Algorytmy i struktury danych I
wykład

Karol Szałowski

WYDZIAŁ FIZYKI
i INFORMATYKI STOSOWANEJ
Uniwersytet Łódzki

2019/2020

karol.szalowski@uni.lodz.pl

Zaliczenie przedmiotu

Ocena końcowa:

- Ocena z wykładu: **wkład 40%**
  - pisemny sprawdzian końcowy
  - **zaliczenie od 50%**
- Ocena z laboratorium: **wkład 60%**
  - obecność - 7 pkt. (14%)
  - aktywność - 7 pkt. (14%)
  - zadania podstawowe (projekty) oddawane w terminie - 18 pkt. (36%)
  - kartkówka - 18 pkt. (36%)
  - **zaliczenie od 25 pkt. (50%)**
- Obowiązkowa obecność na zajęciach.

Literatura

- T.H. Cormen, C.E. Leiserson, R.L. Rivest, **Wprowadzenie do algorytmów**
- A.V. Aho, J.E. Hopcroft, J.D. Ullman, **Projektowanie i analiza algorytmów komputerowych**
- L. Banachowski, K. Diks, W. Rytter, **Algorytmy i struktury danych**
- N. Wirth, **Algorytmy + struktury danych = programy**

- R. Sedgewick, **Algorytmy w C++**
- D. E. Knuth, **Sztuka programowania**
- D. Harel, **Rzecz o istocie informatyki**
- J. Bentley, **Perełki oprogramowania**
- C. Papadimitriou, **Złożoność obliczeniowa**
- P. Wróblewski, **Algorytmy, struktury danych i techniki programowania**
- W. Iszkowski, **Struktury i typy danych**

Algorytm

Dobrze zdefiniowana procedura obliczeniowa, pobierająca oraz zwracająca pewien zbiór wartości.

Dokładny przepis podający sposób rozwiązania określonego zadania w skończonej liczbie kroków; zbiór poleceń odnoszących się do pewnych obiektów, ze wskazaniem porządku, w jakim mają być realizowane.

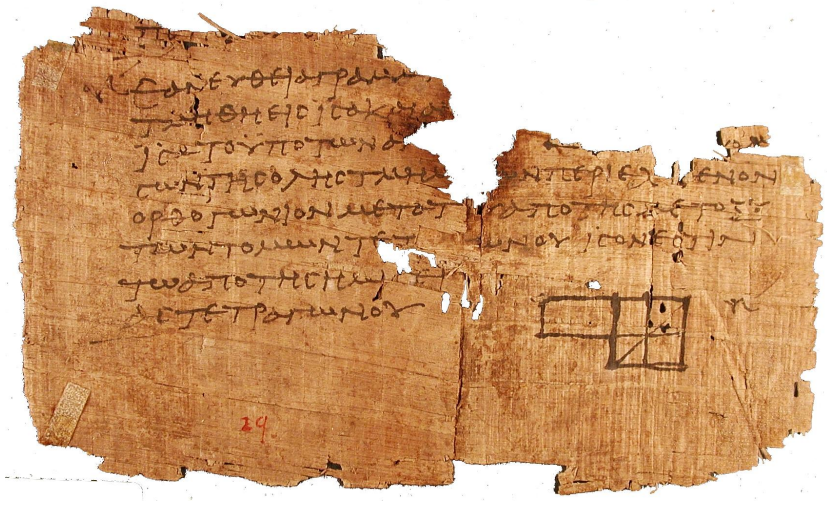
Algorytm

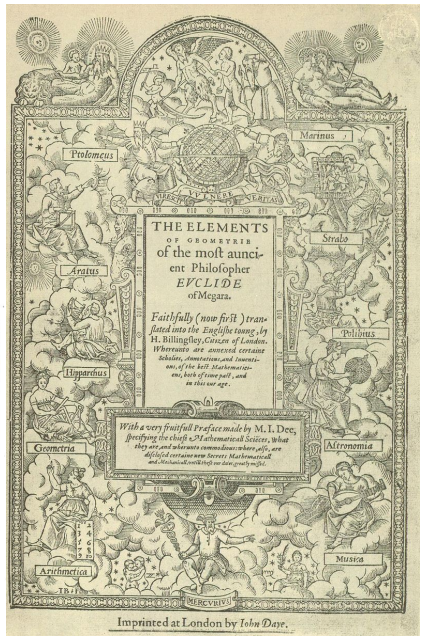
Cechy algorytmów:

- poprawność
  - algorytm powinien zwracać poprawne wyniki dla poprawnych danych wejściowych
- skończoność
  - dla dowolnych poprawnych danych wejściowych algorytm powinien zwracać wyniki wykonując skończoną liczbę kroków
- jednoznaczność
  - algorytm powinien zwracać takie same wyniki dla takich samych danych wejściowych
- efektywność
  - algorytm powinien zwracać wyniki wykonując jak najmniejszą liczbę kroków

Pierwszy algorytm

Algorytm Euklidesa (przełom IV i III w. p.n.e.) - algorytm wyznaczania największego wspólnego dzielnika dwóch liczb - opisany w *Elementach* Euklidesa.





Jak działa algorytm Euklidesa?

Algorytm Euklidesa, którego celem jest wyznaczenie największego wspólnego dzielnika dwóch liczb naturalnych (NWD), bazuje na pewnej prostej obserwacji.

• Jeżeli pewna liczba jest dzielnikiem dwóch liczb całkowitych a, $b < a$, to jest także dzielnikiem liczby $a - b$:
$a = m \cdot k \qquad b = n \cdot k \qquad \Rightarrow \qquad a - b = (m - n) \cdot k$

• Wspólne dzielniki a i b są zatem także wspólnymi dzielnikami b i $a - b$ lub a i $a - b$.

• Własność ta dotyczy też największego wspólnego dzielnika $NWD(a, b)$:
$NWD(a, b) = NWD(b, a - b)$ lub $NWD(a, b) = NWD(a, a - b)$

• Aby zatem znaleźć największy wspólny dzielnik należy odjąć od liczby większej mniejszą i powtórzyć proces dla liczby mniejszej z pary początkowej oraz różnicy liczb początkowych.

• Algorytm należy przerwać, gdy obie liczby są sobie równe.

Jak działa algorytm Euklidesa? (modyfikacja)

Algorytm Euklidesa, którego celem jest wyznaczenie największego wspólnego dzielnika dwóch liczb naturalnych (NWD), bazuje na pewnej prostej obserwacji.

• Jeżeli pewna liczba jest dzielnikiem dwóch liczb całkowitych a, $b < a$, to jest także dzielnikiem liczby $r = a - qb \geqslant 0$:
$a = m \cdot k \qquad b = n \cdot k \qquad \Rightarrow \qquad a - qb = (m - qn) \cdot k$

• Wspólne dzielniki a i b są zatem także wspólnymi dzielnikami b i r lub a i r.

• Własność ta dotyczy też największego wspólnego dzielnika $NWD(a, b)$: $NWD(a, b) = NWD(b, r)$.

• Wybierzmy $r = a \bmod b$ - reszta z dzielenia a przez b. Mamy $0 \leqslant r < b$. Zatem $NWD(a, b) = NWD(b, a \bmod b)$

• Aby zatem znaleźć największy wspólny dzielnik pary liczb, należy obliczyć resztę z dzielenia wiekszej liczby przez mniejszą. Następnie tworzyc parę z mniejszej liczby i reszty z dzielenia.

• Algorytm należy przerwać, gdy reszta z dzielenia jest równa 0.

Metody przedstawiania algorytmów

- Opis słowny
- Lista kroków
- Schemat blokowy
- Pseudokod
- Drzewa algorytmiczne

Lista kroków

Algorytm Euklidesa:

- Wczytaj dwie liczby
- Znajdź wynik odejmowania mniejszej od większej
- Zastąp liczbę większą obliczoną różnicą
- Kiedy obie liczby są równe zakończ

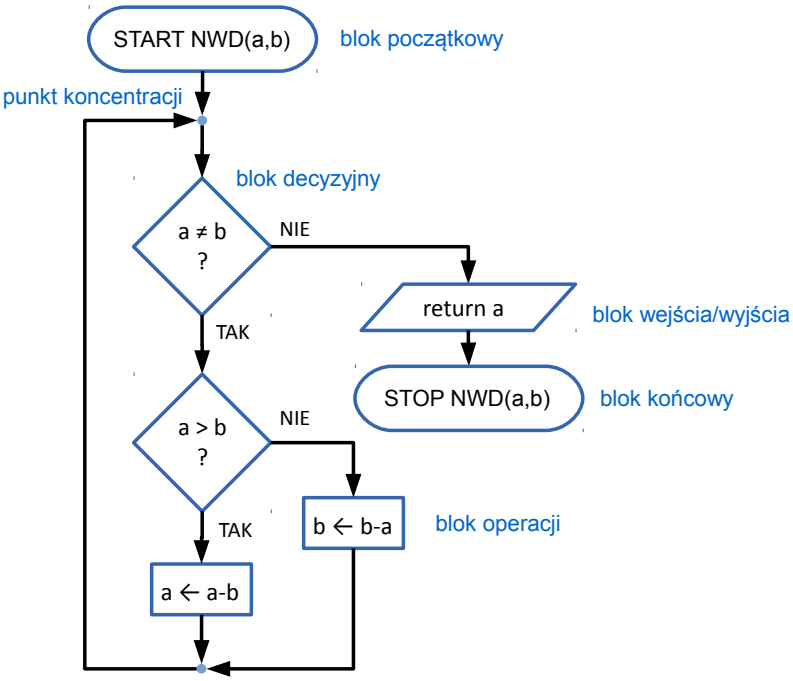
Pseudokod

Algorytm Euklidesa

Wejście: a, b
Wyjście: $NWD(a, b)$

```
1: while a ≠ b do
2:     if a < b then
3:         a ← a − b
4:     else
5:         b ← b − a
6:     end if
7: end while
8: return a
```

Schemat blokowy/pseudokod

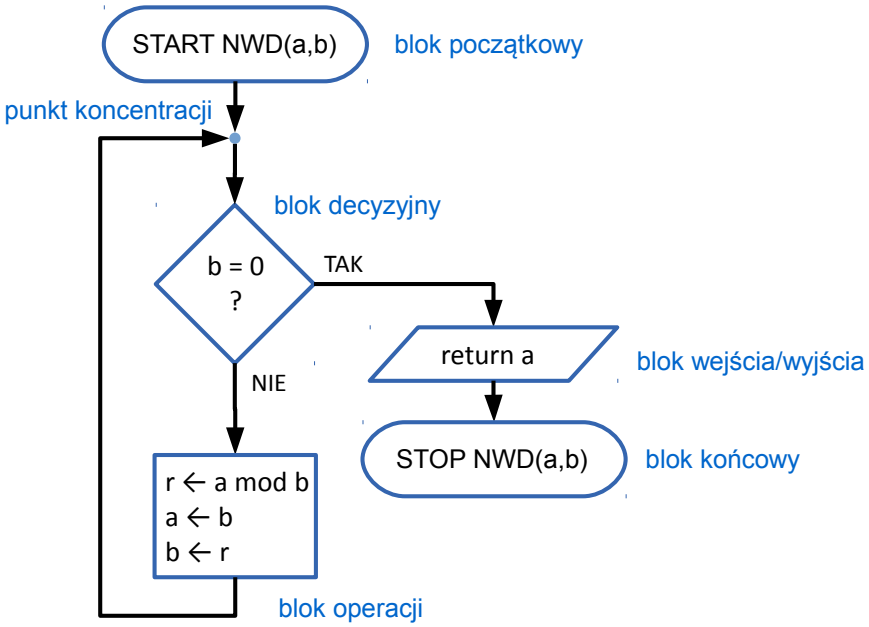


Algorytm Euklidesa

Wejście: a, b
Wyjście: $NWD(a, b)$

```
1: while a ≠ b do
2:     if a < b then
3:         a ← a − b
4:     else
5:         b ← b − a
6:     end if
7: end while
8: return a
```

Schemat blokowy/pseudokod



Algorytm Euklidesa

Wejście: a, b
Wyjście: $NWD(a, b)$

1: **while** $b > 0$ **do**
2: $\quad r \leftarrow a \bmod b$
3: $\quad a \leftarrow b$
4: $\quad b \leftarrow r$
5: **end while**
6: **return** a

Schemat Hornera

- Obliczanie wartości wielomianu

$$P(x) = \sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n$$
$$= a_0 + x\left(a_1 + x\left(a_2 + \cdots + x\left(a_{n-1} + x a_n\right)\right)\right)$$

- Przykład, $n = 3$:

$$P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 =$$
$$= a_0 + x\left(a_1 + x\left(a_2 + x\left(a_3\right)\right)\right)$$

$$y = a_3$$
$$y = a_2 + xy = a_2 + x a_3$$
$$y = a_1 + xy = a_1 + x\left(a_2 + x a_3\right) = a_1 + a_2 x + a_3 x^2$$
$$y = a_0 + xy = a_0 + x\left(a_1 + a_2 x + a_3 x^2\right) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Schemat Hornera

$$P(x) = \sum_{k=0}^{n} a_k x^k = a_0 + x\left(a_1 + x\left(a_2 + \cdots + x\left(a_{n-1} + x a_n\right)\right)\right)$$

Algorytm obliczania wartości wielomianu - schemat Hornera

Wejście: x
Wyjście: $P(x)$

```
1: y ← 0
2: for i = n to 0 do
3:     y ← a_i + x · y
4: end for
5: return y
```

Obliczanie wartości wielomianu

$$P(x) = \sum_{k=0}^{n} a_k x^k$$

Prosty algorytm obliczania wartości wielomianu

Wejście: x
Wyjście: $P(x)$

```
1: y ← 0
2: for i = 0 to n do
3:     y_i ← 1
4:     for j = 1 to i do
5:         y_i ← y_i · x
6:     end for
7:     y ← y + a_i · y_i
8: end for
9: return y
```

Niezmiennik pętli

- Niezmiennik pętli:

  relacja między wartościami zmiennych, która nie ulega zmianie w czasie wykonywania instrukcji zawartych w pętli

Własności do wykazania:

- Czy jest prawdziwy przed pierwszą iteracją?
- Jeśli jest prawdziwy przed daną iteracją, czy jest prawdziwy przed kolejną iteracją?
- Czy wartość niezmiennika pętli po zakończeniu pętli jest przydatna do udowodnienia poprawności działania algorytmu?

Schemat Hornera

Niezmiennik pętli:

- Czy jest prawdziwy przed pierwszą iteracją?
- Jeśli jest prawdziwy przed daną iteracją, czy jest prawdziwy przed kolejną iteracją?
- Czy jest przydatny do udowodnienia poprawności działania algorytmu?

Algorytm obliczania wartości wielomianu - schemat Hornera

```
1: y ← 0
2: for i = n to 0 do
3:     y ← a_i + x · y
4: end for
5: return y
```

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k \quad \text{lub} \quad y = 0 \quad \text{dla} \quad i = n$$

Uwaga: To nie sama wartość y jest niezmiennikiem, $y \neq const$.! Niezmiennikiem jest relacja.

Schemat Hornera

- Niezmiennik pętli:

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k \quad \text{lub} \quad y = 0 \quad \text{dla} \quad i = n$$

- Czy jest on prawdziwy przed pierwszą iteracją?

$$y = 0 \quad \text{dla} \quad i = n$$

Algorytm obliczania wartości wielomianu - schemat Hornera

1: $y \leftarrow 0$
2: **for** $i = n$ to 0 **do**
3: $\quad y \leftarrow a_i + x \cdot y$
4: **end for**
5: **return** y

Schemat Hornera

- Niezmiennik pętli:

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k \quad \text{lub} \quad y = 0 \quad \text{dla} \quad i = n$$

- Jeśli jest prawdziwy przed daną iteracją, czy jest prawdziwy przed kolejną iteracją?

Załóżmy, że jest prawdziwy przed j-ą iteracją (dla tej iteracji licznik pętli ma wartość $i_j = i$):

$$y_j = \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k = \sum_{k=0}^{n-(i_j+1)} a_{k+i_j+1}x^k$$

Jaka będzie wartość przed $j+1$-ą iteracją (dla tej iteracji licznik pętli ma wartość $i_{j+1} = i - 1$):

$$y_{j+1} = a_i + xy_j = a_i + x\sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k = a_i + \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^{k+1} = \sum_{k=0}^{n-i} a_{k+i}x^k =$$

$$= \sum_{k=0}^{n-(i_{j+1}+1)} a_{k+i_{j+1}+1}x^k$$

Schemat Hornera

- Niezmiennik pętli:

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k \quad \text{lub} \quad y = 0 \quad \text{dla} \quad i = n$$

- Czy jest przydatny do udowodnienia poprawności działania algorytmu?

Wartość po zakończeniu ostatniej iteracji, dla $i = -1$:

$$y = \sum_{k=0}^{n} a_k x^k = P(x)$$

Algorytm obliczania wartości wielomianu - schemat Hornera

```
1: y ← 0
2: for i = n to 0 do
3:     y ← a_i + x · y
4: end for
5: return y
```

Algorytm Euklidesa

Algorytm Euklidesa

Wejście: a, b
Wyjście: $NWD(a, b)$

```
1: while b > 0 do
2:     r ← a mod b
3:     a ← b
4:     b ← r
5: end while
6: return a
```

• propozycja niezmiennika:

$$NWD(a, b) = NWD(A, B)$$

A, B - wartości początkowe zmiennych a, b

Złożoność obliczeniowa algorytmu

Parametry opisujące działanie algorytmu zależą od wielkości zbioru danych wejściowych, na którym ten algorytm operuje.

- **Złożoność pamięciowa**
- **Złożoność czasowa**

Złożoność czasowa

- $T(n)$: całkowity czas wykonania algorytmu dla danych o rozmiarze n
- k numeruje różne typy operacji
- t_k: czas potrzebny na jednokrotne wykonanie operacji typu k
- n_k: liczba wykonanych operacji typu k

$$T(n) = \sum_{\mathrm{k}} t_k \cdot n_k$$

Obliczanie wartości wielomianu

Algorytm obliczania wartości wielomianu - schemat Hornera

Wejście: x	czas	liczba operacji
Wyjście: $P(x)$		
1: $y \leftarrow 0$	t_1	1
2: **for** $i = 0$ to n **do**		
3: $\quad y \leftarrow a_i + x \cdot y$	$t_1 + t_2 + t_3$	$n + 1$
4: **end for**	t_4	$n + 1$ ($n + 1$ powtórzeń pętli)
5: **return** y		

- Operacje:
- przypisanie - czas t_1
- mnożenie - czas t_2
- dodawanie - czas t_3
- inkrementacja i sprawdzenie warunku pętli (porównanie) - czas t_4
- $T(n) = (n+2)\,t_1 + (n+1)\,t_2 + (n+1)\,t_3 + (n+1)t_4$
- $T(n) = (t_1 + t_2 + t_3 + t_4)\,n + (2t_1 + t_2 + t_3 + t_4)$

Obliczanie wartości wielomianu

Algorytm obliczania wartości wielomianu - schemat Hornera

Wejście: x
Wyjście: $P(x)$

```
1: y ← 0
2: for i = n to 0 do
3:     y ← a_i + x · y
4: end for
5: return y
```

- Operacje: mnożenia i dodawania
- Jedno mnożenie i jedno dodawanie w każdym przebiegu pętli
- $n + 1$ przebiegów pętli
- Liczba mnożeń: $n + 1$
- Liczba dodawań: $n + 1$

Obliczanie wartości wielomianu

Prosty algorytm obliczania wartości wielomianu

Wejście: x	czas	liczba operacji
Wyjście: $P(x)$		
1: $y \leftarrow 0$	t_1	1
2: **for** $i = 0$ to n **do**		
3: $\quad y_i \leftarrow 1$	t_1	$n+1$
4: $\quad$ **for** $j = 1$ to i **do**		
5: $\quad\quad y_i \leftarrow y_i \cdot x$	$t_1 + t_2$	$\sum_{i=0}^{n} i = \frac{1}{2}n(n+1)$
6: $\quad$ **end for**	t_4	$\sum_{i=0}^{n} i = \frac{1}{2}n(n+1)$ (i powtórzeń pętli wewnętrznej)
7: $\quad y \leftarrow y + a_i \cdot y_i$	$t_1 + t_2 + t_3$	$n+1$
8: **end for**	t_4	$n+1$ ($n+1$ powtórzeń pętli zewnętrznej)
9: **return** y		

- Operacje:
- przypisanie - czas t_1
- mnożenie - czas t_2
- dodawanie - czas t_3
- inkrementacja i sprawdzenie warunku pętli (porównanie) - czas t_4

- $T(n) = \left(\frac{1}{2}n^2 + \frac{5}{2}n + 3\right)t_1 + \left(\frac{1}{2}n^2 + \frac{3}{2}n + 1\right)t_2 + (n+1)t_3 + \left(\frac{1}{2}n^2 + \frac{3}{2}n + 1\right)t_4$
- $T(n) = \left(\frac{1}{2}t_1 + \frac{1}{2}t_2 + \frac{1}{2}t_4\right)n^2 + \left(\frac{5}{2}t_1 + \frac{3}{2}t_2 + t_3 + \frac{3}{2}t_4\right)n + (3t_1 + t_2 + t_3 + t_4)$

Obliczanie wartości wielomianu

Prosty algorytm obliczania wartości wielomianu

Wejście: x
Wyjście: $P(x)$

```
1: y ← 0
2: for i = 0 to n do
3:     y_i ← 1
4:     for j = 1 to i do
5:         y_i ← y_i · x
6:     end for
7:     y ← y + a_i · y_i
8: end for
9: return y
```

- Operacje: mnożenia i dodawania
- Jedno mnożenie w każdym przebiegu wewnętrznej pętli.
- Jedno mnożenie i jedno dodawanie w każdym przebiegu zewnętrznej pętli.
- $n+1$ przebiegów pętli zewnętrznej
- i przebiegów pętli wewnętrznej dla danego $i \Rightarrow i$ mnożeń
- Liczba mnożeń: $\sum_{i=0}^{n}(i+1) = \sum_{k=1}^{n+1} k = \frac{(n+1)(n+2)}{2} = \frac{1}{2}n^2 + \frac{3}{2}n + 1$
- Liczba dodawań: $n+1$

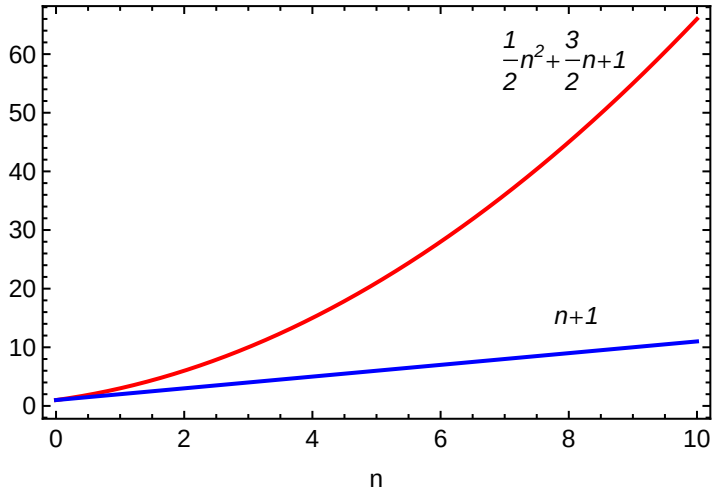
Obliczanie wartości wielomianu

Algorytm obliczania wartości wielomianu - schemat Hornera

Wejście: x
Wyjście: $P(x)$
1: $y \leftarrow 0$
2: **for** $i = n$ to 0 **do**
3: $\quad y \leftarrow a_i + x \cdot y$
4: **end for**
5: **return** y

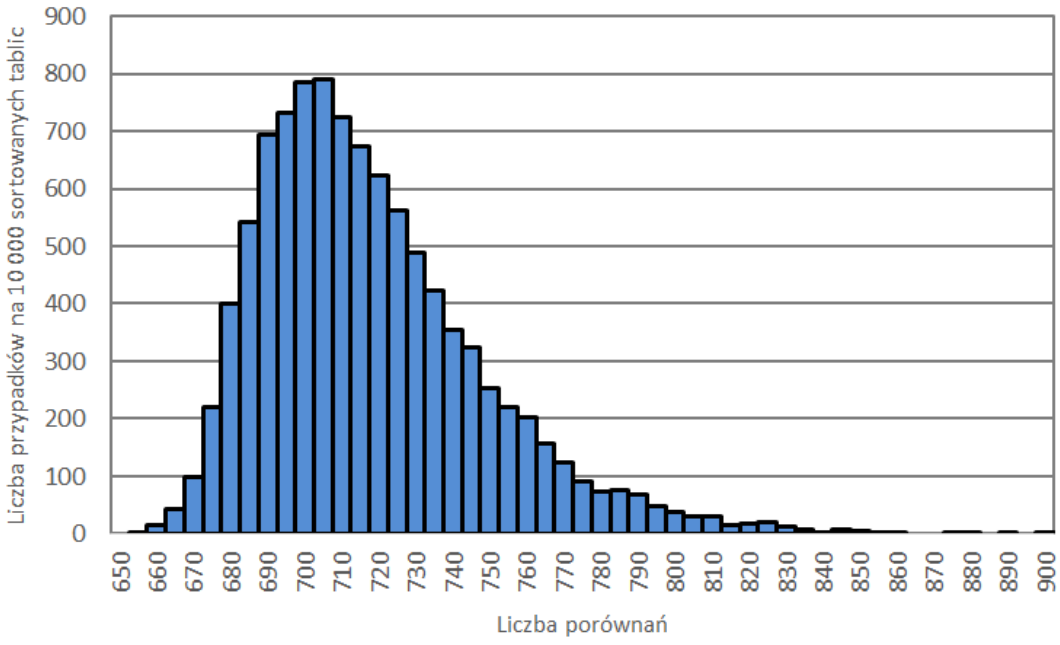
Prosty algorytm obliczania wartości wielomianu

Wejście: x
Wyjście: $P(x)$
1: $y \leftarrow 0$
2: **for** $i = 0$ to n **do**
3: $\quad y_i \leftarrow 1$
4: $\quad$ **for** $j = 1$ to i **do**
5: $\quad\quad y_i \leftarrow y_i \cdot x$
6: $\quad$ **end for**
7: $\quad y \leftarrow y + a_i \cdot y_i$
8: **end for**
9: **return** y



Złożoność czasowa

- oczekiwana (średnia) złożoność czasowa: $\langle T_n \rangle$
- pesymistyczna złożoność czasowa: $\max(T_n)$
- oczekiwana (średnia) wrażliwość czasowa: $\sqrt{(\langle T_n^2 \rangle - \langle T_n \rangle^2)}$ (odchylenie standardowe)
- pesymistyczna wrażliwość czasowa: $\max(T_n - T_m)$



Złożoność czasowa

- oczekiwana (średnia) złożoność czasowa: $\langle T_n \rangle$
- pesymistyczna złożoność czasowa: $\max(T_n)$
- oczekiwana (średnia) wrażliwość czasowa: $\sqrt{(\langle T_n^2 \rangle - \langle T_n \rangle^2)}$ (odchylenie standardowe)
- pesymistyczna wrażliwość czasowa: $\max(T_n - T_m)$