# PAMSI - Sprawozdanie 4

**Filip Guzy 218672**

13 kwietnia 2016

**Algorytm sortowania szybkiego - quicksort**

Algorytm sortowania szybkiego bazuje na zasadzie "Dziel i zwyciężaj". Jego działanie polega na wybraniu jednego elementu z sortowanego zbioru (tak zwanego pivota), który będzie rozgranicznikiem dla dwóch pozdbiorów, na jakie zostaną podzielone sortowane dane: elementów mniejszych lub równych. pivotowi, oraz elementów większych od niego. Następnie owe podzbiory podlegają sortowaniu w ten sam sposób aż do momentu posortowania całości. Złożoność obliczeniowa quicksorta w dużym stopniu zależy od wyboru pivota. Poniżej przedstawiono optymistyczny, średni oraz pesymistyczny przypadek złożoności obliczeniowej tego algorytmu a także sposoby wyboru pivota i ich wpływ na złożoność.

**Złożoność obliczeniowa quicksort**

1. Przypadek optymistyczny:
   Występuje, gdy wybrany pivot zawsze jest medianą sortowanego zbioru. W tym przypadku złożoność obliczeniowa sortowania szybkiego jest rzędu $nlog(n)$. Sytuacja, gdy pivot jest medianą danego zbioru jest w przypadku zwykłego jego losowania praktycznie niemożliwa do zaistnienia (a przynajmniej prawdopodobieństwo tego zdarzenia jest bardzo niskie dla dużych zbiorów). Nie jest możliwe dodanie usprawnienia wyznaczającego medianę, które nie pogorszy złożoności obliczeniowej sortowania.

2. Przypadek średni:
   Występuje dla równomiernego rozkładu prawdopodobieństwa wyboru pivota z tablicy. Złożoność obliczeniowa tego przypadku wynosi w przybliżeniu $1,39nlogn$, czyli jest o 39% większa niż w przypadku optymistycznym. Aby mieć pewność uzyskania przypadku średniego wystarczy wybrać element ze środka tablicy.

3. Przypadek pesymistyczny:
   Występuje w sytuacji, gdy wybrany pivot jest najmniejszym lub największym elementem sortowanego zbioru. Złożoność obliczeniowa dla tego przypadku wynosi w przybliżeniu $\frac{n^2}{2}$. Najgorsza z możliwych opcji przypadku pesymistycznego występuje, gdy wybrany przez nas element jest skrajnym elementem posortowanego w przeciwną stronę zbioru. Wtedy złożoność osiąga rząd $n^2$.

**Sposoby wyboru pivota**

1. Wybór pierwszego lub ostatniego elementu tablicy
   Sposób ten jest najefektywniejszy obliczeniowo, ponieważ jego wykonanie jest możliwe w czasie stałym, zatem jego złożoność należy do O(1). Największą jego wadą jest to, że gdy zbiór jest już uporządkowany, to złożoność sortowania będzie przypadkiem pesymistycznym, czyli rzędu $n^2$.

2. Wybór ze środka lub losowanie elementu tablicy
   Sposób ten jest gwarantem złożoności średniej sortowania. Randomizacja lub wybór ze środka minimalizuje ryzyko wystąpienia sytuacji w której dostajemy złożoność pesymistyczną. Złożoność obliczeniowa wyboru ze środka zawiera się w O(1), natomiast losowanie powoduje wzrost złożoności sortowania.

3. Przybliżona mediana
   Sposób polega na wyborze kilku liczb z sortowanego zbioru, a następnie wyznaczeniu z nich mediany. Gwarantuje, że nigdy nie zajdzie sytuacja, w której zostanie wybrany największy lub najmniejszy element. Usprawnia on sortowanie każdego podzbioru, niestety kosztem złożoności obliczeniowej całego algorytmu. Można wykorzystywać różne algorytmy wyznaczające medianę, jednak w sytuacji dużego ryzyka wyboru najmniejszego elementu zwyczajnie lepiej zastosować inny algorytm sortowania. Złożoność takiego sposobu wyboru pivota zależy od implementacji.

Algorytmy sortowania szybkiego wykorzystujące wybór pivota z końca oraz środka tablicy, a także algorytm z medianą zostały zaimplementowane jako metody klasy Array w pliku algorytmy.cpp.

**Pomiary dla quicksort dla trzech metod pivotowania**

Poniżej przedstawiono w trzech tabelach serie pomiarów oraz ich czasy wykonania (w ms), a także uśrednione wartości kolejno dla pivotowania poprzez wybór środkowego elementu, wybór elementu skrajnego oraz poprzez medianę. Sortowano zbiory zawierające elementy z przedziału 0-1000.

Tabela 1: Quicksort - wybór środkowego elementu

| Ilość danych | 10 | 100 | 1000 | 1000000 | 1000000000 |
|---|---|---|---|---|---|
| Czas [ms] | 0,00100005 | 0,012 | 0,154 | 151,368 | 223659 |
| | 0,00100005 | 0,0120001 | 0,136 | 153,077 | 221248 |
| | 0,000999928 | 0,0109999 | 0,308 | 152,559 | 221008 |
| | 0,00100005 | 0,011 | 0,137 | 150,222 | 217939 |
| | 0,00199997 | 0,012 | 0,137 | 149,222 | 221402 |
| | 0,000999928 | 0,0109999 | 0,159 | 151,14 | 222228 |
| | 0,00199997 | 0,012 | 0,139 | 151,778 | 222051 |
| | 0,00100005 | 0,012 | 0,155 | 152,525 | 219839 |
| | 0,000999928 | 0,011 | 0,154 | 152,466 | 222594 |
| | 0,000999928 | 0,0120001 | 0,151 | 150,932 | 221570 |
| Średnia | 0,0011999852 | 0,0116 | 0,163 | 151,5289 | 221353,8 |

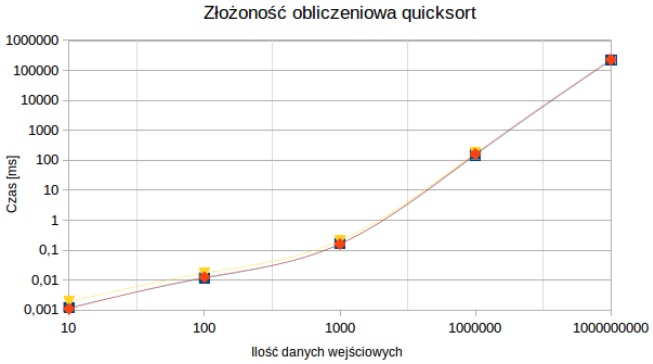
Tabela 2: Quicksort - wybór skrajnego elementu

| Ilość danych | 10 | 100 | 1000 | 1000000 | 1000000000 |
|---|---|---|---|---|---|
| Czas [ms] | 0,00100005 | 0,012 | 0,157 | 161,768 | 227695 |
| | 0,00100005 | 0,012 | 0,157 | 165,162 | 231220 |
| | 0,000999928 | 0,013 | 0,154 | 166,774 | 231345 |
| | 0,000999928 | 0,012 | 0,168 | 160,029 | 228982 |
| | 0,00100005 | 0,012 | 0,158 | 163,658 | 230047 |
| | 0,00100005 | 0,013 | 0,156 | 160,722 | 231599 |
| | 0,00100005 | 0,012 | 0,155 | 157,433 | 230486 |
| | 0,00100005 | 0,012 | 0,175 | 164,203 | 229461 |
| | 0,00100005 | 0,0129999 | 0,158 | 159,69 | 229983 |
| | 0,00199997 | 0,013 | 0,156 | 165,832 | 230990 |
| Średnia | 0,0011000176 | 0,01239999 | 0,1594 | 162,5271 | 230180,8 |

Tabela 3: Quicksort - wybór poprzez medianę

| Ilość danych | 10 | 100 | 1000 | 1000000 | 1000000000 |
|---|---|---|---|---|---|
| Czas [ms] | 0,00199997 | 0,0170001 | 0,194 | 182,761 | |
| | 0,00199997 | 0,0170001 | 0,208 | 184,158 | |
| | 0,00200009 | 0,017 | 0,22 | 183,918 | |
| | 0,00199997 | 0,018 | 0,19 | 179,07 | |
| | 0,00199997 | 0,017 | 0,193 | 179,208 | |
| | 0,00199997 | 0,0170001 | 0,192 | 185,702 | |
| | 0,00199997 | 0,017 | 0,25 | 181,744 | |
| | 0,00199997 | 0,0170001 | 0,209 | 182,57 | |
| | 0,00199997 | 0,018 | 0,19 | 181,608 | |
| | 0,00100005 | 0,016 | 0,251 | 184,466 | |
| Średnia | 0,00189999 | 0,01710004 | 0,2097 | 182,5205 | |

Poniżej przedstawiono również średnie wyniki pomiarów na wykresie w celu porównania złożoności.
Kolorem niebieskim oznaczono quicksort poprzez wybór środkowego elementu, czerwonym - skrajnego, natomiast żółtym - poprzez medianę. Można zauważyć, że usprawnienie algorytmu quicksort liczące medianę nieznacznie pogorszyło jego złożoność obliczeniową. Jak widać z przedstawionych danych, złożoność obliczeniowa sortowania szybkiego wynosi w przybliżeniu $nlogn$. Wszystkie trzy sposoby wyboru pivota dla zróżnicowanych zbiorów gwarantują uzyskanie złożoności średniej. Pomiary dla miliarda w przypadku mediany nie zostały wykonane ze względu na błąd naruszenia ochrony pamięci podczas wykonywania tego algorytmu, którego źródło nie zostało jeszcze wykryte.



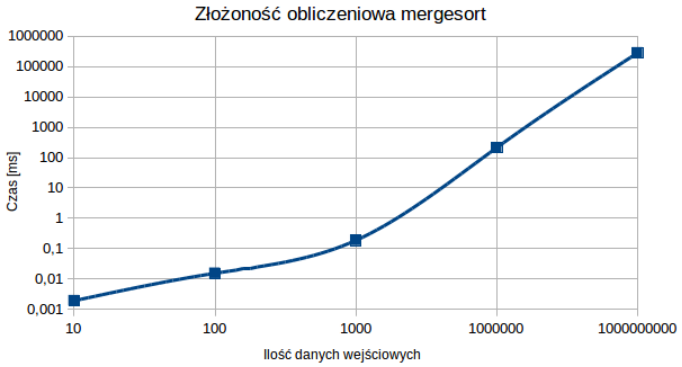

**Algorytm sortowania przez scalanie - mergesort**

Algorytm sortowania przez scalanie to kolejny algorytm bazujący na metodzie "Dziel i zwyciężaj". Jego działanie polega na podziale sortowanej tablicy na połowy, a następnie kontynuację tego działania na podzbiorach do momentu uzyskania zbiorów jednoelementowych. Następnie wykorzystując działanie rekursji następuje scalanie podzbiorów i jednoczesne sortowanie ich elementów aż do momentu powrotu do jednego zbioru. Średnia złożoność algorytmu mergesort wynosi $nlogn$, nastomiast pesymistyczna $n^2$, jednak wystąpienie tej drugiej jest bardzo mało prawdopodobne. Poniżej przedstawiono czasy sortowania (w ms) tym algorytmem zbiorów o różnych rozmiarach. Sortowano zbiory zawierające elementy z przedziału 0-1000.

Tabela 4: Mergesort

| Ilość danych | 10 | 100 | 1000 | 1000000 | 1000000000 |
|---|---|---|---|---|---|
| Czas [ms] | 0,00199997 | 0,015 | 0,175 | 209,028 | 285402 |
| | 0,00199997 | 0,015 | 0,176 | 220,542 | 285827 |
| | 0,00199997 | 0,0139999 | 0,174 | 210,991 | 280365 |
| | 0,00199997 | 0,0140001 | 0,212 | 211,248 | 284051 |
| | 0,00199997 | 0,0189999 | 0,175 | 213,839 | 280504 |
| | 0,00199997 | 0,0140001 | 0,234 | 212,437 | 288233 |
| | 0,00199997 | 0,02 | 0,175 | 214,579 | 289444 |
| | 0,00199997 | 0,0140001 | 0,175 | 211,321 | 287080 |
| | 0,00200009 | 0,0139999 | 0,191 | 212,394 | 288059 |
| | 0,00100005 | 0,014001 | 0,176 | 215,152 | 288223 |
| Średnia | 0,00189999 | 0,01530001 | 0,1863 | 213,1531 | 285718,8 |

Pomiary dla mergesort przedstawiono również na poniższym wykresie.




Pomiary wykonane dla algorytmu mergesort pokazały, że jest on mniej sprawny od quicksort, jednak jego średnia złożoność obliczeniowa rzędu $nlogn$ jest bardziej prawdopodobna do uzyskania. Nieznaczne różnice w szybkości działania obu algorytmów są wynikiem tego, że sortowanie przez scalanie nie jest sortowaniem "w miejscu", gdyż wykorzystuje do tego zewnętrzny kontener przechowujący tymczasowo sortowane dane.

**Wnioski**

Algorytm sortowania szybkiego jest efektywniejszy niż algorytm sortowania przez scalanie, jednak jego złożoność obliczeniowa jest bardziej zagrożona tzw. "ukwadratowieniem", czyli zaistnienem przypadku pesymistycznego. Mergesort z dużym prawdopodobieństwem gwarantuje złożoność średnią, jednak jako algorytm, który nie sortuje "w miejscu" charakteryzuje się niezauważalnie większym czasem sortowania. Sposób wyboru pivota dla sortowania szybkiego ma nieznaczny wpływ na jego złożoność obliczeniową.