

Wykład 1

Wprowadzenie do algorytmów

Zawartość wykładu

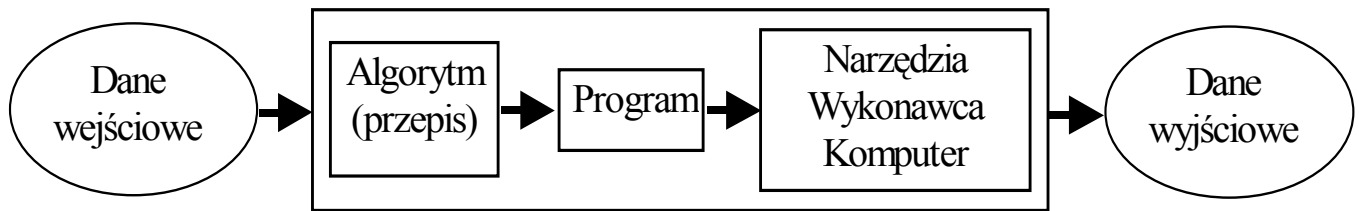
1. Wstęp do algorytmów i struktur danych
2. Algorytmy z rozgałęzieniami

Wykaz literatury

- 1.N. Wirth - *Algorytmy+Struktury Danych = Programy*, WNT Warszawa 1999
- 2.R. Sedgewick - *Algorytmy w C++*, Oficyna Wydawnicza READ ME - Warszawa, 1999
- 3.T.H. Cormen, Ch E. Leiserson, R.L. Rivest - *Wprowadzenie do algorytmów*, WNT Warszawa 1997, 1998
- 4.L Banachowski, K.Diks, W. Rytter - *Algorytmy i struktury danych*, WNT Warszawa, 1999
- 5.D. Harel - *Rzecz o istocie informatyki, Algorytmika*, WNT Warszawa, 1992
- 6.J. Bentley - *Perły oprogramowania*, WNT Warszawa, 1992

1. Wstęp do algorytmów i struktur danych

Algorytm



Definicje algorytmu:

- 1) reguła przekształcania wyrażeń matematycznych przez przetwarzanie tych samych działań na kolejno otrzymywanych wynikach działań poprzednich
- 2) dokładny przepis wykonania szeregu operacji w celu rozwiązania określonego zagadnienia, a może być wykorzystany do rozwiązania całej grupy problemów należących do określonej klasy.

Definicja informatyki:

Informatyka- zespół dyscyplin naukowych i technicznych zajmujących się przetwarzaniem informacji, zwłaszcza przy użyciu środków automatycznych (np. komputerów).

Przetwarzanie informacji - wykonywanie usystematyzowanego ciągu operacji na zbiorze danych; operacjami tymi mogą być: sortowanie, wyszukiwanie, wyliczanie itp.

Przykłady algorytmów:

- przepis na sernik,
- algorytm Euklidesa (znajdowanie największego wspólnego dzielnika).

Budowa algorytmu wymaga zastosowania:

- danych, struktur danych
- instrukcji sterujących i wyrażeń.
















Schematy blokowe, język ELI2D

Laboratorium Informatyki 2.1 - Pomoc

Plik Edycja Zakładka Pomoc

Spis treści Szukaj Wstecz Historia

Klocki podstawowe

-  Początek algorytmu
-  Koniec algorytmu
-  Wykonanie obliczeń
-  Sprawdzenie warunku
-  Wprowadzenie danej
-  Wyprowadzenie wyniku
-  Odczyt z tablicy
-  Zapis do tablicy
-  Odczyt z taśmy
-  Zapis na taśmę
-  Przewinięcie taśmy
-  Przesunięcie wskaźnika na określoną pozycję
-  Sygnał dźwiękowy
-  Przepisanie z tablicy na taśmę
-  Przepisanie z taśmy do tablicy

2. Algorytmy z rozgałęzieniami

Sortowanie trzech danych z wykorzystaniem relacji między ich wartościami

1) Sformułowanie problemu – drzewo algorytmu

a) porządkowanie zbioru liczb przez porównania ich wartości

Liczba danych	Pary	Liczba porównań	Całkowita liczba porównań
n=2	(d1,d2)	1	1=2-1=1
n=3	(d1,d2),(d1,d3) (d2,d3)	3	3=(3-1)+(3-2) =2+1

Liczba porównań jest liczbą 2-kombinacji zbioru n elementowego i jest równa:

$$\frac{n!}{2!(n-2)!} = \frac{n * (n-1) * (n-2)!}{2 * (n-2)!} = \frac{n * (n-1)}{2}$$

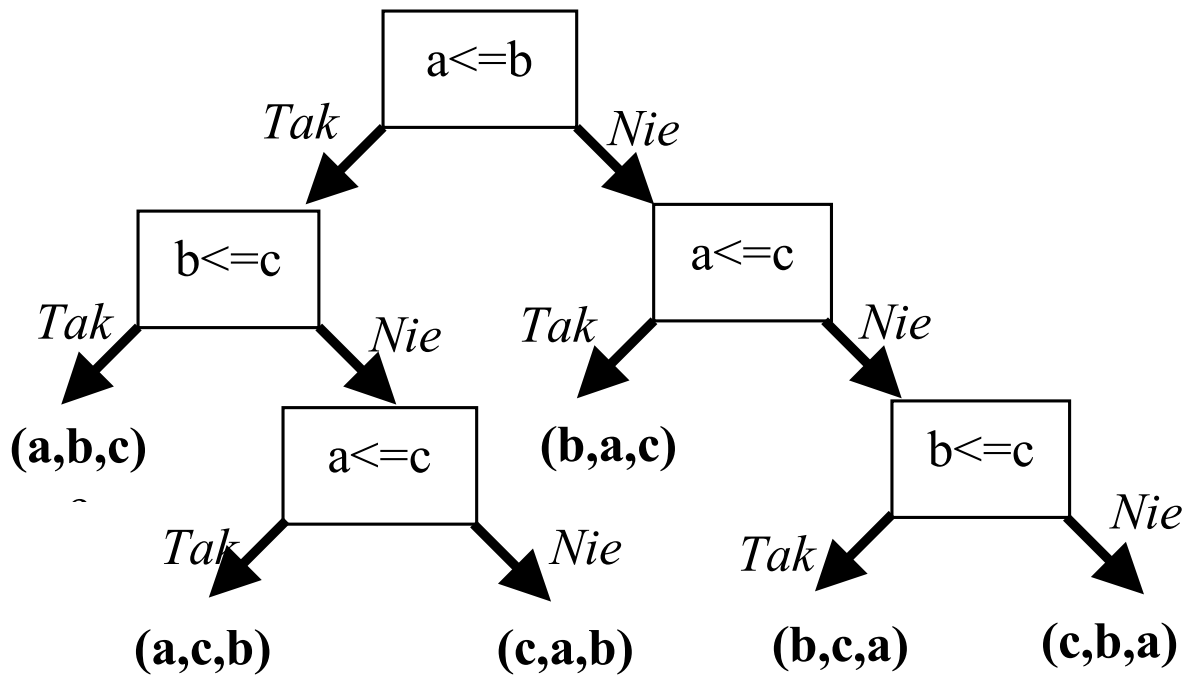
a) liczba uporządkowań trzech danych jest równa $u=1*2*3$, ponieważ każda z danych ustawiona na trzecim miejscu jest łączona z dwoma ustawieniami pary pozostałych liczb ustawionych przed nią.
Jest to liczba wszystkich permutacji trzech danych, czyli 3!.
Dla n danych liczba permutacji jest równa n!, czyli tyle jest różnych uporządkowań n danych

b) drzewo algorytmu pozwala wyznaczyć wszystkie uporządkowania danych oraz określić pracochłonność tego algorytmu czyli złożoność obliczeniową algorytmu:

- najlepszą (dwa porównania)
- najgorszą (trzy porównania)

równą minimalnej i maksymalnej liczbie poziomów drzewa algorytmu

d) drzewo algorytmu



2) Projekt programu

- należy wprowadzić z klawiatury trzy liczby a , b , c
- należy przyjąć, że każde porównanie wykonane w węźle drzewa jest warunkiem instrukcji **if**
- przejsięcie na lewo w drzewie oznacza wykonanie instrukcji po **if**, która jest końcową instrukcją wyprowadzenia trójki uporządkowanych liczb
- przejsięcie na prawo w drzewie oznacza wykonanie alternatywnej instrukcji po **else**, która albo jest kolejną instrukcją **if** albo końcową instrukcją wyprowadzenia trójki uporządkowanych liczb
- liście drzewa odpowiadają instrukcji wyprowadzenia trójki uporządkowanych liczb czyli wyświetleniu na ekranie trójki uporządkowanych liczb zgodnie z ich bieżącymi wartościami

3) Implementacja

3.1) Program w języku Eli2D

Elbox Laboratorium Informatyki 2.01 - liczby1

Plik Edycja Szukaj Widok Wykonanie Opcje Okna Pomoc

The screenshot shows the Elbox software interface. The main window displays a flowchart for an algorithm. The flowchart consists of several vertical columns of yellow boxes connected by arrows, representing a sequence of operations and decision points. The flow starts from the top left and moves downwards through several columns, with decision points (diamonds) branching the flow. The flowchart is set against a green background.

Ślad

Plansza	Poziom	Wynik	Opis
liczby1	1		Początek algorytmu
liczby1	1	3	Podaj liczbę a
liczby1	1	1	Podaj liczbę b
liczby1	1	2	Podaj liczbę c
liczby1	1	Fałsz	$a <= b$
liczby1	1	Fałsz	$a <= c$
liczby1	1	Prawda	$b <= c$
liczby1	1	1	(b,c,a)
liczby1	1	2	(b,c,a)
liczby1	1	3	(b,c,a)
liczby1	1		Koniec algorytmu

Złożoność

Operacja	Liczba
Dodawanie	0
Odejmowanie	0
Mnożenie	0
Dzielenie	0
Moduł	0
Przypisanie	0
Porównanie	3
Wywołanie procedury	0
Funk. trygonometryczne	0
Funk. logarytmiczne	0
Inne funkcje	0
Odwokanie do tablicy	0

Wyrowadzenie wyniku: (c,b,a)

3.2) Program w języku C++

```
#pragma hdrstop
#include <stdio.h>
#include <conio.h>
//-----
#pragma argsused
void main(int argc, char* argv[])
{
    int a,b,c;

    printf("%s","Podaj a: ");
    scanf("%d",&a);
    printf("%s","Podaj b: ");
    scanf("%d",&b);
    printf("%s","Podaj c: ");
    scanf("%d",&c);
    if (a<=b)
        if (b<=c)
            printf("\n a=%i, b=%i, c=%i", a,b,c);           //a<=b, b<=c
        else
            if (a<=c)
                printf("\n a=%i, c=%i, b=%i", a,c,b);       // a<=b, b>c, a<=c
            else
                printf("\n c=%i, a=%i, b=%i", c,a,b);       // a<=b, b>c, a>c
        else
            if (a <= c)
                printf("\n b=%i, a=%i, c=%i", b,a,c);       // a>b, a<=c
            else
                if (b<=c)
                    printf("\n b=%i, c=%i, a=%i", b,c,a);   //a>b, a>c, b<=c
                else
                    printf("\n c=%i, b=%i, a=%i", c,b,a);   //a>b, a>c, b>c
    getch();
}
```

Złożoność obliczeniowa programu w języku C/C++

Miarą oszacowania złożoności obliczeniowej programu może być tzw. czasochłonność operacji, czyli liczba elementarnych jednostek czasowych, jakie są potrzebne do wykonania wszystkich operacji w programie

Operacja	Czasochłonność
Funkcje printf, scanf	1000
Funkcje malloc, free	800
Funkcje trygonometryczne (sin, cos...)	500
Operacje zmiennoprzecinkowe	100
Dzielenie liczb całkowitych	30
Mnożenie liczb całkowitych	20
Wywołanie funkcji	10
Indeksowanie tablicy	6
Operacje przesunięcia	5
Dodawanie/odejmowanie, przypisanie	5
Wyłuskanie wskaźnika	2
Operatory bitowe: &, , ~	1
Operatory logiczne: &&, , !	1