Size and Guesstimation: Computational

EGMOTC 2023 - Rohan

December 13, 2023

Problems

Problem 1. (Newton Iteration)

- Find $\sqrt{2023}$ upto 20 decimal places (without using the $\sqrt{\cdot}$ operation). You are free to write code for this or use a calculator.[1]
- Find the first 10 digits of π. [2]

Problem 2. (Big-O) Order the function by their sizes as $n \mapsto \infty$?

- $f(x) = 2023 \log(n)^{2023}$, $g(x) = \log(\log(F_n^{F_n^2} + 3^{3^n}))^3$, $h(x) = 1.001^n$
- $f(n) = 3f\left(\lfloor n/2 \rfloor\right) + 2023n$ with $f(1) = 1$, $g(n) = 1.01g(n-1) - g(n-2)$ with $g(0) = 0$ and $g(1) = 1$
- $f(n) = 2023^{2023^{2023n}}$, $g(n) = 2^{2^{2^{2n}}}$, and $h(n) = 1.01^{1.01^{1.01^{1.01^{1.01n}}}}$

Problem 3. (Some contest problems)

- Compute $\left\lfloor \sum_{k=2023}^{\infty} \frac{2024! - 2023!}{k!} \right\rfloor$
- For any natural number n, expressed in base 10 , let $s(n)$ denote the sum of all its digits. Find all natural numbers m and n such that $m < n$ and

$$(s(n))^2 = m \text{ and } (s(m))^2 = n$$

[1] 4 iterations of Newton's algorithm are definitely sufficient, 3 iterations might be enough too. Can you argue that 4 iterations are sufficient?

[2] $sin\pi = 0$ and you can use Newton Iteration

[3] F_n is the nth Fibonacci term

Solutions

Problem 1 (Newton Iteration)

- Find $\sqrt{2023}$ upto 20 decimal places (without using the $\sqrt{\cdot}$ operation). You are free to write code for this or use a calculator.
- Find the first 10 digits of π.

Solution. I have written a short python program for both the approximations. You can check the same using a high-precision calculator too.

```
k=float(45)
for i in range(4):
    k=(k+(2023/k))/2
print(k)
```

The answer outputted is: 44.97777228809804. This is because my system can only handle that many bits. So, I put this into a calculator for one more iteration and got:

$$44.977772288098040038527467811867427253575509885245374316455309088$$

Wolfram gives

$\sqrt{2023} = 44.977772288098040038527467811867427237074406112401653066261685806$

You can see that the ouput we get is indeed correct upto 34 points after the decimal in just 5 steps!
Now, moving onto π. We observe that $sin(\pi) = 0$. So, we use our approximation as $\pi_0 = 3$ and

$$\pi_{i+1} = \pi_i - \frac{\sin(\pi_i)}{\cos(\pi_i)}$$

Writing the code for this:

```
import math
pi=float(3)
for i in range(4):
    pi=pi-(math.sin(pi)/math.cos(pi))
print(pi)
```

My system outputs: 3.141592653589793. This is again due to my systme not handling better. I applied the approximation again using a better calculator and got:

$$3.1415926535897932384626433832795028841971693993796258352543$$

Wolfram gives the value of π as:

$$3.1415926535897932384626433832795028841971693993751058209749445923$$

This is correct for 47 digits after the decimal in just 5 steps!! $\square$

Problem 2.

Problem 2. (Big-O) Order the function by their sizes as $n \mapsto \infty$?

- $f(x) = 2023 \log(n)^{2023}$, $g(x) = \log(\log(F_n^{F_n^2} + 3^{3^n}))$[4], $h(x) = 1.001^n$
- $f(n) = 3f\left(\lfloor n/2 \rfloor\right) + 2023n$ with $f(1) = 1$, $g(n) = 1.01g(n-1) - g(n-2)$ with $g(0) = 0$ and $g(1) = 1$
- $f(n) = 2023^{2023^{2023n}}$, $g(n) = 2^{2^{2^{2n}}}$, and $h(n) = 1.01^{1.01^{1.01^{1.01^{1.01n}}}}$

Solution.

- $f(x) = O(\log(x)^{2023})$, $g(x) = O\left(\left(\log\log F_n^{F_n^2}\right) + n\right) = O(n + \log F_n) = O(n)$, $h(x) = O(1.001^n)$. Thus, the first is polylogarithmic, second is linear and third is exponential. Thus, for all large enough n, we have $f(n) < g(n) < h(n)$
- $f(n) = 3f(n/2) + O(n) \implies f(n) = O(n^{3/2})$ by the master theorem. $g(n) = 1.01g(n-1) - g(n-2)$ so the answer is of the form $C_1\alpha^n + C_2\beta^n$ where α and β are the two roots of $x^2 = 1.01x - 1$. Now, both the roots are complex conjugates and have modulus 1. So, $|g(n)|$ is always bounded!.
- We take log twice for all function. The functions become $\log\log f(n) = O(n)$, $\log\log g(n) = O(4^n)$ and finally $\log\log h(n) = O(1.01^{1.01^{1.01n}})$. Now, to make the comparison between $g(n)$ and $h(n)$ even more clear, let's take log twice more. Now, $\log\log\log\log f(n) = O(\log\log n)$, $\log\log\log\log g(n) = O(\log n)$, and $\log\log\log\log h(n) = O(n)$

$\square$

[4] F_n is the nth Fibonacci term

Problem 3

Problem 3. (Contest problems)

- Compute $\left\lceil \sum_{k=2023}^{\infty} \frac{2024! - 2023!}{k!} \right\rceil$
- For any natural number n, expressed in base 10 , let $s(n)$ denote the sum of all its digits. Find all natural numbers m and n such that $m < n$ and

$$(s(n))^2 = m \text{ and } (s(m))^2 = n$$

Solution.

- We write the first part simply as

$$2024 + \left\lceil \sum_{k=2024}^{\infty} \frac{2024!}{(k+1)!} - \frac{2023!}{k!} \right\rceil$$

  Now, for all $k \geq 2024$, we have $\frac{2024!}{(k+1)!} < \frac{2023!}{k!}$ as we have $2024 < k+1$. Thus, the part inside the $\lceil \rceil$ is < 0. Thus, the value is ≤ 2024. Now, we also have using $k = 1$ that the sum is $2023 + \left\lceil \sum_{k=2024} \frac{2024! - 2023!}{k!} \right\rceil \geq 2024$.

  Thus, the final value is simply 2024. $\square$

- This problem is from RMO 2023: We first observe that if n has k digits, we get that $s(m)^2 = n$ and m also has at most k digits. Thus, $10^{k-1} \leq (9k)^2$. This is already false for $n = 5$. Thus, n has at most 4 digits but then $s(m) \leq 36 \implies n \leq 1296$.

  Then,

$$s(m) \leq 27 \implies n \leq 729 \implies s(m) \leq 24 \implies n \leq 576$$
$$\implies s(m) \leq 22 \implies n \leq 484 \implies s(m) \leq 21 \implies n \leq 441$$

  Now, since $n = 441$ doesn't work, we get $n \leq 400$ as it is also a perfect square.

  At this point, we can simply check all n! Possibilities of n are $\{1^2, 2^2, \cdots 20^2\}$ but $s(m) \equiv m \not\equiv 2 \pmod 3$ since m is a square.

  Thus, $n \in \{1^2, 3^3, 4^2, 6^2, 7^2, 9^2, 10^2, 12^2, 13^2, 15^2, 16^2, 18^2, 19^2\}$. Checking tells us that $n = 256$ gives $m = 169$ which indeed works!

  $\square$