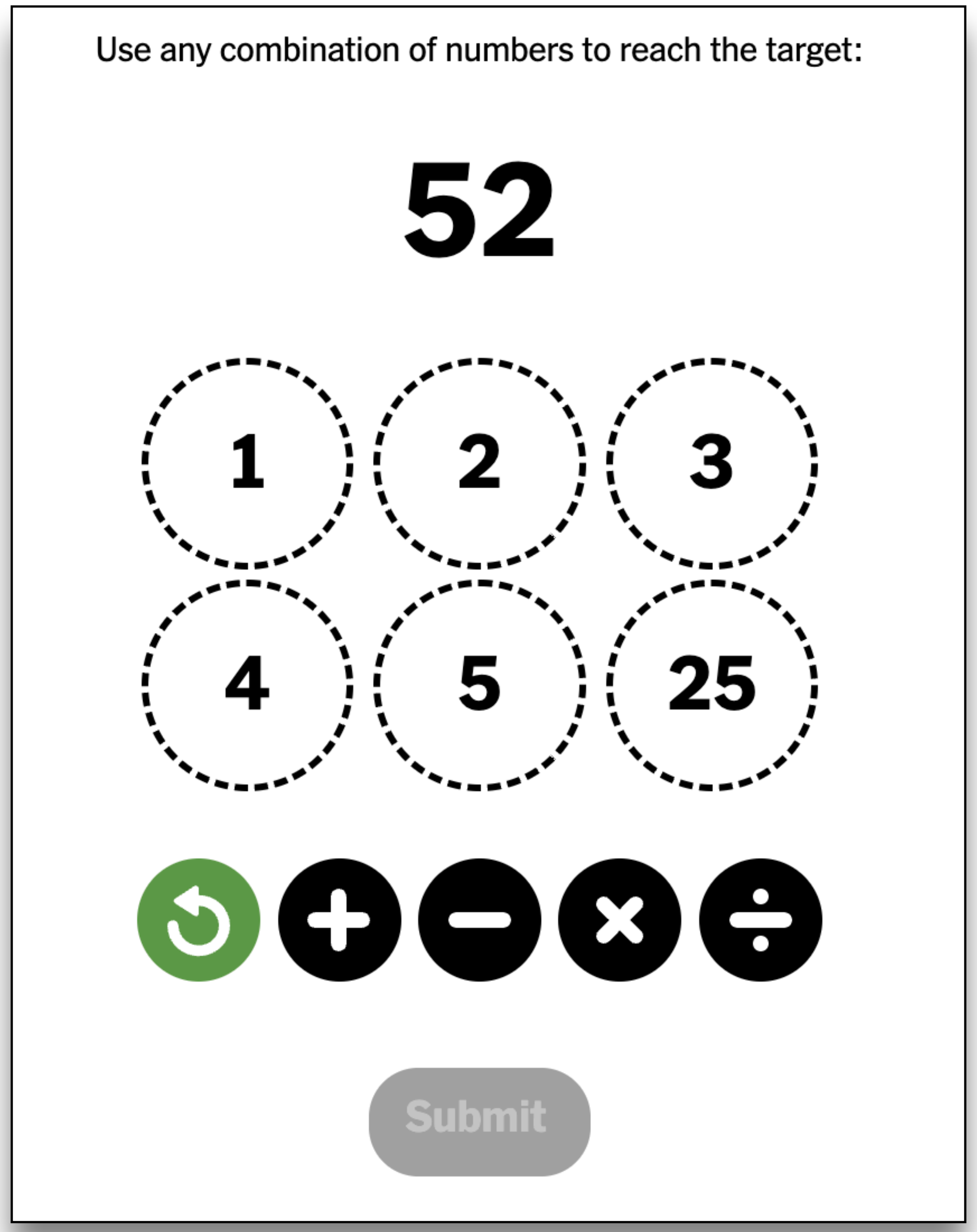
NYT Digits - Algorithms and Mathematics

JFR Journal Club

Mobolaji Williams, July 25 2023

NYT Digits - Rules of the Game



1. You are given six numbers and a target number.

2. The goal is to reproduce the target number from any combination from the set of six numbers using the operations addition, subtraction, division, and multiplication.

3. In combining numbers, divisions cannot have remainders and subtractions cannot result in negative numbers.

Main Question

Can we devise an algorithm to solve this problem?

Supplementary Questions

And what can the algorithm tell us about the nature of the solutions to this problem and this problem's computational complexity?

Presentation Outline

Main Question

Can we devise an algorithm to solve this problem?

Part I

Algorithms

How can we generate all possible solutions to the NYT digits problem?

Part II

Mathematics

How can we count the number of solutions to the puzzle?

$$F(x) = \sum_{k=0}^{N} x^k F_k$$

Part III

Complexity Class

What is the computational complexity class of this problem?

P ⊆ NP ⊆ EXP-TIME

Part IV

ChatGPT

Can Chat GPT solve this problem, or generate an algorithm to solve this problem?

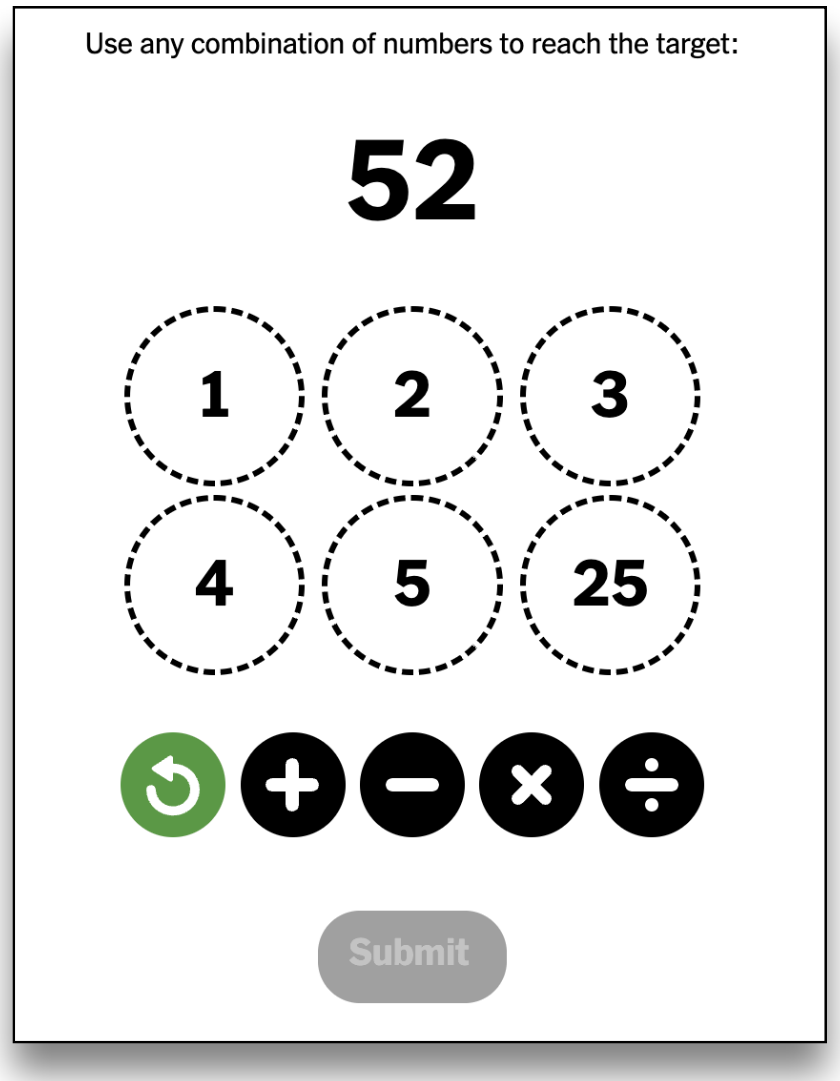
Part I: Algorithms

PartI: Incremental Programming

Part I

Algorithms

How can we generate all possible solutions to the NYT digits problem?



Refined Question: How can we find all valid computational combinations of **six numbers** that yield a target number?

We will use the principle of **incremental development**

> "The goal of **incremental development** is to avoid long debugging sessions by adding and testing only a small amount of code at a time."
> - *How to Think Like a Computer Scientist*

Rather than solving the full "six number" puzzle, we try to solve the "two-number" puzzle, "**three-number**" puzzle, "four number" puzzle and so on.

Incremental Question: How can we find all valid computational combinations of **two numbers** that yield a target number?

Part I: Two-Number Combinations

Incremental Question: How can we find all valid computational combinations of two numbers that yield a target number?

→

If we already have the combinations, we can **easily** find the ones that match a given number. The **difficult** part is generating these combinations

Refined Incremental Question #1: How can we generate all valid computational combinations of two numbers?

Example:
We have the numbers 1 and 2

Possible Combinations:

$$2+1, \quad 2-1, \quad 2\times 1, \quad 2/1$$

But we want to keep track of both the "names" of the combinations and their "results" and so we want a function that does both tasks

Example:

```python
gen_two_combos({'1': 1, '2': 2})
>>> {'(2+1)': 3, '(2*1)': 2, '(2/1)': 2, '(2-1)': 1}
```

Part I: Two-Number Combinations

Refined Incremental Question #1: How can we generate all valid computational combinations of **two numbers**?

Example:
We have the numbers 1 and 2

Possible Combinations:

$$2 + 1, \quad 2 - 1, \quad 2 \times 1, \quad 2/1$$

We want to keep track of both the "names" of the combinations and their "results" and so we want a function that does both tasks.

- Also want to prevent "bad" operations from leading to valid values

```python
def gen_two_combos(value_dict):

    # sort dictionary by value to ensure subtractions are not negative
    # and divisions are greater than 1
    sorted_vals = dict(sorted(value_dict.items(), key=lambda item: item[1], reverse=True))

    # access the list of string names and the list of associated integers
    name_list, num_list = list(sorted_vals.keys()), list(sorted_vals.values())

    # for writing simplicity
    elem1, elem2 = num_list
    name1, name2 = name_list

    # generating all combinations
    # internal if conditional yields a large number if the division
    # and subtraction constraints are violated
    combo_dict = {'('+name1+'+'+name2+')':elem1 + elem2,
                  '('+name1+'*'+name2+')':elem1 * elem2,
                  '('+name1+'/'+name2+')': int(elem1/elem2) if elem1%elem2==0 else 1/3001,
                  '('+name1+'-'+name2+')': elem1- elem2 if elem1- elem2>0 else 1/3001 }

    return combo_dict
```

results

prevention

names

```python
gen_two_combos({'1': 1, '2': 2})
>>> {'(2+1)': 3, '(2*1)': 2, '(2/1)': 2, '(2-1)': 1}
```

Part I: Three-Number Combinations

…solved the two number problem…

…next we have…

Refined Incremental Question #2: How can we generate all valid computational combinations of **three numbers**?

We can use the two number solutions for the **three number** solutions.

A Worked Example

Say we have three numbers `1`, `2`, and `3`. What are the various ways we can combine these numbers using our four operations?

- If we select two numbers `1` and `2` and multiply them, then we would have the number `2` in addition to the original unselected `3`.
- We would now have two numbers. We can then combine this new `2` and the old `3` in all the ways that two numbers can be combined.
- We know there are four ways to combine two numbers, so to find the various ways to combine three numbers, we can first ask how can we form two numbers from three numbers.

Binomial coefficient*

$$\binom{n}{k} \equiv \frac{n!}{k!(n-k)!}$$

Answer: The various ways to combine **three numbers** consists of…

$$= \underbrace{\binom{3}{2}}_{\text{the ways to select two numbers from three}} \otimes \underbrace{(4 \text{ ways})}_{\text{the ways to combine those two selected numbers}} \otimes \underbrace{(4 \text{ ways})}_{\text{the ways to combine the result of the two-number combinations with the third unselected number}}$$

Part I: Three-Number Combinations

Refined Incremental Question #2: How can we generate all valid computational combinations of **three numbers**?

We can use the two number solutions for the three number solutions.

Answer: The various ways to combine **three numbers** consists of…

$=$

the ways to select two numbers from three #1)

$\otimes$

the ways to combine those two selected numbers #2)

$\otimes$

the ways to combine the result of the two-number combinations with the third unselected number. #3)

Python

```python
def gen_three_combos(start_dict):

    # empty dictionary to house combo outputs
    output_dict = dict()

    for key, value in start_dict.items():
        # define copy of three number combination
        copy_dict = start_dict.copy()                      #1)
        del copy_dict[key] # eliminate one number from the three

        # for all combinations of the two remaining numbers  #2)
        for name, elem in gen_two_combos(copy_dict).items():
            # create combinations of the two-number result and the unselected number
            new_dict = {name: elem, key:value }
            result_dict = gen_two_combos(new_dict)         #3)
            output_dict.update(result_dict)

    return output_dict
```

Example:

```
In [6]: gen_three_combos({'1': 1, '2': 2, '3': 3})

Out[6]: {'((3+2)+1)': 6,
         '((3+2)*1)': 5,
         '((3+2)/1)': 5,
         '((3+2)-1)': 4,
         '((3*2)+1)': 7,
         '((3*2)*1)': 6,
         '((3*2)/1)': 6,
         '((3*2)-1)': 5,
         '(1+(3/2))': 1.000333222259247,
         '(1*(3/2))': 0.0003332222592469177,
         '(1/(3/2))': 0.0003332222592469177,
         '(1-(3/2))': 0.9996667777407531,
         '((3-2)+1)': 2,
```

Part I: Four-Number Combinations

Refined Incremental Question #3: How can we generate all valid computational combinations of **four numbers**?

We use the same logic as that for the **three number** and **two number** combinations…

#1. Choose subsets of numbers
#2. Build up the solution up from the lower-order solutions

This procedure is reminiscent of something that is common in combination and programming problems…

Recursive programming: Build up a solution by referring to solutions to sub-problems

This helps us formulate the general solution

Python

```python
from itertools import combinations

def gen_four_combos(start_dict):

    output_dict = dict()

    # split four numbers into three numbers and one number
    for key, value in start_dict.items():
        copy_dict = start_dict.copy() #1)
        del copy_dict[key]

        for name, elem in gen_three_combos(copy_dict).items():
            new_dict = {name: elem, key:value }              #2)
            result_dict = gen_two_combos(new_dict)
            output_dict.update(result_dict)

    # split four numbers into two numbers and two numbers
    for elem in list(combinations((start_dict.keys()), 2)):
        copy_dict = start_dict.copy()
        del copy_dict[elem[0]]        #1)
        del copy_dict[elem[1]]
        comp_copy_dict = {elem[0]:start_dict[elem[0]], elem[1]:start_dict[elem[1]]}

        for key1, value1 in  gen_two_combos(copy_dict).items():
            for key2, value2 in gen_two_combos(comp_copy_dict).items():
                new_dict = {key1: value1, key2:value2 }
                result_dict = gen_two_combos(new_dict)  #2)
                output_dict.update(result_dict)

    return output_dict
```

Part I: General Combinations

Refined Incremental Question #4: How can we find all valid computational combinations of **any list of numbers***?

*We are generalizing the problem to any list of numbers

```python
from itertools import combinations
from math import comb

# generates all manipulation combinations of a dictionary of elements
def generator_combos(start_dict: dict):

    # defining dict_len to determine pathing
    dict_len = len(start_dict)

    # defining output dictionary
    output_dict = dict()

    count = 0

    # output the dictionary if there is just one element
    if dict_len==1:
        output_dict = start_dict

    # output all ways to add, subtract, multiply, or divide two numbers
    elif dict_len==2:
        # sort to ensure division is >= 1 and subtraction is >= 0
        sorted_vals = dict(sorted(start_dict.items(), key=lambda item: item[1], reverse=True))

        # list of names and numbers
        name_list, num_list = list(sorted_vals.keys()), list(sorted_vals.values())

        # getting elements and there names
        elem1, elem2 = num_list
        name1, name2 = name_list

        # set of all possible operations with keys for operation name
        # Note: digits game doesn't allow fractions or negative numbers
        # so we set the result to a large
        output_dict = {'('+name1+'+'+name2+')':elem1 + elem2,
                       '('+name1+'*'+name2+')':elem1 * elem2,
                       '('+name1+'/'+name2+')': int(elem1/elem2) if elem1%elem2==0 else 1/3001,
                       '('+name1+'-'+name2+')': elem1- elem2 if elem1- elem2>0 else 1/3001 }

    # for dict_len >=3, recursively build sub solutions
    else:

        # only go up to half the number of elements in dict
        # to prevent double counting (e.g., n choose k = n choose n-k)
        for ix in range(1, dict_len//2+1):

            # number of ways to select 'ix' elements from 'dict_len' total
            num_comb = comb(dict_len, ix)

            # limiting number of combinations considered
            # when there is an even split; this prevents double counting
            if ix == dict_len/2:
                lim = num_comb//2
            else:
                lim = num_comb

            # generating combinations of elements
            # '[:lim]' prevents double consideration of combos
            for elem in list(combinations(start_dict.keys(), ix))[:lim]:
                copy_dict  = start_dict.copy()
                comp_copy_dict = dict() # complement of copy_dict
                for j in range(ix):
                    del copy_dict[elem[j]]
                    comp_copy_dict[elem[j]] = start_dict[elem[j]]

                # generating set of possible operations for elements
                for key1, value1 in generator_combos(copy_dict).items():
                    for key2, value2 in generator_combos(comp_copy_dict).items():
                        new_dict = {key1: value1, key2: value2 }
                        result_dict = generator_combos(new_dict)
                        output_dict.update(result_dict)
                        count += len(result_dict)

    return output_dict
```

Recursion

Part I: NYT Digits Solver (Easy Part)

Example:

```
In [9]: generator_combos({'1':1, '2': 2, '3': 3, '4': 4, '5':5 })

Out[9]: {'((((5+4)+3)+2)+1)': 15,
         '((((5+4)+3)+2)*1)': 14,
         '((((5+4)+3)+2)/1)': 14,
         '((((5+4)+3)+2)-1)': 13,
         '((((5+4)+3)*2)+1)': 25,
         '((((5+4)+3)*2)*1)': 24,
         '((((5+4)+3)*2)/1)': 24,
         '((((5+4)+3)*2)-1)': 23,
         '((((5+4)+3)/2)+1)': 7,
         '((((5+4)+3)/2)*1)': 6,
         '((((5+4)+3)/2)/1)': 6,
         '((((5+4)+3)/2)-1)': 5,
         '((((5+4)+3)-2)+1)': 11,
         '((((5+4)+3)-2)*1)': 10,
         '((((5+4)+3)-2)/1)': 10,
```

Original Refined Question: How can we find all valid computational combinations of **any list of numbers** that yield a target number?

"If we already have the combinations, we can **easily** find the ones that match a given number. The **difficult** part is generating these combinations"

Python

```python
def nyt_digits_solver(num_list: list, target: int, soln_num=None):

    # generating full list of combinations of all sizes for the input numbers
    soln_set_dict = dict()
    for i in range(2, len(num_list) + 1):
        for combo in list(combinations(num_list, i)):
            dict_conv = {str(num):num for num in list(combo)}
            soln_dict = generator_combos(dict_conv)
            soln_set_dict.update(soln_dict)

    # enumerating all solutions (i.e., combinations that result in target)
    solns = [key for key in soln_set_dict if soln_set_dict[key]==target]

    # find the `soln_num`-shortest solutions (by char count)
    if soln_num:
        solns_copy = list()
        for k in sorted(solns, key=len)[:soln_num]:
            solns_copy.append(k)
        solns = solns_copy

    return solns
```

"past 10 slides"

"1 min"

DEMO

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Tue Jul 25 12:39 AM

Number Gar | Home Page | algorithm_c | New Tab

localhost:8889/noteboo...

UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter algorithm_clean

Logout

Not Trusted | Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help

Run | Code

NYT Digits Puzzle - Algorithm

(May 23, 2023)

Problem Statement: Can you come up with some code that finds the solution to the NYT digits puzzle given six numbers and a target number?

In [1]:

```
from itertools import combinations
from math import comb
```

In [2]:

```
# generates all manipulation combinations of a dictionary of ele
def generator_combos(start_dict: dict):

    # defining dict_len to determine pathing
    dict_len = len(start_dict)

    # defining output dictionary
    output_dict = dict()

    count = 0

    # output the dictionary if there is just one element
    if dict_len==1:
        output_dict = start_dict

    # output all ways to add, subtract, multiply, or divide two
```

Digits: A Daily Math Puzzle - T

nytimes.com/games...

Digits

BETA

A daily numbers puzzle from The New York Times.

This game is going away on **August 8th.**

Play

July 25, 2023

Part I: Hints at Something Deeper Solutions

When we count the number of generated combinations for various lists of numbers we find…

In [10]:
```
for ell in range(1,7):
    dict_ = {str(num):num for num in range(1,ell+1)}
    all_combos = generator_combos(dict_)
    print(f'Number of combinations with {ell} numbers:', len(all_combos))
```

```
Number of combinations with 1 numbers: 1
Number of combinations with 2 numbers: 4
Number of combinations with 3 numbers: 48
Number of combinations with 4 numbers: 960
Number of combinations with 5 numbers: 26880
Number of combinations with 6 numbers: 967680
```

Is there a way that we can show why this is the case? And also use this to determine the complexity of this algorithm?

Part II: Mathematics

Part II

Mathematics

How can we count the number of solutions* to the puzzle?

$$F(x) = \sum_{k=0}^{N} x^k F_k$$

*We are actually counting the number of "generated combinations" and not exact solutions

Refined Question:
How can we explain the following sequence?

```
In [10]: for ell in range(1,7):
             dict_ = {str(num):num for num in range(1,ell+1)}
             all_combos = generator_combos(dict_)
             print(f'Number of combinations with {ell} numbers:', len(all_combos))
```

```
Number of combinations with 1 numbers: 1
Number of combinations with 2 numbers: 4
Number of combinations with 3 numbers: 48
Number of combinations with 4 numbers: 960
Number of combinations with 5 numbers: 26880
Number of combinations with 6 numbers: 967680
```

Let's formulate this more abstractly

20. NYT Digits Puzzle

[combinatorics - numbers]

We have N integers $d_1 < d_2 < \ldots < d_N$. We have four possible operations $\times$, $\div$, $+$, and $-$. Say that Q_N is the number of different real numbers (that are greater than 1) we can form from combining all N integers using the available operations.

(Definition) Q_N is the number of combinations created by this algorithm when there are N numbers (i.e., $Q_1 = 1, Q_2 = 4, Q_3 = 48$, etc.)

*"closed-form" means we can compute values directly as a function

Further Refined Question: What is the closed-form* expression for Q_N?

Part II: Counting Combinations

20. NYT Digits Puzzle

[combinatorics - numbers]

We have N integers $d_1 < d_2 < \ldots < d_N$. We have four possible operations $\times$, $\div$, $+$, and $-$. Say that Q_N is the number of different real numbers (that are greater than 1) we can form from combining all N integers using the available operations.

Further Refined Question: What is the closed-form expression for Q_N?

We will proceed as before (i.e., incrementally)

Two-Number Combinations:

$$d_2 + d_1, \quad d_2 - d_1, \quad d_2 \times d_1, \quad d_2 / d_1$$

$$Q_2 = 4$$

Three-Number Combinations:

$$Q_3 = Q_2 \binom{3}{2} Q_2 = 48$$

number of ways to combine the results of sub-combinations

number of ways to choose two numbers from three

number of ways to combine those two numbers

Four-Number Combinations:

number of ways to choose two numbers from four

$$Q_4 = Q_2 \binom{4}{3} Q_3 + \frac{1}{2} Q_2 \binom{4}{2} Q_2 Q_2 = 960$$

correction for double counting

number of ways to combine the results of sub-combinations

number of ways to combine those two chosen numbers

number of ways to combine the two unchosen numbers

Or with $Q_1 \equiv 1$

$$Q_4 = \frac{1}{2} Q_2 \left[\binom{4}{3} Q_3 Q_1 + \binom{4}{2} Q_2 Q_2 + \binom{4}{1} Q_1 Q_3 \right]$$

Part II: Counting Combinations

20. **NYT Digits Puzzle**

[combinatorics - numbers]

We have N integers $d_1 < d_2 < \ldots < d_N$. We have four possible operations $\times$, $\div$, $+$, and $-$. Say that Q_N is the number of different real numbers (that are greater than 1) we can form from combining all N integers using the available operations.

Four-Number Combinations:

$$Q_1 \equiv 1$$

$$Q_4 = \frac{1}{2} Q_2 \left[\binom{4}{3} Q_3 Q_1 + \binom{4}{2} Q_2 Q_2 + \binom{4}{1} Q_1 Q_3 \right]$$

Further Refined Question: What is the closed-form expression for Q_N?

N-Number Combinations:

$$Q_N = \frac{1}{2} Q_2 \sum_{k=1}^{N-1} \binom{N}{k} Q_{N-k} Q_k$$

We can use **dynamic programming** to generate solutions to this equation

```
In [10]: for ell in range(1,7):
             dict_ = {str(num):num for num in range(1,ell+1)}
             all_combos = generator_combos(dict_)
             print(f'Number of combinations with {ell} numbers:', len(all_combos))

Number of combinations with 1 numbers: 1
Number of combinations with 2 numbers: 4
Number of combinations with 3 numbers: 48
Number of combinations with 4 numbers: 960
Number of combinations with 5 numbers: 26880
Number of combinations with 6 numbers: 967680
```

```python
from math import comb
import numpy as np

Q, Q2 = dict(), 4 # define dictionary and Q2 value
Q[1] = 1 # define first dictionary value

def Q_calc(N):

    if N in Q.keys():
        return Q[N]
    else:
        return (Q2//2)*np.sum([comb(N,k)*Q_calc(N-k)*Q_calc(k) for k in range(1,N)])
```

```
In [4]: for k in range(1, 7):
            print(f'QN for N={k}: {Q_calc(k)}')

QN for N=1: 1
QN for N=2: 4
QN for N=3: 48
QN for N=4: 960
QN for N=5: 26880
QN for N=6: 967680
```

These solutions match the previous counting of results…

…however this expression is not in closed-form. We still need to solve for Q_N.

Part II: Closed form solution

20. **NYT Digits Puzzle**

[combinatorics - numbers]

We have N integers $d_1 < d_2 < \ldots < d_N$. We have four possible operations $\times, \div, +,$ and $-$. Say that Q_N is the number of different real numbers (that are greater than 1) we can form from combining all N integers using the available operations.

Further Refined Question: What is the closed-form expression for Q_N?

#1

$$Q_N = \frac{1}{2} Q_2 \sum_{k=1}^{N-1} \binom{N}{k} Q_{N-k} Q_k$$

…This expression is not in closed form. We still need to solve for Q_N.

Generating functions approach: Convert a recursive equation into an algebraic equation by summing over combinatorial coefficients

"Generating Function" $Q(x) \equiv \sum_{\ell=1}^{\infty} \frac{x^\ell}{\ell!} Q_\ell$ #2 and #1

$$Q(x) - x = \frac{Q_2}{2} Q(x)^2$$

Quadratic formula

$$Q(x) = \frac{1}{Q_2}\left(1 - \sqrt{1 - 2Q_2 x}\right)$$

Taylor series expansion

$$Q(x) = \sum_{k=1}^{\infty} C_{k-1} \left(\frac{Q_2}{2}\right)^{k-1} x^k$$

$$C_n \equiv \frac{1}{n+1}\binom{2n}{n}$$

Catalan Numbers

Which (with #2) gives us

$$Q_N = N! \left(\frac{Q_2}{2}\right)^{N-1} C_{N-1},$$

This reproduces the series we found

A052714 a(n) = 2^(n-1) * n! * Catalan(n-1) for n > 0 with a(0) = 0. 22

0, 1, 4, 48, 960, 26880, 967680, 42577920, 2214051840, 132843110400, 9033331507200, 686533194547200, 57668788341964800, 5305528527460761600, 530552852746076160000, 57299708096576225280000, 6646766139202842132480000, 824199001261152424427520000 (list; graph; refs; listen; history; text; internal format)

OFFSET 0,3

https://oeis.org/A052714

See "NYT Digits Puzzle - Mathematics" for derivation details

Part III: Complexity

Part III

Complexity Class

What is the computational complexity class of this problem?

P ⊆ NP ⊆ EXP-TIME

20. NYT Digits Puzzle

[combinatorics - numbers]

We have N integers $d_1 < d_2 < \ldots < d_N$. We have four possible operations $\times$, $\div$, $+$, and $-$. Say that Q_N is the number of different real numbers (that are greater than 1) we can form from combining all N integers using the available operations.

$$Q_N = N!\left(\frac{Q_2}{2}\right)^{N-1} C_{N-1},$$

$$C_n \equiv \frac{1}{n+1}\binom{2n}{n}$$

Catalan Numbers

$$Q_N = \left(\frac{Q_2}{2}\right)^{N-1} \frac{(2N-2)!}{(N-1)!}$$

With Stirling's approximation
$N! \sim (N/e)^N$

$$Q_N \sim O\left((2Q_2)^N\right)$$

$$Q_2 = 4$$

Exponential Time Computation

$$\boxed{Q_N \sim O(2^{3N})}$$

"As N increases it eventually becomes impossible to list every solution"

NYT Digits Puzzle ⊆ EXP-TIME

Presentation Outline

Main Question

Can we devise an algorithm to solve this problem?

Part I

Algorithms

How can we generate all possible solutions to the NYT digits problem?

```
def nyt_digits_solver(num_list:

    # generating full list of co
    soln_set_dict = dict()
    for i in range(2, len(num_li
```

Part II

Mathematics

How can we count the number of solutions to the puzzle?

$$Q_N = N!\left(\frac{Q_2}{2}\right)^{N-1} C_{N-1},$$

Part III

Complexity Class

What is the computational complexity class of this problem?

NYT Digits Puzzle $\subseteq$ EXP-TIME

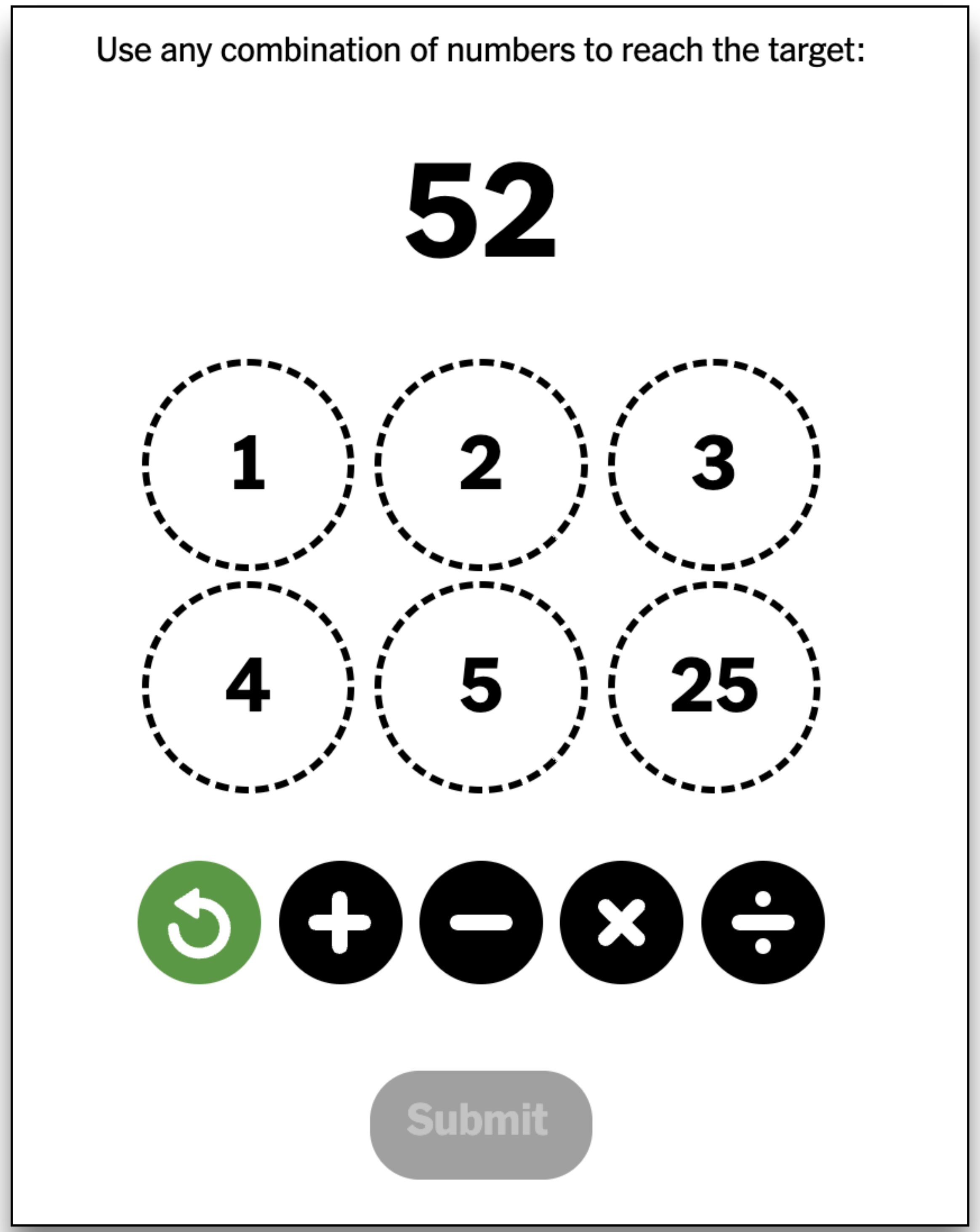
Part IV

ChatGPT

Can Chat GPT solve this problem, or generate an algorithm to solve this problem?

Part IV: ChatGPT

General Algorithm



1. You are given six numbers and a target number.

2. The goal is to reproduce the target number from any combination from the set of six numbers using the operations addition, subtraction, division, and multiplication.

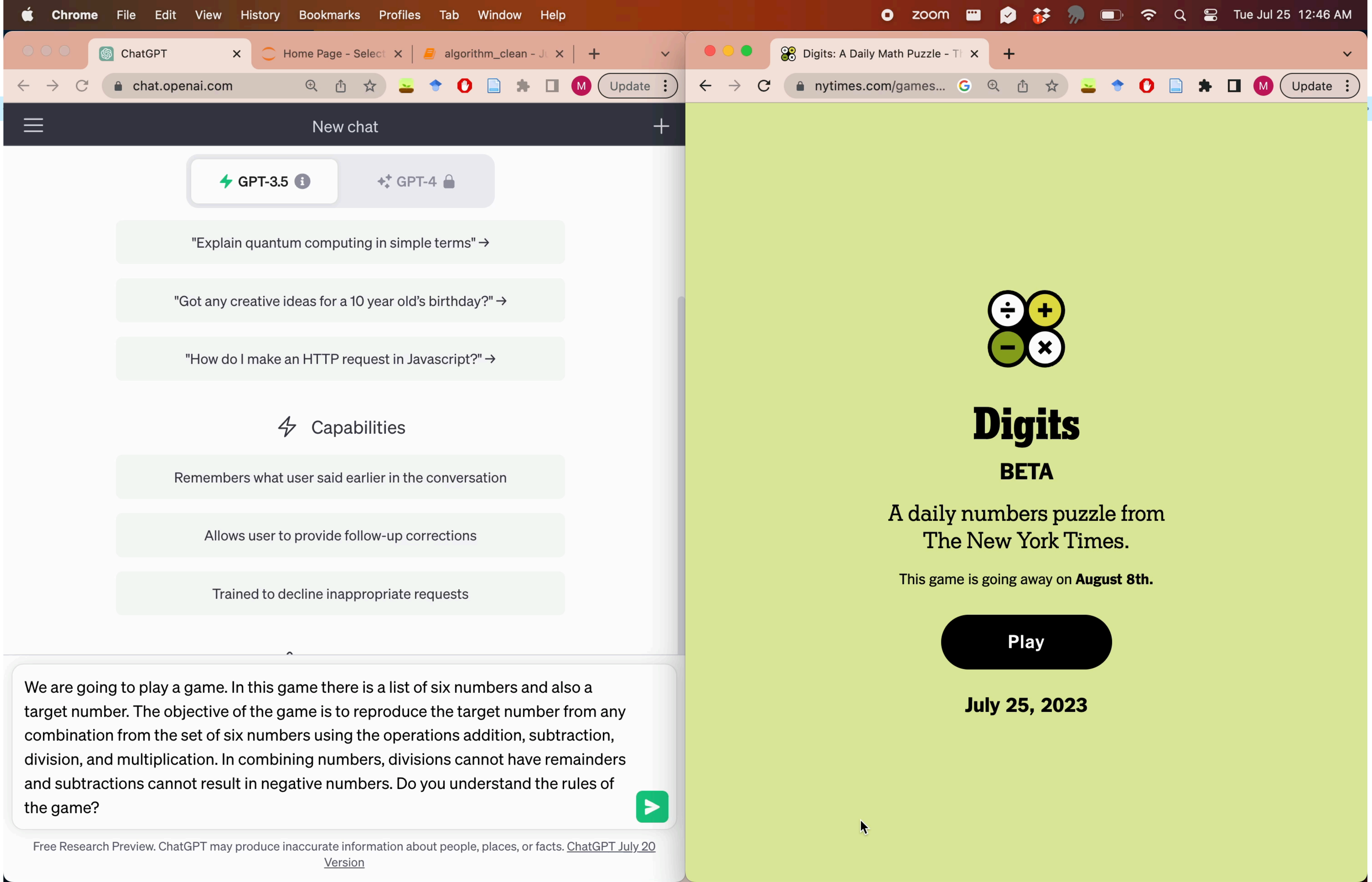
3. In combining numbers, divisions cannot have remainders and subtractions cannot result in negative numbers.

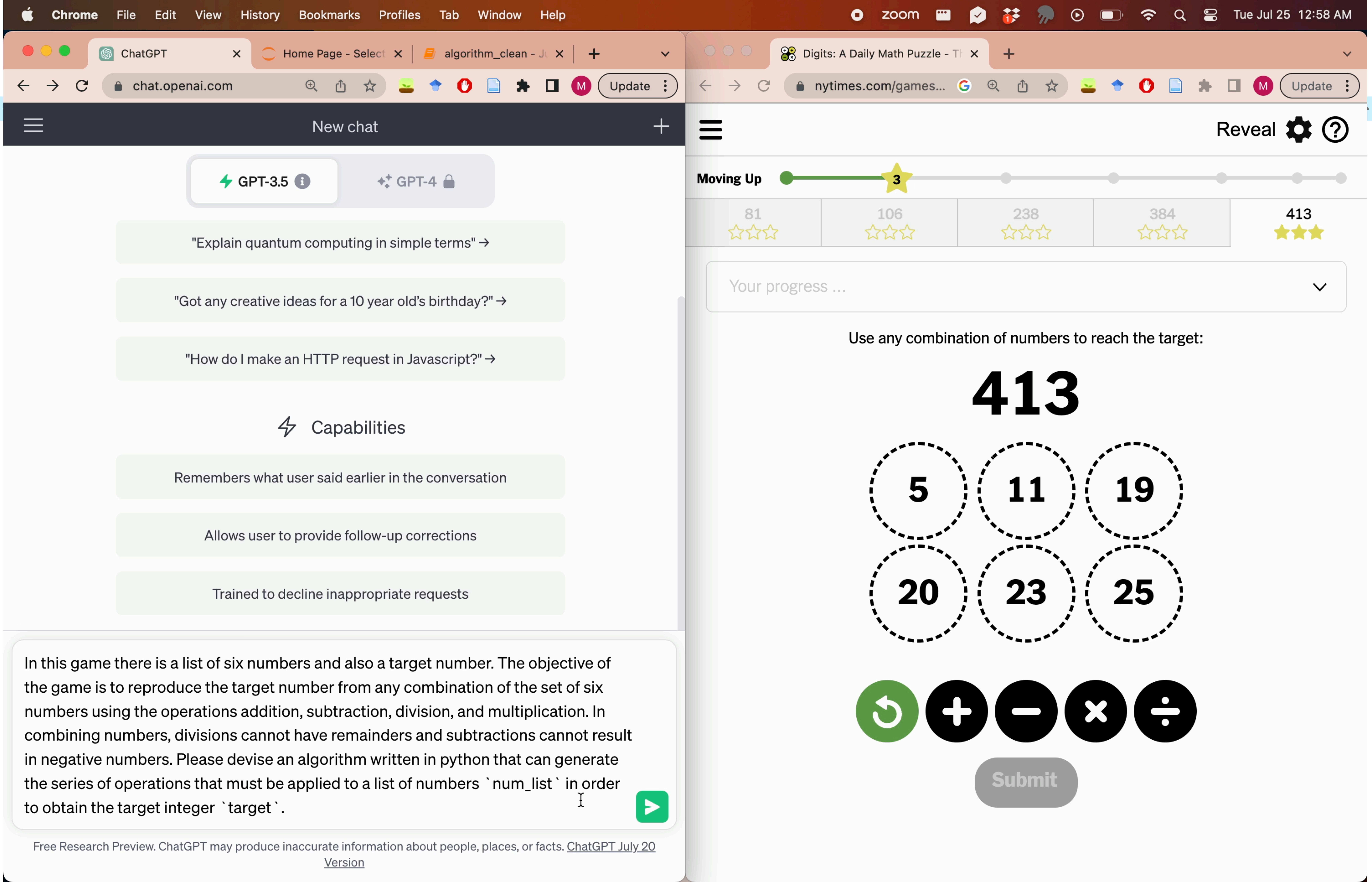
Part IV

ChatGPT

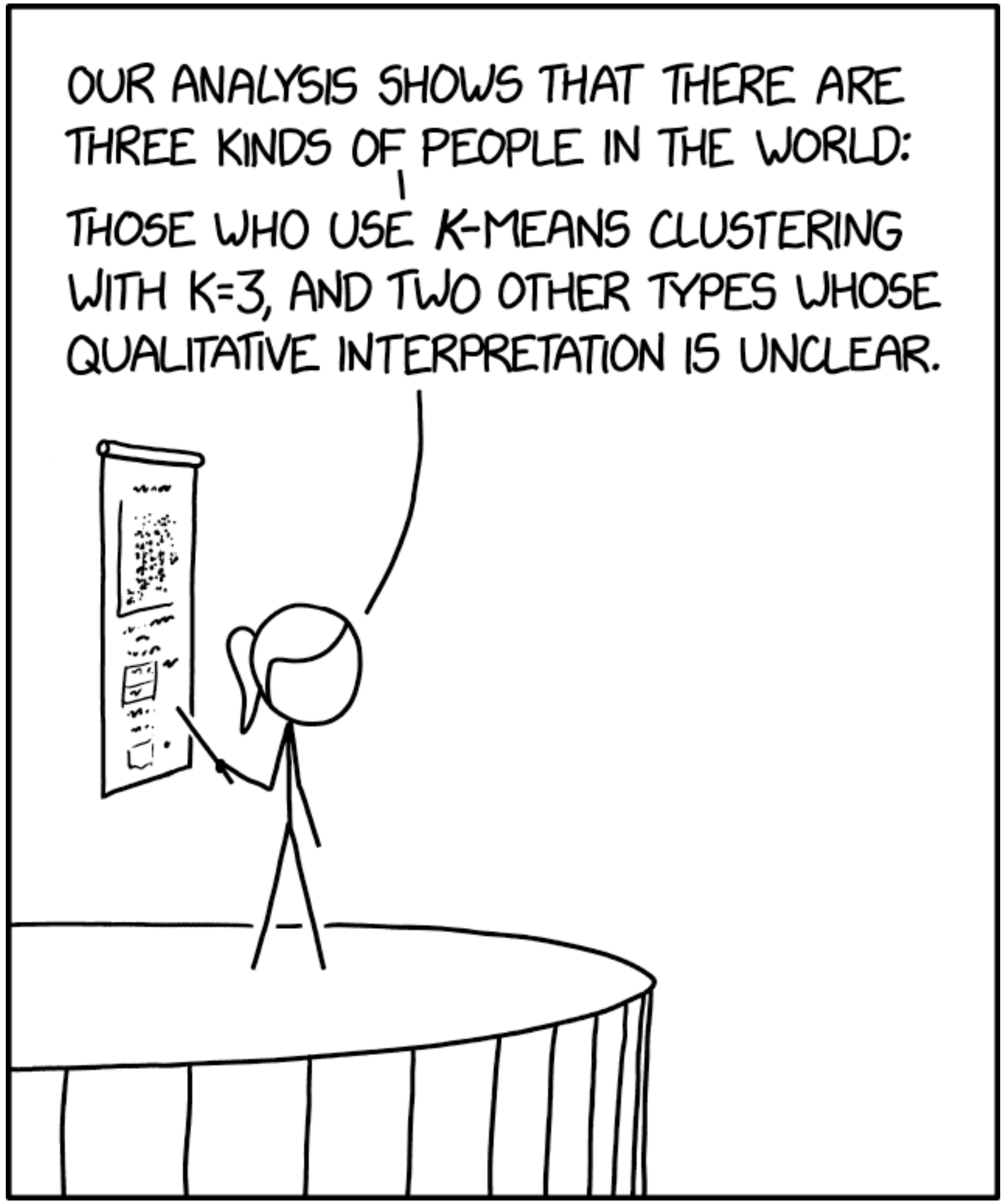
Can Chat GPT solve this problem, or generate an algorithm to solve this problem?

DEMO


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue Jul 25 12:46 AM
ChatGPT
Home Page - Select
algorithm_clean - J
chat.openai.com
Update
New chat
GPT-3.5
GPT-4
"Explain quantum computing in simple terms" →
"Got any creative ideas for a 10 year old's birthday?" →
"How do I make an HTTP request in Javascript?" →
Capabilities
Remembers what user said earlier in the conversation
Allows user to provide follow-up corrections
Trained to decline inappropriate requests
We are going to play a game. In this game there is a list of six numbers and also a target number. The objective of the game is to reproduce the target number from any combination from the set of six numbers using the operations addition, subtraction, division, and multiplication. In combining numbers, divisions cannot have remainders and subtractions cannot result in negative numbers. Do you understand the rules of the game?
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT July 20 Version
Digits: A Daily Math Puzzle - T
nytimes.com/games...
Update
Digits
BETA
A daily numbers puzzle from The New York Times.
This game is going away on August 8th.
Play
July 25, 2023


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Tue Jul 25 12:58 AM
ChatGPT
Home Page - Select
algorithm_clean - J
chat.openai.com
Update
New chat
GPT-3.5
GPT-4
"Explain quantum computing in simple terms" →
"Got any creative ideas for a 10 year old's birthday?" →
"How do I make an HTTP request in Javascript?" →
Capabilities
Remembers what user said earlier in the conversation
Allows user to provide follow-up corrections
Trained to decline inappropriate requests
In this game there is a list of six numbers and also a target number. The objective of the game is to reproduce the target number from any combination of the set of six numbers using the operations addition, subtraction, division, and multiplication. In combining numbers, divisions cannot have remainders and subtractions cannot result in negative numbers. Please devise an algorithm written in python that can generate the series of operations that must be applied to a list of numbers `num_list` in order to obtain the target integer `target`.
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT July 20 Version
Digits: A Daily Math Puzzle - T
nytimes.com/games...
Update
Reveal
Moving Up
3
81
106
238
384
413
Your progress ...
Use any combination of numbers to reach the target:
413
5
11
19
20
23
25
Submit

Requisite XKCD comic



https://xkcd.com/2731/

Thanks for listening!

References

https://mobowill.com/nyt-digits-puzzle-solution-algorithm/

https://mobowill.com/nyt-digits-puzzle-mathamtics/

https://math.mit.edu/~rstan/transparencies/miami_catalan.pdf

[JFR Research Day] Programmatic Solution to NYT Connections