PIERWSZA ŁAWKA · KURS INFORMATYKA · INFORMATYKA · MATURA PR · ROZSZERZONY

# Algorytmy na liczbach

Algorytm Euklidesa (iteracyjnie i rekurencyjnie), NWW i działania na ułamkach, badanie pierwszości, sito Eratostenesa, rozkład na czynniki pierwsze, szybkie potęgowanie, schemat Hornera, pierwiastek kwadratowy metodą Herona, ciąg Fibonacciego iteracyjnie, podejście zachłanne, szyfrowanie z kluczem publicznym i efektywność algorytmów. Sześć kart pracy zgodnych z podstawą 2024 i maturą 2027 – z zadaniem przy komputerze na plikach danych (archiwum ZIP), kluczem, zasadami oceniania i numerami wymagań.

| Karta | Temat | Zadania |
|---|---|---|
| 1 | **Algorytmy na liczbach – Teoria krok po kroku**<br>Wszystko, co trzeba wiedzieć, z przykładami | 2 zad.<br>6 pkt |
| 2 | **Algorytmy na liczbach – Rozgrzewka**<br>Proste zastosowania – sprawdź, czy rozumiesz | 3 zad.<br>6 pkt |
| 3 | **Algorytmy na liczbach – Zadania maturalne, część 1**<br>Analiza algorytmów, tabele i zapis algorytmu – jak w arkuszu | 5 zad.<br>15 pkt |
| 4 | **Algorytmy na liczbach – Zadania maturalne, część 2**<br>Trudniejsze zadania i uzasadnienia | 5 zad.<br>17 pkt |
| 5 | **Algorytmy na liczbach – Zadanie z danymi – jak na egzaminie**<br>Praca z plikami danych przy komputerze – wyniki w pliku wyniki1.txt | 1 zad.<br>9 pkt |
| 6 | **Algorytmy na liczbach – Sprawdź się**<br>30 minut · 15 pkt | 5 zad.<br>15 pkt |

## JAK KORZYSTAĆ Z ZESTAWU

- Karta 1 – teoria z przykładami. Karty 2–4 – zadania w formie maturalnej: analiza algorytmów, tabele, zapis algorytmu ze specyfikacją (Dane, Wynik) i ograniczeniami z uwagi pod treścią.
- Karta 5 – zadanie przy komputerze: w archiwum ZIP pełne dane i pliki *_przyklad.txt z odpowiedziami kontrolnymi; wyniki w pliku wyniki1.txt z numerami zadań, kod w osobnych plikach. Karta 6 – zestaw kontrolny na czas.
- Ramki „Uwaga na maturze” – błędy, przez które zdający najczęściej tracą punkty (sprawozdania CKE 2023–2026).
- Klucz: rozwiązania wzorcowe (Python; C++ i Java są równoprawne), zasady oceniania i wymagania podstawy (Dz.U. 2024 poz. 1019): I.n, II.n, I+II.n – zakres rozszerzony, P.… – podstawowy. Wyniki dla plików danych policzono programem.

Materiał oryginalny zgodny z podstawą programową informatyki w brzmieniu z 2024 r. (Dz.U. 2024 poz. 1019) i z formułą egzaminu maturalnego z informatyki w 2027 r. Nie zawiera zadań z arkuszy CKE. Do użytku własnego i pracy z własnymi uczniami.

# Algorytmy na liczbach – Teoria krok po kroku

Wszystko, co trzeba wiedzieć, z przykładami

KARTA 1/6

Imię i nazwisko: ..... Data: ..... Wynik: ..... / 6 pkt

## PRZYPOMNIJ SOBIE

- **Algorytm Euklidesa**: NWD(a, b) = NWD(b, a mod b), NWD(a, 0) = a. Iteracyjnie: dopóki $b \neq 0$: $(a, b) \leftarrow (b, a \bmod b)$. Rekurencyjnie: nwd(a, b) = a, gdy b = 0, w przeciwnym razie nwd(b, a mod b). Wersja z odejmowaniem jest poprawna, ale dla (1000000, 1) wykonuje prawie milion kroków – wersja z mod tylko O(log min(a, b)).
- **NWW i ułamki**: NWW(a, b) = a div NWD(a, b) · b (dzielimy *przed* mnożeniem – mniejsze liczby pośrednie). Ułamek skracamy, dzieląc licznik i mianownik przez ich NWD; ułamki dodajemy, sprowadzając je do mianownika NWW.
- **Badanie pierwszości**: liczba $n \geq 2$ jest pierwsza, gdy nie ma dzielnika d z przedziału $2 \leq d \leq \sqrt{n}$. Warunek zapisujemy jako $d \cdot d \leq n$ – bez pierwiastka zmiennoprzecinkowego i bez błędów zaokrągleń. Złożoność $O(\sqrt{n})$.
- **Sito Eratostenesa** znajduje wszystkie liczby pierwsze ≤ n: dla kolejnych niewykreślonych p ($p \cdot p \leq n$) wykreślamy $p^2$, $p^2 + p$, $p^2 + 2p$, … Liczby mniejsze od $p^2$ o dzielniku p mają mniejszy czynnik pierwszy i są już wykreślone.
- **Rozkład na czynniki pierwsze**: dzielimy n przez d = 2, 3, 4, … dopóki $d \cdot d \leq n$ (każdy dzielnik tyle razy, ile się da); jeśli na końcu $n > 1$, to n jest ostatnim czynnikiem pierwszym. Dzielniki złożone nigdy nie „zadziałają”.
- **Szybkie potęgowanie**: $a^n = (a^{n \operatorname{div} 2})^2$ dla n parzystego i $a \cdot a^{n-1}$ dla nieparzystego. Wersja iteracyjna przegląda bity wykładnika: około $2 \log_2 n$ mnożeń zamiast $n - 1$. Z resztą modulo m po każdym mnożeniu liczby pośrednie pozostają małe.
- **Schemat Hornera**: $W(x) = (\ldots((a_n x + a_{n-1})x + a_{n-2})x + \ldots)x + a_0$ – n mnożeń i n dodawań. Ten sam schemat zamienia zapis w systemie p na liczbę (temat 1).
- **Pierwiastek kwadratowy (metoda Herona – Newtona)**: $x_{k+1} = (x_k + a / x_k) / 2$, np. od $x_0 = a$. Kończymy, gdy $|x_k^2 - a| < \varepsilon$. Wynik jest przybliżeniem (błąd przybliżenia metody + błąd zaokrągleń).
- **Ciąg Fibonacciego iteracyjnie**: pamiętamy tylko dwa ostatnie wyrazy: $(a, b) \leftarrow (b, a + b)$ – n kroków, stała pamięć.
- **Podejście zachłanne**: w każdym kroku wybieramy to, co w tej chwili wygląda najkorzystniej, i nie cofamy decyzji. Algorytm jest szybki, ale nie zawsze daje rozwiązanie optymalne – trzeba to uzasadnić albo obalić kontrprzykładem.
- **Szyfrowanie z kluczem publicznym (RSA)**: klucz publiczny (e, n), prywatny (d, n), $n = p \cdot q$ (p, q – duże liczby pierwsze). Szyfrowanie $c = m^e \bmod n$, deszyfrowanie $m = c^d \bmod n$ – obie operacje wykonuje szybkie potęgowanie modulo. Bezpieczeństwo opiera się na trudności rozkładu n na czynniki. **Podpis elektroniczny**: nadawca szyfruje skrót dokumentu kluczem prywatnym, każdy może go sprawdzić kluczem publicznym.

| Algorytm | Liczba kroków |
|---|---|
| Euklides z mod | O(log min(a, b)) |
| pierwszość do √n | O(√n) |
| sito Eratostenesa do n | O(n log log n) |
| szybkie potęgowanie | O(log n) |
| schemat Hornera, stopień n | n mnożeń |

- **Jak zapisać algorytm, żeby dostać punkty**: pseudokod albo wybrany język (Python, C++, Java) – oba zapisy są równoprawne. Trzymaj się nazw ze *Specyfikacji* (Dane/Wynik), na końcu jawnie ustal wynik (zwróć go albo wypisz), przestrzegaj ograniczeń z *Uwagi* pod treścią i sprawdź algorytm „na sucho" na przykładzie z treści oraz na przypadku brzegowym (np. n = 1, liczba 0).

## PRZYKŁADY Z WYJAŚNIENIEM

**Przykład 1. Oblicz NWD(84, 36) algorytmem Euklidesa.**
84 mod 36 = 12 → (36, 12); 36 mod 12 = 0 → (12, 0). NWD = 12, NWW = 84 div 12 · 36 = 252.

**Przykład 2. Dlaczego dzielniki sprawdzamy tylko do √n?**
Gdyby n = a · b i oba czynniki były większe od √n, to a · b > n – sprzeczność. Jeśli więc n ma dzielnik, to ma też dzielnik ≤ √n.

**Przykład 3. Dodaj ułamki 5/12 + 7/18 i skróć wynik.**
NWW(12, 18) = 36: 15/36 + 14/36 = 29/36; NWD(29, 36) = 1, więc ułamek jest nieskracalny.

**Przykład 4. Ile mnożeń wykona szybkie potęgowanie dla $a^{16}$?**
$a^2$, $a^4$, $a^8$, $a^{16}$ – cztery podnoszenia do kwadratu zamiast piętnastu mnożeń. Ogólnie około $\log_2 n$ kwadratów i najwyżej tyle samo dodatkowych mnożeń.

## ZADANIA

**Zadanie 1. Oblicz NWD(126, 84) i NWW(126, 84), zapisując kolejne kroki algorytmu Euklidesa.**

___ / 3 pkt

**Zadanie 2. Uzupełnij rozkłady na czynniki pierwsze.**

___ / 3 pkt

1. 84 = _____
2. 225 = _____
3. 1000 = _____

# Algorytmy na liczbach – Zadania maturalne, część 1

Analiza algorytmów, tabele i zapis algorytmu – jak w arkuszu

KARTA 3/6

Imię i nazwisko: ..... Data: ..... Wynik: ..... / 15 pkt

**Zadanie 1. Przeanalizuj algorytm i uzupełnij, co wypisze dla podanych danych (wynik a oraz liczbę wykonanych odejmowań k).** ___ / 3 pkt

```
a, b ← dane liczby całkowite dodatnie
k ← 0
dopóki a ≠ b wykonuj
    jeżeli a > b
        a ← a - b
    w przeciwnym razie
        b ← b - a
    k ← k + 1
wypisz a, k
```

1. a = 12, b = 18: wypisze a = _____, k = _____
2. a = 35, b = 14: wypisze a = _____, k = _____
3. a = 100, b = 1: wypisze a = _____, k = _____
4. a = 21, b = 21: wypisze a = _____, k = _____

**Zadanie 2. Dla jakich danych powyższy algorytm działa bardzo długo? Zaproponuj zmianę, która przyspieszy go bez zmiany wyniku, i podaj jej złożoność.** ___ / 2 pkt

Podgląd • Pierwsza Ławka

# Algorytmy na liczbach – Zadanie z danymi – jak na egzaminie

Praca z plikami danych przy komputerze – wyniki w pliku wyniki1.txt

KARTA 5/6

Imię i nazwisko: ..... Data: ..... Wynik: ..... / 9 pkt

**Zadanie 1. W pliku liczby.txt zapisano 1000 liczb całkowitych z zakresu od 2 do 1 000 000, każdą w osobnym wierszu. Napisz program (lub kilka programów), który da odpowiedzi do poniższych zadań. Odpowiedzi zapisz w pliku wyniki1.txt, poprzedzając każdą numerem zadania.** ___ / 9 pkt

**Pierwsze wiersze pliku liczby.txt**

```
558778
505234
172465
116403
174910
…
```

Do dyspozycji masz także **liczby_przyklad.txt** (30 liczb) – odpowiedzi dla nich podano w treściach zadań. Twój program musi ostatecznie działać na pełnych danych (**liczby.txt**).

**Do oceny oddajesz:** plik **wyniki1.txt** z odpowiedziami do zadań 1.1–1.3 (każda poprzedzona numerem zadania) oraz pliki z kodem źródłowym, np. **zad1_1.py**, **zad1_2.py**… (lub .cpp, .java). Brak plików z kodem oznacza brak rozwiązania.

Dane przygotowane na potrzeby kursu (archiwum ZIP dołączone do opracowania, katalog DANE)

**UWAGA NA MATURZE** Każdą odpowiedź zapisz w wyniki1.txt z numerem zadania, a kod – w pliku o nazwie wskazującej zadanie (zalecenie ze sprawozdania CKE 2026).

**1.1. Podaj, ile liczb z pliku jest pierwszych. Odpowiedź dla danych przykładowych: 5** ___ / 2 pkt

**UWAGA NA MATURZE** Dzielenie aż do n − 1 dla tysiąca liczb rzędu miliona to nawet miliard dzieleń. Pętla while d * d <= n wykona najwyżej tysiąc dzieleń na liczbę i nie wymaga liczb rzeczywistych (brak błędów zaokrągleń pierwiastka).

**1.2. W pliku jest dokładnie jedna liczba, która ma najwięcej różnych czynników pierwszych. Podaj tę liczbę oraz jej rozkład na czynniki pierwsze w postaci iloczynu (np. 360 = 2*2*2*3*3*5). Odpowiedź dla danych przykładowych: 2310 = 2*3*5*7*11** ___ / 3 pkt

**UWAGA NA MATURZE** „Różne czynniki” – 360 ma trzy różne czynniki (2, 3, 5), choć w rozkładzie jest ich sześć.

Podgląd • Pierwsza Ławka

# Algorytmy na liczbach – Sprawdź się

30 minut · 15 pkt

KARTA 6/6

Imię i nazwisko: ..... Data: ..... Wynik: ..... / 15 pkt

**Zadanie 1. Oblicz NWD(1071, 462) algorytmem Euklidesa z resztą z dzielenia i podaj liczbę wykonanych dzieleń.** ___ / 3 pkt

**Zadanie 2. Zapisz algorytm wypisujący wszystkie dzielniki liczby n, wykonujący około √n obiegów pętli.** ___ / 3 pkt

**Uwaga:** Twój algorytm może używać tylko zmiennych przechowujących pojedyncze liczby całkowite, operacji arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie całkowite div i reszta z dzielenia mod), porównań, operatorów logicznych, instrukcji warunkowych, pętli i przypisań oraz samodzielnie napisanych funkcji. Nie wolno używać funkcji wbudowanych, tablic ani list, ani zamieniać liczby na napis (i odwrotnie).

**Specyfikacja**
**Dane:** n – liczba całkowita dodatnia
**Wynik:** wszystkie dzielniki liczby n (w dowolnej kolejności, bez powtórzeń)

**Zadanie 3. Oceń prawdziwość zdań.** ___ / 4 pkt

| | | |
|---|---|---|
| 1. NWD(a, 0) = a dla każdej liczby całkowitej dodatniej a. | **P** | **F** |
| 2. Liczba 1 jest liczbą pierwszą. | **P** | **F** |
| 3. Liczba mnożeń w szybkim potęgowaniu rośnie jak log n. | **P** | **F** |
| 4. Wybierając do jednej sali zawsze zajęcia, które zaczynają się najwcześniej (i nie kolidują z wybranymi), otrzymamy największą możliwą liczbę zajęć. | **P** | **F** |

Podgląd • Pierwsza Ławka

CO JESZCZE JEST W ŚRODKU

# Algorytmy na liczbach – opracowanie maturalne z informatyki (6 kart pracy z kluczem)

Podgląd pokazał 6 z 19 stron. Pełna wersja: 19 stron A4, PDF do druku i DOCX do edycji.

- Karta 1: Teoria krok po kroku – 2 zadania, 6 pkt
- Karta 2: Rozgrzewka – 3 zadania, 6 pkt
- Karta 3: Zadania maturalne, część 1 – 5 zadań, 15 pkt
- Karta 4: Zadania maturalne, część 2 – 5 zadań, 17 pkt
- Karta 5: Zadanie z danymi – jak na egzaminie – 1 zadanie, 9 pkt
- Karta 6: Sprawdź się – 5 zadań, 15 pkt Razem 19 stron A4 w PDF do druku oraz ten sam materiał w DOCX do edycji w Wordzie, a do tego archiwum ZIP z plikami danych do zadania z karty

Kup pełną wersję – plik od razu na e-mail
pierwszalawka.pl/material/matura-informatyka-algorytmy-na-liczbach-opracowanie