

Algorytmy i struktury danych

Algorytmy i struktury danych #1

Wprowadzenie



Wojciech Complak
Instytut Informatyki, Wydział Informatyki, Politechnika Poznańska

e-mail: Wojciech.Complak@wsb.poznan.pl



1.06

Algorytmy i struktury danych

Plan wykładu

- literatura podstawowa i uzupełniająca
- czym jest algorytm ?
- jak zapisać i ocenić wydajność algorytmu ?
- metody oceny złożoności
- wyszukiwanie liniowe
- wyszukiwanie binarne

Algorytmy i struktury danych (2/50)

Algorytmy i struktury danych

Literatura podstawowa (#1/4)



Algorytmy, struktury danych i techniki programowania. Wydanie IV
Piotr Wróblewski
Helion, 2009
Stron: 352

Algorytmy i struktury danych (3/50)

Algorytmy i struktury danych

Literatura podstawowa (#2/4)




Algorytmy + struktury danych = programy
Niklaus Wirth
Wydawnictwa Naukowo-Techniczne, 2000
Stron: 385


Algorytmy i struktury danych (4/50)

Algorytmy i struktury danych

Literatura podstawowa (#3/4)



Wprowadzenie do algorytmów. Wydanie VII
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
Wydawnictwo Naukowe PWN, 2014
Stron: 1400

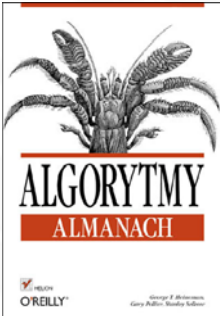


Algorytmy bez tajemnic. Wydanie I
Thomas H. Cormen
Helion, 2013
Stron: 223

Algorytmy i struktury danych (5/50)

Algorytmy i struktury danych

Literatura podstawowa (#4/4)

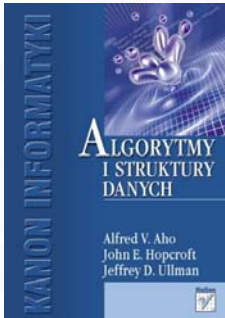


Algorytmy. Almanach.
George Heineman, Gary Pollice, Stanley Selkow
Helion, 2012
Stron: 351

Algorytmy i struktury danych (6/50)

Algorytmy i struktury danych

Literatura uzupełniająca (#1/4)

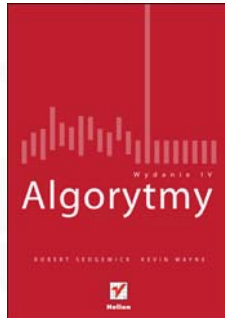


Algorytmy i struktury danych
**Alfred V. Aho, John E. Hopcroft,
 Jeffrey D. Ullman**
Helion, 2003
Stron: 448

Algorytmy i struktury danych (7/50)

Algorytmy i struktury danych

Literatura uzupełniająca (#2/4)

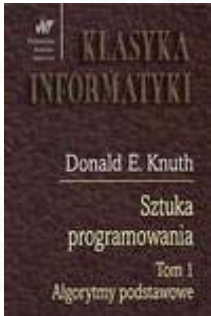


Algorytmy.
Wydanie IV
Robert Sedgwick, Kevin Wayne
Helion: 2012
Stron: 952

Algorytmy i struktury danych (8/50)

Algorytmy i struktury danych

Literatura uzupełniająca (#3/4)

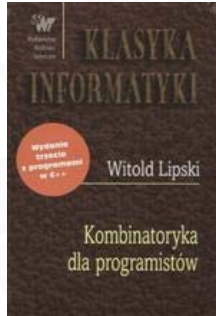


**Sztuka programowania
 Tom 1: Algorytmy podstawowe**
Donald E. Knuth
**Wydawnictwa Naukowo-
 Techniczne, 2002**
Stron: 679

Algorytmy i struktury danych (9/50)

Algorytmy i struktury danych

Literatura uzupełniająca (#4/4)




Kombinatoryka dla programistów
Witold Lipski
**Wydawnictwa Naukowo-
 Techniczne, 2004**
Stron: 276

Algorytmy i struktury danych (10/50)

Algorytmy i struktury danych

Czym jest algorytm ?

- recepta, przepis na wykonanie jakiejś czynności
- w matematyce/informatyce: skończony ciąg (sekwencja) reguł/czynności, które po zastosowaniu do skończonego zbioru danych, pozwalają rozwiązywać zbliżone do siebie klasy problemów (algorytm ma być możliwie uniwersalny)
- słowo algorytm pochodzi od nazwiska perskiego matematyka z IX w. Abū 'Abdallāh Muḥammad ibn Mūsā al-Khwārizmī (عَبْدَ اللَّهِ مُحَمَّدُ بْنُ مُوسَى الْخَوَارِزْمِي), al-Khwarizmi -> Al-Khwarithmi -> Algoritmi




Algorytmy i struktury danych (11/50)

Algorytmy i struktury danych

Jak zapisać algorytm ?

- język naturalny (<http://www.kwestiasmaku.com/>):
- jak zrobić omelet ?
 - składniki:
 - 2 jaja
 - 1 - 2 łyżki masła
 - 1 łyżka zimnej wody
 - sól, pieprz
 - całe jaja dobrze rozbić widelcem, dodając wodę, sól i pieprz
 - patelnię postawić na bardzo małym ogniu, potem zwiększyć płomień, rozgrzać masło, ale nie rumienić, wlać masę jajeczną.
 - smażyć bez przykrycia na dużym ogniu, aż brzeży omeletu się zetną
 - ... omelet smaży się nie dłużej niż 2 minuty, z większej ilości jajek trochę dłużej



Algorytmy i struktury danych (12/50)

Algoritmy i struktury danych


Jak zapisać algorytm ?

- pseudokod:
algorytm Forda-Fulkersona obliczania maksymalnego przepływu w sieci:

Wejście: Graf G o przepustowości c , węzeł źródłowy s i węzeł ujścia t

Wyjście: Maksymalny przepływ w fz s do t

- $f(u,v) \leftarrow 0$ dla wszystkich krawędzi (u,v)
- dopóki w G_f istnieje ścieżka p z s do t , taka że $c_f(u,v) > 0$ dla wszystkich krawędzi $(u,v) \in p$ wykonuj:
 - Znajdź $c_f(p) = \min\{c_f(u,v) \mid (u,v) \in p\}$,
 - Dla każdej krawędzi $(u,v) \in p$:
 - $f(u,v) \leftarrow f(u,v) + c_f(p)$
 - $f(v,u) \leftarrow f(v,u) - c_f(p)$



Algoritmy i struktury danych (13/50)


Algoritmy i struktury danych

Jak zapisać algorytm ?

- język formalny
 - zapis matematyczny, precyzyjna definicja ale czasami trudna w implementacji

$$\text{Silnia}(n) = \begin{cases} 1 & \text{dla } n = 0 \\ \text{Silnia}(n-1) * n & \text{dla } n > 0 \end{cases}$$

- język programowania
 - Pascal/Delphi
 - Java
 - C/C++/C#
 - VB, Fortran, Lisp ...




Algoritmy i struktury danych (14/50)

Algoritmy i struktury danych

Jak ocenić wydajność algorytmu ?

- kryteria:
 - (nie)zadowolenie użytkownika
 - częstotliwość uruchamiania
 - koszty implementacji (usprawnienia)
 - problem komunikacji
- czas wykonania („zawiesił się czy jeszcze liczy ?“)
- wymagania pamięciowe
- miara złożoności obliczeniowej powinna być niezależna od komputera (procesora), platformy programistycznej, opcji i bieżącego stanu systemu
- poszukujemy abstrakcyjnej miary odzwierciedlającej charakter zależności i pomijającej mniej istotne szczegóły




Algoritmy i struktury danych (15/50)

Algoritmy i struktury danych

Jak ocenić wydajność algorytmu ?

- zwykle wydajność algorytmu opisuje się jako zależność wzrostu czasu wykonywania od rozmiaru danych wejściowych (np. liczby elementów do przeszukania albo posortowania),
© jak opisać złożoność algorytmu sprawdzania czy dana liczba jest pierwsza ?
- kluczowe znaczenie ma identyfikacja operacji dominujących (mających największy wpływ na koszt) np. porównanie, zamiana elementów
- konieczna jest analiza zachowania dla różnych rozkładów danych wejściowych z uwzględnieniem identyfikacji najgorszego, typowego i ewentualnie najlepszego przypadku,
- określenie złożoności obliczeniowej algorytmu zwykle obejmuje złożoności:
 - pesymistyczną (maksymalną),
 - średnią (typową/oczekiwaną) oraz ewentualnie
 - optymistyczną (minimalną)
- złożoność algorytmu może być większa niż złożoność rozwiązywanego problemu




Algoritmy i struktury danych (16/50)

Algoritmy i struktury danych

Jak ocenić wydajność algorytmu ?

typy złożoności:

- pesymistyczna (maksymalna): stanowi górną granicę możliwego czasu działania algorytmu dla dowolnych danych wejściowych (algorytm na pewno nigdy nie będzie działał gorzej) w niektórych zastosowaniach najgorszy przypadek występuje stosunkowo często – np. wyszukiwanie informacji, której nie ma w danych, w systemach uwarunkowanych czasowo jej uwzględnienie jest konieczne, dla wielu algorytmów przypadek średni nie różni się istotnie od pesymistycznego
- średnia: pozwala ocenić zachowanie algorytmu w typowych sytuacjach (problemy: co to są typowe sytuacje, jaki jest rozkład prawdopodobieństwa wystąpienia określonych danych ?)
- minimalna: istotna w nietypowych sytuacjach, np. bezpieczeństwo szyfru – minimalny czas złamania szyfru określonym algorytmem




Algoritmy i struktury danych (17/50)

Algoritmy i struktury danych

Jak ocenić wydajność algorytmu ?

- zazwyczaj analizując złożoność przyjmujemy uproszczenia dotyczące mniej znaczących składników:
 - pomijamy rzeczywisty koszt poszczególnych instrukcji używając abstrakcyjnych stałych
 - koncentrujemy się na rzędzie wielkości funkcji – najwyższa potęga to najbardziej znaczący, najszybciej rosnący składnik w formule – inne składniki wraz ze wzrostem n stają coraz mniej istotne
- przeważnie przyjmujemy, że lepszy algorytm ma czas działania opisany funkcją niższego rzędu – może to być niesłuszne w przypadkach szczególnych takich jak wykorzystywanie algorytmu dla małych egzemplarzy danych wejściowych (warto rozważyć kombinację dwóch lub więcej algorytmów)
- w niektórych zastosowaniach stałe i składniki niższych rzędów mogą się okazać istotne (po to ulepszymy komputery)




Algoritmy i struktury danych (18/50)

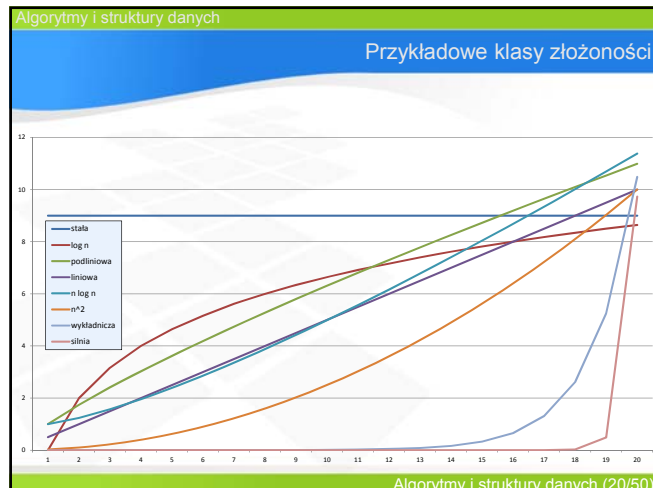
Algoritmy i struktury danych

Jak ocenić wydajność algorytmu ?

- złożoność obliczeniowa algorytmu jest mierzona jego wymaganiami czasowymi $T(n)$ i przestrzennymi $S(n)$, które są funkcjami zależnymi od rozmiaru problemu n
- funkcja ta jest nazywana złożonością teoretyczną i opisuje klasę algorytmu najczęściej za pomocą notacji „dużego O”
- notacja $O()$ ma charakter asymptotyczny (zachowanie wartości funkcji wraz z wzrostem jej argumentów):
 $f(n) = O(g(n))$ oznacza, że istnieją takie stałe c i n_0 , że:
 $f(n) \leq c \cdot g(n)$ dla $n \geq n_0$
- w notacji $O(n)$ zaniedbujemy stałe oraz niższe potęgi
- typowe klasy algorytmów:
 - wielomianowe: stała – $O(1)$, logarytmiczna – $O(\log n)$, liniowa – $O(n)$, kwadratowa – $O(n^2)$
 - wykładnicze: $O(2^n)$, $O(n!)$



Algoritmy i struktury danych (19/50)



Algoritmy i struktury danych

Zalety notacji „dużego O”


dla dużych n złożoność algorytmu może być przyczyną wielkich różnic w czasie odpowiedzi
 przyjmując, że komputer działa z prędkością operacji/μs:

Klasa	Złożoność	Liczba operacji dla $n = 10^6$	Czas rzeczywisty
Wielomianowa:			
stała	$O(1)$	1	1 μs
liniowa	$O(n)$	10^6	1 s
kwadratowa	$O(n^2)$	10^{12}	10 dni
sześcienne	$O(n^3)$	10^{18}	27,397 lat
Wykładnicza	$O(2^n)$	10^{301030}	10^{301016} lat

dla porównania:

1. Ziemia ma ca $4,5 \cdot 10^9$ lat
2. prawo Moore'a mówi, że moc komputerów podwaja się co 2 lata (szybsze komputery nie są rozwiązaniem wykładniczej złożoności)

ale wykładnicza złożoność obliczeniowa jest podstawą bezpieczeństwa informacji (np. szyfrów) ☺



Algoritmy i struktury danych (21/50)

Algoritmy i struktury danych


Wyszukiwanie

Problem: sprawdzenie czy określona liczba występuje w jednowymiarowej tablicy (wektorze) ?

1. na określonej pozycji
2. w całym wektorze
 - a) jeśli elementy nie są uporządkowane
 - b) jeśli elementy są posortowane rosnąco

Zadanie:

- ocenić w każdym przypadku złożoność minimalną, maksymalną i średnią



Algoritmy i struktury danych (22/50)

Algoritmy i struktury danych

Wyszukiwanie

Problem: czy liczba x występuje w wektorze w na pozycji y ?

Wejście:


- wektor w zawierający n liczb całkowitych

I_1	I_2	I_3	...	I_{n-2}	I_{n-1}	I_n
-------	-------	-------	-----	-----------	-----------	-------

- liczba x
- pozycja y

Wyjście:

- TAK – liczba x znajduje się w wektorze w na pozycji y
- NIE – liczba x nie znajduje się w wektorze w na pozycji y



Algoritmy i struktury danych (23/50)

Algoritmy i struktury danych


Wyszukiwanie

Problem: czy liczba x występuje w wektorze w na pozycji y ?

```
bool IsEqual(int[] Vector, int Number, int Position)
{
    return Vector[Position] == Number;
}
```

operacja dominująca (elementarna): operacja charakterystyczna dla danego algorytmu mająca bezpośredni wpływ na czas jego wykonania – porównanie ($==$)

złożoność czasowa: to zależność liczby operacji dominujących w funkcji rozmiaru danych wejściowych



Algoritmy i struktury danych (24/50)

Algorytmy i struktury danych

Wyszukiwanie

Problem: czy liczba x występuje w wektorze w na pozycji y ?

Przykład: czy liczba 5 występuje w wektorze w na pozycji 3. (2) ?

NIE

7	13	4	2	8	9	1
---	----	---	---	---	---	---

```
bool IsEqual(int[] Vector, int Number, int Position)
{
    return Vector[Position] == Number;
}
// ...
int[] w = { 7, 13, 4, 2, 8, 9, 1 };
Console.WriteLine(IsEqual(w, 5, 2));
```

liczba operacji (porównań): 1

Algorytmy i struktury danych (25/50)

Algorytmy i struktury danych

Wyszukiwanie

Problem: czy liczba x występuje w wektorze w na pozycji y ?

Przykład: czy liczba 8 występuje w wektorze w na pozycji 5. (4) ?

TAK

7	13	4	2	8	9	1
---	----	---	---	---	---	---

```
bool IsEqual(int[] Vector, int Number, int Position)
{
    return Vector[Position] == Number;
}
// ...
int[] w = { 7, 13, 4, 2, 8, 9, 1 };
Console.WriteLine(IsEqual(w, 8, 4));
```

liczba operacji (porównań): 1

Algorytmy i struktury danych (26/50)

Algorytmy i struktury danych

Wyszukiwanie

Problem: czy liczba x występuje w wektorze w na pozycji y ?

Analiza:

Czy liczba czynności zależy od rozmiaru wektora ?

Czy liczba czynności zależy od pozycji liczby ?

Czy liczba czynności zależy od wartości elementów ?

Algorytmy i struktury danych (27/50)

Algorytmy i struktury danych

Ocena wydajności

Problem: czy liczba x występuje w wektorze w na pozycji y ?

Analiza:

Czy liczba czynności zależy od rozmiaru wektora ?

Czy liczba czynności zależy od pozycji liczby ?

Czy liczba czynności zależy od wartości elementów ?

Wniosek:

Złożoność jest stała: $O(1)$ (minimalna = maksymalna = średnia)

Algorytmy i struktury danych (28/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

Wejście:

- wektor w zawierający n liczb całkowitych

l_1	l_2	l_3	...	l_{n-2}	l_{n-1}	l_n
-------	-------	-------	-----	-----------	-----------	-------

- liczba x

Wyjście:

- TAK – liczba x znajduje się w wektorze w
- NIE – liczba x nie znajduje się w wektorze w

Algorytmy i struktury danych (29/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

Algorytm:

- rozpoczynając od początku wektora porównujemy kolejne liczby
- zatrzymujemy się po znalezieniu liczby x lub osiągnięciu końca wektora w

l_1	l_2	l_3	...	l_{n-2}	l_{n-1}	l_n
-------	-------	-------	-----	-----------	-----------	-------

Algorytmy i struktury danych (30/50)

Algorytmy i struktury danych


Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

```
bool IsPresent(int[] Vector, int Number)
{
    for (int i = 0; i < Vector.Length; i++)
        if (Vector[i] == Number) return true;
    return false;
}
```

co jest operacją dominującą?

- przypisanie ($=$)
- porównanie ($<$)
- porównanie ($==$)
- inkrementacja ($++$)



Algorytmy i struktury danych (31/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?


Przykład: czy liczba 8 występuje w wektorze w ?

TAK

1	2	3	4	5			
7	13	4	2	8	9	1	

```
bool IsPresent(int[] Vector, int Number)
{
    for (int i = 0; i < Vector.Length; i++)
        if (Vector[i] == Number) return true;
    return false;
}
// ...
int[] w = { 7, 13, 4, 2, 8, 9, 1 };
Console.WriteLine(IsPresent(w, 8));
```

liczba operacji dominujących (porównań $==$): 5



Algorytmy i struktury danych (32/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

Jaka jest złożoność minimalna?


poszukiwana liczba jest na pierwszej pozycji

TAK

1							
7	13	4	2	8	9	1	

```
bool IsPresent(int[] Vector, int Number)
{
    for (int i = 0; i < Vector.Length; i++)
        if (Vector[i] == Number) return true;
    return false;
}
// ...
int[] w = { 7, 13, 4, 2, 8, 9, 1 };
Console.WriteLine(IsPresent(w, 7));
```

liczba operacji dominujących (porównań $==$): 1



Algorytmy i struktury danych (33/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

Jaka jest złożoność maksymalna?


poszukiwana liczba jest na ostatniej pozycji albo nie ma jej w wektorze

1	2	3	4	5	6	7	
7	13	4	2	8	9	1	

liczba na ostatniej pozycji: $IsPresent(w, 1)$

liczby nie ma w wektorze: $IsPresent(w, 5)$

liczba operacji (porównań $==$): 7 = liczba elementów tablicy = n



Algorytmy i struktury danych (34/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Problem: czy liczba x występuje w nieuporządkowanym wektorze w ?

Analiza:


Czy liczba czynności zależy od rozmiaru wektora?

Czy liczba czynności zależy od pozycji liczby?

Czy liczba czynności zależy od wartości elementów?

Ocena złożoności:

- analitycznie – znajdujemy funkcję $f(n)$
- empirycznie:
 - instrumentacja kodu
 - pomiar czasu, profilowanie



Algorytmy i struktury danych (35/50)

Algorytmy i struktury danych

Wyszukiwanie liniowe

Analityczna ocena średniej złożoności algorytmu

założenia:

- tablica ma rozmiar n (zawiera n elementów)
- element znajduje się w tablicy (jeśli nie to mamy złożoność pesymistyczną)
- każdy element występuje w tablicy z jednakowym prawdopodobieństwem

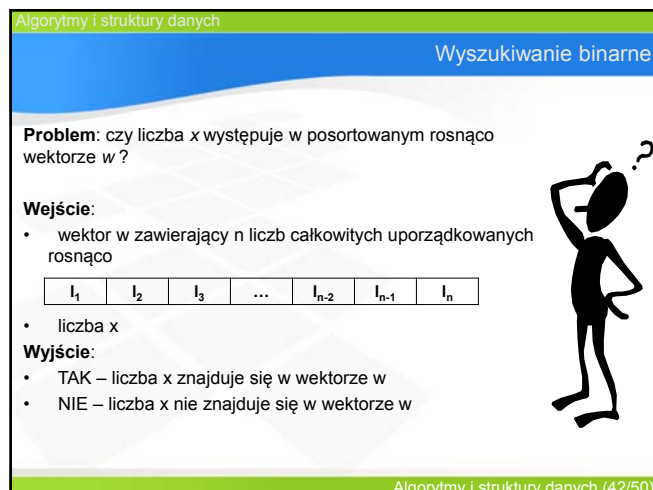
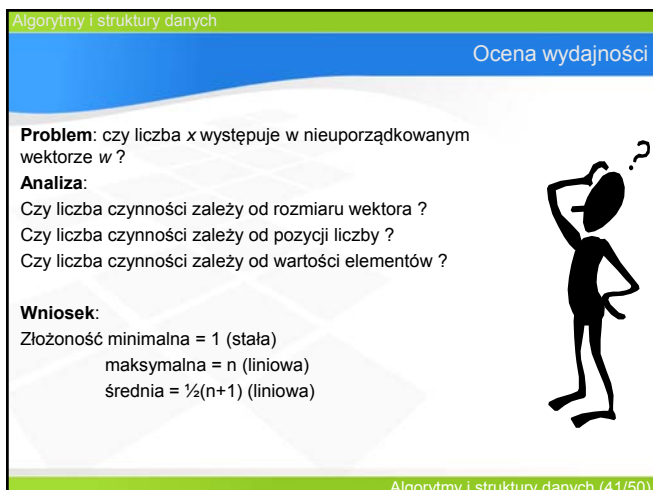
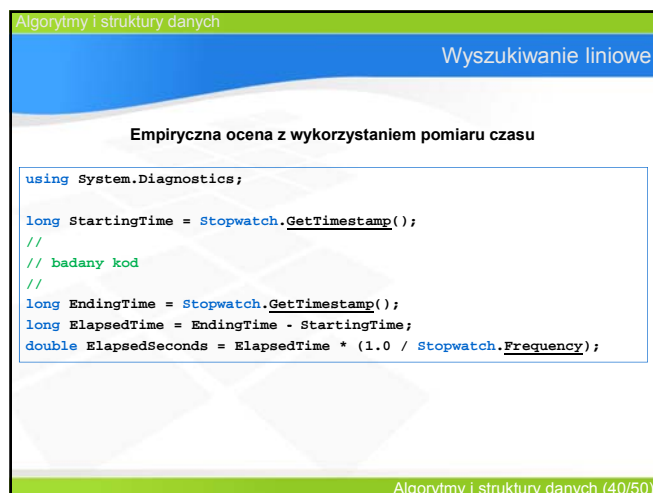
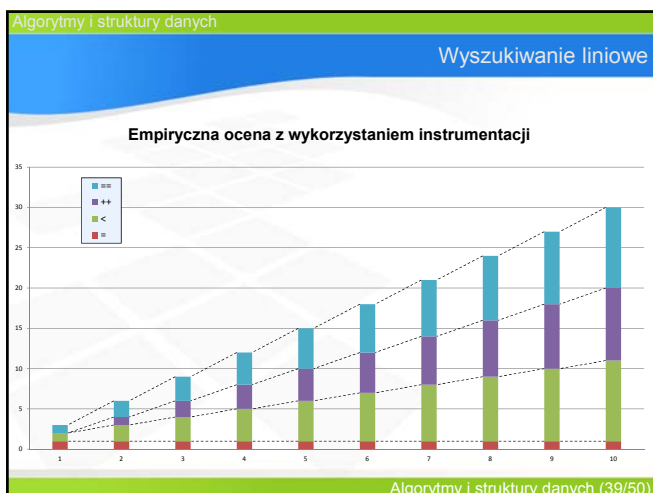
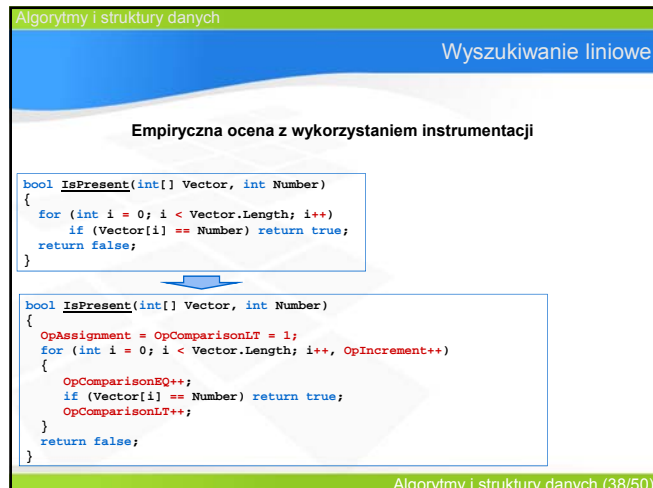
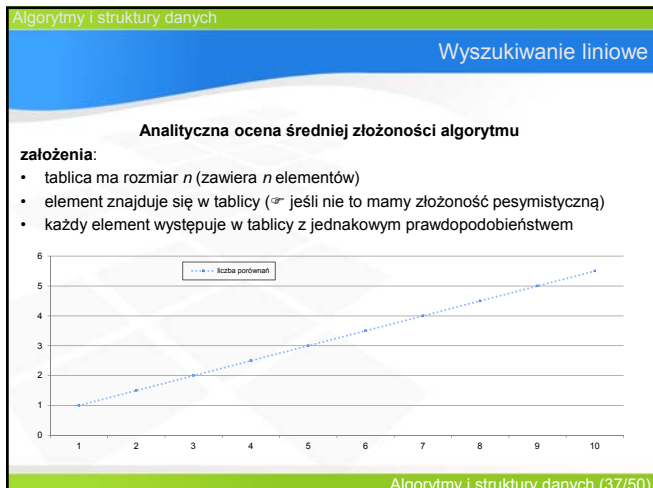
średnia liczba porównań: $\frac{\sum_{i=1}^n i}{n} = \frac{1+2+\dots+n}{n}$

ciąg arytmetyczny:

n -ty wyraz: $a_n = a_1 + (n-1)r$ suma elementów ciągu: $S_n = \frac{a_1 + a_n}{2} n$

średnia liczba porównań: $\frac{\sum_{i=1}^n i}{n} = \frac{1+2+\dots+n}{n} = \frac{1}{n} * \frac{1+(1+(n-1)*1)}{2} * n = \frac{1+n}{2} = \frac{1}{2} + \frac{n}{2}$

Algorytmy i struktury danych (36/50)



Algorytmy i struktury danych

Wyszukiwanie binarne

Problem: czy liczba x występuje w posortowanym rosnąco wektorze w ?

Algorytm:

- porównujemy liczbę x z elementem w połowie wektora, stop jeśli jest równa lub lewy indeks wektora jest większy od prawego
- jeśli liczba x jest większa od badanego elementu powtarzamy algorytm dla prawej połowy wektora, jeśli mniejsza – dla lewej

liczba porównań: (n = 7)

Algorytmy i struktury danych (43/50)

Algorytmy i struktury danych

Wyszukiwanie binarne

Problem: czy liczba x występuje w posortowanym rosnąco wektorze w ?

Algorytm:

- porównujemy liczbę x z elementem w połowie wektora, stop jeśli jest równa lub lewy indeks wektora jest większy od prawego
- jeśli liczba x jest większa od badanego elementu powtarzamy algorytm dla prawej połowy wektora, jeśli mniejsza – dla lewej

poziomy węzłów:

(zrównoważone) drzewo wyszukiwań binarnych

liczba porównań: (n = 7)

(in-order)

Algorytmy i struktury danych (44/50)

Algorytmy i struktury danych

Wyszukiwanie binarne

Problem: czy liczba x występuje w posortowanym rosnąco wektorze w ?

Jaka jest złożoność maksymalna?

Przykład: czy liczba 8 występuje w wektorze w ?

liczba operacji (porównań ==): 3

Algorytmy i struktury danych (45/50)

Algorytmy i struktury danych

Wyszukiwanie binarne

Problem: jaka jest średnia złożoność wyszukiwania binarnego?

$\frac{1+2}{2} = \frac{3}{2} = 1,5$

$\frac{1+2+2+3+3+3}{6} = \frac{14}{6} \approx 2,33$

średnia złożoność wyszukiwania binarnego to:

$\frac{\text{suma poziomów węzłów drzewa BST}}{\text{liczba elementów}}$

Algorytmy i struktury danych (46/50)

Algorytmy i struktury danych

Wyszukiwanie binarne

Problem: czy liczba x występuje w posortowanym rosnąco wektorze w ?

```

bool IsPresent(int[] Vector, int Number)
{
    int Left = 0, Right = Vector.Length - 1, Middle;
    while(Left <= Right)
    {
        Middle = (Left + Right) / 2;
        if (Vector[Middle] == Number) return true;
        else if (Vector[Middle] > Number) Right = Middle - 1;
        else Left = Middle + 1;
    }
    return false;
}
    
```

przyjmujemy, że operacją dominującą jest porównanie elementów (==)

Algorytmy i struktury danych (47/50)

Algorytmy i struktury danych

Porównanie metod wyszukiwania

porównanie maksymalnych (L_{max}) i średnich (L_{avg}) kosztów wyszukiwania liniowego i binarnego (odpowiednio B_{max} , B_{avg}):

Algorytmy i struktury danych (48/50)

Problem: czy liczba x występuje w posortowanym rosnąco wektorze w ?

Analiza:

Czy liczba czynności zależy od rozmiaru wektora?

Czy liczba czynności zależy od pozycji liczby?

Czy liczba czynności zależy od wartości elementów?

Wniosek:

Złożoność minimalna = $O(1)$ (stała)

maksymalna = $O(\log n)$ (logarytmiczna $\log_2 n$)

średnia = $O(\log n)$



dla uporządkowanych (posortowanych) danych przewagę wyszukiwania binarnego nad liniowym dla rosnących n można zademonstrować na przykładzie kosztów wyszukiwania w książkach telefonicznych o różnej liczbie numerów:

Liczba numerów	lsearch max	lsearch avg	bsearch max	bsearch avg
2 (Luboszów)	2	1,5	2	1,5
$2 \cdot 10^5$ (Poznań)	200 000	100 000,5	18	16,69
10^6 (Warszawa)	1 000 000	500 000,5	20	18,95

Wnioski:

- warto uporządkować dane jeśli operacja wyszukiwania będzie wykonywana wielokrotnie
- ulepszony algorytm (wyszukiwanie binarne) może dla większych rozmiarów problemów osiągać znaczącą przewagę nad prostszym (wyszukiwanie liniowe)
- wyrafinowany algorytm dla relatywnie małych rozmiarów problemu może mieć porównywalną wydajność