Rekurencja, algorytm Euklidesa, szukanie liczb pierwszych

Zadanie 1

Zaimplementuj rekurencyjną metodę `fibonacci` wyliczającą wartość danego elementu ciągu Fibonnacciego.

Algorytm Euklidesa

Algorytm Euklidesa służy do wyznaczania największego wspólnego dzielnika dwóch liczb. Najprostsza wersja algorytmu rozpoczyna się od wybrania dwóch liczb naturalnych, dla których należy wyznaczyć największy wspólny dzielnik. Następnie z tych dwóch liczb tworzymy nową parę: pierwszą z liczb jest liczba mniejsza, natomiast drugą jest różnica liczby większej i mniejszej. Proces ten jest powtarzany aż obie liczby będą sobie równe – wartość tych liczb to największy wspólny dzielnik wszystkich par liczb wcześniej wyznaczonych (cyt. wiki).

Zadanie 2

Zaimplementuj rekurencyjną metodę obliczającą NWD dla zadanych liczb.

Szukanie liczb pierwszych

Zadanie 3

Napisz funkcję `is_prime_3` testującą czy dana liczba jest pierwszą. Funkcją ma działać naiwnie i sprawdzać czy którakolwiek z liczb od 2 do N-1 dzieli zadaną liczbę.

Zadanie 4

Napisz funkcję `is_prime_4` którą będzie działała w trochę bardziej zawansowany sposób i sprawdzała tylko liczby w zakresie od 2 do pierwiastka z N.

Zadanie 5

Napisz funkcję `eratosthenes_sieve`, która zwróci wszystkie liczby pierwsze z zadanego zakresu [2,n].

Sito Eratostenesa

Algorytm sita Eratostenesa służy do wyznaczania liczb pierwszych z zadanego przedziału [2,n].

Algorytm rozpoczyna swoje działanie od wybrania najmniejszej dostępnej w przedziale liczby, czyli początkowo jest to 2, po czym wykreśla się ze zbioru wszystkich jej wielokrotności. Następnie takie samo działanie wykonuje się dla pozostałych w zbiorze. Wszystkie pozostałe liczby w zbiorze są pierwszymi.

Zadanie 6

Napisz funkcję `is_prime_6` która będzie ulepszoną wersją funkcji z zadania 4 i zamiast sprawdzać wszystkie liczby z przedziału, będzie sprawdzała tylko podzielność przez liczby pierwsze wykorzystując. Do wygenerowania ciągu liczb pierwszych mniejszych niż pierwiastek z N użyj funkcji z poprzedniego zadania.

Zadanie 7

Napisz funkcję `is_prime_7`, która będzie zoptymalizowaną wersją funkcji z poprzedniego zadania. By zyskać na wydajności raz użyj sita by przygotować listę liczb pierwszych i dalej wykorzystuj jej zapamiętaną wartość. Korzystaj z sita ponownie tylko wtedy, gdy okaże się iż zapamiętana lista jest za krótka.

Zadanie 8

Przetestuj zaimplementowane funkcje pod kątem wydajności. Wygeneruj losową listę liczb z wybranego przedziału a następnie zmierz i przeanalizuj średni czas działania każdego algorytmu. Wykorzystaj kod z poprzednich zadań.