ADT & Algorithms
Rev 1.

Dr. NOURI Nabil

September 30, 2024

1. Number Theory Exercises

1.1 Prime Number Check:

Write a function that takes an integer N ($1 \leq N \leq 10^9$) and returns whether N is a prime number or not.

1.2 Greatest Common Divisor (GCD):

Given two integers A and B, write an algorithm to compute their greatest common divisor using the Euclidean algorithm.

1.3 Least Common Multiple (LCM):

Write a function that calculates the least common multiple of two integers A and B ($1 \leq A, B \leq 10^9$). Use the relationship between LCM and GCD to optimize your solution.

1.4 Sieve of Eratosthenes:

Write an algorithm to generate all prime numbers less than or equal to a given number N ($1 \leq N \leq 10^6$) using the Sieve of Eratosthenes.

1.5 Modular Exponentiation:

Given three integers A, B, and M, write an algorithm to compute $(A^B) \mod M$ efficiently for large values of B using modular exponentiation.

1.6 Euler's Totient Function:

Given an integer N, write an algorithm to compute the number of integers from 1 to N that are coprime with N (i.e., their GCD with N is 1). This is also known as Euler's Totient Function $\phi(N)$.

1.7 Fermat's Little Theorem:

Write a function that takes three integers A, B, and P (where P is a prime) and computes $(A^B) \mod P$ using Fermat's Little Theorem for efficient computation.

1.8 Sum of Divisors:

Write an algorithm to compute the sum of all divisors of a given number N ($1 \leq N \leq 10^6$).

1.9 Linear Diophantine Equation:

Given three integers A, B, and C, write an algorithm to determine if the equation $Ax + By = C$ has integer solutions and, if so, find one such solution.

1.10 Chinese Remainder Theorem:

Write a function that solves a system of simultaneous congruences. Given several pairs of integers $(n_1, r_1), (n_2, r_2), \ldots, (n_k, r_k)$, find the smallest integer X such that:

$$X \mod n_1 = r_1$$

$$X \mod n_2 = r_2$$

$$\ldots$$

$$X \mod n_k = r_k$$

2. Series Exercises

2.1 Arithmetic Series Sum:

Write a function to compute the sum of the first N terms of an arithmetic series with the first term a and common difference d.

2.2 Geometric Series Sum:

Given the first term a, the common ratio r, and the number of terms N, write an algorithm to compute the sum of the first N terms of a geometric series.

2.3 Fibonacci Series:

Write a function to generate the first N numbers in the Fibonacci series. The series starts with $0, 1$ and each subsequent number is the sum of the previous two.

2.4 Sum of Squares of the First N Natural Numbers:

Write a function that computes the sum of the squares of the first N natural numbers. The result should be calculated as:

$$S = 1^2 + 2^2 + 3^2 + \cdots + N^2$$

2.5 Triangular Numbers Series:

The n-th triangular number is the sum of the first n natural numbers. Write a function that takes an integer n and returns the n-th triangular number. The series is: 1, 3, 6, 10, 15, ...

2.6 Sum of a Harmonic Series:

Write a function that computes the sum of the first N terms of a harmonic series:

$$H(N) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$$

2.7 Factorial Series:

Write a function that computes the sum of the factorials of the first N natural numbers. The sum is:

$$S = 1! + 2! + 3! + \cdots + N!$$

2.8 Alternating Series Sum:

Write an algorithm to compute the sum of the first N terms of an alternating series. For example:

$$S(N) = 1 - 2 + 3 - 4 + 5 - \cdots + (-1)^{N+1}N$$

2.9 Exponential Series:

Given a real number x and an integer N, write a program to compute the sum of the first N terms of the exponential series:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^N}{N!}$$

2.10 Sum of an Arithmetic-Geometric Series:

Write a function that computes the sum of the first N terms of an arithmetic-geometric series. The n-th term of such a series is given by the product of an arithmetic sequence and a geometric sequence:

$$T_n = (a + (n-1)d) \cdot r^{n-1}$$

Where a is the first term, d is the common difference, and r is the common ratio.

3. Vector Exercises

3.1 Vector Addition:

Write a function that takes two vectors of size n and returns their sum. For example, given $A = [a_1, a_2, a_3]$ and $B = [b_1, b_2, b_3]$, return $[a_1 + b_1, a_2 + b_2, a_3 + b_3]$.

3.2 Dot Product of Two Vectors:

Write a function to compute the dot product of two vectors of size n. The dot product is defined as:

$$A \cdot B = a_1b_1 + a_2b_2 + \cdots + a_nb_n$$

3.3 Cross Product of Two 3D Vectors:

Write a function that takes two 3D vectors and computes their cross product. Given $A = [a_1, a_2, a_3]$ and $B = [b_1, b_2, b_3]$, the cross product is:

$$A \times B = [(a_2b_3 - a_3b_2), (a_3b_1 - a_1b_3), (a_1b_2 - a_2b_1)]$$

3.4 Magnitude of a Vector:

Write a function to compute the magnitude (or length) of a vector of size n. The magnitude is calculated as:

$$|A| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2}$$

3.5 Vector Normalization:

Write an algorithm that normalizes a given vector A of size n, making its magnitude equal to 1 while preserving its direction. The normalized vector B is given by:

$$B = \frac{A}{|A|}$$

3.6 Angle Between Two Vectors:

Write a function to compute the angle (in radians or degrees) between two vectors A and B of size n. Use the formula:

$$\cos(\theta) = \frac{A \cdot B}{|A||B|}$$

3.7 Projection of One Vector onto Another:

Write an algorithm to compute the projection of vector A onto vector B. Use the formula:

$$\text{proj}_B(A) = \frac{A \cdot B}{B \cdot B} B$$

3.8 Vector Reflection:

Given a vector A and a normal vector N, write a function to compute the reflection of A off the surface represented by N. The formula for reflection is:

$$R = A - 2(A \cdot N)N$$

3.9 Scalar Multiplication of a Vector:

Write a function that takes a vector A of size n and a scalar k, and returns the vector resulting from multiplying A by k. For example, if $A = [a_1, a_2, a_3]$ and $k = 3$, return $[3a_1, 3a_2, 3a_3]$.

3.10 Distance Between Two Vectors:

Write an algorithm to compute the Euclidean distance between two vectors A and B of size n. The distance is given by:

$$d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$