem się jeszcze bardziej zagłębić w temat zoptymalizowanego algorytmu quicksort i wykonać kilka testów.

5 Dalsze testy

Po wykonaniu zadań, na miejsce początkowych pytań, pojawiły się nowe.

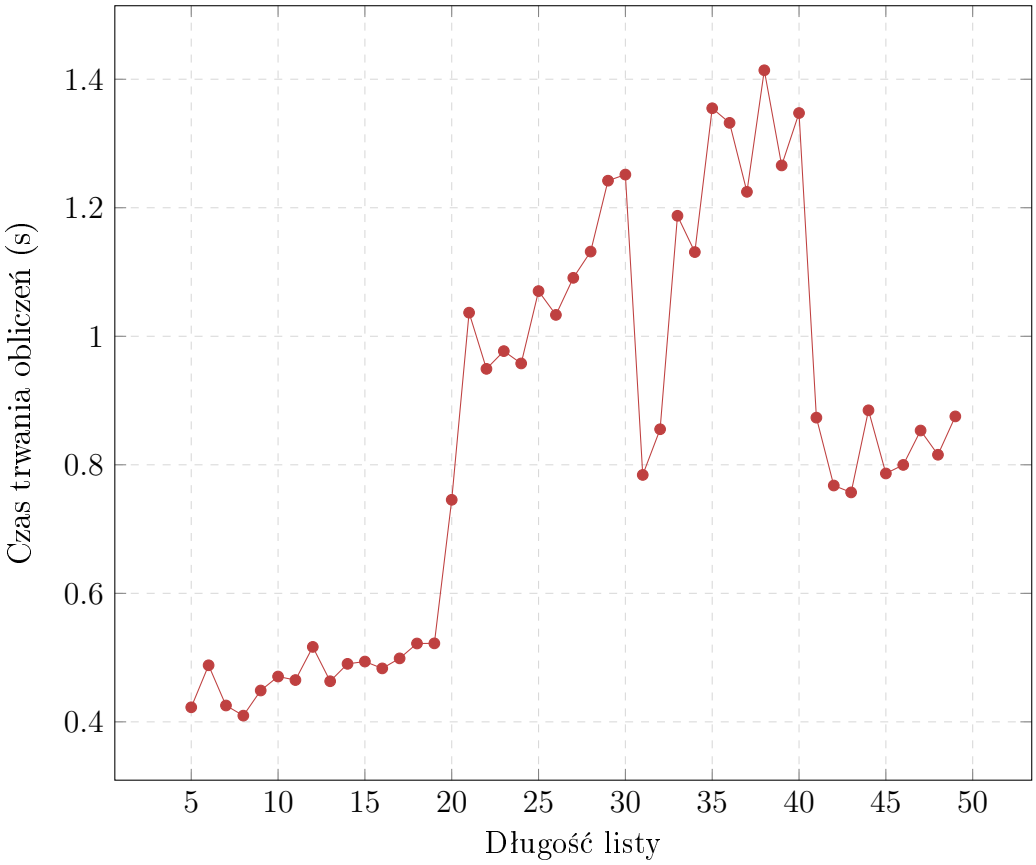
- *Jaka jest optymalna wartość granicznego n?*
- *Jak przedstawiają się zależności dla większych długości list?*

Żeby odpowiedzieć na te pytania, zdecydowałem się zbadać dwie zależności:

1. Jak zmienia się czas obliczeń dla n w zakresie 5 - 50 dla tablicy o długości 10^5.
2. Jak względem siebie wypadają algorytmy quicksort, zmodyfikowany quickosort z $n = 10$ i zmodyfikowany quicksort z optymalnym n wynikającym z pierwszej zależności, dla list o długościach 10^3 - 10^6, wypełnionymi liczbami w zakresie 1 - 10^{12}.

Poszukiwania optymalnego n

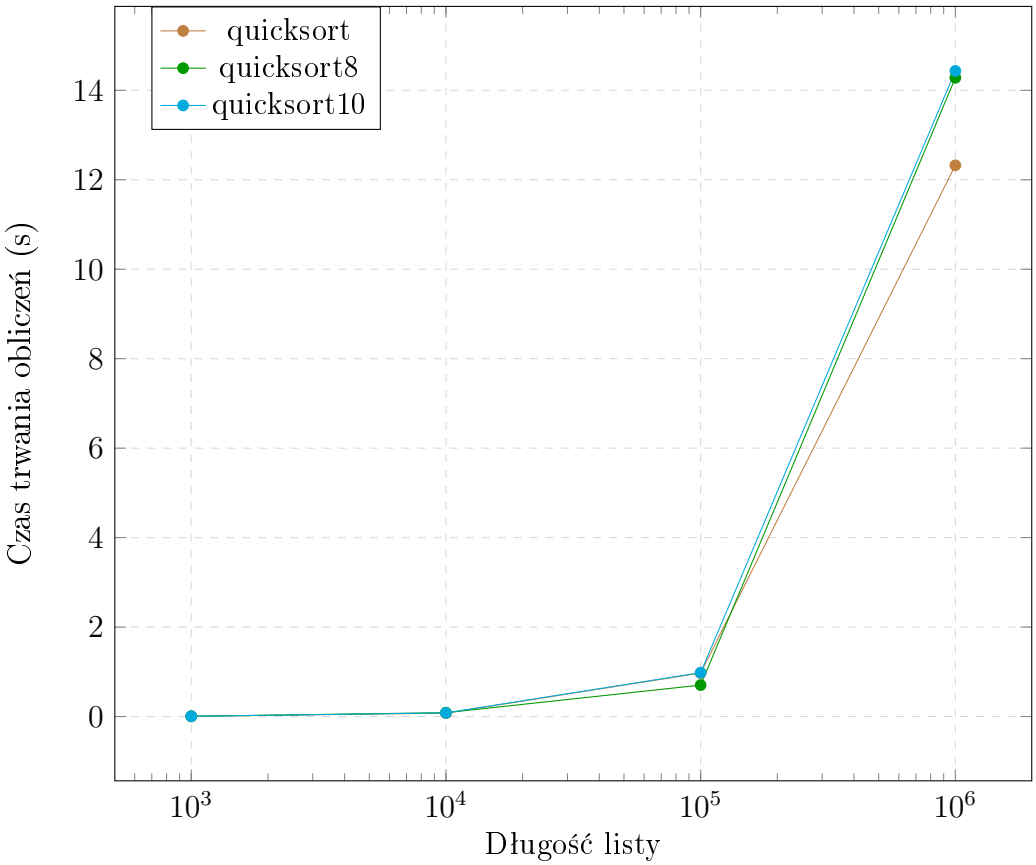
Po zmodyfikowaniu mojego programu, tak żeby stworzył od razu plik `.csv` z interesującymi mnie danymi, otrzymałem następujący wykres. Na podsta-



wie tego wykresu zauważamy, że dla $n \geqslant 20$, czas obliczeń znacząco wzrasta. Dodatkowo, możemy odczytać, że opytymalnie $n = 8$. W takim razie w następnym kroku porównam algorytmy, które będę określał jako quicksort, quicksort8 i quicksort10.

Czy optymalizacja quicksorta jest w ogóle opłacalna?

Żeby odpowiedzieć na to pytanie, porównam quicksort, optymalny quicksort8 oraz quicksort10, którego używałem w początkowej części eksprymentu.



Zauważmy, że dla długości list do $2 \cdot 10^5$, quicksort8 jest faktycznie szybszy niż zwykły quicksort. Z kolei, dla list dłuższych, quicksort jest zdecydowanie lepszy niż obie modyfikacje quicksortu.

6 Wnioski

Podsumowując przeprowadzone doświadczenia, dochodzę do wniosku, że zoptymalizowane wersje algrytmu quicksort, są użyteczne jedynie dla list długości krótszej niż $2 \cdot 10^5$. Ponadto, na wydajność poszczególnych algorytmów wpływa też to, jak bardzo nieuporządkowana jest wyjściowa lista, co można

stwierdzić na podstawie samych teoretycznych współczynników $O(n)$ dla sortowania bąblekowego oraz sortowania szybkiego.