

Algorytmy

DOKŁADNA NAUKA
WARZENIA PIWA.

WEDŁUG METODY ŁATWEJ, STWIERDZONEJ
OSMIOLETNIEM DOSWIADCZENIEM.

DO WYNAŁAZKÓW NAYNOWSZYCH
ZASTOSOWANA,

Z PRZYDANEM OPISANIEM APPARATU DO STUDDZENIA, ZASTR-
PIĄCEGO ZWIĘZAJĄC KILKATKI, ZA POMOCĄ KTÓREGO PI-
WO WLIĄCIE, W PRZECIĄGU IEDNEJ MINUTY DO TEMPERATURY
WODY STUDDZENNEJ OCIECZONE BYDŃ MOŻE;

PRZEDSTAWIONA W SPOSOBIE PRAKTYCZNYM,

PRZEZ KAROLA WILHELMA SCHMIDT,
AUTORA ROZMAITYCH DZIEŁ TECHNOLOGICZNYCH.

PRZEŁOŻONA Z JĘZYKA NIEMIECKIEGO,
DLA UŻYTKU ZIEMIANY POLSKICH.



WARSZAWA.

NAKLAD I Druk N. GLÜCKSBERGA,
KSIĘGARNIA I TYPOGRAFIA KRÓL: WARSZAW: UNIWERSYT.

1830.

Chambers udziela w swych pismach odmienny spo-
sób robienia Piwa Brunświckiego, odpisany iak twier-
dzi, z pierwotný Recepty, zachowywauący na Ratuszu
miejskim w Brunświku.

Według Recepty téy wziąć należy 200 miarek
zwanych *Kanne* wody, i te tak długo gotować, po-
kąd 1/3 części iéy nie ubędzie. Sypie się potem do
téy wody siedm Szefłów siodu Pszennego i jeden Sze-
fel drobný Fasoli; gdy odleie się do beczki dla wy-
robienia, nie należy zapelniać iéy całkowicie, bo sko-
ro fermentacya nastąpi, kładzie się do beczki we-
wnętrzný kory z Sosniny funtów trzy, pączków Brzo-
zowych i kotków (pączków) Sosnowych po funkcie
jednym, trzy garści ziela *Cardi-Benedicti*, garść je-
dną lub dwie kwiatu zwanego *Sonthenaublüthe*, *Pim-
pinelli*, *Betoniki*, *Majeranu*, ziela zwanego *Poley* i dzi-
kiego *Tymianku*, każdego pół garści, albo garść ca-
łą, kwiatu *Bzowego* garści dwie, owocu róży polný
uncyi 30, utłuczonych. — Po nakładzeniu tych zapraw
do beczki, dolewa się ta do pełności, zaczęm skutkiem
fermentacyi Piwo naciągnie nieco zapachu i smaku,
z zapraw przydanych. — Przy ukończeniu fermentacyi
wybiją się do beczki io jay świeżo zniesionych i ta
szpuntuje się. We dwa roki dopiero po zaszpuntowa-
niu ściąga się Piwo do picia.

- składniki (dane wejściowe): woda, słód, itd.
- wynik: beczka piwa
- sprzęt: beczka, piwowar (mielcarz)
- przepis: oprogramowanie, algorytm
- instrukcje: dodawanie składników, gotowanie, odlewanie, dolewanie, fermentacja, szpuntowanie

Chambers udziela w swych pismach odmienny sposób robienia Piwa Brunświckiego, odpisany iak twierdzi, z pierwotnéj Recepty, zachowywanéj na Ratuszu mieyskim w Brunświku.

Według Recepty téj wziąć należy 200 miarek zwanych *Kanne* wody, i te tak długo gotować, pókiż 1/3 części iéy nie ubędzie. Sypie się potem do téj wody siedm Szeflów siodu Pszennego i jeden Szeffel drobnéj Fasoli; gdy odleie się do beczki dla wyrobienia, nie należy zapełniać iéy całkowicie, bo skoro fermentacya nastąpi, kładzie się do beczki wewnętrzny kory z Sośniny funtów trzy, pączków Brzozowych i kotków (pączków) Sosnowych po funcie jednym, trzy garści ziela *Cardi-Benedicti*, garść jedną lub dwie kwiatu zwanego *Sonnenhaublütthe*, *Pimpinelli*, *Betoniki*, *Majeranu*, ziela zwanego *Poley* i dzikiego *Tymianku*, każdego pół garści, albo garść całą, kwiatu *Bzowego* garści dwie, owocu róży polnéj uncyi 30, utłuczonych.— Po nakładzeniu tych zapraw do beczki, dolewa się ta do pełności, zaczém skutkiem fermentacyi Piwo naciągnie nieco zapachu i smaku, z zapraw przydanych.— Przy ukończeniu fermentacyi wybiją się do beczki io jay świeżo zniesionych i ta szpuntuje się. We dwa roki dopiero po zaszpuntowaniu ściąga się Piwo do picia.

ALGORYTM

jednoznacznie zdefiniowany ciąg operacji prowadzący w skończonej liczbie kroków do rozwiązania zadania.

ALGORYTM EUKLIDESA, OK. 300 P.N.E.

algorytm znajdowania największego wspólnego dzielnika (NWD) dwóch liczb całkowitych dodatnich. Uznawany za pierwszy kiedykolwiek wymyślony niebanalny algorytm.

Algorytmy to rozwiązania pewnych zadań - **zadań algorytmicznych**.

SPECYFIKACJA ZADANIA ALGORYTMICZNEGO

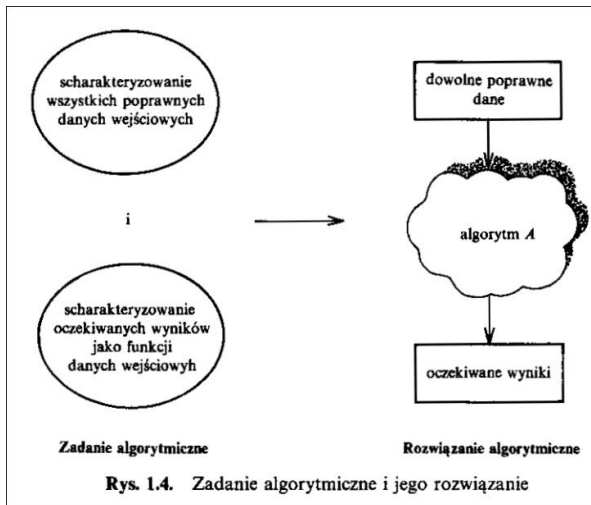
- warunki jakie muszą spełniać dane wejściowe
- określenie oczekiwanych wyników jako funkcji danych wejściowych

Dodatkowe ograniczenia algorytmu, np. dotyczące ilości operacji.

PRZYKŁAD SPECYFIKACJI: ALGORYTM EUKLIDESA

Dane wejściowe: liczby całkowite $x, y > 0$

Wynik: $NWD(x, y)$



CECHY ALGORYTMU

- **skończoność** - ograniczona liczba kroków
- **poprawność** - zgodny ze specyfikacją
- **uniwersalność** - poprawny dla klasy problemów
- **efektywność** - niska złożoność to gwarancja użyteczności
- **określoność** - zrozumiałe polecenia, możliwe do wykonania w jednoznacznej kolejności
- określony stan początkowy i wyróżniony koniec

Ciąg struktur sterujących definiuje kolejność wykonywanych operacji.

- bezpośrednie następstwo: *wykonaj A, potem B*
- wybór warunkowy (rozgałęzienie): *jeśli Q to wykonaj A, w przeciwnym razie wykonaj B*
- iteracja ograniczona: *wykonaj A dokładnie N razy*
- iteracja warunkowa: *dopóki Q, wykonuj A*
- instrukcja skoku: *skocz do G*
- podprogram - wyodrębniony fragment programu, funkcja

Struktury sterujące można dowolnie składać: np. pętle zagnieżdżone. „Nie ma granic złożoności algorytmów”.

- Opis językiem naturalnym
- Lista kroków
- Schematy blokowe
- Pseudo-języki
- Języki wysokiego poziomu

PROBLEM:

znalezienie największej liczby całkowitej dzielącej bez reszty liczby całkowite dodatnie a i b

POMYŚL:

Zauważmy, że

$$NWD(a, b) = k \implies a = nk, b = mk \implies a - b = (m - n)k$$

Stąd dla $a > b$ zachodzi

$$NWD(a, b) = NWD(a - b, b) = NWD(a - b, a)$$

.

ALGORYTM EUKLIDESA

Dane są dwie liczby całkowite. Odejmij od większej liczby mniejszą liczbę a większą liczbę zastąp uzyskaną różnicą. Powtarzaj tę czynność tak długo aż obie liczby będą równe. Otrzymana liczba jest największym wspólnym dzielnikiem liczb wejściowych.

SPECYFIKACJA

Dane wejściowe: liczby całkowite $a, b > 0$

Wynik: liczba całkowita a stanowiąca największy wspólny dzielnik

LISTA KROKÓW

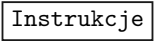
- 1 jeśli a jest równe b to jest to największy dzielnik
- 2 jeśli $a > b$ to zastąp a wartością $a - b$ i wróć do punktu 1
- 3 jeśli $a < b$ to zastąp b wartością $b - a$ i wróć do punktu 1

Algorytm zakłada istnienie operacji $-$, $=$ (porównanie) oraz $>$.




Stan

Blok graniczny: start, stop, przerwanie, opóźnienie.



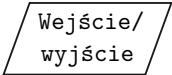
Instrukcje

Blok operacyjny: zmiana wartości, postaci lub miejsca zapisu danych.




Decyzja

Blok decyzyjny, rozgałęzienie.



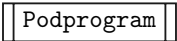
Wejście/
wyjście

Wprowadzanie danych i wyprowadzenia wyników.



Łącznik

Połączenie z innym fragmentem diagramu.

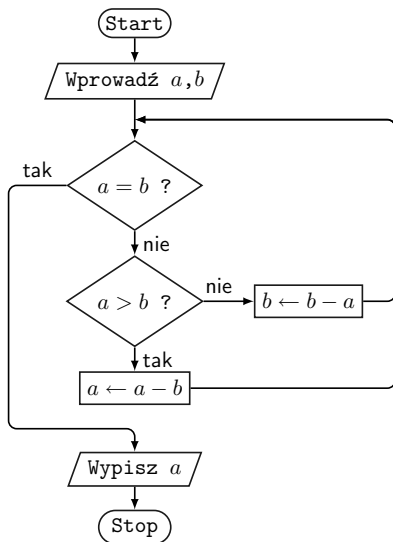


Podprogram

Wywołanie podprogramu.

PRZYKŁAD: SCHEMAT BLOKOWY

ALGORYTM EUKLIDESA Z ODEJMOWANIEM



Algorithm Algorytm Euklidesa

dopóki $a \neq b$ **wykonuj**

jeżeli $a > b$ **wykonaj**

$$a \leftarrow a - b$$

w przeciwnym wypadku

$$b \leftarrow b - a$$

- struktura kodu języka wysokiego poziomu (często Pascal)
- uproszczona składnia na rzecz prostoty i czytelności
- formuły matematyczne, język naturalny, podprogramy
- nie zawiera szczegółów implementacji
„Dla ludzi, nie dla maszyn”.

```
1  /* Algorytm Euklidesa. */
2
3  #include <stdio.h>
4
5  int main()
6  {
7      int a, b;
8
9      printf("Podaj dwie liczby calkowite: ");
10     scanf("%d %d", &a, &b);
11
12     while (a != b)
13         if (a > b) a = a - b;
14         else     b = b - a;
15
16     printf("NWD = %d\n", a);
17
18     return 0;
19 }
```

```
1 program NWD(input,output);
2 { Algorytm Euklidesa. }
3 var
4   A, B : Integer;
5 begin
6   Writeln('Podaj dwie liczby calkowite: ');
7   Readln(a,b);
8
9   while a <> b do
10    begin
11     if a > b then a := a - b
12     else b := b - a;
13    end;
14   Writeln('NWD = ', a);
15 end.
```

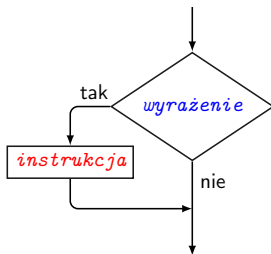


```
1      PROGRAM EUCLID1
2  c    Algorytm Euklidesa w jezyku Fortran 77
3
4      WRITE (*,*) 'Podaj dwie liczby calkowite: '
5      READ (*,*) N, M
6
7      DO WHILE ( N .NE. M )
8          IF ( N .GT. M ) THEN
9              N = N - M
10         ELSE
11             M = M - N
12         ENDIF
13     ENDDO
14     WRITE (*,*) 'NWD=', N
15     END
```

 euclid1.f

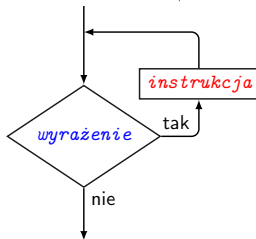
WARUNEK IF (JEŻELI)

```
if ( wyrażenie )  
    instrukcja
```



PĘTLA WHILE (DOPÓKI)

```
while ( wyrażenie )  
    instrukcja
```



- Czy algorytm jest poprawny? Dla jakich danych?
- Problem stopu. Czy algorytm posiada skończoną ilość kroków?
- Efektywność algorytmu. Ile iteracji należy się spodziewać dla różnych danych?

Algorithm Algorytm Euklidesa

1: **dopóki** $b \neq 0$ **wykonuj**

2: $c \leftarrow a \bmod b$

3: $a \leftarrow b$

4: $b \leftarrow c$

- wymaga operacji dzielenie modulo oraz \neq
- złożoność: dla $a > b \geq 0$ co najwyżej $2 \log_2(a + b)$ iteracji

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int a, b, c;
6
7     printf("Podaj dwie liczby calkowite: ");
8     scanf("%d %d", &a, &b);
9     printf("NWD(%d,%d) = ", a, b);
10
11     while (b != 0)
12     {
13         c = a % b;
14         a = b;
15         b = c;
16     }
17     printf("%d\n", a);
18
19     return 0;
20 }
```

- **Podprogram** - funkcja lub procedura, wydzielona część programu, która może być wielokrotnie użyta
- ekonomiczność - ujednocila powtarzające się bloki programu - mniej kodowania
- przejrzystość, nawet przy złożonych i obszernych algorytmach
- podprogram staje się nową instrukcją elementarną
w języku C brak wbudowanych funkcji, tylko `main()`
- uproszczenie problemu poprzez rozbitcie na mniejsze pod-problemy
 - programowanie zstępujące (*top-down*)
 - programowanie wstępujące (*bottom-up*)
- podprogram uruchamiający sam siebie - rekurencja

```
1 int nwd(int a, int b)
2 {
3     int c;
4     while (b != 0)
5     {
6         c = a % b;
7         a = b;
8         b = c;
9     }
10    return a;
11 }
```

 euclid3.c

```
1 int nwd(int a, int b)
2 {
3     if (b == 0) return a;
4     return nwd(b, a % b );
5 }
```

 euclid3rekurencja.c

Algorithm Sortowanie przez wybieranie (selection sort)

Dane wejściowe: ciąg n liczb $A = \{a_1, a_2, \dots, a_n\}$

Wynik: uporządkowany ciąg $a_1 \leq a_2 \leq \dots \leq a_n$

1: $i \leftarrow 1$

2: **dopóki** $i < n$ **wykonuj**

3: $k \leftarrow \text{minind}(\{a_i, a_{i+1}, \dots, a_n\})$

▷ Podprogram

4: $a_i \longleftrightarrow a_k$

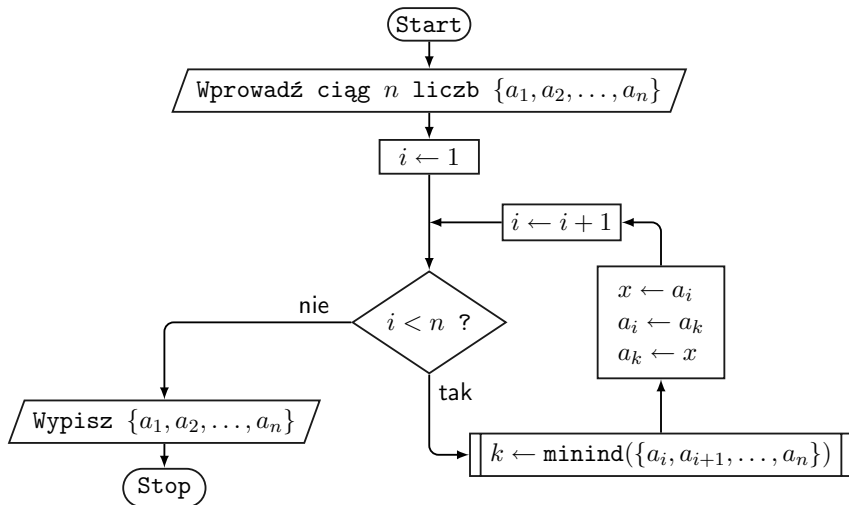
5: $i \leftarrow i + 1$

`minind()` wyszukuje indeks elementu o najmniejszej wartości.

Wizualizacja algorytmów sortowania:  AlgoRythmics

PRZYKŁAD: SCHEMAT BLOKOWY

SORTOWANIE PRZEZ WYBIERANIE



ZŁOŻONOŚĆ OBLICZENIOWA

liczba operacji wykonywanych przez algorytm. Zazwyczaj wyznaczana względem ilości danych lub ich rozmiaru.

Ile operacji wymaga ...

- porównanie dwóch liczb całkowitych?
- obliczenie $NWD(a, b)$?
- znalezienie pewnego elementu wśród N elementów?
- ile operacji wymaga posortowanie listy N elementów?
- Problem komiwojażera: znalezienie najkrótszej drogi łączącej wszystkie miasta?

PROBLEM TSP

TRAVELLING SALESMAN PROBLEM



Tabela 1.1. Czasy znalezienia przez komputer, wykonujący 100 miliardów operacji na sekundę, najkrótszej trasy podróży premiera (zob. ćwiczenie 1.3)

| Liczba województw | Czas znajdowania najkrótszej trasy |
|-------------------|------------------------------------|
| 17 | 3.5 minuty |
| 25 | $2 \cdot 10^5$ lat |
| 49 | $4 \cdot 10^{42}$ lat |

Rysunek 1.1. Najkrótsza trasa premiera przebiegająca przez wszystkie miasta województwie w podziale administracyjnym z 1975 roku (A. Adrabiński)

[1] M.M. Sysło, „Algorytmy”

PROBLEMY O „ROZSĄDNYCH” ROZWIĄZANIACH

PROBLEMY P I NP

| Funkcja \ N | 10 | 50 | 100 | 300 | 1000 |
|---------------------|---------------------------|----------------------|-----------------------|-------------------------|---------------------------|
| $5N$ | 50 | 250 | 500 | 1500 | 5000 |
| $N \times \log_2 N$ | 33 | 282 | 665 | 2469 | 9966 |
| N^2 | 100 | 2500 | 10 000 | 90 000 | 1 milion (7 cyfr) |
| N^3 | 1000 | 125 000 | 1 milion (7 cyfr) | 27 milionów (8 cyfr) | 1 miliard (10 cyfr) |
| 2^N | 1024 | liczba 16-cyfrowa | liczba 31-cyfrowa | liczba 91-cyfrowa | liczba 302-cyfrowa |
| $N!$ | 3,6 miliona (7 cyfr) | liczba 65-cyfrowa | liczba 161-cyfrowa | liczba 623-cyfrowa | niewyobrażal- nie duża |
| N^N | 10 miliardów (11 cyfr) | liczba 85-cyfrowa | liczba 201-cyfrowa | liczba 744-cyfrowa | niewyobrażal- nie duża |

Dla porównania: liczba protonów w znanym wszechświecie ma 126 cyfr,
liczba mikrosekund od „wielkiego wybuchu” ma 24 cyfry.

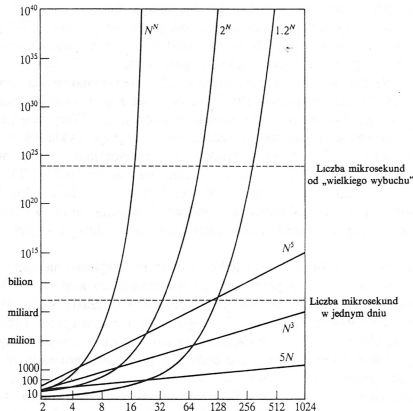
Rys. 7.3. Niektóre wartości pewnych funkcji

[2] D. Harel, *Rzecz o istocie informatyki. Algorytmika.*

PROBLEMY O „ROZSĄDNYCH” ROZWIĄZANIACH

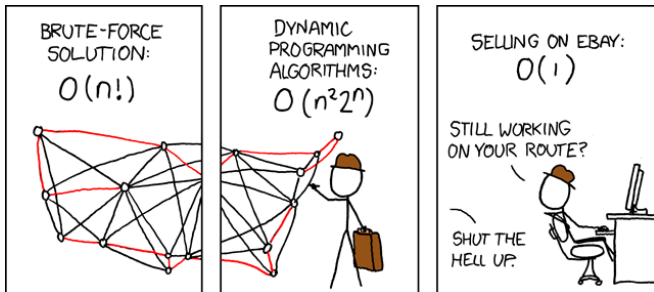
PROBLEMY P I NP

- Rozsądne rozwiązania, wykonywalne w czasie wielomianowym
 $\log N$, N , $N \log N$, $N^7 + N^3 + 2$
- Nerozsądne, niepraktyczne, czas ponad-wielomianowy
 2^N , $1.001^N + N^7$, N^N , $N!$



Rys. 7.4. Tempo wzrostu pewnych funkcji

[2] D. Harel, *Rzecz o istocie informatyki. Algorytmika.*








Reprezentacja danych kształtuje algorytm!

- zmienne: liczby, znaki, $a = 3$
- tablice, wektory, listy, $a[3]=3$
- tablice wielowymiarowe, tablice tablic, $a[3,3]=3$ $a[3][3]=3$
- rekordy, struktury, $a.b=3$ $a.c='a'$
- kolejki, stosy, drzewa
- pliki, bazy danych

Złożoność pamięciowa algorytmu - miara ilości wykorzystanej pamięci

OD PROBLEMU DO ROZWIĄZANIA

- analiza problemu
- specyfikacja zadania:
 - dopuszczalny zestaw danych wejściowych
 - pożądane wyniki jako funkcja danych wejściowych
- algorytm - rozwiązanie zadania
- poprawność algorytmu:
 - względem specyfikacji
 - problem stopu
 - efektywność (złożoność): „*Każda akcja zajmuje czas!*”
- implementacja algorytmu \Rightarrow program

-  Maciej M. Sysło, „*Algorytmy*”, WSiP, Warszawa, 2002.
-  D. Harel, *Rzecz o istocie informatyki. Algorytmika.*, WNT, Warszawa, 1992.
-  Fulmański Piotr, Sobieski Ścibór, „*Wstęp do informatyki*”, Uniwersytet Łódzki, 2004