

Wstęp do algorytmiki

- Marek Gajęcki, Katedra Informatyki AGH D-17, pok. 1.23, e-mail: mag@agh.edu.pl
- Slajdy z wykładu i inne materiały dostępne w systemie Moodle

1

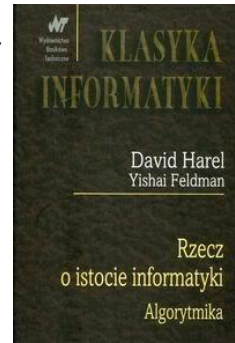
Zakres wykładu

- Pojęcia podstawowe
 - Algorytm, złożoność algorytmu
 - Język programowania, składnia i semantyka języka
- Podstawowe mechanizmy języka Python
- Typy skalarne , struktury danych
- Funkcje, rekurencja
- Zaawansowane mechanizmy języka Python
- Przykładowe biblioteki

2

Literatura

- D. Harel „Rzecz o istocie informatyki - algorytmika”
- J.G. Brookshear „Informatyka w ogólnym zarysie”
- J. Mieścicki „Wstęp do informatyki nie tylko dla informatyków”
- Mark Lutz „Python. Wprowadzenie”
- T.H. Cormen „Wprowadzenie do algorytmów”



3

Podstawowe pojęcia

Zadanie algorytmiczne – polega na określeniu:

- wszystkich poprawnych danych wejściowych
- oczekiwanych wyników jako funkcji danych wejściowych

Algorytm - specyfikacja ciągu elementarnych operacji, które przekształcają dane wejściowe na wyniki.

Algorytm można przedstawić w postaci:

- werbalnej (*opis słowny*)
- symbolicznej (*schemat blokowy*)
- wizualnej (*Scratch*)
- programu

4

Przykład zapisu algorytmu

Problem: równanie kwadratowe

Dane: współczynniki **a,b,c**

Wyjście: pierwiastki **x1,x2** albo informacja o ich braku

Postać werbalna algorytmu:

wczytaj wartości współczynników a,b,c

oblicz wyróżnik równania według wzoru $d = b^2 - 4ac$

jeżeli d jest nieujemne to :

oblicz pierwiastek z delty jako p

oblicz wartości pierwiastków według wzorów:

$$x1 = (-b-p)/(2*a)$$

$$x2 = (-b+p)/(2*a)$$

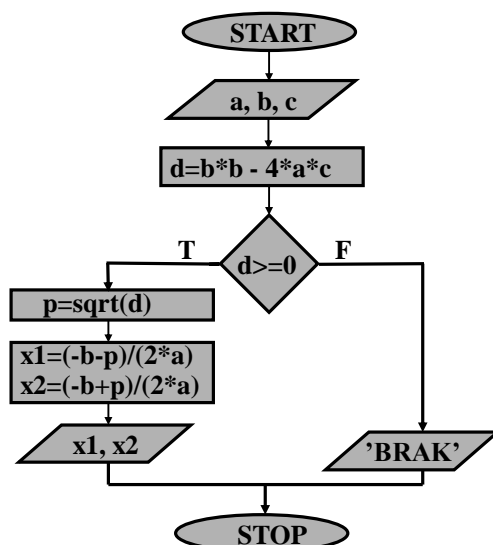
wypisz wartości x1, x2

w przeciwnym przypadku :

wypisz "BRAK PIERWIASTKÓW"

5

Zapis symboliczny a program

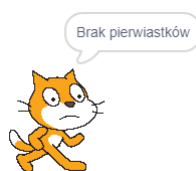
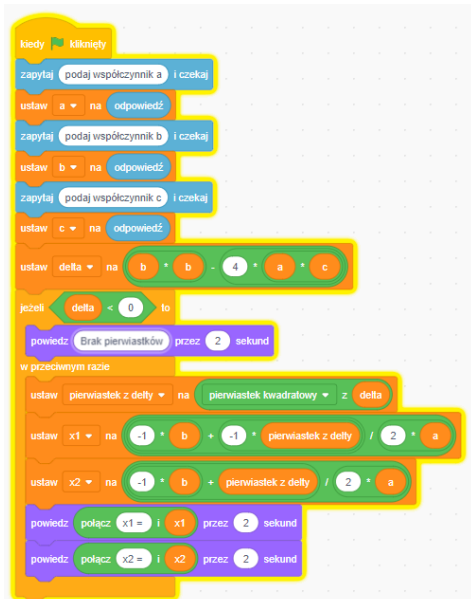


```
from math import sqrt

a=float(input("a="))
b=float(input("b="))
c=float(input("c="))
d=b*b-4*a*c
if d>=0:
    p=sqrt(d)
    x1=(-b-p)/(2*a)
    x2=(-b+p)/(2*a)
    print("x1=",x1)
    print("x2=",x2)
else:
    print("BRAK")
```

6

Wizualny język programowania



7

Przykład zadania

Problem: Rozkład liczby na czynniki pierwsze

Proszę napisać program, który dla wczytanej liczby naturalnej wypisuje jej rozkład na czynniki pierwsze.

Przykład:

120 : 2, 2, 2, 3, 5

8

Rozkład na czynniki pierwsze

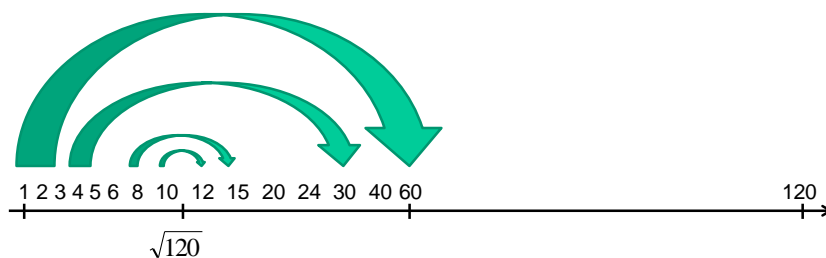
Rozwiązanie pierwsze

```
n=int(input("n="))
b=2
while n>1:
    if n % b == 0:
        print(b)
        n = n // b
    else:
        b = b+1
    # end if
# end while
print("koniec")
```

9

Położenie podzielników liczby

Podzielniki liczby 120



10

Rozkład na czynniki pierwsze

Rozwiązanie drugie

```
n=int(input("n="))
b=2
while n>1 and b<=sqrt(n):
    if n % b == 0:
        print(b)
        n = n // b
    else:
        b = b+1
    # end if
# end while
if n>1: print(n)
print("koniec")
```

11

Porównanie rozwiązań

Liczba cyfr	Algorytm pierwszy	Algorytm drugi
6	0.07s	0.0s
7	0.58s	0.0s
8	7.75s	0.0s
9	1m7.2s	0.01s
10	9m43s	0.01s
11	1h23m	0.04s
12	12h27m	0.13s
13	4d9h	0.42s
14	37d12h	1.23s

Czy można jeszcze szybciej?

12

Algorytm

Algorytm jako technologia

- Współczesne zaawansowane technologie informatyczne:
 - sprzęt (procesory, karty graficzne)
 - sieci (lokalne, globalne)
- Algorytmy są równie istotne co powyższe technologie

Analiza algorytmów:

- to dział informatyki zajmujący się szukaniem najlepszych algorytmów dla zadań komputerowych,
- odpowiada na pytania:
 - Czy dany problem może być rozwiązany na komputerze w dostępnym czasie i pamięci?
 - Który algorytm zastosować w danych okolicznościach?
 - Czy jest lepszy algorytm (czy jest on optymalny)?
 - Jak uzasadnić, że algorytm rozwiązuje postawione zadanie?

13

Złożoność obliczeniowa algorytmu

Definiuje się ją jako:

- ilość zasobów komputerowych potrzebnych do jego wykonania (czas procesora, wielkość pamięci).

Wyróżnia się:

- złożoność pesymistyczną (ilość zasobów potrzebnych przy „najgorszych” danych wejściowych o rozmiarze n),
- złożoność oczekiwaną (ilość zasobów potrzebnych przy „typowych” danych wejściowych o rozmiarze n),

14

Funkcja złożoności obliczeniowej

Funkcja złożoności obliczeniowej algorytmu rozwiązującego dany problem to funkcja przyporządkowująca każdej wartości rozmiaru konkretnego problemu maksymalną liczbę kroków elementarnych (lub jednostek czasu) komputera potrzebnych do rozwiązania za pomocą tego algorytmu konkretnego problemu o tym rozmiarze.

15

Notacja asymptotyczna

Mówimy, że funkcja g **jest co najwyżej rzędu** f ,
gdy istnieją takie stałe $n_0 > 0$, oraz $c_2 > 0$, że

$$\forall n \geq n_0 : g(n) \leq c_2 \cdot f(n)$$

Co zapisujemy: $g(n) \in O(f(n))$

Mówimy, że funkcja g **jest dokładnie rzędu** f ,
gdy istnieją takie stałe $n_0 > 0$, oraz $c_1, c_2 > 0$, że

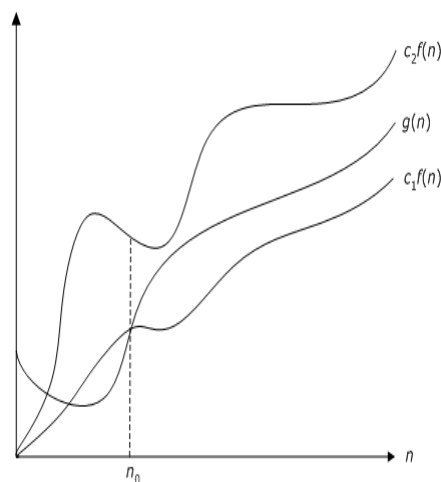
$$\forall n \geq n_0 : c_1 \cdot f(n) \leq g(n) \leq c_2 \cdot f(n)$$

Co zapisujemy: $g(n) \in \theta(f(n))$

Mówimy, że funkcja g **jest co najmniej rzędu** f ,
gdy istnieją takie stałe $n_0 > 0$, oraz $c_1 > 0$, że

$$\forall n \geq n_0 : c_1 \cdot f(n) \leq g(n)$$

Co zapisujemy: $g(n) \in \Omega(f(n))$



16

Rząd złożoności obliczeniowej

Funkcja $g(n)$ jest rzędu $f(n)$, co zapisujemy $O(f(n))$, jeżeli istnieje taka stała c_2 , że $g(n) \leq c_2 * f(n)$ dla prawie wszystkich wartości n .

Przykłady:

$O(1)$	stała
$O(\log_2 N)$	logarytmiczna
$O(N)$	liniowa
$O(N * \log_2 N)$	liniowo-logarytmiczna
$O(N^2)$	kwadratowa (wielomianowa)
$O(2^N)$	wykładnicza
$O(N!)$	silnia

Problem stałej:

$$f_1(N) = N$$

$$f_2(N) = 100 * \log_2 N$$

17

Złożoność czasowa algorytmu

Algorytm wielomianowy (o złożoności czasowej wielomianowej) to taki, którego złożoność jest $O(p(n))$, gdzie $p(n)$ jest wielomianem, a n jest rozmiarem problemu.

Algorytm wykładniczy (o złożoności czasowej wykładniczej) to taki, który nie jest wielomianowy.

18

Złożoność czasowa a czas wykonania

Rozmiar problemu Funkcja złożoności	10	20	30	40	50
n	$10 \cdot 10^{-6}$ sekundy	$20 \cdot 10^{-6}$ sekundy	$30 \cdot 10^{-6}$ sekundy	$40 \cdot 10^{-6}$ sekundy	$50 \cdot 10^{-6}$ sekundy
$n \log_2 n$	$33,2 \cdot 10^{-6}$ sekundy	$86,4 \cdot 10^{-6}$ sekundy	$147,2 \cdot 10^{-6}$ sekundy	$212,9 \cdot 10^{-6}$ sekundy	$282,5 \cdot 10^{-6}$ sekundy
n^2	$0,1 \cdot 10^{-3}$ sekundy	$0,4 \cdot 10^{-3}$ sekundy	$0,9 \cdot 10^{-3}$ sekundy	$1,6 \cdot 10^{-3}$ sekundy	$2,5 \cdot 10^{-3}$ sekundy
n^3	$1 \cdot 10^{-3}$ sekundy	$8 \cdot 10^{-3}$ sekundy	$27 \cdot 10^{-3}$ sekundy	$64 \cdot 10^{-3}$ sekundy	$125 \cdot 10^{-3}$ sekundy
2^n	0,001 sekundy	1 sekunda	17,9 minuty	12,7 dnia	35,7 lat
3^n	0,059 sekundy	58,1 minuty	6,53 roku	3 855 wieków	$2,3 \cdot 10^8$ wieków
10^n	2,8 godziny	31710 wieków	$3,17 \cdot 10^{14}$ wieków	$3,17 \cdot 10^{24}$ wieków	$3,17 \cdot 10^{34}$ wieków

19

Początek nauki algorytmiki

- Proste programy z pętlami
- Zadania z tablicami (listami)
- Funkcje, przekazywanie parametrów
- Zastosowania rekordów
- Operacje na plikach
- Rekurencja
- Wskaźniki, alokacja pamięci
- Podstawowe algorytmy

20

Project Euler

https://projecteuler.net/problem=10

Project Euler.net

Logged in as mag1964
Sat, 24 Aug 2019, 09:14

AboutArchivesRecentProgressAccountNewsFriendsStatisticsSign Out


Summation of primes

Problem 10

The sum of the primes below 10 is $2 + 3 + 5 + 7 = 17$.
Find the sum of all the primes below two million.

Answer:

Confirmation Code:


Click image for new code

Project Euler: Copyright Information | Privacy Policy

21

Popularność języków programowania

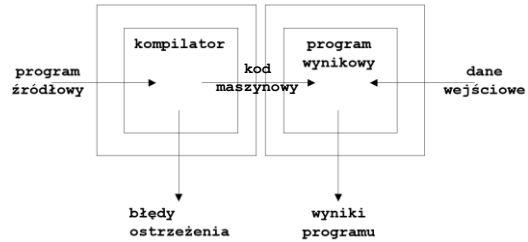
Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	▲	Python	10.020%	+3.03%
4	3	▼	C++	6.057%	-1.41%
5	6	▲	C#	3.842%	+0.30%
6	5	▼	Visual Basic .NET	3.695%	-1.07%
7	8	▲	JavaScript	2.258%	-0.15%
8	7	▼	PHP	2.075%	-0.85%
9	14	▲▲	Objective-C	1.690%	+0.33%
10	9	▼	SQL	1.625%	-0.69%
11	15	▲▲	Ruby	1.316%	+0.13%
12	13	▲	MATLAB	1.274%	-0.09%
13	44	▲▲	Groovy	1.225%	+1.04%
14	12	▼	Delphi/Object Pascal	1.194%	-0.18%
15	10	▼▼	Assembly language	1.114%	-0.30%

www.tiode.com

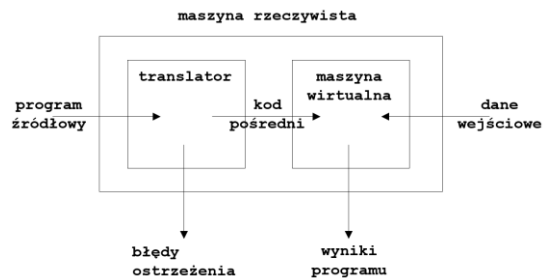
22

Wykonanie programu

Język C++



Język Python



23

Podstawowe pojęcia

Aspekty języka programowania:

- **Syntaktyka** (składnia) - zbiór reguł określający formalnie poprawne konstrukcje językowe
- **Semantyka** - opisuje znaczenie konstrukcji językowych, które są poprawne składniowo

24

Gramatyka języka w notacji EBNF

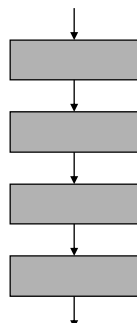
```
<pgm> ::= <pgmHeading> <pgmDeclarations> <codeBlock> "."
<pgmHeading> ::= program <pgmIdentifier> ","
<pgmDeclarations> ::= { <pgmDeclaration> }
<pgmDeclaration> ::= <varDeclaration> | <typeDeclaration> | <procDeclaration>
....
<codeBlock> ::= begin { <statement> } end
<statement> ::= <assignStatement> | <ifStatement> | <whileStatement> | <procCall>
<assignStatement> ::= <variable> "=" <expression> ";"
<ifStatement> ::= if <expression> then <codeBlock> [ else <codeBlock> ] ";"
<whileStatement> ::= while <expression> do <codeBlock> ";"
....
<identifier> ::= <letter> { <letter> | <digit> }
<longint> ::= <digit> { <digit> }
<relOperator> ::= "=" | "<" | ">" | "<=" | ">" | ">="
<addOperator> ::= "+" | "-" | or
<multOperator> ::= "*" | div | and
<letter> ::= "A" | ... | "Z" | "a" | ... | "z"
<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

25

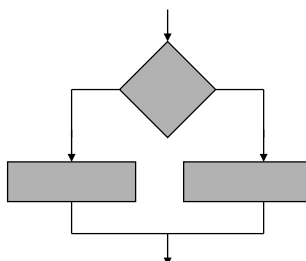
Budowa algorytmów

- 1) Bezpośrednie następstwo
- 2) Wybór warunkowy
- 3) Iteracja warunkowa

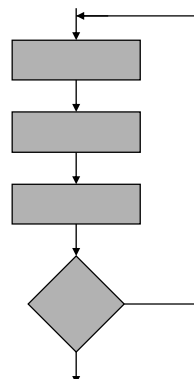
1)



2)



3)



26