5-23-19

Mental Model

- Thinking : Abstraction
- Mathematical Thinking : Abstraction + Assertion [Proof properties of ~~idea~~ model we have developed.
- Computational Thinking : Abstraction + Assertion + Action [State - to capture idea of dynamics]
- Algorithmic Thinking : Abstraction + Assertion + Action + Optimization [optimize time & space]

☆ Question : [Mathematical Thinking]

Q: Write numbers in a circle ex:- from 1 to 1000. Tick mark numbers one by one with a specific jump or gap i.e. ex: 1, 16, 31..... until you encounter a number that is already marked and then you stop. Find total marked and unmarked elements by the time you have stopped.

<u>Solution</u> :- We do this for total 1000 numbers with a jump of 15.

<u>Things we observe</u>

<u>#1 Round</u>: 1, 16, 31 .... 991 [last number before starting again]

<u>#2 Round</u>: 6, 21, 36, ... 996

<u>#3 Round</u>: 11, 26, 41, ... 986, (1001 == 1)

We encounter 1 again so we stop.

We observe they are all multiples of 5 plus 1

i.e. of the form $5k + 1$

Multiples of 5 are:- 0, 5, 10, 15 ..... 995, 1000

Here:- 1, 6, 11, 16 996

$\Rightarrow$ Marked 200 Numbers, Unmarked 800 Numbers

→ How is $5k+1$ related to $15k'+1$

k can be 3α, $3\alpha+1$, $3\alpha+2$

→ $5(3\alpha)+1 = 15\alpha+1$

$5(3\alpha+1)+1 = 15\alpha+6$

$5(3\alpha+2)+1 = 15\alpha+11$

- We use modular arithmetic to find pattern.
- Modular Arithmetic is also known as Clock Arithmetic
- We find pattern in the problem to tackle that problem.

Question: Mathematical Thinking [Impossibility Result]

- How to tackle a problem with no solution

Q For some $F(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots\ldots\ldots a_1 x + a_0$ if $F(a)=b$, $F(b)=c$, $F(c)=a$. Is it possible when $a \neq b \neq c$. Given Int coefficients.

$\Rightarrow$ <u>Proof by Contradiction</u> $\because$ We assume its possible for some $a \neq b \neq c$

$\rightarrow F(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots\ldots a_1 x + a_0$

$\rightarrow F(y) = a_n y^n + a_{n-1} y^{n-1} + \ldots\ldots a_1 y + a_0$

$$F(x) - F(y) = a_n(x^n - y^n) + a_{n-1}(x^{n-1} - y^{n-1}) + \ldots\ a_1(x-y)$$

$$= k(x-y)$$

$\because k$ ~~is~~ some constant you get after taking $(x-y)$ common from every term.

So ~~for~~ we can write

$$f(a) - f(b) = k_1(a-b) = b-c \quad (\text{given}) \quad -①$$

$$f(b) - f(c) = k_2(b-c) = c-a \quad -②$$

$$f(c) - f(a) = k_3(c-a) = a-b \quad -③$$

Multiply ① × ② × ③ & we get

$$k_1 \times k_2 \times k_3 = 1$$

So now we need to prove

$$a-b = b-c$$

$$b-c = c-a$$

$$c-a = a-b$$

When solved we see $a=b=c$, which is against our assumption. So its not possible.

Question:- Computational Thinking

Q You have three jugs with a, b & c litres of water in them. These are the rules you have to follow ~~for~~ ~~iff~~ $(a>b>c)$.

Put from bigger to smaller jug. You can go from state $(a,b,c) \rightarrow (a-b, 2b, c)$

$\rightarrow (a-c, b, 2c)$

$\rightarrow (a, b-c, 2c)$

Goal :- Only using above steps empty one of the jugs.

⇒ Solution :- The solution is based on the principle called as Well ordering principle and the fact that remainder is always smaller than the number.

Ex: 10, 186, 329

Removing $10 \times 18 = 180$ from 186, brings capacity to 6 as 186 modulus $10 = 6$.

18 represented in binary form is 10010.

⇒ ~~10×18~~ $10 \times 2^4 + 0 + 0 + 10 \times 2 + 0 = 180$

→ 10 186 329 → 320, 9, 199

→ 20 186 319

→ 40 166 319

→ 80 166 279

→ 160 166 199

To solve we find $q = b/c$. Here it is $\frac{186}{10} = 18$.

Write q in binary form.

Now if 0 transfer from ~~a~~ $a \to c$.

if 1 transfer from ~~b~~ $b \to c$.

After you complete this process, minimum would be 6. Keep on repeating until you reach zero.

☆ Can also be solved by making largest jug even larger. Think on your own?

Go from some state $(a, b, c) \to (a', b', c')$ where $Max(a', b', c') > Max(a, b, c)$

☆ Odd-Even Transposition Sort

It is a sorting algorithm developed for use on parallel processors.

Here you consider index of array start from 1.

Algorithm: [Given array of n elements]

1. Consider pair of every odd ~~d~~ position element and its next element. Compare & swap if necessary.
2. Do same thing but for every even & its next element
3. Repeat 1 & 2 $\lceil n/2 \rceil$ times.

Task to do:- Prove the correctness of algorithm. Check proof after attempting it.

Question [Adaptive Incremental Problem].

Q. There are two people A & B. A thinks of a positive integer a, & B thinks a positive integer b. They give it to a refree R. R gives them 2 numbers back ~~which~~ which are $(a+b, r)$; r is any random number. R asks A if you can guess number of B, then he asks B if he can guess number of A. They say no. This keeps on going on, till one person says they can find the number. Each no gives the other person a new hint. Can you figure out how does a person figures out the others number.

Concept

- Incremental Order: Bunching, Order, Adaptive
- Decremental Design: where output is subset of input. Ex: Tower of Hanoi
- Pruning :- Design by elimination.

Problem :- Group of n people.

A person A is a celebrity in a group if

1. If A does not know anyone
2. Everyone knows A.

Only One celebrity possible because of constraint.

Operation you can do: Ask A if they know B.

Solution:

Naive Way:- Ask everyone if they know all other people and store in a matrix. Check for the person who knows no one but is known by all. [$^nC_2$ time]

Optimal Way:-

For some A & B

→ if (A knows B) = Yes then Remove A

if (A knows B) = No then Remove B.

[$3(n-1)$ time]

Question:- N*N matrix of positive integers, assume all are distinct positive integers. Identify the value which is larger than all its neighbours. Probe is seeing the value of a cell in matrix. Find ~~no~~ value in minimum numbers of Probe.

Question: We have 2n consecutive integers. Distribute them randomly in n boxes, 2 per box. Identify content of the boxes.

Operation you can do:- You can give any amount of numbers and get back the total number of different boxes those numbers belongs to.

Find all pairs in minimum operations.

Solution:- Divide all numbers in 2 sets A & B.

Ask system how many boxes there are in A, and how many boxes there are in set {A, number whose pair you want to find}.

→ if {A, number} > {A}

remove A

→ else

remove B.

→ Repeat these till you are left with 1 number. That number is a pair of your number.

→ Repeat whole process for leftover numbers.