



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Algorytmy cd...

Dalsze przykłady podstawowych algorytmów

M. Rad

17.12.2015

Plan

- Powtórka
- Segregowanie
- Poszukiwanie przez połowienie
- Wstawianie
- Inne algorytmy segregowania
- Algorytm Euklidesa

Powtórka

Jak porównywać ze sobą algorytmy:

- Złożoność obliczeniowa:
Jest to funkcja określająca ilość zasobów potrzebnych do wykonania danego algorytmu w zależności od rozmiaru danych wejściowych
- Inne własności: stopień skomplikowania (np. ilość linii w zapisie), przydatność do określonego zagadnienia, itp...

Algorytm poszukiwania przez połowienie

Dane: uporządkowany ciąg x_n , gdzie $n = \langle 0 \ k \rangle$

Szukane: znaleźć n takie, że $x_n = s$

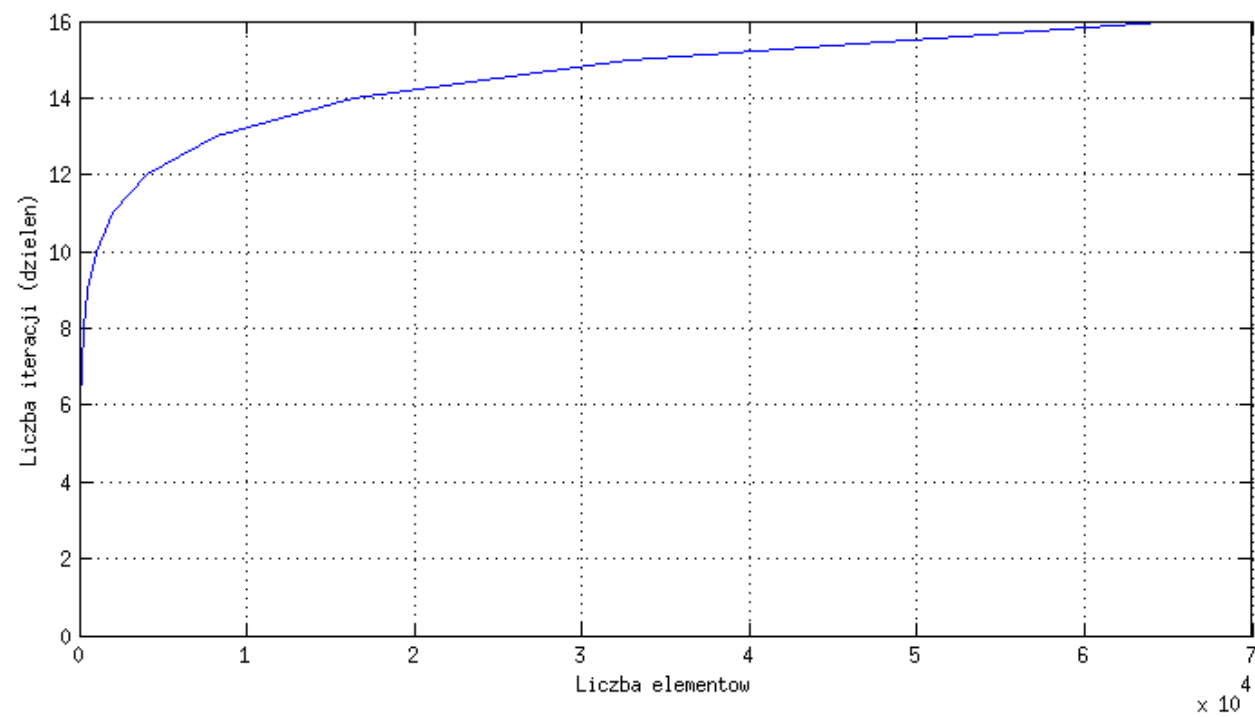
Algorytm:

1. Oznacz lewy_koniec=0 prawy_koniec=k
2. Podziel zbiór $L = (\text{prawy_koniec} + \text{lewy_koniec}) \div 2$ (czyli L trafia w środek przedziału)
3. Jeśli $x_L = s$ to koniec, szukane n równa się L
4. Jeśli $x_L < s$ to lewy_koniec= $L+1$ w przeciwnym przypadku (czyli gdy $x_L > s$) prawy_koniec= $L-1$
5. Wróć do punktu 2.

Algorytm poszukiwania przez połowienie

- Czy istnieją prostsze algorytmy?
- Czemu ten jest dobry?
- A jaka jest złożoność obliczeniowa?
- Jeśli elementów jest 16 to należy wykonać co najwyżej 4 podziały
- Jeśli elementów jest 32 to 5 podziałów itd.
- Czyli złożoność $O(n) = \log_2 N$ gdzie N to potęga liczby 2 nie mniejsza niż n

$\text{Log}_2 N$



Poszukiwanie jeszcze szybsze

- Metoda poszukiwania interpolacyjnego
 - Połowienie dokładniejsze bo zakładające interpolację , a nie zawsze dzielenie na pół.

Wstawianie

- Algorytm wstawiania elementu na miejsce tak aby zbiór pozostał uporządkowany jest analogiczne do szukania
- Może być zrealizowane jako szukanie z połowieniem czyli złożoność $O(n) = \log_2 N$

Inne algorytmy segregowania

Sortowanie przez wstawianie:

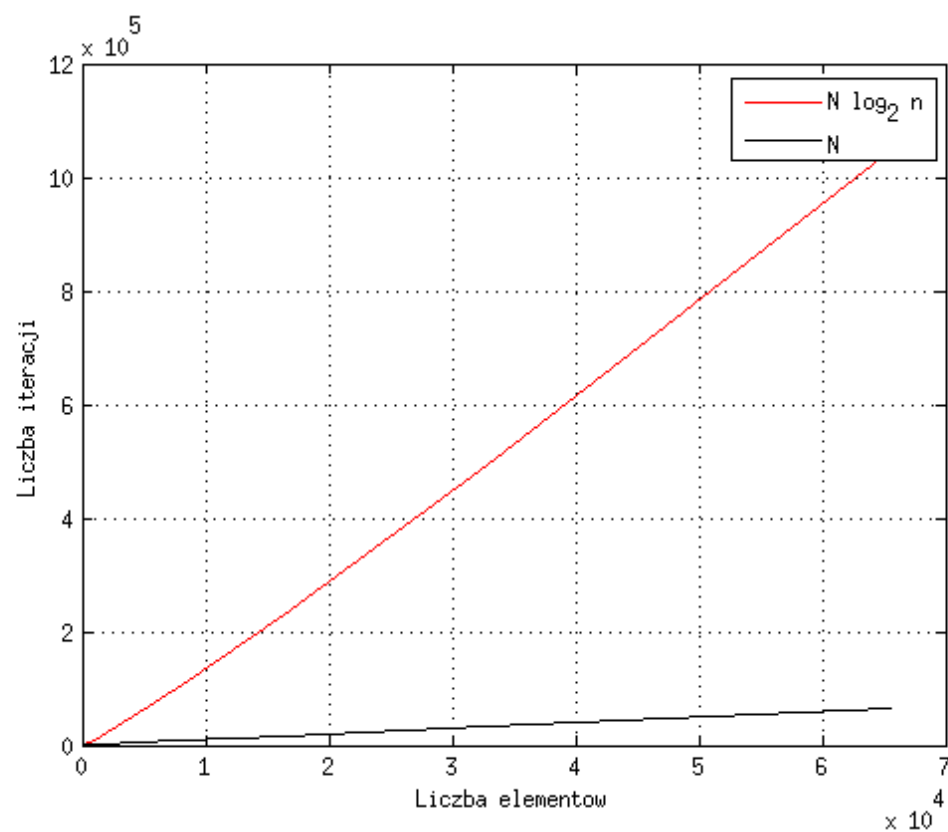
Dane: nieposortowany zbiór x_i

Szukane: zbiór posortowany x_n

- Algorytm
- Weź pierwszy element ze zbioru wejściowego i umieść w wyjściowym
- Weź kolejny i wstaw w „odpowiednie” miejsce (odpowiednie to znaczy po mniejszym ale przed większym)
- Postępuj tak aż do końca zbioru wejściowego

Sortowanie przez wstawianie

- Jaka jest złożoność obliczeniowa?
- Trzeba wziąć każdy z n elementów i umieścić w odpowiednim miejscu posegregowanego ciągu
- Jeśli zrobić to „liniowo” to jest n porównań czyli $O(n) = n * n = n^2$.
- Ale do „umieszczania” można użyć algorytmu „umieszczania z połowieniem” którego złożoność wynosi $O(n) = \log_2 N$
- I wtedy $O(n) = n * \log_2 n$



Inne algorytmy sortowania

- Poprzez scalanie (wykorzystuje rekurencję)
Co to jest rekurencja?
- Sortowanie bąbelkowe $O(n) = n^2$
- I jeszcze kilka innych...

Rekurencja

- Rekurencja to odwołanie funkcji do niej samej. Przykład w C – funkcja która liczy silnię:

```
int silnia(int N)
{
    if(N <= 2) return N;
    else return ( N*silnia(N-1) );
}
```

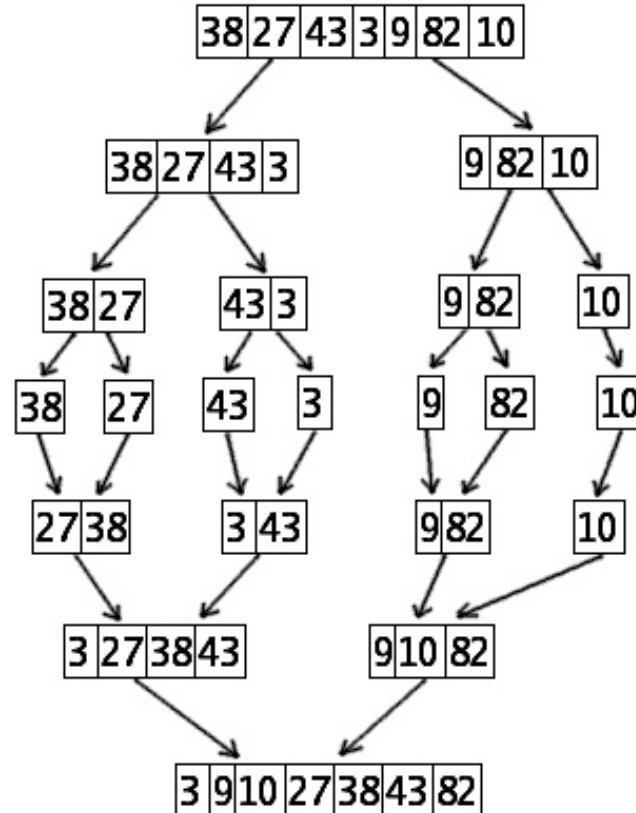
Scalanie

- Cel: mamy dwa uporządkowane zbiory i chcemy otrzymać zbiór połączony, ale tak aby dalej był posortowany
- Algorytm:
 - Porównaj pierwsze elementy ze zbiorów
 - Do zbioru wynikowego wstaw mniejszy (bądź większy – zależy od sposobu uporządkowania) i usuń ten element ze zbioru wejściowego
 - Powtarzaj aż oba zbiory staną się puste

Sortowanie poprzez scalanie

- Algorytm: (nazwijmy go SCALAJ)
 - 1) Podziel zbiór na dwie części chyba że pozostał tylko jeden element
 - 2) Jeśli po podziale zbiory mają więcej niż jeden element na każdym ze zbiorów wykonaj funkcję SCALAJ
 - 3) jeśli zbiory mają po jednym elemencie to scal zbiory (używając scalania)

Sortowanie poprzez scalanie - przykład



Sortowanie bąbelkowe

- Porównaj dwa sąsiednie elementy tablicy i zamień je miejscami jeśli nie są uporządkowane właściwie
- Przejdź w ten sposób przez całą tablicę
- Powtórz od początku jeśli w danym przebiegu tablicy dokonano choć jednej zmiany

Złożoność czasowa $O(n^2)$ ale pamięciowa $O(1)$

Algorytm Euklidesa

- Znajduje największy wspólny dzielnik (NWD) dwóch liczb
- Opiera się na założeniu, że NWD nie zmienia się gdy od liczby większej odejmiemy mniejszą

$$\begin{array}{rcl} 21 & 14 & | \ 21 - 14 = 7 \\ 14 & 7 & | \ 14 - 7 = 7 \\ 7 & 7 & | \text{NWD} = 7 \end{array}$$

$$\begin{array}{rcl} 21 & 30 & | \ 30 - 21 = 9 \\ 21 & 9 & | \ 21 - 9 = 12 \\ 9 & 12 & | \ 12 - 9 = 3 \\ 9 & 3 & | \ 9 - 3 = 6 \\ 3 & 6 & | \ 6 - 3 = 3 \\ 3 & 3 & | \text{NWD} = 3 \end{array}$$

Algorytm Euklidesa

