ENUMERATING LABELED GRAPHS THAT REALIZE A FIXED DEGREE SEQUENCE

ATABEY KAYGUN

ABSTRACT. A finite non-increasing sequence of positive integers \(d = (d_1 \geq \cdots \geq d_n)\) is called a degree sequence if there is a graph \(G = (V,E)\) with \(V = \{v_1, \ldots, v_n\}\) and \(\text{deg}(v_i) = d_i\) for \(i = 1, \ldots, n\). In that case we say that the graph \(G\) realizes the degree sequence \(d\). We show that the exact number of labeled graphs that realize a fixed degree sequence satisfies a simple recurrence relation. Using this relation, we then obtain a recursive algorithm for the exact count. We also show that in the case of regular graphs the complexity of our algorithm is better than the complexity of the same enumeration that uses generating functions.

INTRODUCTION

A finite non-increasing sequence of positive integers \(d_1 \geq \cdots \geq d_n\) is called a degree sequence if there is a graph \((V, E)\) with \(V = \{v_1, \ldots, v_n\}\) and \(\text{deg}(v_i) = d_i\) for \(i = 1, \ldots, n\). In that case, we say that the graph \(G\) realizes the degree sequence \(d\). In this article, in Theorem 1.1 we give a remarkably simple recurrence relation for the exact number of labeled graphs that realize a fixed degree sequence \((d_1, \ldots, d_n)\). We also give an algorithm and a concrete implementation to explicitly count three classes of labeled graphs for a moderate number of vertices.

There is an extensive volume of research on the asymptotics of the number of graphs that realize a fixed degree sequence. We refer the reader to Wormald’s excellent ICM lecture [18] for a comprehensive survey, and references therein. As for the exact number of graphs that realize a fixed degree sequence, Read obtains enumeration formulas in [14] and [15] as applications of Polya’s Hauptsatz [12, 13]. However, Read himself admits

“It may readily be seen that to evaluate the above expressions in particular cases may involve an inordinate amount of computation.” [15, Sect.7]

But we encountered no explicit complexity analysis of Read’s formulas in our search in the literature. On the other hand, in [9] McKay writes explicit generating polynomials whose complexities can readily be calculated, and in which coefficients of certain monomials yield the exact number of different classes of labeled graphs. In particular, he writes a generating polynomial (see Equation (1.3)), whose computation complexity is \(O(2^{n^2/2})\), in which the coefficient of the monomial \(x_1^m \cdots x_n^m\) gives the exact count of labeled \(m\)-regular graphs on \(n\)-vertices.

Our recurrence relation works for all degree sequences, but for those degree sequences where there is a uniform upper limit \(m\) for the degrees, our algorithm has the worst-case complexity of \(O(n^{m^2})\). This means, in the specific case of \(m\)-regular graphs we achieve a better complexity than generating polynomials.

While factorial-like worst-case complexity of the enumeration \(1.2\) may render practical calculations difficult, the fact that it is recursive allows us to employ computational tactics such as dynamic programming \([1, 8]\) or memoization \([10]\) to achieve better average complexity. We explore this avenue in our implementation given in the Appendix. To demonstrate of the versatility of our recurrence relation and the resulting implementation, we tabulate the number of \(m\)-regular labeled graphs, the number of labeled graphs that realize the same degree sequence with binary trees, and the number of labeled graphs that realize the same degree sequence with complete bipartite graphs.
One can also read a given degree sequence \((d_1, \ldots, d_n)\) as a partition of \(N = \sum d_i\). The Erdős-Gallai Theorem [4, 2] tells us when such a partition is realizable as a degree sequence, or one can also use Havel-Hakimi algorithm to decide whether the given partition is realizable [6, 5]. Now, one can also use our enumeration to decide whether given a degree sequence is realizable, but admittedly, the Havel-Hakimi algorithm has a much better complexity.

**Plan of the article.** We prove our recurrence relation, analyze its complexity and compare it with generating functions for regular graphs in Section \([1]\) We present explicit calculations we made in Section \([2]\) and the code we used performing the calculations in the Appendix.

**Notations and conventions.** We assume all graphs are simple, labeled, and undirected throughout the article.

**Acknowledgments.** This article was written while the author was on academic leave at Queen’s University in Canada from Istanbul Technical University. The author would like to thank both universities for their support.

1. **Enumerating Graphs that Realize a Fixed Degree Sequence**

1.1. **The recurrence relation.** Assume \(d = (d_1, \ldots, d_n)\) is a non-increasing sequence of strictly positive integers \(d_i > 0\). Let us consider the trivial cases first: It is clear that there is a single graph on the empty sequence \(e\): the empty graph. Also, in case \(n = 1\), the only case for which the sequence \((d_1)\) is realized by a graph is when \(d_1 = 0\) which is excluded by our assumption. So, the count is 0 for all \((d_1)\) for \(d_1 > 0\). We also exclude the case where the sum \(\sum_i d_i\) is odd since such sequences cannot be realized as degree sequences because of the Hand-Shake Lemma. We assume \(n > 1\) and the sum \(\sum_i d_i\) is even. If we consider the vertex \(x_n\) we see that it needs to be connected to exactly \(d_n\) vertices in the set \(\{x_1, \ldots, x_{n-1}\}\). We need to consider the set of all subsets of \(\{x_1, \ldots, x_{n-1}\}\) of size \(d_n\) to enumerate all possibilities. So, let \(S\) be an arbitrary subset of \(\{1, \ldots, n-1\}\) of size \(d_n\), and let \(\chi_S\) be the characteristic function of the set \(S\). Every graph in which \(x_n\) is connected to each vertex in \(\{x_i \mid i \in S\}\) realizes the same degree sequence

\[
(d_1 - \chi_S(1), \ldots, d_{n-1} - \chi_S(n - 1))
\]

if we remove \(x_n\) and all the edges connected to \(x_n\). Let us write \((d_1, \ldots, d_{n-1})/S\) for the sequence \((1,1)\) after we reorder the sequence in descending order and remove all 0’s. Let \(C((d_1, \ldots, d_n))\) be the number of graphs that realize the same degree sequence \((d_1, \ldots, d_n)\). Thus we obtain:

**Theorem 1.1.** The total number of labeled graphs that realize the degree sequence \((d_1, \ldots, d_n)\) satisfies the recurrence relation

\[
C((d_1, \ldots, d_n)) = \sum_{S \in \binom{\{1, \ldots, n-1\}}{d_n}} C((d_1, \ldots, d_{n-1})/S)
\]

where we write \(\binom{X}{k}\) for the set of all subsets of size \(k\) of a set \(X\).

1.2. **The complexity analysis.** In [9] McKay writes a generating polynomial

\[
f(x) = \prod_{1 \leq i < j \leq n} (1 + x_i x_j).
\]

in which the coefficient of the monomial \(x_1^m \cdots x_n^m\) yields the number of \(m\)-regular graphs on \(n\)-vertices. If we assume the complexity of the calculation is given by the number of multiplications in the product, then the computational complexity of the generating polynomial is \(O(2^n/2)\).
Let \( d = (d_1, \ldots, d_n) \) be a degree sequence, and let us use \( \#C(d) \) for the total number of summands (which is the number of leaves in the recursion tree) in \( C(d) \) in Equation (1.2) which will be the complexity measure for our enumeration.

**Proposition 1.2.** Assume there is a fixed upper bound \( m \) for the degrees in \( d \). Then the complexity of the enumeration given in (1.1) is \( O(n^{mn}) \). In particular, the enumeration complexity for \( m \)-regular graphs is also \( O(n^{mn}) \).

**Proof.** As long as \( 2m \leq n \), the function \( \binom{n}{m} \) is increasing in \( m \). Then

\[
\#C((d_1, \ldots, d_n)) = \sum_{S \in \mathcal{P}(\{1, \ldots, n-1\})} \#C((d_1, \ldots, d_{n-1})/S)
\]

\[
\leq \binom{n}{m} \max_{S \in \mathcal{P}(\{1, \ldots, n-1\})} \#C((d_1, \ldots, d_{n-1})/S)
\]

\[
\leq \cdots \leq \binom{n}{m} \cdot \binom{2m}{m} \max_{S \in \mathcal{P}(\{1, \ldots, 2m-1\})} \#C((d_1, \ldots, d_{2m-1})/S)
\]

\[
\leq \binom{n}{m}^{n-2m} C_m
\]

for some constant \( C_m \). Since \( m \) is fixed and \( \binom{n}{m} \) is of order \( n^m \) we get that the number of summands in \( C((d_1, \ldots, d_n)) \) is \( O(n^{m(n-2m)}) = O(n^{mn}) \). \( \square \)

One can easily see that the enumeration complexity of (1.2) we obtained in Proposition 1.2 is better than the complexity of generating polynomial for regular graphs. However, we still have to work around the fact that the worst-case complexity is factorial-like with a constant exponent. Fortunately, one can employ powerful computational tactics such as dynamic programming or memoization to improve average complexity of the enumeration since it is recursive. See the Appendix for how we used memoization to improve average complexity of our calculations.

## 2. Explicit Calculations

Let us start with calculating an explicit example by hand. The degree sequence of the complete graph \( K_n \) on \( n \)-vertices is the constant sequence \( n - 1 \) of length \( n \). Since one has only one subset of \( \{1, \ldots, n-1\} \) of size \( n - 1 \) we get that

\[
C((n - 1, \ldots, n - 1)) = C((n - 2, \ldots, n - 2)) = \cdots = C((1, 1)) = C(\epsilon) = 1.
\]

In other words, the labeled complete graph \( K_n \) is the only graph with that specific degree sequence.

### 2.1. Enumerating regular graphs

An \( m \)-regular graph on \( n \)-vertices is similar to \( K_{m+1} \) in that it is a graph on \( n \)-vertices where every vertex has the same constant degree \( m \)

\[
(m, \ldots, m).
\]

Now, let us write

\[
R(n, m) = C((m, \ldots, m)).
\]

We need to note that \( R(n, m) = 0 \) when \( m \geq n \), or when both \( n \) and \( m \) are odd since in these cases there are no graphs that can realize the sequences given in (2.2).
The number of labeled \( m \)-regular graphs on \( n \)-vertices for \( m = 1, \ldots, 8 \). We calculated \( R(n, m) \) for \( 1 \leq n \leq 30 \) and \( 2 \leq m \leq 8 \). The results for \( 1 \leq m \leq 5 \) took about 2 minutes while the cases for \( 6 \leq m \leq 8 \) took about 10 minutes on a moderate computer. These calculations strongly indicate the average complexity of the enumeration algorithm with memoization is much better than the worst-case complexity.

We tabulated the results for \( 2 \leq n \leq 15 \) in Table 1. A smaller version of the tables can be found at [3, pg. 279], and as the sequence A295193 at OEIS [7]. The individual sequences for \( m = 1, \ldots, 6 \) in Table 1 are respectively the sequences A001147, A001205, A002829, A005815, A338978 and A339847 at OEIS.

2.2. Enumerating graphs that realize the same degree sequences as binary trees. Any binary tree with \( k + 1 \)-leaves will have \( k - 1 \) internal vertices of degree 3. Thus any such tree has the degree sequence

\[
(3, \ldots, 3, 1, \ldots, 1).
\]

Now, let

\[
T(k) = \binom{2k}{k-1} C((3, \ldots, 3, 1, \ldots, 1)).
\]

\footnote{On an Intel i5-8250U CPU working at 1.60GHz with 8Gb of RAM on a Linux operating system.}
be the number of labeled graphs that has the same degree sequence given in (2.4). Notice that we put a correction factor \((\frac{2^k}{k!})\) since in the original enumeration \(C(d)\) vertices are not allowed to change degree. In the case of regular graphs, one does not need a correction factor since every vertex has the same degree.

Using our implementation of the enumeration algorithm we calculated these numbers on the same setup we described above. The results are calculated almost immediately and they are given in Table 2.

| \(k\) | \(R(k)\) |
|---|---|
| 1 | 1 |
| 2 | 4 |
| 3 | 90 |
| 4 | 8400 |
| 5 | 1426950 |
| 6 | 366153480 |
| 7 | 134292027870 |
| 8 | 67095690261600 |
| 9 | 43893900947947050 |
| 10 | 36441011093916429000 |
| 11 | 37446160423265535041100 |
| 12 | 4666935764700872700474400 |
| 13 | 69367722399061403579194432500 |
| 14 | 121238024532751529573125745790000 |
| 15 | 246171692450596203263023527657431250 |

Table 2. The number of labeled graphs that realize the same degree sequence as any binary tree on \(2k\) vertices.

In [11, pg.6] the number of labeled trees on \(n\) vertices that realize a fixed degree sequence \((d_1, \ldots, d_n)\) is calculated as

\[
(2.6) \quad \binom{n - 2}{d_1 - 1, \ldots, d_n - 1}
\]

which is different than our calculations for the case \((d_1, \ldots, d_n)\) given as (2.4). But note that Moon’s formula enumerates only trees that realize a particular degree sequence while we count all graphs.

2.3. **Enumerating graphs that realize the same degree sequences as complete bipartite graphs.**
We fix two positive integers \(n \leq m\). A complete bipartite graph \(K_{n,m}\) contains \(n + m\) vertices which is split into two disjoint sets, say black and white. Black vertices are connected to every white vertex and vice versa, but vertices of the same color are not connected. Any such graph would have the degree sequence

\[
(2.7) \quad (\underbrace{m, \ldots, m}_{n\text{-times}}, \underbrace{n, \ldots, n}_{m\text{-times}})
\]

Let us write

\[
(2.8) \quad K(n, m) = \begin{cases} 
\binom{n+m}{n} C((\underbrace{m, \ldots, m, n, \ldots, n}_{n\text{-times}}, \underbrace{n, \ldots, n}_{m\text{-times}})) & \text{if } n \neq m \\
C((\underbrace{n, \ldots, n}_{2n\text{-times}})) & \text{if } n = m
\end{cases}
\]

for the number of labeled graphs that realize the degree sequence given in (2.7) for every \(m \geq 2\) and \(1 \leq n \leq m\). We tabulated the results for \(2 \leq m \leq 10\) and \(2 \leq n \leq 6\) in Table 3. The first column of Table 3 is A002061 at OEIS.
The number of labeled graphs that realize the same degree sequence as the complete bipartite graph $K_{n,m}$.

Appendix: The Code

Since or implementation is simple and short, we opted to list the code we used to make our calculations here in an Appendix in Figure 1. This way, our results can be reproduced and verified.

We implemented our enumeration using Common Lisp [17] and a suitable memoization to control the depth of the recursive calls. However, due to efficiency issues of the data structures we use, our degree sequences are non-decreasing instead of being non-increasing. We used SBCL version 2.0.11 to run the lisp code [16].

Our enumeration calculation requires us to calculate a finite number of shorter degree sequences in each call. When we employ memoization, we use a global table of already calculated results. If an enumeration on a shorter degree sequence is needed, and if the result is already calculated for another branch of the recursive call we recall the result instead of calculating it from scratch.

```
(defun subsets (k xs)
  (cond ((null xs) 'nil)
        ((= 1 k) (loop for x in xs collect (list x)))
        (t (union (subsets k (cdr xs))
                  (mapcar (lambda (x) (cons (car xs) x))
                           (subsets (1- k) (cdr xs)))))))

(defun new-degree-sequence (ds is)
  (let ((ys (copy-list (cdr ds))))
    (dolist (i is) (decf (nth i ys)))
    (delete 0 (sort ys #'<))))

(defun graph-count (ds)
  (cond ((null ds) 1)
        ((oddp (reduce #'+ ds)) 0)
        (t (or (gethash ds table)
               (setf (gethash ds table)
                     (let* ((index-set (loop for i from 0 below (1- (length ds))
                                             collect i))
                             (all-subsets (subsets (car ds) index-set)))
                              (loop for xs in all-subsets sum
                                    (graph-count (new-degree-sequence ds xs))))))))
```

Figure 1. Common Lisp implementation of the enumeration algorithm given in Theorem 1.1.
REFERENCES

[1] R. Bellman. *Dynamic programming*. Princeton University Press, Princeton, N. J., 1957.

[2] S. A. Choudum. A simple proof of the Erdős-Gallai theorem on graph sequences. *Bull. Austral. Math. Soc.*, 33(1):67–70, 1986.

[3] L. Comtet. *Advanced combinatorics*. D. Reidel Publishing Co., Dordrecht, enlarged edition, 1974.

[4] P. Erdős and T. Gallai. Gráfok előírt fokszámú pontokkal. *Matematikai Lapok*, 11:264–274, 1960.

[5] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10:496–506, 1962.

[6] V. Havel. A remark on the existence of finite graphs. Časopis pro pěstování matematiky (in Czech), 80:477–480, 1955.

[7] The OEIS Foundation Inc. The on-line encyclopedia of integer sequences, 2020.

[8] A. Lew and H. Mauch. *Dynamic Programming: A Computational Tool*. Studies in Computational Intelligence 38. Springer-Verlag Berlin Heidelberg, 2007.

[9] B. D. McKay. Applications of a technique for labelled enumeration. In *Proceedings of the fourteenth Southeastern conference on combinatorics, graph theory and computing (Boca Raton, Fla., 1983)*, volume 40, pages 207–221, 1983.

[10] D. Michie. 'Memo' functions and machine learning. *Nature*, 218:19–22, 1968.

[11] J. W. Moon. *Counting labelled trees*, volume 1969 of *From lectures delivered to the Twelfth Biennial Seminar of the Canadian Mathematical Congress (Vancouver)*. Canadian Mathematical Congress, Montreal, Que., 1970.

[12] G. Pólya. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Math.*, 68(1):145–254, 1937.

[13] G. Pólya and R. C. Read. *Combinatorial enumeration of groups, graphs, and chemical compounds*. Springer-Verlag, New York, 1987. Pólya's contribution translated from the German by Dorothee Aeppli.

[14] R. C. Read. *Some enumeration problems in graph theory*. PhD thesis, University of London, 1959.

[15] R. C. Read. The enumeration of locally restricted graphs. II. *J. London Math. Soc.*, 35:344–351, 1960.

[16] C. Rhodes. SBCL: A sanely-bootstrappable common lisp. In Robert Hirschfeld and Kim Rose, editors, *Self-Sustaining Systems, First Workshop, S3 2008, Potsdam, Germany, May 15-16, 2008, Revised Selected Papers*, volume 5146 of *Lecture Notes in Computer Science*, pages 74–86. Springer, 2008.

[17] Guy Steele. *Common LISP: the language*. Elsevier, 1990.

[18] N. Wormald. Asymptotic enumeration of graphs with given degree sequence. In *Proceedings of the International Congress of Mathematicians—Rio de Janeiro 2018. Vol. IV. Invited lectures*, pages 3245–3264. World Sci. Publ., Hackensack, NJ, 2018.

Istanbul Technical University, Istanbul, Turkey.

Email address: kaygun@itu.edu.tr