Enumerating All Graphical Sequences

(Extended Abstract)

Yosuke Kikuchi*, Katsuhisa Yamanaka†and Shin-ichi Nakano‡

1 Introduction

The degree sequence of simple graph G is a sequence of degrees of vertices in G in decreasing order, whereas a given integer sequence D is the degree sequence for some simple graph, then D is a *graphical sequence* (also *graphic sequence*). In this paper degree sequence and graphical sequence are decreasing order sequence. Then for a given graph, we can obtain the unique degree sequence, and for a given graphical sequence D, we may obtain some graphs whose degree sequences are D. For example, Figure 1 shows that there are two graphs for the degree sequence $D = (5, 3, 3, 3, 2, 2)$. Furthermore the sequence $(3, 2, 2, 2)$ is not graphical. In this paper we consider to generate all graphical sequences.

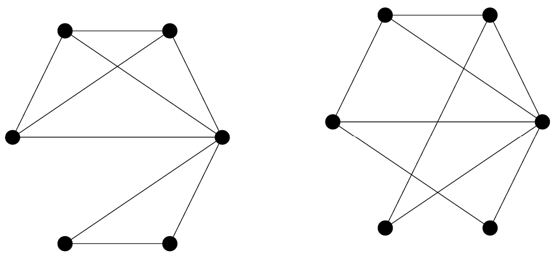

Figure 1: Two graphs sharing the degree sequence $(5, 3, 3, 3, 2, 2,)$.

A generating algorithm is *CAT* (Constant Amotized Time) if its running time is propotional to the number of generating objects. If experimental behavior of a generating algorithm is observed to be CAT, then the algorithm is called *alley CAT* [3]. An alley CAT algorithm has no proof (home) of CAT but its experimental behavior shows it seemed to be a CAT algorithm.

Ruskey et al. [3] gave an alley CAT algorithm that generates all degree sequences of length n, where n is a given positive integer. By modifying their algorithm, we design a CAT algorithm. That is, we give a "good home" to the algorithm and show that it is no longer "an alley CAT".

This paper first provides a simple algorithm to generate all graphical sequences of at most n positive integers. The algorithm generates each sequence in constant time for each without repetition. Then, by modifying the algorithm, we give an algorithm that generates every graphical sequence of n integers in constant time on average

A basic idea of the algorithm is as follows: 1. we define a tree structure on the set of graphical sequences, 2. we traverse the tree efficiently. Some algorithms based on the similar but other ideas are known [1, 4, 5, 6].

2 Tree Structure

Havel gave an algorithmic characterization of graphical sequence and the chracterization is rediscovered by Hakimi, and Erdös and Gallai gave a combinatorial characterization[2, pp.12–15]. Our tree structure is based on Havel and Hakimi's characterization.

Let S_n be the set of degree sequences of graphs that have at most n vertices, thus the set of graphical sequences of length at most n. For example, there are seven sequences in $S_3 = \{(0), (0, 0), (1, 1), (0, 0, 0), (2, 1, 1), (1, 1, 0), (2, 2, 2)\}$.

A tree structure among S_n is defined as follows. Suppose a sequence $D \in S_n \setminus \{(0)\}$. Since the graphical sequence of length one is (0), the sequence D contains $k > 1$ integers, then D can be denoted by $D = (d_1, d_2, \ldots, d_k)$. Note that $d_1 \geq d_2 \geq \cdots \geq d_k$. Let $P(D)$ be an integer sequence obtained from D. Then we will consider the following two cases to obtain $P(D)$ from D:

Case 1: $d_k = 0$. $P(D)$ is $(d_1, d_2, \ldots, d_{k-1})$.

Case 2: $d_k \neq 0$. $P(D)$ is the sorted sequence of $(d_2 - 1, d_3 - 1, \ldots, d_{1+d_1} - 1, d_{2+d_1}, d_{3+d_1}, \ldots, d_k)$.

If $d_1 \geq k$, D is not a graphical sequence. Then $1 + d_1 \leq k$ is held. In Case 2, if $d_{1+d_1} > d_{2+d_1}$ then, $(d_2 - 1, d_3 - 1, \ldots, d_{1+d_1} - 1, d_{2+d_1}, d_{3+d_1}, \ldots, d_k)$ is a decreasing sequence. Then we do not sort it. If $d_{1+d_1} = d_{2+d_1}$, $(d_2 - 1, d_3 - 1, \ldots, d_{1+d_1} - 1, d_{2+d_1}, d_{3+d_1}, \ldots, d_k)$ is not a decreasing sequence. Then we need to sort it. However we can obtain $P(D)$ from D by the following way instead of sorting. Let $(d_a, \ldots, d_b)$ be the maximum subsequence of D such that $d_a = d_{1+d_1} = d_b$ and set $c = 1 + d_1 - a + 1$. Then we can obtain $P(D) = (d_2 - 1, d_3 - 1, \ldots, d_{a-1} - 1, d_a, d_{1+a}, \ldots, d_{b-c}, d_{b-c+1} - 1, d_{b-c+2} - 1, \ldots, d_b - 1, d_{1+b}, \ldots, d_k)$ from D without sorting. Thus, after removing d_1 from D, we reduce each value from two subsequences from d_2 to d_{a-1} and from d_{b-c+1} to d_b by 1. Then we obtain $P(D)$ from D. If $a = 2$ or $b = 1 + d_1$, we reduce each value from one subsequence. Moreover d_{b-c+1} is equal to d_b in both D and $P(D)$.

In Case 1 and Case 2, $P(D)$ is a graphical sequence by

*Dept. Comp. and Info. Eng., Tsuyama National Coll. Tech., Okayama, JAPAN, kikuchi@tsuyama-ct.ac.jp

†Graduate School of Information Systems, University of Electro-Communications, Tokyo, JAPAN, yamanaka@is.uec.ac.jp

‡Dept. Comp. Sci., Gunma University, Gunma, JAPAN nakano@cs.gunma-u.ac.jp

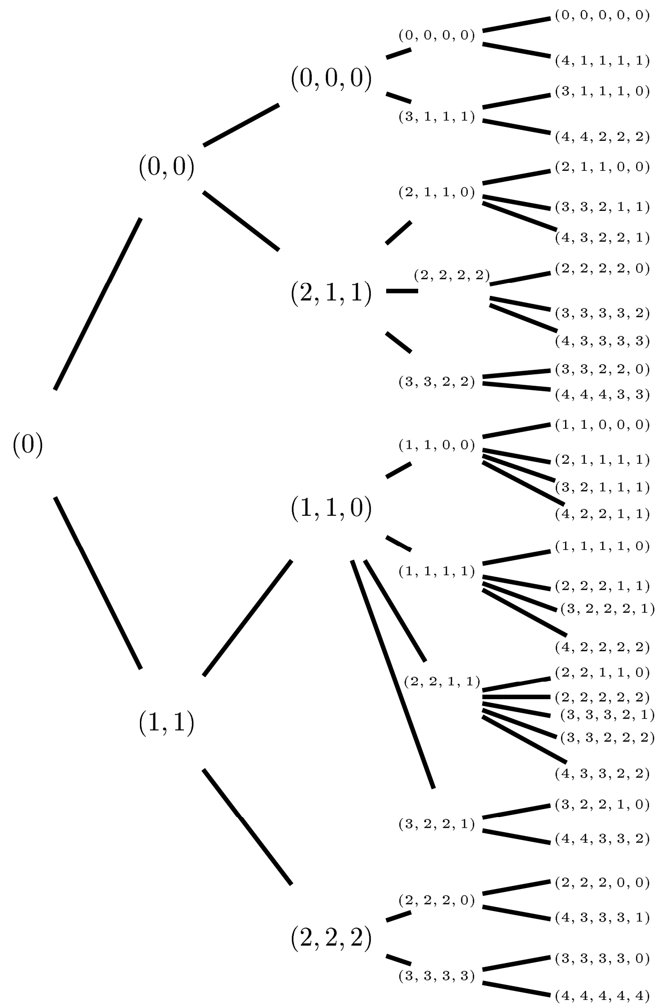

Figure 2: The family tree T_5.

Lemma 1, and the length of $P(D)$ is equal to the length of D minus one.

Lemma 1 [2, p.13] *A sequence $D = (d_1, d_2, \ldots, d_k)$ is graphical if and only if $D = (d_2 - 1, d_3 - 1, \ldots, d_{1+d_1} - 1, d_{2+d_1}, d_{3+d_1}, \ldots, d_k)$ is graphical.*

So, let $P(D)$ be the parent sequence of D and D be a child sequence of $P(D)$. Note that D has the unique parent sequence $P(D)$, and on the other hand $P(D)$ may have some child sequences.

For an arbitrary $D \in S_n \setminus \{(0)\}$, repeatedly finding the parent sequence of the derived sequence produces the unique sequence $D, P(D), P(P(D)), \ldots$ of graphical sequences in S_n, which eventually ends with the root sequence (0). By merging these sequences we have *the family tree* of S_n, denoted by T_n, in which each vertex at the depth k corresponds to the graphical sequence of length $k+1$ in S_n, and each edge corresponds to the pair of each D and $P(D)$. For instance, T_5 is shown in Fig. 2.

3 Algorithm

If an algorithm can generate all child sequences of a given graphical sequence in S_n, then the algorithm traverses T_n in a recursive manner, and generates all graphical sequences in S_n. This section gives such a generating algorithm. Let $D = (d_1, d_2, \ldots, d_{k-1})$ be a graphical sequence. The child sequences of D can be classified the following four types by the way of adding one or zero to each digit in D.

Type 0: $C[0] = (d_1, d_2, \ldots, d_{k-1}, 0)$.

Type 1: $C[x] = (x, 1 + d_1, 1 + d_2, \ldots, 1 + d_x, d_{1+x}, d_{2+x}, \ldots, d_{k-1})$. Note that $x \le k-1$.

Type 2: $C[x, s] = (x, d_1, d_2, \ldots, d_{s-1}, 1 + d_s, 1 + d_{s+1}, \ldots, 1 + d_{s+x-1}, d_{s+x}, d_{s+x+1}, \ldots, d_{k-1})$. Note that $d_1 = d_{s-1} = 1 + d_s$ and $x < k-1$.

Type 3: $C[x, r, s] = (x, 1 + d_1, 1 + d_2, \ldots, 1 + d_r, d_{1+r}, d_{2+r}, \ldots, d_{s-1}, 1 + d_s, 1 + d_s, \ldots, 1 + d_t, d_{t+1}, d_{t+2}, \ldots d_{k-1})$, where $s > r + 1$ and $t = s + x - r - 1$. Note that d_{r+1} and d_{s-1} are invariable and $x < k-1$.

Each child of D is one of these three type. Note that D may have no child for some type.

By the above classification we can have the following theorem.

Theorem 1 *By the above classification, one can generate each graphical sequence in S_n. The algorithm uses $O(n)$ space and runs in $O(|S_n|)$ time.*

By Theorem 1 the algorithm generates each sequence in $O(1)$ time on average. So, each sequence cannot be generated in $O(1)$ time in worst case. However, a simple modification [7] improves the algorithm to generate each sequence in $O(1)$ time.

By modifying the algorithm such that it outputs only sequences corresponding to the leaves in T_n, we have the following theorem.

Theorem 2 *One can generate all graphical sequences of length exactly n in $O(1)$ time on average.*

References

[1] D. Avis and K. Fukuda, *Reverse search for enumeration*, Discrete Appl. Math., 6, 21–46, 1996.

[2] G. Chartrand and L. Lesniak, *Graphs & Digraphs*, 4th ed., Chapman & Hall, 2005.

[3] F. Ruskey, P. Eades, B. Cohen and A. Scott, *Alley CATs in search of good homes*, Congressus Numerantium., 102, 97–110, 1994.

[4] S. Nakano, *Enumerating floorplans with n rooms*, Proc. ISAAC2001, LNCS 2223, 104–115, 2001.

[5] S. Nakano, *Efficient generation of plane trees*, Inf. Process. Lett., 84, 167–172, 2002.

[6] S. Nakano, *Efficient generation of triconnected plane trianguration*s, Computational Geometry, Theory and Applications, 27, 109–122, 2004.

[7] S. Nakano and T. Uno, *Constant time generation of trees with specified diameter*, Proc. WG2004, LNCS 3353, 33–45, 2004.