Summation Formula For the Geometric Progression and Its Applications

Iydon Liang

SUSTech and Mathlang Organization[*]

October 31, 2019

Abstract

We introduce the geometric progression through an ancient story called *The rice and the chessboard*, then we give a proper and accurate definition of geometric progression, from which we derive the summation formula for the geometric progression in both finite and infinite cases. Furthermore, we list some useful applications of the geometric progression and implement them in `Python`.

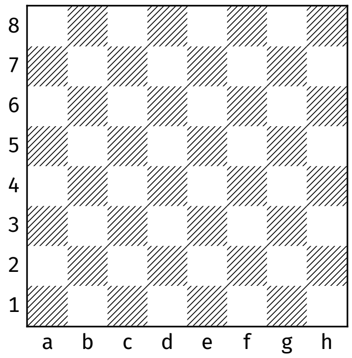
[*] `https://github.com/mathlang-org`

Contents

1 Introduction

There is a story about geometric progression called *The rice and the chessboard*[1]:

> There was once a king in India who was a big chess enthusiast and had the habit of challenging wise visitors to a game of chess. One day a traveling sage was challenged by the king. The sage having played this game all his life all the time with people all over the world gladly accepted the king's challenge. To motivate his opponent the king offered any reward that the sage could name. The sage modestly asked just for a few grains of rice in the following manner: the king was to put a single grain of rice on the first chess square and double it on every consequent one. The king accepted the sage's request.



> Having lost the game and being a man of his word the king ordered a bag of rice to be brought to the chessboard. Then he started placing rice grains according to the arrangement: 1 grain on the first square, 2 on the second, 4 on the third, 8 on the fourth and so on.

Let us just calculate how many grains of rice did the king need to be a man of his word,

$$\begin{aligned}&1+2+2^2+\cdots+2^{63}\\=&2^0+2^1+2^2+\cdots+2^{63}\\=&\sum_{n=0}^{63}2^n\end{aligned}$$

Without calculator, it will be a boring process to calculate the result. Fortunately, we live in the age of computers, with the help of `Python`, we can easily get the result through a for-loop: 18,446,744,073,709,551,615.

That is to say, the amount of grains the king needs is equal to about 210 billion tons and is allegedly sufficient to cover the whole territory of India with a meter thick layer of rice[1].

In fact, we can easily get the result through the summation formula for geometric progression, and calculating rice is equivalent to the summation of the first n terms in a geometric progression. Popularly speaking, a geometric progression is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number, we will give a mathematical definition and derivation process in the next section.

2 The Derivation Process

In this section, we will give a proper and accurate definition of geometric progression, from which we will derive the summation formula for the geometric progression in both finite and infinite cases.

2.1 Definition

Definition 1 ▸ Geometric Progression[2]

A geometric sequence is a sequence $\{a_k\}, k = 0, 1, \ldots$, such that each term is given by a multiple r of the previous one. Another equivalent definition is that a sequence is geometric iff[a] it has a zero series bias. If the multiplier is r, then the kth term is given by

$$a_k = ra_{k-1} = r^2 a_{k-2} = \cdots = a_0 r^k.$$

Taking $a_0 = 1$ gives the simple special case

$$a_k = r^k.$$

[a] if and only if

Accordingly, we can define the arithmetic progression in advance to facilitate future introductions.

[1] A grain of rice is approximately 0.2 inches long. Converting 0.2 inches to feet.

Definition 2 ▸ Arithmetic Progression[3]

An arithmetic progression, also known as an arithmetic sequence, is a sequence of n numbers $\{a_0+kd\}_{k=0}^{n-1}$ such that the differences between successive terms is a constant d. That is,

$$a_k = a_{k-1} + d = a_{k-2} + d = \cdots = a_0 + kd.$$

Taking $a_0 = 0$ gives the simple special case

$$a_k = kd.$$

2.2 Derivation

2.2.1 Summation Formula For the Geometric Progression

According to the definition of the geometric progression, we have,

$$S_n = \sum_{k=1}^{n} a_k = \sum_{k=1}^{n} a_1 r^{k-1}. \tag{1}$$

Multiply both sides of the equation by r, then we have,

$$rS_n = \sum_{k=1}^{n} a_1 r^k = \sum_{k=1}^{n} a_{k+1} = \sum_{k=2}^{n+1} a_k. \tag{2}$$

Subtract equation (2) from equation (1), we have,

$$(1 - r)S_n = a_1 - a_{n+1} = a_1(1 - r^n). \tag{3}$$

Therefore, we need a classification discussion, if the r equals to 1, both sides of the equation cannot divide $1 - r$ simultaneously, but we can derivate the formula from equation (1), $S_n = na_1$.

Above all, we derivate the summation formula for the geometric progression,

$$S_n = \begin{cases} na_1 & \text{if } r = 1, \\ \frac{a_1(1-r^n)}{1-r} & \text{if } r \neq 1. \end{cases} \tag{4}$$

Moreover, if we use the form of the limit, we can simplify the formula,

$$S_n = \lim_{q \to r} \frac{a_1(1 - q^n)}{1 - q}. \tag{5}$$

Next we consider what happens when $n \to \infty$. $1 - q$ is finite, the convergence of S_n is equivalent to convergence of $\lim_{q \to r} 1 - q^n$. Therefore, S convergence if and only if $|r| < 1$.

$$S = \lim_{n \to \infty} S_n = \lim_{\substack{n \to \infty \\ q \to r}} \frac{a_1(1 - q^n)}{1 - q} = \lim_{n \to \infty} \frac{a_1(1 - r^n)}{1 - r} = \frac{a_1}{1 - r}. \tag{6}$$

In summary, we get the summation formula for the geometric progression,

$$S_n = \begin{cases} na_1 & \text{if } r = 1, \\ \frac{a_1(1-r^n)}{1-r} & \text{if } r \neq 1. \end{cases}$$

and S convergence if and only if $|r| < 1$,

$$S = \lim_{n \to \infty} = \frac{a_1}{1 - r}.$$

2.2.2 Summation Formula For the Arithmetic progression

The summation formula for the arithmetic progression is slightly different from the geometric progression, and it relates back to a famous mathematician, Gauss. The main idea of the formula comes from

$$1 + 2 + \cdots + n = \frac{n(n + 1)}{2}.$$

The proof part is left to readers as a practice question, we just provide clues below,

$$\begin{aligned} S_{n+1} &= \sum_{k=0}^{n} a_k \\ &= \sum_{k=0}^{n} (a_0 + kd) \\ &= \sum_{k=0}^{n} a_0 + d \sum_{k=0}^{n} k. \end{aligned} \tag{7}$$

3 Applications

3.1 Computer Field

In the field of computers, the summation formula for the geometric progression is quite important for reducing the computing complexity. For example, we can reduce the summation complexity of the arithmetic progression from $O(n)$ to $O(1)$, and the geometric progression from $O(n^2)$ to $O(n)$, which maximizes memory utilization. You may refer to appendix A for more information. Here comes an example written in `IPython`[4]:

Comparison

```
base = 2
number = 1024

%%timeit
s = 0
for i in range(number):
    s += base**i

%%timeit
s = sum(base**i for i in range(number))

%%timeit
s = base**number - 1
```

The running time of the first block is $656\mu s \pm 10.4\mu s$ per loop, the second block is $628\mu s \pm 15.8\mu s$, and the third block is $765ns \pm 9.5ns$ per loop. Obviously, the formula works efficiently.

4 Acknowledgments

We are grateful to Dr. Vian Lee of Shenzhen University for many useful discussions. And this work was supported in part by *Mathematics for Beginners*.

5 References

References

[1] DEJAN. The rice and the chessboard story — the power of exponential growth [EB/OL]. 2018. https://purposefocuscommitment.com/the-rice-and-the-chess-board-story/.

[2] WEISSTEIN, W. E. Geometric sequence[EB/OL]. 2019. http://mathworld.wolfram.com/GeometricSequence.html.

[3] WEISSTEIN, W. E. Arithmetic progression[EB/OL]. 2019. http://mathworld.wolfram.com/ArithmeticProgression.html.

[4] PÉREZ F, GRANGER B E. IPython: a system for interactive scientific computing[J/OL]. Computing in Science and Engineering, 2007, 9(3):21-29. https://ipython.org. DOI: 10.1109/MCSE.2007.53.

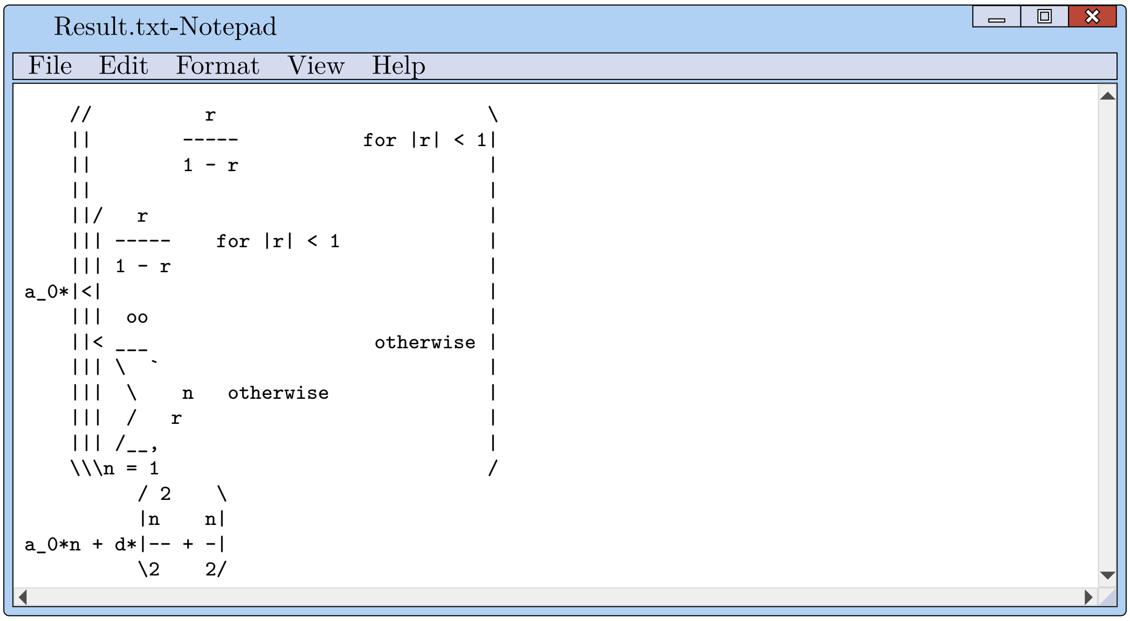
[5] 汤涛, 丁玖. 数学之英文写作[M]. 北京: 高等教育出版社, 2013.

A Python Code

Sum of Progression

```python
# -*- encode: utf-8 -*-
from sympy import symbols, Sum, oo, pretty


n = symbols('n', integer=True)
a_0 = symbols('a_0')


def sum_progression(a_n, end):
    r'''\sum^{end}_{n=1}␣a_n
␣␣␣␣'''
    expr = Sum(a_n, (n, 1, end))
    return expr.doit()


def sum_geometric_progression(end=oo):
    r'''\sum^{end}_{n=1}␣a_0␣r^n
␣␣␣␣'''
    r = symbols('r')
    a_n = a_0 * r**n
    return sum_progression(a_n, end)


def sum_arithmetic_progression(end=n):
    r'''\sum^{end}_{n=1}␣a_0␣+␣nd
␣␣␣␣'''
    d = symbols('d')
    a_n = a_0 + n*d
    return sum_progression(a_n, end)


def echo_expression_without_unicode(expr, echo=print):
    '''sympy.pretty_print
    '''
    result = pretty(expr,
        use_unicode=False,
        use_unicode_sqrt_char=False)
    echo(result)
```

```python


if __name__ == '__main__':
    expr_gp = sum_geometric_progression()
    echo_expression_without_unicode(expr_gp)

    expr_ap = sum_arithmetric_progression()
    echo_expression_without_unicode(expr_ap)
```



```
    //          r                        \
    ||        -----           for |r| < 1|
    ||        1 - r                      |
    ||                                   |
    ||/   r                              |
    ||| -----    for |r| < 1             |
    ||| 1 - r                            |
a_0*|<|                                  |
    |||  oo                              |
    ||< ___                    otherwise |
    ||| \  `                             |
    |||  \    n   otherwise              |
    |||  /   r                           |
    ||| /__,                             |
    \\\n = 1                             /
          / 2    \
          |n    n|
a_0*n + d*|-- + -|
          \2    2/
```