AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Jeszcze o algorytmach

Przykłady podstawowych algorytmów

M. Rad

17.01.2019

Plan

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

I termin egzaminu z informatyki: 01.02.2019 (piątek) godzina 8.30 sala 121 budynek B1

II termin egzaminu z informatyki: 08.02.2019

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...

Znajdowanie najmniejszego elementu

- Dany jest zbiór liczb, należy znaleźć najmniejszy element

Znajdowanie najmniejszego elementu

Rozwiązanie:

1. Weź pierwszy element i chwilowo uznaj że jest najmniejszy
2. porównaj z kolejnym, jeśli kolejny jest mniejszy to uznaj, że to jest najmniejszy, jeśli nie to nic nie rób
3. Powtarzaj punkt 2 aż do wyczerpania zbioru
4. Wartość która po przejściu całości jest uważana za najmniejszą jest rzeczywiście najmniejszą w zbiorze

Znajdowanie najmniejszego elementu

- Taki algorytm jest najczęściej nazywany przeszukiwaniem liniowym.
- Jaka jest jego złożoność obliczeniowa?
- Czy istnieje algorytm prostszy?
- Jak zmodyfikować algorytm aby znajdował największy element?
- Czemu mówimy o tak prostym zagadnieniu?

Segregowanie

- Segregowanie lub uporządkowanie informacji to dla np. dla zbioru liczb ustawienie od najmniejszej do największej
- Czy algorytm znajdowania min. da się wykorzystać w algorytmie segregowania?

  Tak, jest to nazywane segregowaniem przez wybór
- Jaka będzie złożoność obliczeniowa takiego łączonego algorytmu?
- A po co w ogóle segregować?

Algorytm poszukiwania przez połowienie

Dane: uporządkowany ciąg x_n , gdzie n=<0 k>

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

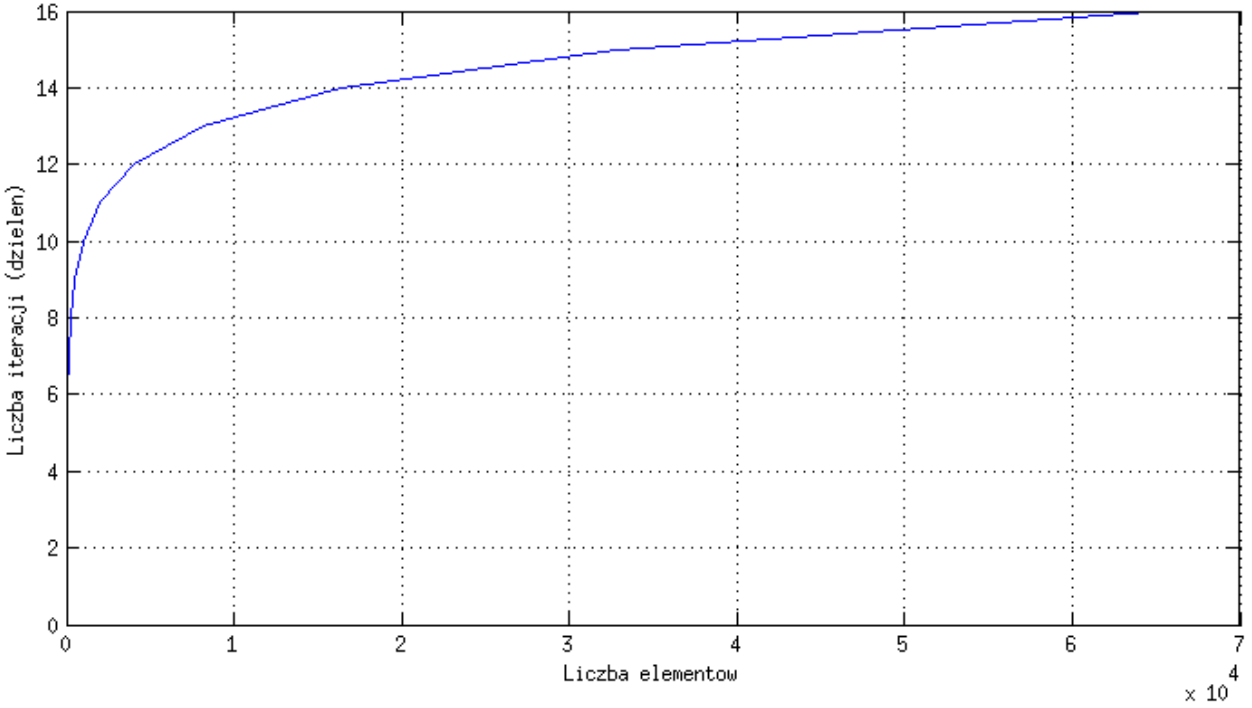
Algorytm:

1. Oznacz lewy_koniec=0 prawy_koniec=k
2. Podziel zbiór L=(prawy_koniec+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

Log_2N


Liczba iteracji (dzielen)
Liczba elementow
x 10^4
0
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7

Poszukiwanie jeszcze szybsze

- Metoda poszukiwania interpolacyjnego albo ze słownikiem

A wstawianie

- Algorytm wstawiania elementu na miejsce tak aby zbiór pozostał uporządkowany jest analogiczne do szukania z połowieniem
- Czyli złożoność $O(n) = \log_2 N$

Inne algorytmy segregowania

Sortowanie przez wstawianie:

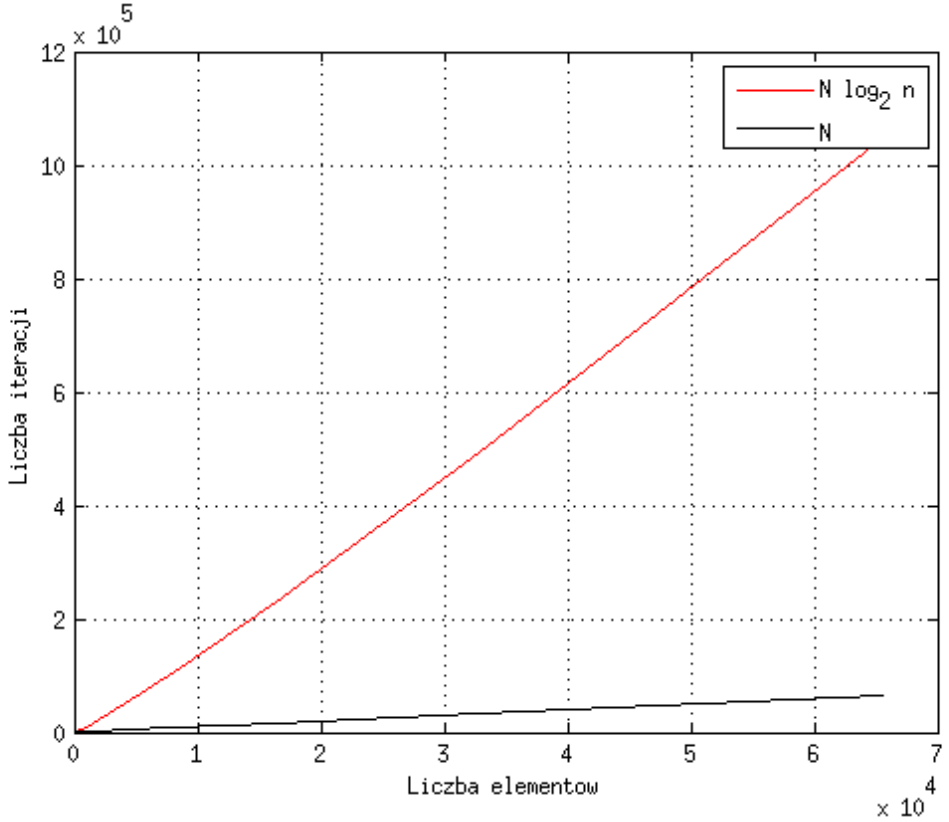
Dane: nieposortowany zbiór xi

Szukane: zbiór posortowany xn

- 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 $O(n)= n*\log_2 n$


x 10^5
12
10
8
6
4
2
0
Liczba iteracji
N log2 n
N
0
1
2
3
4
5
6
7
Liczba elementow
x 10^4

Inne algorytmy sortowania

- Poprzez scalanie (wykorzystuje rekurencję)

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

Algorytm Euklidesa

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

```
21 14  | 21 – 14 = 7
14  7  | 14 – 7 = 7
7   7  | NWD = 7
```

```
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  |  NWD =3
```

Algorytm Euklidesa