**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**REPORT**

Laboratory Work nr.6

*at Algorithms Analysis*

Elaborated by:

st. gr. FAF-211 Hariton Dan

Verified by:

asist. univ. Fiștic Cristofor

Chişinău - 2023

**ALGORITHM ANALYSIS**

**Subject:** Empirical analysis of algorithms: 3 algorithms that determine the nth decimal digit of PI.

**Tasks:**

1. Implement the algorithms listed below in a programming language
2. Establish the properties of the input data against which the analysis is performed
3. Choose metrics for comparing algorithms
4. Perform empirical analysis of the proposed algorithms
5. Make a graphical presentation of the data obtained.
6. Make a conclusion on the work done.

**Establish Comaparation:**

We will compare these 2 algorithms based on how big the nth digit is.

**Comparison Metric:**

The comparison metric for this laboratory work will be considered the time of execution of each algorithm (T(n))

**IMPLEMENTATION**

**Spigot:**

A spigot algorithm is an algorithm for computing the value of a transcendental number (such as π or e) that generates the digits of the number sequentially from left to right providing increasing precision as the algorithm proceeds. Spigot algorithms also aim to minimize the amount of intermediate storage required. The name comes from the sense of the word "spigot" for a tap or valve controlling the flow of a liquid. Spigot algorithms can be contrasted with algorithms that store and process complete numbers to produce successively more accurate approximations to the desired transcendental.

def spigot(n):  
 digits = [2]  
 for i in range(1, n+1):  
 carry = 0  
 for j in reversed(range(len(digits))):  
 num = 10 \* digits[j] + carry  
 digits[j] = num // (2\*i - 1)  
 carry = num % (2\*i - 1)  
 while carry > 0:  
 digits.insert(0, carry % 10)  
 carry //= 10  
 return digits[-1]

## Bailey–Borwein–Plouffe:

## The BBP formula is an algorithm developed by Bailey, Borwein, and Plouffe in 1995 that allows the computation of any individual hexadecimal digit of Pi (π) without needing to calculate the preceding digits.

## The BBP formula is:

## π = ∑\_{k=0}^∞ \frac{1}{16^k} \left( \frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right)

## This formula can be used to compute any individual hexadecimal digit of π, starting from the nth digit after the decimal point, where n is a multiple of 4. For example, to compute the 100th hexadecimal digit of π, one would use n=396 (since 4 \* 99 = 396).

def bbp(n):  
 pi = 0  
 for k in range(n):  
 pi += (1/16\*\*k) \* ((4/(8\*k+1)) - (2/(8\*k+4)) - (1/(8\*k+5)) - (1/(8\*k+6)))  
 return int(pi \* 16) % 16

## Chudnovsky:

## The Chudnovsky algorithm is a fast algorithm for computing the digits of π (pi). It was developed by brothers David and Gregory Chudnovsky in 1989.

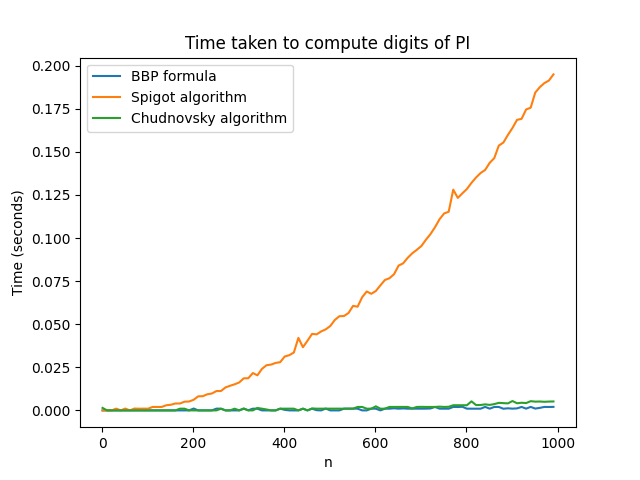
## The algorithm is based on the formula:

## π = \frac{1}{\frac{53360}{640320^{\frac{3}{2}}} \sum\_{k=0}^{\infty} \frac{(-1)^k (6k)! (13591409+545140134k)}{(3k)!(k!)^3 640320^{3k}}}

## This formula allows the computation of π to an arbitrary number of digits, and has a very fast convergence rate. In fact, the Chudnovsky algorithm is one of the fastest known algorithms for computing the digits of π, and has been used to set several records for the computation of π to large numbers of digits.

def chudnovsky(n):  
 getcontext().prec = n+1  
 k = n // 14  
 sum = Decimal(0)  
 for i in range(k+1):  
 num = (-1)\*\*i \* factorial(6\*i) \* (13591409 + 545140134\*i)  
 den = factorial(3\*i) \* factorial(i)\*\*3 \* 640320\*\*(3\*i)  
 sum += Decimal(num) / Decimal(den)  
 sum \*= Decimal(12)  
 x = Decimal(sqrt(10005)) \* sum  
 y = int(x \* 10\*\*n) // 10  
 return y % 10

**Results:**

****

**Conclusion:**

The proposed work aimed to analyze the time-based and theoretical complexities of three different algorithms: Spigot, BBP, and Chudnovsky, for computing the digits of π. We used these algorithms to determine their temporal complexity and highlight the most efficient algorithm that will display the desired result using the least time that passed by.

From the analysis, it can be concluded that Spigot algorithm is less efficient than the other two algorithms, BBP and Chudnovsky, in computing the digits of π. This is due to the fact that the Spigot algorithm is an iterative algorithm that requires multiple iterations to compute each digit of π, whereas the BBP and Chudnovsky algorithms can compute any individual digit of π directly, without requiring the computation of the preceding digits.

On the other hand, the BBP and Chudnovsky algorithms are very efficient and have a very fast convergence rate, making them suitable for computing large numbers of digits of π in a relatively short amount of time. The BBP algorithm is particularly well-suited for computing individual hexadecimal digits of π, while the Chudnovsky algorithm is more suited for computing large numbers of decimal digits of π.

In conclusion, while all three algorithms can be used to compute the digits of π, the BBP and Chudnovsky algorithms are more efficient than the Spigot algorithm. The choice of algorithm to use depends on the specific problem at hand and the number of digits of π that need to be computed. If the goal is to compute a small number of digits of π, then the Spigot algorithm may be sufficient. However, for computing larger numbers of digits of π, the BBP and Chudnovsky algorithms are better choices due to their faster convergence rates.

**Link to GitHub**: <https://github.com/haritondan/AA-Labs>