Listing up Combinations of Resistances

ISOKAWA Yukinao*

(Received 27 October, 2015)

Abstract

Circuits, nested connections of resistances in series and/or in parallel, are studied. We give a computer program which lists up all circuits. Furthermore we discuss construction of the minimal circuit that has a given resistance value.

Keywords : Series, Parallel, Scheme, Continued fraction

1 Introduction

Electrical resistances may be placed in a circuit either in series or in parallel, or in various combinations of them. Although there are circuits such as the Wheatstone bridge, which are neither series nor parallel arrangements, we exclude investigations of them in this paper. Excatly we define circuits in a recursive way as follows:

1. a resistance is a circuit,
2. a combination of two resistances in series is a circuit, and a combination of two resistances in parallel is also a circuit,
3. a combination of two circuits in series is a circuit, and a combination of two circuits in parallel is also a circuit.

These circuits have been studied from old times (see [1],[2]). Since those times they have continued to attract many researchers. In particlar, the number of circuits has been studied both etensively and thoroughly (see [3],[4]).

Let us deote by a_n the number of circuits composed of n resistances. The sequence of numbers a_n $(n = 1, 2, 3, \cdots)$ are computed to be (see [5]) :

$1, 2, 4, 10, 24, 66, 180, 522, 1532, 4624, 14136, 43930, 137908, 437502, 1399068, 4507352, 14611576, \cdots\cdots\cdots$

In this paper we discuss other aspects of circuits than their numbers a_n. In the section 2 we give a computer program which list up all circuits as long as memory of enough size is available. In the section 3 we discuss how to construct a circuit which has a given resistance value. This constrction problem seems to have escaped from the eyes of previous works.

* Professor of Kagoshima University, Research Field in Education, isokawa@edu.kagoshima-u.ac.jp

2 A computer program which list up all circuits

2.1 An algorithm and data strucrure

Our algorithm goes away back to MacMahon.

By definition every circuit is either a series combination of two circuits or a parallel combination of two circuits. We call the former S-type and the latter P-type. Consider circuits which is compose of n resistances. Let C_n, S_n, P_n stand for the set of all circuits, all S-type circuits, all P-type circuits respectively. Needless to say, $C_n = S_n \cup P_n$.

MacMahon' algorithm is as follows:

- Any circuit of S_n can be constructed by connecting smaller circuits in series. More precisely,
  1. Let $n = m_1 + m_2 + \cdots m_k$ $(m_1 \geqq m_2 \geqq \cdots \geqq m_k)$ be a parition of an integer n. Consider any circuit $c_1 \in C_{m_1}, c_2 \in C_{m_2}, \cdots, c_k \in C_{m_k}$. If we connect them in series, a resultant circuit belongs to S_n.
  2. However, in the above construction, parts $c_1, c_2, \cdots, c_k$ must be diffrent each other. For example, if $m_1 = m_2$, then it is inadmissible that c_1 and c_2 are the same.
  3. Conversely, any circuit of S_n can be constructed in the above way.
- Any circuit of P_n can be constructed by "conjugation". Here the "conjugation" of a circuit c is the other circuit c' which is constructe by substitution "series" ans "parallel" connection in c with "parallel" and "series" connection respectively.

In order to represent any circuit, we adopt "list" as data structure.

1. a resistance is represented by a list `(r)`,
2. a series connection of several resistances is represented by a list `(s r r ...)`, and a parallel connection of several resistances is represented by a list `(p r r ...)` respectively,
3. a series connection of several smaller circuits with representation `L1, L2, ...` is represented by a list `(s L1 L2 ...)`, and paralllel connection of several smaller circuits with representation `L1, L2, ...` is represented by a list `(p L1 L2 ...)`.

Thus, in general, a circuit will have representation aa a nested list (list of list of list ...). For example,

$$
\begin{aligned}
S_4 &= \{\texttt{((s (p r r) (p r r)),(s r (p r (s r r))),} \\
&\quad \texttt{(s r (p r r r)),(s r r (p r r)),(s r r r r))}\} \\
P_4 &= \{\texttt{((p (s r r) (s r r)),(p r (s r (p r r))),} \\
&\quad \texttt{(p r (s r r r)),(p r r (s r r)),(p r r r r))}\}
\end{aligned}
$$

To use list fully, our program will be written by Scheme (a major dialect of Lisp) language. [1]

[1] At first the author started to write a program by C language. But he can no complete the task bacause realization of data structure by C language is very complex. The following implementation is due to S.I. who is a student at Hiroshima University. He first implements a program by Haskell language, and later ports it to Scheme. If you are interested in his Haskell program, please contact me (isokawa@edu.kagoshima-u.ac.jp).

2.2 An implementation

utility functions (1)

```
; for partial application
(define-syntax pa$
   (syntax-rules ()
      ((_ f arg ...) (lambda (x) (f arg ... x)))))

(define (1+ x) (+ x 1))

(define (1- x) (- x 1))

; A list of integer between m and n (m <= n)
(define (range+ m n)
   (if (> m n)
      '()
      (cons m (range+ (1+ m) n))))

; A list of integer between m and n (m >= n)
(define (range- m n)
   (if (< m n)
      '()
      (cons m (range- (1- m) n))))

; sum :: [Integer] -> Integer
(define (sum xs) (apply + xs))

; concat :: [[a]] -> [a]
; e.g. [ [1,2], [3,4,5], [], [6] ] -> [1,2,3,4,5,6]
(define (concat xss)
   (if (null? xss)
      '()
      (append (car xss) (concat (cdr xss)))))

; tails :: [a] -> [[a]]
; e.g. [1,2,3,4,5] -> [ [1,2,3,4,5], [2,3,4,5], [3,4,5], [4,5], [5] ]
(define (tails xs)
   (if (null? xs)
      '()
      (cons xs (tails (cdr xs)))))
```

utility functions (2)

```
; fmap :: (a -> b) -> [a] -> [b]
(define fmap map)

; return :: a -> [a]
(define (return x)
   (cons x '()))

; (>>=) :: [a] -> (a -> [b]) -> [b]
(define (>>= xs f)
   (if (null? xs)
      '()
      (concat (map f xs))))

; e.g. (split prime? '(2 11 7 1 9)) -> ( (2 11 7) (1 9) )
(define (split p xs)
   (letrec ((recfn (lambda (p xs acc)
      (if (null? xs)
         (list (reverse acc) '())
         (let ((x (car xs)) (xs' (cdr xs)))
            (if (p x)
               (recfn p xs' (cons x acc))
               (list (reverse acc) xs)))))))
   (recfn p xs '())))

; e.g. (pack '(1 1 2 2 2 2 3)) -> ( (1 1) (2 2 2 2) (3) )
(define (pack xs)
   (if (null? xs)
      '()
      (let* ((ys/zs (split (pa$ eqv? (car xs)) xs))
             (ys (car  ys/zs))
             (zs (cadr ys/zs)))
         (cons ys (pack zs)))))

; e.g. (encode '(1 1 2 2 2 2 3)) -> ( (2 . 1) (4 . 2) (1 . 3) )
(define (encode xs)
   (map (lambda (xs) (cons (length xs) (car xs))) (pack xs)))
```

utility functions (3)

```
; A list of list of integer, the sum of elements is equal to given
; e.g. (split-number 3) -> ( (3) (1 2) (1 1 1) )
(define (split-number n)
   (letrec ((recfn (lambda (i j)
      (if (zero? i)
         '(())
         (>>= (range- i j) (lambda (x)
            (fmap (pa$ cons x) (recfn (- i x) x))))))))
   (recfn n 1)))

; e.g. (combinations 3 '(1 2 3))
        -> ( (1 1 1) (1 1 2) (1 1 3) (1 2 2) (1 2 3)
;             (1 3 3) (2 2 2) (2 2 3) (2 3 3) (3 3 3) )
(define (combinations n xs)
   (if (zero? n)
      '(())
      (>>= (tails xs) (lambda (ys)
         (fmap (pa$ cons (car ys)) (combinations (1- n) ys))))))

; e.g. (cartesian-product '( (1 2 3) (4 5) ))
        -> ( (1 4) (1 5) (2 4) (2 5) (3 4) (3 5) )
(define (cartesian-product xss)
   (if (null? xss)
      '(())
      (>>= (car xss) (lambda (x)
         (fmap (pa$ cons x) (cartesian-product (cdr xss)))))))

; conjugate :: Circuit -> Circuit
;    Find a conjugate of a circuit
(define (conjugate c)
   (case (car c)
      ('r c)
      ('s (apply Par (map conjugate (cdr c))))
      ('p (apply Ser (map conjugate (cdr c))))))

; resistance :: Circuit -> Double
;    Calculate a resistance of a circuit
(define (resistance c)
   (case (car c)
      ('r (cadr c))
      ('s (sum (map resistance (cdr c))))
      ('p (/ 1 (sum (map (pa$ / 1) (map resistance (cdr c))))))))
```

main program

```
; Circuit = Res Double | Ser [Circuit] | Par [Circuit]

(define (Res . v)
   (cons 'r v))

(define (Ser . cs)
   (cons 's cs))

(define (Par . cs)
   (cons 'p cs))

; circuits :: Integer -> [Circuit]
;    A list of a circuit containing n resistances
(define (circuits n)

   (define (template series/parallel Par/Ser n)
      (if (= n 1)
         (return (Res))
         (>>= (cdr (split-number n)) (lambda (ns)
            (let* ((f (lambda (xs)
                        (combinations (car xs) (series/parallel (cdr xs)))))
                   (as (map f (encode ns)))
                   (bs (cartesian-product as))
                   (cs (map concat bs)))
               (fmap Par/Ser cs))))))

   (define series (pa$ template parallel (pa$ apply Ser)))
   (define parallel (pa$ template series (pa$ apply Par)))

   (cond
      ((= n 0) '())
      ((= n 1) (return (Res)))
      (else (let* ((ss (series n))
                   (ps (map conjugate ss)))
               (append ss ps)))))
```

2.3 A result

Reault for only $n = 5$ will be shown.

$$
\begin{aligned}
S_5 \;=\; & \{\texttt{((s (p r r) (p r (s r r))),(s (p r r) (p r r r)),(s r (p (s r r) (s r r))),}\\
& \texttt{(s r (p r (s r (p r r)))),(s r (p r (s r r r))),(s r (p r r (s r r))),}\\
& \texttt{(s r (p r r r r)),(s r (p r r) (p r r)),(s r r (p r (s r r))),}\\
& \texttt{(s r r (p r r r)),(s r r r (p r r)),(s r r r r r))}\}\\
P_5 \;=\; & \{\texttt{((p (s r r) (s r (p r r))),(p (s r r) (s r r r)),(p r (s (p r r) (p r r))),}\\
& \texttt{(p r (s r (p r (s r r)))),(p r (s r (p r r r))),(p r (s r r (p r r))),}\\
& \texttt{(p r (s r r r r)),(p r (s r r) (s r r)),(p r r (s r (p r r))),}\\
& \texttt{(p r r (s r r r)),(p r r r (s r r)),(p r r r r r))}\}
\end{aligned}
$$

3 A contruction problem

In this section we assume that all resistances have the value 1. Then, a circuit $c = (\mathrm{p}\overbrace{1\cdots1}^{m})$ have resistance $1/m$. Therefore, if we consider a circuit which is a combination in series of n copies of c, it has resistance n/m. Thus, given a positive rational number n/m, we can construct a circuit whose resistance value is equal to the rational number.

However the above circuit is composed of the number mn resistances. It seems to be unnecessarily large. Then we may ask what is the minimal circuit when given n/m.

For this problem we will give a plausible solution. For example, when $n/m = 221/71$ is given, it can be represented as

$$\frac{221}{71} = 3 + \cfrac{1}{8 + \cfrac{1}{1 + \cfrac{1}{7}}}.$$

From this expression we construct a circuit

```
(s 1 1 1 (p 1 1 1 1 1 1 1 1 (s 1 (p 1 1 1 1 1 1 1))))
```

It is easy to check that this circuit, which is composed of only 19 resistances, has the given value as its resistance.

We conjecture the following.

Theorem

Suppose that a rational number w can be expressed as

$$w = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}.$$

Then a circuit

$$(\mathrm{s}\overbrace{1+\cdots+1}^{a_0}\,(\mathrm{p}\overbrace{1+\cdots+1}^{a_1}\,(\mathrm{s}\overbrace{1+\cdots+1}^{a_2}\,(\mathrm{p}\cdots)\cdots)))$$

is the minimal circuit which has resistance value w.

References

[1] MacMahon,P.A., "Yoke-chains and multipartite compositions in connexion with the analytical forms called 'Trees'", *Proceedings of the London Mathematical Society* vol.22 (1891) pp.330346. doi:10.1112/plms/s1-22.1.330.

[2] MacMahon,P.A., "Combinations of resistances", *The Electrician* vol.28 (1892) pp.601602. Reprinted in *Discrete Applied Mathematics* vol.54 (1994) pp.225228. doi:10.1016/0166-218X(94)90024-8.

[3] Lomnicki,Z.A., "Two-terminal series-parallel networks", *Adv. Appl. Prob.* vol.4 (1972) pp.109-150.

[4] Moon,J.W., "Some enumerative results on series-parallel networks", *Annals Discrete Math.* vol.33 (1987) pp 199-226.

[5] Sloane,N.J.A., "Table of n, a(n) for n=1..1001", *http://oeis.org/A000084/b000084.txt*