Indian Journal of History of Science (2023) 58:96–102
https://doi.org/10.1007/s43539-023-00082-4

ARTICLE

Recursion and iteration in combinatorics of *Chandaśśāstra*

Amba Kulkarni[1]

Received: 6 October 2022 / Accepted: 24 March 2023 / Published online: 24 July 2023

Abstract
Piṅgaḷa in his book on *Chandaśśāstra*, a text related to the description and analysis of meters in poetic work, describes algorithms that deal with Combinatorial Mathematics and are tail-recursive in nature. Later after almost a millennium in around 800 CE, Kedāra Bhaṭṭa provides iterative algorithms for the same operations. Another major difference between the two works is stylistic. Piṅgaḷa uses a cryptic style of *sūtras* while Kedāra Bhaṭṭa uses a verse style. The purpose of this paper is to highlight the methodological differences between Piṅgaḷa's algorithms and the corresponding algorithms of Kedāra Bhaṭṭa. We look at the algorithms described by both scholars, and express them in modern mathematical notation or in algorithmic style in order to understand the differences. Recursive algorithms are easy to conceptualise. However, the iterative algorithms are easy from the learner's point of view. The transition from *sūtras* to verses and from recursion to iteration in the later period might be due to pedagogy.

Keywords Recursion · Iteration · Piṅgaḷa · Kedāra Bhaṭṭa · *Chandaśśāstra* · Indian Mathematics

1 Introduction

The discovery of a binary number system by Indians escaped the attention of Western scholars, may be because *Chandaśśāstra* was considered mainly a text related to the description and analysis of meters in poetic literary work, totally unrelated to mathematics. Only recently in the twentieth century India's contribution to combinatorics was brought to the limelight by various scholars such as Ludwig (1991), Nooten (1993), and Sridharan (2005) to name a few.

Chandaśśāstra by Piṅgaḷa is the earliest treatise found on the Vedic Sanskrit meters. Piṅgaḷa defines different meters on the basis of a sequence of what are called *laghu* and *guru* (short and long) syllables and their count in the verse. The description and analysis of the sequence of the *laghu* and *guru* syllables in a given verse is the major topic of Piṅgaḷa's work. He has described different sequences that can be constructed with a given number of syllables and has also named them. At the end of his book on *Chandaśśāstra*, Piṅgaḷa gives rules to list all possible combinations of *laghu* and *guru* (L and G) in a verse with n syllables, rules to find out the *laghu-guru* combinations corresponding to a given index, the total number of possible combinations of n L-G syllables, and so on. In short Piṅgaḷa describes the 'combinatorial mathematics' of meters in *Chandaśśāstra*. The six operations Piṅgaḷa describes are termed *prastāra, naṣṭa, uddiṣṭa, eka-dvi-ādi-la-ga-kriyā, saṅkhyā*, and *adhvayoga*, and collectively these are termed as *pratyayas*.[1] Later around the eighth century CE Kedāra Bhaṭṭa wrote *Vṛttaratnākara* a work on non-vedic meters. This seems to be independent of Piṅgaḷa's work, in the sense that it is not a commentary on Piṅgaḷa's work, and the last chapter gives the rules related to combinatorial mathematics which are totally different from Piṅgaḷa's approach.

In what follows we take up the first five *pratyaya*s and explain the corresponding *sūtra* from Piṅgaḷa's *Chandaśśāstra* and the verses from Kedāra Bhaṭṭa's *Vṛttaratnākara*, and highlight how Piṅgaḷa's procedures are tail-recursive while those of Kedāra Bhaṭṭa's are iterative in nature. The sixth one viz. *adhvayoga* has the same treatment in both texts, and hence we do not cover it here.

✉ Amba Kulkarni
ambakulkarni@uohyd.ac.in

1 Department of Sanskrit Studies, University of Hyderabad, Hyderabad 500046, India

[1] प्रस्तारो नष्टमुद्दिष्टम् एकद्व्यादिलगक्रिया ।
सङ्ख्यानमध्वयोगश्च षडेते प्रत्ययाः स्मृताः ॥(६.१)॥
prastāro naṣṭamuddiṣṭam ekadvyādilagakriyā |
saṅkhyānamadhvayogaśca ṣaḍete pratyayāḥ smṛtāḥ ‖ *(6.1)* ‖.

Table 1 Mixed with G

G	G
L	G

Table 2 Mixed with L

G	L
L	L

2 *Prastāraḥ*

Literally *prastāraḥ* means expansion or a spread. In the present context, it means sequential enumeration of all possible permutations of *laghu* and *guru*.

The *sūtra*s[2] in Piṅgaḷa's *Chandaśśāstra* are as follows:

द्विकौ ग्लौ ||८.२०||
मिश्रौ च ||८.२१||
पृथग्ला मिश्राः ||८.२२||
वसवस्त्रिकाः ||८.२३||

dvikau glau ||8.20||
miśrau ca ||8.21||
pṛthaglā miśrāḥ ||8.22||
vasavastrikāḥ ||8.23||

G and L are two letters (8.20)
(They are) also combined (8.21)
G and L are combined separately (8.22)
There are eight triplets (8.23)

2.1 Explanation

Now let us try to understand these *sūtras*. There is some redundancy since we now interpret each sūtra word by word.

1. Skt: द्विकौ ग्लौ ||८.२०||
   Skt: ***dvikau glau*** ||8.20||
   Gloss: consisting_of_two g_and_l.
   Eng: G and L are two (letters).

   This *sūtra* means *prastāra* of 'one syllable(*akṣara*)' has two possible elements viz. 'G (*ga*) and L (*la*)'.

   $$\begin{bmatrix} G \\ L \end{bmatrix}$$

2. Skt: मिश्रौ च ||८.२१||
   Skt: ***miśrau ca*** ||8.21||
   Gloss: combined also.
   Eng: (they are) also combined.

   It is not clear what they are to be combined with. This is clear from the next *sūtra*.

3. Skt: पृथग्ला मिश्राः ||८.२२||
   Skt: ***pṛthaglā miśrāḥ*** ||8.22||
   Gloss: Separately_g_l are_combined.
   Eng: G and L are combined separately.

   Thus, 'G-L' is mixed with 'G' followed by 'L' (Tables 1 and 2). This results in the permutations shown in Table 3.

4. Skt: वसवस्त्रिकाः ||८.२३||
   Skt: ***vasavastrikāḥ*** ||8.23||
   Gloss: *Vasus* triplets.
   Eng: Triplets are eight.

   Here the term *vasu*, a *bhūtasaṅkhyā*, refers to the number 8. Indeed, when the above *prastāra* of two G-Ls are mixed with one more, it results into eight permutations, as shown in Table 4.

Table 3 Two syllable combinations

G	G
L	G
G	L
L	L

Table 4 Three syllable *prastāra*

G	G	G
L	G	G
G	L	G
L	L	G
G	G	L
L	G	L
G	L	L
L	L	L

Thus the first of the four *sūtras* is for initialisation. The second and the third *sūtras* together tell how to get the permutations of a pair of G-Ls from that of single G-Ls. The fourth *sūtra* states the size of the permutations of a triplet of G-Ls, and that's all. Though it is not explicitly mentioned by any word but is inferred from the plural usage of the words in the third *sūtra* that this process (the second and the third *sūtras* together) is to be repeated to get permutations of higher order.

2.2 Recursion in *prastāra*

Let us now represent this in modern Mathematical notation. We represent the matrix in step 1 as

[2] The *sūtra*s are numbered with two parts separated by a '.'. The integer part denotes the chapter number, and the fractional part denotes its index in the chapter.

$$A^1_{2*1} = \begin{bmatrix} G \\ L \end{bmatrix}$$

Then the matrix in step 2 is

$$A^2_{4*2} = \begin{bmatrix} G & G \\ L & G \\ G & L \\ L & L \end{bmatrix} = \begin{bmatrix} A^1_{2*1} & G_{2*1} \\ A^1_{2*1} & L_{2*1} \end{bmatrix}$$

where G_{m*n} is an $m * n$ matrix with all elements equal to G and L_{m*n} is an $m * n$ matrix with all elements equal to L.

Continuing further, the matrix in step 3 is

$$A^3_{8*3} = \begin{bmatrix} A^2_{4*2} G_{4*1} \\ A^2_{4*2} L_{4*1} \end{bmatrix}$$

The generalisation of this leads to

$$A^n_{2^n*n} = \begin{bmatrix} A^{n-1}_{2^{n-1}*(n-1)} G_{2^{n-1}*1} \\ A^{n-1}_{2^{n-1}*(n-1)} L_{2^{n-1}*1} \end{bmatrix}$$

We notice that the algorithm for the generation of all permutations of n G-Ls given by Piṅgaḷa is thus recursive.

2.3 Kedāra Bhaṭṭa's algorithm for *prastāraḥ*

Kedāra Bhaṭṭa in his *Vṛttaratnākara* has given a different procedure to get the *prastāra* for a given number of G-Ls. His procedure goes like this:

Table 5 Kedāra Bhaṭṭa's procedure

G	G	G	Start with all Gs
L	G	G	Place L below the first G of the above line, copy the remaining GLs to the right from the above line.
G	**L**	G	Place L below the first G of the above line, copy the remaining GLs to the right from the above line, and place G in the remaining places to the left of L.
L	L	G	Place L below the first G of the above line, copy the remaining GLs to the right from the above line.
G	G	**L**	place L below the first G of the above line and Gs in the remaining places to the left.
L	G	L	Place L below the first G of the above line, copy the remaining GLs to the right from the above line.
G	**L**	L	Place L below the first G of the above line, copy the remaining GLs to the right from the above line, and place G in the remaining place to the left.
L	L	L	Place L below the first G of the above line, copy the remaining GLs to the right from the above line, and stop the process since all are Ls.

पादे सर्वगुरावाद्यात् लघुं न्यस्य गुरोरधः ।
यथोपरि तथा शेषं भूयः कुर्यादमुं विधिम् ॥ (६.२) ॥
ऊने दद्यात् गुरूनेव यावत् सर्वलघुर्भवेत् ।
प्रस्तारोऽयं समाख्यातः छन्दोविचितिवेदिभिः ॥ (६.३) ॥

pāde sarvagurāvādyāt laghuṃ nyasya guroradhaḥ |
yathopari tathā śeṣaṃ bhūyaḥ kuryādamuṃ vidhim ‖ *(6.2)* ‖
ūne dadyāt gurūneva yāvat sarvalaghurbhavet |
prastāro'yaṃ samākhyātaḥ chandovicitivedibhiḥ ‖ *(6.3)* ‖

"[In the beginning], in a line (*pāde*) all are *gurus*(G) (*sarvagurau*). [In the second line], place (*nyasya*) an L (*laghu*) below (*adhaḥ*) the first G (*ādyāt guroḥ*) [of the previous line]. [Copy the] remaining (*śeṣam*) [right ones] as in the above line(*yathā-upari tathā*). Place (*dadyāt*) only Gs (*gurūn eva*) in all the remaining places (*ūne*) [to the left (if any) of the first G]. Repeat this process (*bhūyaḥ kuryāt amuṃ vidhim*) till all of them become laghu(*yāvat sarve laghuḥ bhavet*). This (*ayaṃ*) [process] is known (*samākhyātaḥ*) as *prastāra* by the experts in the *chanda*s (*chandoviciti vedibhiḥ*)."

This method, as we notice, is iterative in nature. Table 5 provides an illustration with a triplet, explaining the above procedure.

3 *Naṣṭam*

Naṣṭa literally means lost or vanished or disappeared. In ancient days sand was used as a medium for writing and hence there was a possibility of a row getting erased by the wind. Let us now see the Piṅgaḷa's and Kedāra Bhaṭṭa's procedures for recovering the lost rows in the *prastāra*.

3.1 Piṅgaḷa's *sūtras* for *Naṣṭam*

The *sūtra*s are as follows:

लर्द्धे ॥ (८.२४)॥
स-एके ग् ॥ (८.२५)॥
larddhe ‖ (8.24)‖
sa-eke g ‖ (8.25)‖

If (the given number) can be halved, (then write) L (8.24)
If after adding one (to the given number, it can be halved) (then write) G (8.25)

Now let us see the word by word meaning for these *sūtras*.

Skt: ल् अर्द्धे(८.२४)
Skt: *l-arddhe*
Gloss: *l* if_can_be_halved.
Eng: If (the given number) can be halved, (then write) L.
Skt: स-एके ग् (८.२५)

Table 6 *Naṣṭam*

5	$\rightarrow$	$\frac{5+1}{2}=3$	G
3	$\rightarrow$	$\frac{3+1}{2}=2$	G G
2	$\rightarrow$	$\frac{2}{2}=1$	G G L

Skt: *sa-eke g*
Gloss: if_after_adding_one *g*.
Eng: If after adding one (to the given number, it can be halved) (then write) G.

For example, suppose the fifth row in the expansion of triplets is lost. We start with the number 5 (See Table 6). Since it is an odd number, we add 1 to it and write 'G'. After dividing 5+1(=6) by 2, we get 3. Again this is an odd number, and hence we add 1 to it, and write 'G'. After dividing 3+1(=4) by 2 we get 2. Since it is an even number we write 'L'. After dividing 2 by 2, we get 1. And we terminate the process here, as we have obtained the three G-Ls. Piṅgaḷa does not specify when to terminate the process. It is indeed not needed since the assumption is that the process will continue till one gets the desired number of G-Ls.

So the fifth row in the *prastāra* with three letters is G G L. The algorithm may be written as a recursive function as below:

Get_Binary(n) =
Print L; Get_Binary(n/2), if n is even,
Print G; Get_Binary(n+1/2), if n is odd,
Print G; if n=1. (Terminating condition).

We have named this function Get_Binary since, in fact, this procedure provides a binary equivalent of a decimal number. However, there is a small deviation from the binary representation.

3.2 Deviation from binary representation

The binary equivalent of 5 is 101. If we replace G by 0 and L by 1 in 'GGL' we get 001. Since the numbers are written with higher place value digits to the left of the lower place value digits, we take the mirror image. The mirror image of '001' is '100'. Thus, by Piṅgaḷa's method we get the equivalent of 5 as '100' which is the previous number of '101'. This difference of 1 is attributed to the fact that the counting in Piṅgaḷa's method starts with '1'.

Thus we notice two major differences between the Piṅgaḷa's representation and the modern representation of binary numbers. In Piṅgaḷa's system,

- as has been initially observed by Nooten (1993), the numbers are written with the higher place value digits to the right of lower place value digits, and
- the counting starts with 1.

3.3 Kedāra Bhaṭṭa on *Naṣṭam*

If one knows Kedāra Bhaṭṭa's procedure for *prastāra* then the lost row can be recovered easily from the previous or the next one. Nevertheless, he has given an algorithm to recover the lost row. The verse is

नष्टस्य यो भवेदङ्कः तस्यार्धे च समे च लः ।
विषमे चैकमाधाय तदर्धे च गुरुर्भवेत् ॥ (६.४) ॥
naṣṭasya yo bhavedaṅkaḥ tasyārdhe ca same ca laḥ |
viṣame caikamādhāya tadardhe ca gururbhavet ‖ *(6.4)* ‖

"Whatever (*yo*) is (*bhavet*) the missing (*naṣṭasya*) number(*aṅkaḥ*), if it can be halved (*tasya ardhe same*), half it and write L(*laḥ*), if the number is odd(*viṣame*), adding 1(*caikam*) to it, and after halving it(*tadardhe*), becomes(*bhavet*) G(*guru*)."

This procedure is just a versification of the procedure due to Piṅgaḷa.

4 *Uddiṣṭam*

The third algorithm is to obtain the position of the desired (*uddiṣṭa*) row in a given *prastāra*, without counting its position from the top, i.e. to get the row index corresponding to a given permutation of G and Ls. Thus this is the inverse operation of *naṣṭam*.

4.1 Piṅgaḷa's sūtras for *uddiṣṭam*

Two *sūtra*s viz. (8.26) and (8.27) from Piṅgaḷa's *Chandaśśāstra* describe this algorithm. The *sūtra*s are as follows:

प्रतिलोमगुणं द्विर्लाद्यम् ॥ (८.२६) ॥
ततो ग्येकंजह्यात् ॥ (८.२७) ॥
pratilomaguṇaṁ dvirlādyam ‖ *(8.26)* ‖
tato gyekaṁjahyāt ‖ *(8.27)* ‖

Multiply by 2 in the reverse order,
starting with L. (8.26)
If it is G, (after multiplying by 2)
subtract one from it. (8.27)

Skt: प्रतिलोमगुणं द्विः ल्-आद्यम् ||(८.२६)||
Skt: *pratilomaguṇaṃ dviḥ l-ādyam*
Gloss: in the reverse order, multiply, by two, starting with *la*.
Eng: Multiply by 2 in the reverse order, starting with L.

Skt: ततः गि-एकं जह्यात् ||(८.२७)||
Skt: *tataḥ gi ekaṁ jahyāt*
Gloss: from that, in the case of G, one, subtract.

Eng: If it is G, (after multiplying by 2) subtract one from it.

The word *dviḥ* is not repeated in the second line, but is borrowed from the previous line in order to understand the second line. Since it is not mentioned what the starting number would be, and the operation of multiplication is involved, we start with the multiplicative identity viz. 1.

The algorithm may be written formally as below.

Let S_i denote the position of the sequence of *i* G-Ls to the right in the *prastāra*.

S_i	= 1 if the first *laghu* occurs in the i^{th} position from the right.
S_{i+1}	= 2 * S_i if $(i + 1)^{th}$ position from the right has L,
	= 2 * S_i - 1 if $(i + 1)^{th}$ position from the right has G,

We illustrate this algorithm with an example. Let the input sequence be 'G L G'. Table 7 describes the application of the above *sūtras*.

Thus the row 'G L G' is in the third position in the expanded triplets of G-Ls.

4.2 Kedāra Bhaṭṭa's verse for *uddiṣṭam*

Kedāra Bhaṭṭa's version of *uddiṣṭam* differs from that of Piṅgaḷa. His version goes like this:

उद्दिष्टं द्विगुणानाद्यात् उपर्यङ्कान् समालिखेत् |
लघुस्था ये तु तत्राङ्काः तैः सैकैर्मिश्रितैर्भवेत् ॥ (६.५) ॥
uddiṣṭaṁ dviguṇānādyāt uparyaṅkān samālikhet |
laghusthā ye tu tatrāṅkāḥ taiḥ saikairmiśritairbhavet ‖ (6.5) ‖

" Starting from the beginning (*ādyāt*) write the numbers (*aṅkān samālikhet*) double (*dviguṇān*) [the previous one] on the top (*upari*) [of each *laghu-guru*]. [Then all] those (*ye*) numbers (*aṅkāḥ taiḥ*) which are (*ye tu*) on top of laghu(*laghusthā*) added (*miśritaiḥ*) with 1 (*sa-ekaiḥ*) becomes (*bhavet*) the row number corresponding to the given laghu guru combination (*uddiṣṭam*)"

Since the starting number is not mentioned, we start with 1, the multiplicative identity, since the operation involved is that of multiplication.

We illustrate this with an example.
Let the row be 'G L L'.

We start with 1, write it on top of the first G. Then multiply it by 2, and write 2 on the top of L, and similarly (2*2=) 4 on top of the last L. Then we add all the numbers which are on top of L viz. 2 + 4. To this add 1. This results in 7. Hence the row index of the given L-G combination is 7 (Table 8).

Table 7 *Uddiṣṭam*

G	L	G	remark
	1		start with the first L from the right, with the number 1
	2		multiply it by 2
2			continue with the previous result i.e. 2
4			multiply it by 2
3			subtract 1 from it, since it is a guru.

Table 8 *Uddiṣṭam*

1	2	4
G	L	L

5 *Eka-dvi-ādi la-ga-kriyā*

The term *eka-dvi-ādi la-ga-kriyā* offers a method to get the number of combinations of 1 L, 2 L, etc. (or 1 G, 2 G, etc.) among all possible combinations of *n* L-Gs. In other words, it gives a procedure to calculate ^nC_r. Kedāra Bhaṭṭa's work describes explicit rules to get such number of combinations. Piṅgaḷa's *sūtra* is very cryptic and it is only through the Halāyudha's commentary on it, one can interpret the *sūtra* as a *meru* which resembles Pascal's triangle. We give here only the Kedāra Bhaṭṭa's algorithm since he uses it in the next *pratyaya* - *Saṅkhyā*. The Piṅgaḷa's cryptic algorithm, since is not relevant here, is skipped.

5.1 Kedāra Bhaṭṭa's algorithm

The procedure for *eka-dvi-adi-la-ga-kriyā* in *Vṛttaratnākara* is described as follows:

वर्णान् वृत्तभवान् सैकान् औत्तराधर्यतः स्थितान् |
एकादिक्रमतश्चैतान् उपर्युपरि निक्षिपेत् ॥ (६.६) ॥
उपान्त्यतो निवर्तेत त्यजेदेकैकम् ऊर्ध्वतः|
उपर्याद्यात् गुरोरेवम् एकद्व्यादिलगक्रिया ॥ (६.७) ॥
varṇān vṛttabhavān saikān auttarādharyataḥ sthitān |
ekādikramataścaitān uparyupari nikṣipet ‖ (6.6) ‖
upāntyato nivarteta tyajedekaikam ūrdhvataḥ|
uparyādyāt gurorevam ekadvyādilagakriyā ‖ (6.7) ‖

Whatever the given number of syllables is(*varṇān vṛttabhavān*), placing (*sthitān*) those many 1 s (*sa-ekān*) from the left to right as well as top to bottom (*auttarādharyataḥ*), starting from one (*ekādikramataḥ*) these (*etān*) are to be placed (*nikṣipet*) in the top (*upari-upari*). Proceed (*nivartet*) till the last but one (*upāntyataḥ*) removing (*tyajate*) one at a time(*ekaikam*) from the top (*ūrdhvataḥ*). On the top in the beginning (*upari ādyāt*) Gs(*guroḥ*), like this (*evaṁ*) *eka-dvi-ādi-la-ga-kriyā*.

We explain this algorithm with an example.

Table 9 *Meru* aka Pascal's Triangle

	1	1	1	1	1	**1**
1	2	3	4	5	**6**	
1	3	6	10	**15**		
1	4	10	**20**			
1	5	**15**				
1	**6**					
1						

Let the number be 6. Write six 1 s horizontally as well as vertically (See Table 9). Elements are populated row-wise by writing the sum of numbers in immediately preceding row and column.

The numbers in bold in Table 9 viz. 1, 6, 15, 20, 15, 6, 1 give number of combinations with all gurus, one laghu, two laghus, three laghus, four laghus, five laghus, and finally all laghus.

This process describes the method of getting the number of ways one can choose *r* from *n* things viz. ${}^{n}C_{r}$. This Pascal's triangle is termed *meru* (hill) in Indian literature.

6 *Saṅkhyā*[3]

Saṅkhyā stands for the number of possible permutations of *n* L-Gs. Piṅgaḷa and Kedāra give different methods to calculate this *saṅkhyā* for a given *n*. Kedāra Bhaṭṭa uses the results of previous operations (*uddiṣṭam* and *eka-dvi-ādi-la-ga-kriya*), whereas Piṅgaḷa describes a totally independent algorithm.

6.1 Piṅgaḷa's sūtras for *Saṅkhyā*

Piṅgaḷa describes the procedure with four *sūtras* as below.

द्विः अर्द्धे ||(८.२८)||
रूपे शून्यम् ||(८.२९)||
द्विः शून्ये ||(८.३०)||
तावदर्द्धे तद्गुणितम्||(८.३१)||

dviḥ arddhe ||(8.28)||
rūpe śūnyam ||(8.29)||
dviḥ śūnye ||(8.30)||
tāvadarddhe tadguṇitam ||(8.31)||

If the number is divisible by 2(*arddhe*), [divide it by 2 and write] 2 (*dviḥ*). If subtracting 1 (*rūpe*) [from it makes it divisible by 2 divide it by 2, and write] 0 (*śūnyam*). If the answer were 0 (*śūnye*), multiply by 2 (*dviḥ*), [and] if [the answer were] 2 (*arddhe*), multiply (*tad guṇitam*) it by self (*tāvad*).

This process of getting the *saṅkhyā* consists of two parts. In the first part, the information about the divisibility of the number by 2 is noted down as either 2 or 0, halving the number at each step. In the second step, this information is used to either multiply by 2 or by itself resulting in the desired power of 2.

So for example, consider 8. To get the *saṅkhyā* (that is the total number of entries in the *prastāra* of 8 LGs), the first step is as described below.

8
4 2 (if even, divide by 2 and write 2)
2 2 (if even, divide by 2 and write 2)
1 2 (if even, divide by 2 and write 2)
0 0 (if odd, subtract 1 and write 0).

The second column stores the information about the decomposition of 8. This information is used to get the power of 2, which corresponds to the total number of entries in the *prastāra*. Now we start from the bottom of the second column and move upwards each step to get the desired number.

0 1 * 2 = 2 (if 0, multiply by 2; we start with 1)
2 2 * 2 = 4 (if 2, multiply by itself)
2 4 * 4 = 16 (if 2, multiply by itself)
2 16 * 16 = 256 (if 2, multiply by itself)

This algorithm may be expressed formally as

$$\begin{aligned} &power_2(n) \\ &\quad = [power_2(n/2)]^2, \text{ if n is even,} \\ &\quad = power_2((n-1) * 2), \text{ if n is odd,} \\ &\quad = 1, \text{ if } n = 0. \end{aligned}$$

Note that the results after each call of the function are 'stacked' and may also be treated as 'tokens' carrying the information for the next action (whether to multiply by 2 or by itself). It still remains unclear to the author which part of the *sūtra* codes information about the 'stack'. Or, in other words, how does one know that the information-carrying tokens are to be used in the reverse order? There is no information about this in the *sūtra*s anywhere either explicit or implicit. This algorithm of calculating n^{th} power of 2 is a recursive algorithm and its complexity is $O(\log_2 n)$, whereas the complexity of calculating power by normal multiplication is O(n). Knuth (1981, p. 399) has referred to this algorithm as a 'binary method'.

6.2 Kedāra Bhaṭṭa's verses for *Saṅkhyā*

Kedāra Bhaṭṭa gives the following *śloka* in his sixth chapter of the book *Vṛttaratnākara*.

[3] alternately this is also called ***saṅkhyāna***

लगक्रियाङ्कसन्दोहे भवेत् सङ्ख्या विमिश्रिते |
उद्दिष्टाङ्कसमाहारः सैको वा जनयेदिमाम् ॥ (६.८) ॥
lagakriyāṅkasandohe bhavet saṅkhyā vimiśrite |
uddiṣṭāṅkasamāhāraḥ saiko vā janayedimām ॥ (6.8) ॥

1. The totality of the addition (*vimiśrite sandohe*) of *la-ga-kriyā-aṅka* becomes (*bhavet*) the total combinations (*saṅkhyā*).
2. Alternately (*vā*) aggregation (*samāhāraḥ*) of the numbers (*aṅka*) at the top in the *uddiṣṭa* by adding 1 (*sa-ekaḥ*) generates (*janayet*) it (*imām*).

So for example, the total possible permutations of 6 G-Ls is calculated as follows.

- The numbers in the *eka-dvi-ādi-la-ga-kriyā* are 1, 6, 15, 20, 15, 6, 1 (see Table 9). Adding these we get 1 + 6 + 15 + 20 + 15 + 6 + 1 = 64.
- The *uddiṣṭa* numbers in case of 6 G-Ls are 1, 2, 4, 8, 16, 32 and adding 1 to their sum, we get 64.

Here Kedāra Bhaṭṭa is not giving any new procedure but uses the earlier results to get the total number of rows in the spread for a given number of G-Ls. However, from these two descriptions, it is obvious that he was aware of the following two well-known formulae.

$$2^{n} = \sum_{r=0}^{n} {}^{n}C_{r}$$

(Sum of the numbers in *eka-dvi-ādi-la-ga-kriyā*)

and

$$2^{n} = \sum_{i=0}^{n-1} 2^{i} + 1$$

(sum of *uddiṣṭa* numbers +1).

7 Conclusions

Piṅgaḷa used recursion extensively to describe the algorithms. The use of stack to store the information of intermediate operations, in Piṅgaḷa's algorithms is also worth mentioning. All these algorithms use a terminating condition, ensuring that the recursion terminates. Recursive algorithms are easy to conceptualise, and implement mechanically. Further tail-recursive algorithms ensure that the process is not memory hungry. The iterative algorithms, on the other hand, are easy from the learner's point of view. They are directly executable for a given value of inputs, without the requirement of any stacking of variables. Hence the latter commentators such as Kedāra Bhaṭṭa might have used only iterative algorithms.

The *sūtra* style was prevalent in India, and unlike modern mathematics, the Indian mathematics was passed from generation to generation orally, through *sūtra*s and verses. *Sūtra*s being very brief and compact were easy to memorise and also to communicate orally. But they being cryptic, are very difficult to comprehend. The verses on the other hand are comparatively easy to comprehend. Further, the verses are set to various metrical patterns, and thus are easy to memorise, as they are rhythmic in nature. Further the verse style also provides a space to state the algorithms more lucidly than the *sūtra*s.

The conceptual lucidity and the ease to memorise might have been the two reasons for the shift from *sūtra* to verse style and from the recursion to the iterative style of the algorithms.

Acknowledgements The material in this paper was evolved while teaching a course on "Glimpses of Indian Mathematics" to the first-year students of the Integrated Masters course at the University of Hyderabad in 2007. The author acknowledges the counsel of her father late A. B. Padmanabha Rao for useful pointers and suggestions.

References

Colebrooke, H. T. (1817). *Algebra with Arithmetic mensuration from the Sanskrit of Brahmagupta and Bhaskar*. Motilal Banarasidas.

Knuth, D. E. (1981). *Seminumerical algorithms: The art of programming* (Vol. 2). Addison-Wesley.

Ludwig, A. (1991). The *Pratyayās*: Indian contribution to combinatorics (translated from german by Sreeramula Rajeswar Sarma). *Indian Journal of History of Science* **26**(1).

Nooten, V. B. (1993). Binary numbers in Indian antiquity. *Journal of Indian Philosophy, 21*, 31–50.

Seidenberg, A. (1978). The origin of mathematics. In: *Archive for history of exact sciences*.

Shah, J. (2013). A history of Pingala's combinatorics. *Ganita Bharati* **35**(1-2).

Sharma, K. N. (1986). *Vṛttaratnākara, Nārāyanī. Manimayī Vyākhyādyayopetaḥ*. Chaukhamba Sanskrit Sansthan.

Sharma, A. (2001). *Piṅgalāchārya Praṇītam Chandaśāstram*. Parimal Publication.

Sridharan, R. (2005). In: G. G. Emch, R. Sridharan, and M. D. Srinivas (Eds.) *Sanskrit prosody, Piṅgaḷa sūtras and binary arithmetic* (pp. 33–62). Hindustan Book Agency.