An Arithmetic for trees

Fabrizio Luccio

November 2015

Ref: arXiv1510.05512
We refer to rooted unordered trees

The vertices have any number of children, and if a tree $A$ coincides with $B$ by reshuffling the subtrees rooted at the children of any of its vertices we have $A = B$
Basic notation

♦ 0 denotes the empty tree.
♦ 1 denotes the tree containing exactly one vertex.
♦ 2 denotes the tree containing exactly two vertices.

♦ In a tree $T \neq 0$, $r(T)$ and $n_T$ denote the root and the number of vertices of $T$, respectively.

♦ The subtrees routed at the children of a vertex $x$ are called the subtrees of $x$. 
Tree representation as a binary sequence

all the prefixes of $S_T$, except for the whole sequence, have more 1's than 0's
Tree enumeration

- The trees are grouped into consecutive families $F_0, F_1, \ldots, F_i$, where $F_i$ contains the trees of $i$ vertices.

- Two trees obtained from one another by changing the order of the subtrees of any vertex appear once in $F_i$.

- For two trees $U, T$ with $n_U < n_T$ we have $S_U < S_T$ if the sequences are interpreted as binary numbers.
The canonical form
<table>
<thead>
<tr>
<th>0</th>
<th>10</th>
<th>18</th>
<th>110101010100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>19</td>
<td>110101011000</td>
</tr>
<tr>
<td>2</td>
<td>1100</td>
<td>20</td>
<td>110101101000</td>
</tr>
<tr>
<td>3</td>
<td>110100</td>
<td>21</td>
<td>110101110000</td>
</tr>
<tr>
<td>4</td>
<td>111000</td>
<td>22</td>
<td>110110011000</td>
</tr>
<tr>
<td>5</td>
<td>11010100</td>
<td>23</td>
<td>110110101000</td>
</tr>
<tr>
<td>6</td>
<td>11011000</td>
<td>24</td>
<td>110110110000</td>
</tr>
<tr>
<td>7</td>
<td>11101000</td>
<td>25</td>
<td>110111010000</td>
</tr>
<tr>
<td>8</td>
<td>11110000</td>
<td>26</td>
<td>110111100000</td>
</tr>
<tr>
<td>9</td>
<td>1101010100</td>
<td>27</td>
<td>111001101000</td>
</tr>
<tr>
<td>10</td>
<td>1101011000</td>
<td>28</td>
<td>111001110000</td>
</tr>
<tr>
<td>11</td>
<td>1101011000</td>
<td>29</td>
<td>111010101000</td>
</tr>
<tr>
<td>12</td>
<td>1101011100</td>
<td>30</td>
<td>111010110000</td>
</tr>
<tr>
<td>13</td>
<td>1101011100</td>
<td>31</td>
<td>111010110000</td>
</tr>
<tr>
<td>14</td>
<td>1101011100</td>
<td>32</td>
<td>111011100000</td>
</tr>
<tr>
<td>15</td>
<td>1101011100</td>
<td>33</td>
<td>111100110000</td>
</tr>
<tr>
<td>16</td>
<td>1101011100</td>
<td>34</td>
<td>111101010000</td>
</tr>
<tr>
<td>17</td>
<td>1111000000</td>
<td>35</td>
<td>111101100000</td>
</tr>
<tr>
<td>18</td>
<td>1111000000</td>
<td>36</td>
<td>111110100000</td>
</tr>
<tr>
<td>19</td>
<td>1111000000</td>
<td>37</td>
<td>111110000000</td>
</tr>
</tbody>
</table>
How many trees?

- **Doubling Rule.** From each tree $T$ in $F_{n-1}$ build two trees $T_1, T_2$ in $F_n$ by adding a new vertex as the leftmost child of $r(T)$, or adding a new root and appending $T$ to it as a unique subtree.

- Let $f_n$ be the number of trees in $F_n$:
  
  we immediately have: $f_n \geq 2^{n-2}$ for $n \geq 2$. 
More strictly:

**Proposition 1.** $f_n > 2^{11n/10} - 2$ for $n \geq 11$.

E.g. $f_{11} > 2^{11}$, $f_{21} > 2^{22}$, $f_{31} > 2^{33}$, \ldots

**Proposition 2.** $f_n \leq 2^{2n-5}$ for $n \geq 3$.

**Open problem 1.** Express $f_n$ exactly as a function of $n$. 
Proposition 3. A tree $T$ of $n$ vertices can be transformed in canonical form in time $O(n^2)$. 

```
algorithm CF($T, n$)
1. for any vertex $x \in T$
   count the number of vertices $n_1, \ldots, n_k$ of its subtrees;
   reorder these subtrees for non-decreasing values of the $n_i$;
   let $G_1, \ldots, G_r$ be the groups of subtrees with the same number $g_1, \ldots, g_r$
      of vertices, with all $g_i > 2$;
   // reordering is necessary but not sufficient for having $T$ in canonical form
   // the trees in all $G_i$ must be arranged in canonical order
2. for any $x \in T$, down-top from the vertices closest to the leaves
   for any group $G_i = \{T_1, \ldots, T_s\}$
      compute the representing sequences $S_1, \ldots, S_s$;
      order $S_1, \ldots, S_s$ for increasing binary value;
      permute $T_1, \ldots, T_s$ accordingly.
```
The three operators

- **add**
- **add-plus**
- **mult**
\[ T = A + B \]
The roots \( r(A) \) and \( r(B) \) are merged.
\[ A + 1 = 1 + A = A \]
Addition with 0 is not defined.

\[ T = A \oplus B \]
A new root \( r(T) \) is created, and \( A \) and \( B \) become subtrees of \( r(T) \).
\[ A \oplus 0 = 0 \oplus A \neq A. \]

\[ T = A \odot B \]
\( B \) is merged with each vertex of \( A \) (the subtrees of \( r(B) \) become new subtrees of \( r(A) \)).
\[ A \odot 0 = 0 \odot A = 0 \] (with some abuse of the definition of multiplication in the second term since 0 has no vertices).
\[ A \odot 1 = 1 \odot A = A. \]
Number of vertices

Proposition 4 (Immediate)

\[ T = A + B \Rightarrow n_T = n_A + n_B - 1 \]
\[ T = A \oplus B \Rightarrow n_T = n_A + n_B + 1 \]
\[ T = A \cdot B \Rightarrow n_T = n_A n_B \]
+ and \( \oplus \): commutativity and associativity

**Proposition 5.** For \( A, B, C \neq 0 \):
\[
A + B = B + A
\]
\[
(A + B) + C = A + (B + C)
\]

**Proposition 6.**
\[
A \oplus B = B \oplus A \text{ for all } A, B
\]
\[
(A \oplus B) \oplus C = A \oplus (B \oplus C) \text{ if and only if } A = C
\]
Multiplicity: “product” of a tree $A$ by an integer $k > 1$

$kA = A + A + \ldots + A \quad k \text{ times}$

$k^\oplus A = A \oplus A \oplus \ldots \oplus A \quad k \text{ times}$

$M = kA \Rightarrow n_M = kn_A - k + 1$

$M = k^\oplus A \Rightarrow n_M = kn_A + k - 1$

Note: the number of trees obtained as $kA$ or $k^\oplus A$ is $f_{n_A}$

e.g. : “even” ($k = 2$) trees of $n$ vertices are exponentially less than all trees of $n$ vertices
Multiplication: commutativity and associativity

Proposition 7. Associativity:
\[(A \cdot B) \cdot C = A \cdot (B \cdot C)\] for all \(A, B, C\).

Proposition 8. Commutativity 1:
For \(n_A = n_B\), \(A \cdot B = B \cdot A\) if and only if \(A = B\).

Proposition 9. Commutativity 2:
For \(n_A > n_B\), \(A \cdot B = B \cdot A\) only if
(i) \(B\) is a proper subtree of \(A\)
(ii) \(n_A/e_A = n_B/e_B\), where \(e_A\), \(e_B\) are numbers of leaves of \(A, B\).

For \(n_A > n_B\) several other necessary conditions for commutativity exist. An iff condition has not yet been found.
Commutative product

\[ A \cdot B = B \cdot A \]

for B subtree of A

\[ Z = A \cdot B \]

in canonical form
Power: product of a tree $A$ by itself $k > 1$ times

$$A^k = A \cdot A \cdot \ldots \cdot A \quad k \text{ times}$$

$$P = A^k \quad \Rightarrow \quad n_P = n_{A^k}$$

Note: the number of trees obtained as $A^k$ is $f_{n_A}$

In the previous slide $A = B^2$, then $Z = B^3$. 
Finally multiplication is not distributive over addition and addition-plus, that is in general:

\[(A + B) \cdot C \neq A \cdot C + B \cdot C\]

\[(A \oplus B) \cdot C \neq A \cdot C \oplus B \cdot C\]
Generating all trees

from the single generator $0$, using $+$ and $\oplus$

• the empty tree $0$ is the generator of itself
• tree 1 can be generated as $0 \oplus 0$
• tree 2 can be generated as $1 \oplus 0$
• assuming inductively that each of the trees in $F_i$ with $1 \leq i \leq n - 1$ can be generated by the trees of the preceding families, then each tree $T$ in $F_n$ can also be generated . . . . .
Both $+$ and $\oplus$ are needed
Prime trees

Euclid’s Elements:

\[ \pi\rho\omicron\omicron\varsigma \alpha\rho\iota\mu\omicron\omicron\varsigma = \text{prime number} \]
the concept is significant under multiplication

Mocking Euclid:

\[ \pi\rho\omicron\omicron\varsigma \delta\epsilon\nu\delta\rho\omicron\varsigma = \text{prime tree} \]
the concept is now significant under addition, addition-plus, and multiplication
Euler (1751):
“There are mysteries that we will be never able to understand. It is sufficient to take a look at the distribution of prime numbers”

Gauss (observation when he was a teenager, 1792):
“Primzahlen unter $a ( = \infty) a / \log a$”

Now Prime number theorem, proved independently by Hadamard and de le Vallee-Poussin (1896)

Riemann hypothesis (1859):
“It would be beautiful to have a rigorous proof of this . . .”
Consider a tree \( T \) with more than one vertex

Definition

\( T \) is prime under addition (or add-prime) if can be generated by addition only if the two terms are 1 and \( T \).

\( T \) is prime under addition-plus (or plus-prime) if cannot be generated by addition-plus of any pair of trees.

\( T \) is prime under multiplication (mult-prime) if can be generated by multiplication only if the two factors are 1 and \( T \).
Proposition 10. \( T \) is add-prime if and only if \( r(T) \) has only one subtree.

Proposition 11. \( T \) is plus-prime if and only if \( r(T) \) has more than two subtrees.

Then:

There are infinite add-prime, add-composite, plus-prime, and plus-composite trees.

The number of add-prime trees of \( n \) vertices is \( f_{n-1} \).

The number of plus-prime trees of \( n \) vertices depends on the values of all the \( f_{i \leq n} \).
Primality testing

From Propositions 10 and 11, deciding if a tree is add-prime or plus-prime is computationally “easy” (in fact if the trees are accessed from the root the decision is taken in constant time).

Testing mult-primality is much more difficult.
Proposition 12. If n is a prime number all the trees with n vertices are mult-prime.

Then mult-primality can be decided with an integer primality test if n is prime, but the test is insufficient if n is composite.

Proposition 13. For any tree T we have:

- if \( r(T) \) has only one subtree, T is mult-prime;
- if \( r(T) \) has two subtrees with \( n_1 = n_2 \) vertices, then T is mult-prime;
- if \( r(T) \) has two subtrees with \( n_1 \leq n_2 \) vertices and \( n_1 + 1 \) does not divide \( n_2 \), then T is mult-prime.

Testing these conditions is “easy” but still insufficient.
Further properties of mult-prime trees have been found. We restrict our discussion to the following:

Proposition 14. Let \( T = A \cdot B \) with \( A, B \neq 0 \) and \( A, B \neq 1 \), and let \( Y \) be a subtree of \( r(B) \) with maximum number \( n_Y \) of vertices. Then the subtrees of \( r(B) \) are exactly the subtrees of \( r(T) \) with at most \( n_Y \) vertices.
Notation. For an arbitrary tree $T$:

$G_1, \ldots, G_r$ are the groups of subtrees of $r(T)$ with the same number $g_1, \ldots, g_r$ of vertices, $g_1 < g_2 < \cdots < g_r$;

$H_i$ is the union of $G_1, \ldots, G_i$, i.e. each $H_i$ is the group of subtrees of $r(T)$ with up to $g_i$ vertices.
Structure of Algorithm MP for deciding if a tree T of n vertices is mult-prime.

algorithm MP(T, n)
1. CF(T, n);
   // transform T in canonical form with Algorithm CF
2. let H_1, ..., H_r be the groups of subtrees of r(T);
3. for 1 \leq i \leq r - 1
   copy T into Z;
   traverse Z in preorder
   for any vertex x encountered in the traversal
   if x has all the subtrees of H_i erase these subtrees in Z
   else exit from the i-th cycle;
   return MULT-COMPOSITE;
4. return MULT-PRIME.

A rough analysis shows that MP runs in time O(n^4)
If \( T \) is mult-composite Algorithm MP allows to find a pair of factors \( A, B \) at no extra cost. This implies that \( n \) is factorized in time polynomial in \( n \), in agreement with the factorization in ordinary arithmetic that requires time exponential in \( \log n \).

Counting the number of mult-prime trees seems to be very hard:

Open problem 2. For any given \( n \), determine the number of mult-prime trees of \( n \) vertices.


