Sprawozdanie - 3 sortowania w LaTeX-u

Anna Plywaczyk, 200340

30.03.2014

1 Wstep

Polaczylam 2 sprawozdania w jedno - z zajec 20.03 i 27.03, gdyz poprzedniego nie udalo mi sie wyslac na czas.

Zajecia 1:
Zadaniem, ktore powinnismy wykonac jest sprawdzenie zlozonosci obliczeniowej roznych sortowan. Wybor sortowania zalezy od oceny jakza chcemy uzyskac. Wybralam sortowanie szybkie, przez kopcowanie i scalanie. Przy kazdym sortowaniu musze sprawdzic w jakim czasie wykonuje sortowanie w zaleznosci od ilosci danych oraz jak czy zlozonosc obliczeniowa algorytmu zgodna jest z teoria.

Zajecia 2:
Zadaniem na zajecia drugie bylo porownanie sortowania szybkiego przy uzyciu dwoch roznych elementow mediany:

- Mediana jako wartosc losowa
- Mediana jako pierwszy element tablicy

2 Sortowania

2.1 Sortowanie przez kopcowanie(heap)

1. Dzialanie funkcji Jest to jeden z algorytmow dosc szybkich i niepochlaniajacych zbyt wiele pamieci. Zlozonosc obliczeniowa wynosi $O(nlogn)$, wynika to z tego, ze czas przywracania wlasnosci kopca jest $O(logn)$. W praktyce algorytm ten jest nieco wolniejszy od sortowania szybkiego. Podstawa algorytmu jest uzycie binarnego kopca zupelnego. Przez caly czas posiadamy dostep do elementu maksymalnego. Sortowanie to sklada sie z dwoch czesci.

    - Tworzenie kopca - mozemy wykorzystac ta sama tablice, w ktorej znajduja sie nieposortowane elementy.Poczatkowo do kopca nalezy pierwszy elemnt, nastepnie kopiec rozszerzamy o kolejne elemnty tablicy.Istotnym elementem drzewa binarnego jest fakt, ze wartosc w

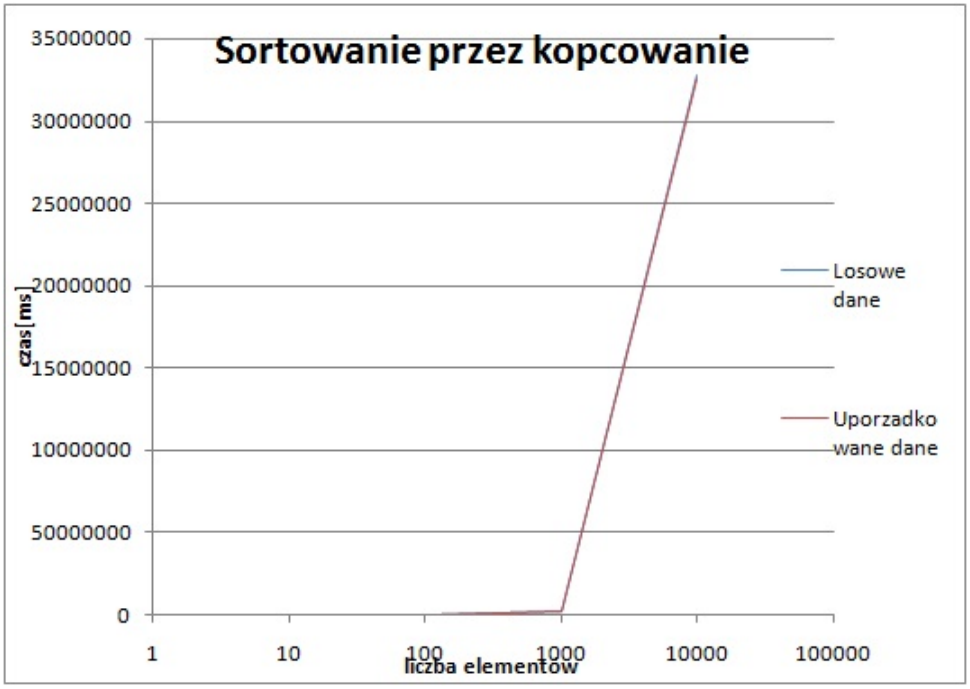
wezle jest nie mniejsza niz u jego potomkow. Kopiec przegladamy rekurencyjnie od korzenia w dol. Szukamy wiekszego elementu potomkow korzenia, jezeli syn jest wiekszy to zamieniamy go z korzeniem. I tak do przechodzimy przez wszystkie galezie i sprawdamy aby najwieksza wartosc byla w wierzcholku.

- Sortowanie - polega na cyklicznej wymianie liscia z korzeniem, odcinaniu liscia i przywroceniu wlasnosci kopca, odcinane liscie umieszczane sa w nowo stworzonym wektorze.

2. Test sortowania

   Czasy zmierzone odpowiednio dla wartosci:10, 100, 1000, 10000, 100000, 1000000

   - Losowe dane: 226,2; 16948,1; 2131090; 327717000
   - Dane uporzadkowane: 216,2; 16716,4; 2123260; 325362000
   - Wykres zlozonosci obliczeniowej.



Rysunek 1: Wykres czasu wykonania algorytmu od liczby probek

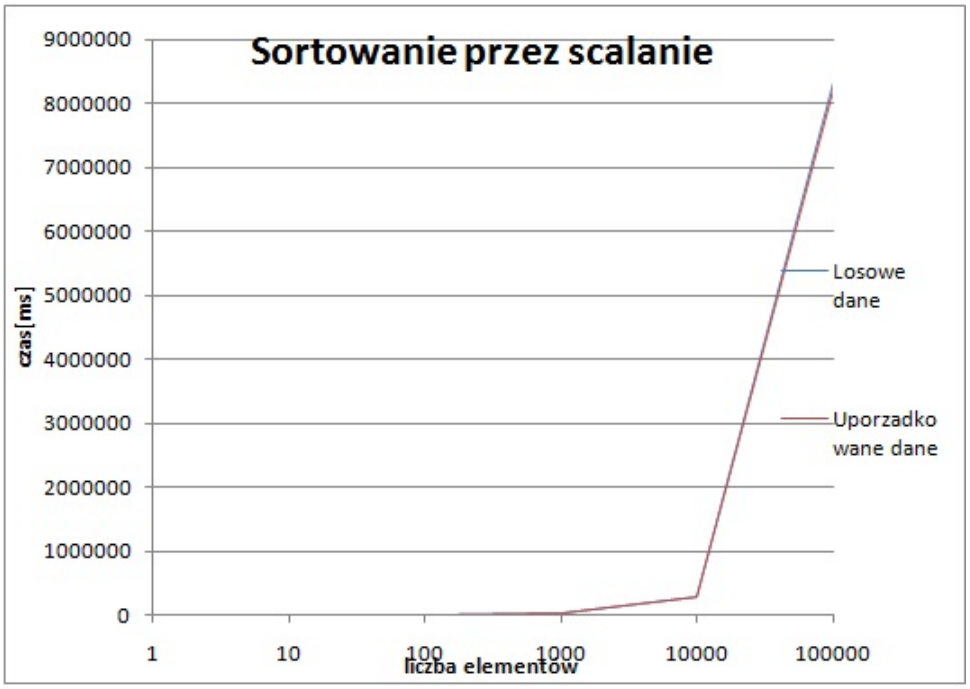
2.2 Sortowanie przez scalanie (merge)

1. Dzialanie funkcji Sortowanie przez scalanie nalezy do algorytmow, ktore wykorzystuja zalozenie "dziel i zwyciezaj". Zlozonosc obliczeniowa algorytmu wynosi $O(nlogn)$. Glownym zalozeniem sortowania jest podzielenie ciagu na dwie rowne czesci, nastepnie kazda z czesci uprzadkowujemy i dwa uprzadkowane ciagi laczymy w jeden. Podzielone czesci sortujemy w sposob rekurencyjnie wykorzytujac ten sam algorytm dla kazdego z podciagow. Musimy pamietac aby okreslic, kiedy zaprzestajemy podzialu ciagu.

2. Test sortowania

   Czasy zmierzone odpowiednio dla wartosci:10, 100, 1000, 10000, 100000, 1000000

   - Losowe dane: 207,5; 2161,9; 24210,9; 276170; 8352300
   - Dane uporzadkowane: 214; 2146,9; 22577; 277291; 8238310
   - Wykres zlozonosci obliczeniowej.



Rysunek 2: Wykres czasu wykonania algorytmu od liczby probek

2.3 Sortowanie szybkie(quicksort)

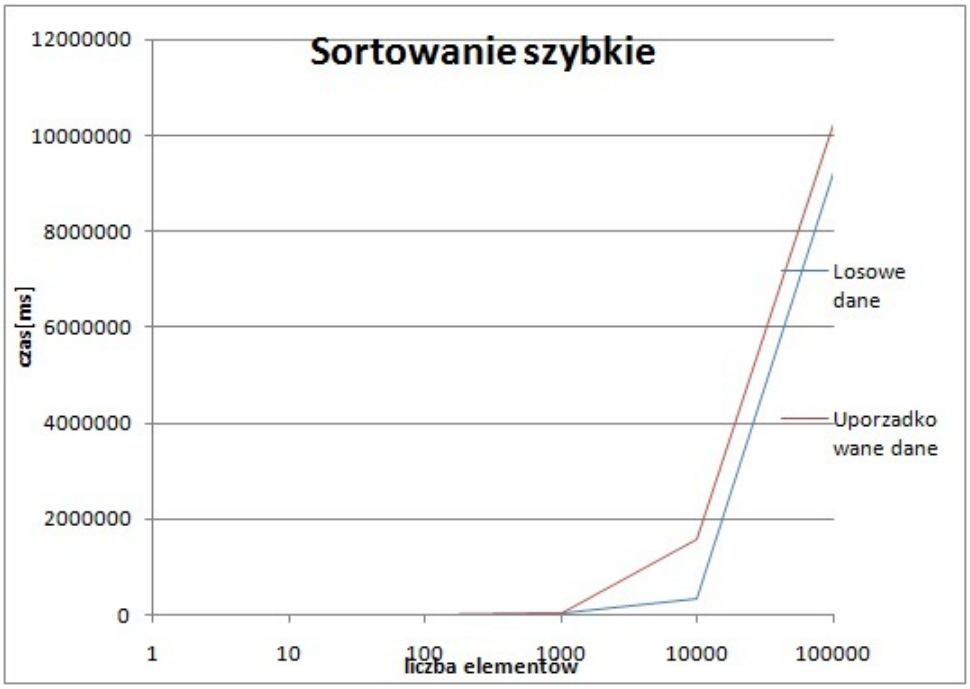
1. Dzialanie funkcji Jedno z najpopularniejszych sortowan wykorzystujacych metode dziel i zwyciezaj. Metoda dziel i zwyciezaj to metoda, w ktorej dany problem dzielony jest na kilka mniejszych problemow. Nastepnie problemy te rozwiazywane sa rekurencyjnie, a rozwiazania wszystkich mniejszych problemow laczone sa w celu utworzenia rozwiazania calego problemu. W sortowaniu wyrozniamy 4 glowne czesci programu.

    - Podzielenie elementu ciagu na dwie czesci wzgledem pewnego ustalonego elemntu (jest to ostatni element tablicy) tak aby na lewo od mediany znajdowaly sie elementy mniejsze a na prawo wieksze.
    - Sortujemy elementy po lewej stronie.
    - Sortujemy elementy po prawej stronie.
    - Rekursja konczy sie gdy kolejny fragment zawiera pojedynczy element (jednoelementowa podtablica nie wymaga sortowania).

2.3.1 Mediana jako pierwszy argument tabilcy

Test sortowania Czasy zmierzone odpowiednio dla wartosci:10, 100, 1000, 10000, 100000, 1000000

- Losowe dane: 619,9; 3139,7; 29049,5; 330319; 9240370
- Dane uporzadkowane: 292,9; 2975; 42177; 1561650; 10263410
- Wykres zlozonosci obliczeniowej.

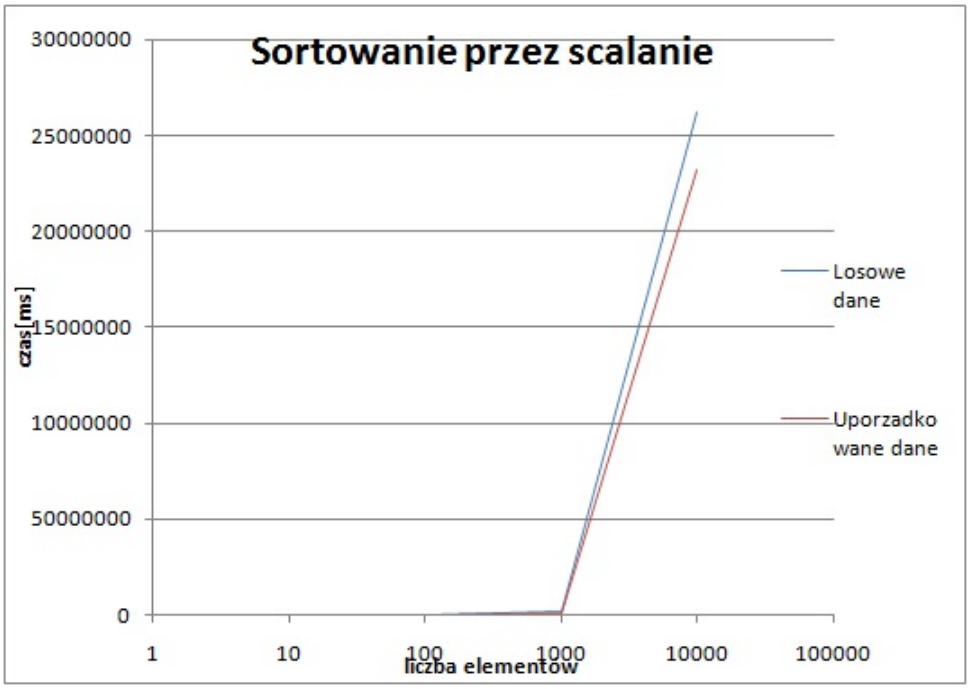


Rysunek 3: Wykres czasu wykonania algorytmu od liczby probek

2.3.2 Losowa wartosc elementu tablicy jako mediana

Test sortowania Czasy zmierzone odpowiednio dla wartosci:10, 100, 1000, 10000, 100000

- Losowe dane: 37,8; 2677,2; 1799380; 262613100
- Dane uporzadkowane: 26,7; 1521; 1022940; 232613100
- Wykres zlozonosci obliczeniowej.



Rysunek 4: Wykres czasu wykonania algorytmu od liczby probek

3 Wnioski

Wykresy w kazdym przypadku przypominaja funkcje kwadratowa wiec prawidlowe. W skali logarytmicznej najlepiej widac, gdyz przy skali lin-lin wykres byl funkcja prosta. W kazdym przypadku czas zmierzony byl 10 razy i srednia zostala wzieta pod uwage.
Natomiast przy uzyciu sortowania przez scalanie roznica jest niewidoczna, a wyniki w zaleznosci od liczby elemntow sa rozne. Nie mozemy jednoznacznie okreslic jaki wplywa ma uporzadkowanie danych na szybkosc sortowania.
Dla sortowania przez kopcowanie zmniejszylam liczbe elementow do 10000, gdyz przy tej liczbie widac jak szybko rosnie czas wykonywania algorytmu. W tym sortowaniu mozemy zauwazyc, iz rowniez na wykresie nie ma znaczenia czy sortujemy liczby uporzadkowane czy losowe. Natomiast w danych liczby te niewiele sie od siebie roznia, sortowanie liczb uporzadkowanych poprzez kopcowanie jest szybsze niz losowych.

W sortowaniu szybkim gdy mediana jest pierwszy element wektora widac znaczaca roznice przy sortowaniu liczb uporzadkowanych badz losowych. Gdy liczby sa posortowane czas dzialania algorytmu znacznie sie wydluza. Natomiast przy losowym wyborze elementow algorytm szybciej sortuje uporzadkowane dane. Dodatkowo zauwazyc mozemy, ze dla mniejszych danych metod ta szybciej

wykonala sortowanie.
Zauwazyc mozemy ze wszystkie algorytmy poza sortowaniem szybkim czas wykonywania dzialania maja zblizony dla danych uporzadkowanych i losowych.
Najdluzej wykonujacym dzialanie okazal sie algorytm sortowania przez kopcowanie, natomiast algorytm sortowania przez scalanie wykonal to najszybciej.
Wywnioskowac mozemy, iz wybor metody ma znaczacy wplyw na szybkosc sortowania. Nawet przy tym samym typie sortowania a roznym wyborze mediane otrzymane wyniki moga sie roznic.